UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE ESTUDIOS

Título

Métodos matemáticos para preservar la privacidad en ciencia de datos

Autor/es

Iker Zubillaga Ruiz

Director/es

JÓNATAN HERAS VICENTE

Facultad

Facultad de Ciencia y Tecnología

Titulación

Grado en Matemáticas

Departamento

MATEMÁTICAS Y COMPUTACIÓN

Curso académico

2022-23

publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

UNIVERSIDAD DE LA RIOJA

Facultad de Ciencia y Tecnología

TRABAJO FIN DE GRADO

Grado en Matemáticas

Métodos matemáticos para preservar la privacidad en ciencia de datos

Realizado por:

Iker Zubillaga Ruiz

Tutelado por:

Jónathan Heras Vicente

26 de julio de 2023

Resumen

El presente Trabajo Final de Grado tiene como objetivo abordar los métodos matemáticos utilizados para preservar la privacidad en el campo de la ciencia de datos. En un mundo cada vez más interconectado y tecnológico, el manejo seguro y confidencial de datos se ha vuelto crucial para proteger la privacidad de los individuos. A pesar de los grandes avances en ciencia de datos, uno de los mayores problemas actualmente es la falta de avances en materia de privacidad. En el trabajo, se explorarán diferentes enfoques criptográficos y técnicas de privacidad, con énfasis en el aprendizaje federado, el sistema RSA y el cifrado homomórfico en el que se hará especial hincapié en el sistema criptográfico de Paillier. Estos métodos permiten realizar cálculos en datos cifrados o distribuidos, sin necesidad de revelar información confidencial. Se presentará cada uno de estos enfoques, detallando sus fundamentos matemáticos, ventajas y limitaciones. Además, se incluirán ejemplos prácticos donde estos métodos han demostrado su eficacia en mantener la privacidad de los datos. En conclusión, este Trabajo Final de Grado busca brindar una visión integral de los métodos matemáticos para preservar la privacidad en ciencia de datos, destacando su relevancia en un mundo cada vez más digitalizado y la importancia de abordar adecuadamente la protección de datos personales en un contexto global.

Abstract

This Final Degree Project aims to address the mathematical methods used to preserve privacy in the field of data science. In an increasingly interconnected and technological world, secure and confidential data handling has become crucial to protect the privacy of individuals. Despite great advances in data science, one of the biggest problems today is the lack of progress on privacy. In the paper, different cryptographic approaches and privacy techniques will be explored, with emphasis on federated learning, RSA and homomorphic encryption with special emphasis on the Paillier cryptographic system. These methods allow computations to be performed on encrypted or distributed data, without the need to disclose confidential information. Each of these approaches will be presented, detailing their mathematical foundations, advantages and limitations. In addition, practical examples will be included where these methods have proven to be effective in maintaining data privacy. In conclusion, this Final Degree Project seeks to provide a comprehensive overview of privacy-preserving mathematical methods in data science, highlighting their relevance in an increasingly digitised world and the importance of adequately addressing the protection of personal data in a global context.

Índice general

Capítulo 1

Introducción

La ciencia de datos es un campo interdisciplinar que involucra métodos científicos, procesos y sistemas para extraer conocimiento o un mejor entendimiento de datos en sus diferentes formas, ya sea estructurados o no estructurados. La ciencia de datos es una continuación de algunos campos de análisis de datos como la estadística, la minería de datos, el aprendizaje automático, y la analítica predictiva [1]. Hoy en día las aplicaciones de la ciencia de datos en nuestro día a día son muy amplias y el alcance que puede llegar a tener en un futuro cercano es inimaginable; gran parte de la culpa de esto es la sociedad tecnológica en la que vivimos hoy en día y los millones de datos que manejamos cada instante con nuestros dispositivos electrónicos (*smartphones*, ordenadores, etc) [2].

Muchos de los problemas que queremos resolver hoy con la ciencia de datos requieren acceso a información personal confidencial, ya sea nuestro historial médico, registros financieros o hábitos privados [3]. Todos los días, las personas producimos una gran cantidad de datos en nuestros teléfonos inteligentes, dispositivos electrónicos o equipos médicos; sin embargo, debido a cuestiones de privacidad o de propiedad, los datos para abordar problemas significativos pueden ser limitados y de difícil acceso. Es por esto por lo que surge la siguiente pregunta ¿podemos realizar ciencia de datos sin entrometernos en nuestra privacidad individual? Si es así, ¿qué tecnologías podemos combinar para hacerlo posible?

Tradicionalmente, era necesario transferir los datos a un servidor central para aplicar las técnicas de ciencia de datos, pero esto genera numerosas preocupaciones sobre la privacidad y seguridad de los datos. Los riesgos de fugas de datos y uso indebido han llevado a varias partes del mundo a legislar leyes de protección de datos [4]. Para llevar a cabo la ciencia de datos en dominios que requieren datos privados mientras se respetan las leyes de privacidad de datos y se minimizan los riesgos, los investigadores de aprendizaje automático han aprovechado las soluciones de la investigación de privacidad y seguridad, desarrollando el campo de la ciencia de datos privados y seguros gracias a los avances matemáticos en ese área.

El aprendizaje automático (en inglés *Machine Learning*) privado y seguro está fuertemente inspirado en la criptografía y la investigación de la privacidad [5]. El aprendizaje automático

privado y seguro consiste en una colección de técnicas que permite trabajar en ciencia de datos sin tener acceso directo a los datos y que evita que se almacene inadvertidamente información sensible sobre los mismos.

El aprendizaje automático privado y seguro se lleva a cabo en la práctica mediante una combinación de técnicas, aunque cada método tiene limitaciones y costos. Algunas técnicas serían demasiado onerosas en contextos en los que los propietarios de datos y modelos ya confían entre sí (por ejemplo, cuando los empleados de una empresa son los que hacen uso de datos internos de la propia empresa), mientras que otras serían insuficientemente seguras para contextos que necesitan proteger datos y modelos de la acciones de actores maliciosos (por ejemplo, cuando una empresa de seguros médicos trabaja con los datos privados de los pacientes). Una combinación adecuada de técnicas para un proyecto específico solo puede decidirse una vez que las diversas ventajas y desventajas de las técnicas se comunican claramente a los titulares de los datos y a las partes interesadas clave del proyecto.

El objetivo de mi trabajo es dar a conocer los diferentes métodos matemáticos para preservar la privacidad en ciencia de datos y tratar de explicar los conceptos e ideas que surgen alrededor de ellos. Para ello iré desarrollando punto por punto los diferentes métodos e iré introduciendo los conceptos ligados a cada método, empezando por los más básicos y llegando hasta algunos más técnicos.

La fuente principal que he utilizado para desarrollar este trabajo ha sido un artículo [6] de un blog llamado *OpenMined*, en el que se trataban la mayoría de los temas contenidos en mi trabajo y en el que había abundantes referencias a otros muchos temas relacionados.

He organizado la memoria del trabajo de modo que en este primer capítulo se contextualiza la problemática a la que nos enfrentamos. En este primer capítulo también se dan a conocer algunos conceptos que se trabajarán más adelante, y se da una pequeña explicación de cual será la estructura del trabajo. En el capítulo 2 se introducen nociones sobre todo teóricas a cerca del aprendizaje federado y el aprendizaje automático. En este también se explican los dos tipos de aprendizaje automático que existen. En el capítulo 3 se presenta el sistema RSA, para el cual se necesita introducir previamente varias nociones matemáticas que van desde definiciones de estructuras algebraicas, hasta bases para el cifrado, pasando por la teoría de números. A continuación se da una prueba matemática de el correcto funcionamiento el algoritmo de cifrado del sistema RSA. En el último capítulo, el capítulo 4, se explica teóricamente qué es el cifrado homomórfico en sus diferentes formas y se hace especial hincapié en el criptosistema de Paillier y su importancia en el contexto del aprendizaje automático seguro. El criptosistema de Paillier es un sistema de cifrado homomórfico parcial, es decir, pertenece a uno de los 3 tipos de cifrado homomórfico que existen.

Por último, la memoria concluye con un resumen de las principales conclusiones derivadas del trabajo realizado. y la mención de toda la bibliografía utilizada para haberlo podido llevar a cabo.

Capítulo 2

Aprendizaje federado

La idea de este capítulo es comenzar a entender qué tipo de soluciones (métodos) se pueden utilizar para preservar la privacidad en ciencia de datos. Para ello, nos haremos conocedores de los conceptos más básicos de esta teoría y con ellos, nos intentaremos acercar de la forma más simple posible al aprendizaje federado.

2.1. Introducción

Antes de empezar, podríamos resumir el aprendizaje federado como: entrenar un modelo de aprendizaje automático con datos que se almacenen en diferentes dispositivos o servidores en todo el mundo, sin tener que recopilarlos de forma centralizada.

El aprendizaje federado es una solución algorítmica que permite el entrenamiento de modelos de aprendizaje automático mediante el envío de copias de un modelo al lugar donde residen los datos, realizando allí el entrenamiento, y eliminando así la necesidad de mover grandes cantidades de datos a un servidor central con fines de formación.

2.2. Nociones previas

Antes de meternos en materia y explicar el funcionamiento del aprendizaje federado enunciaremos una serie de conceptos previos.

Definición 2.1 (**Modelo**). *Un **modelo** en aprendizaje automático es una función que ante unas entradas produce unas salidas.*

Por ejemplo un modelo de regresión puede predecir el precio, p, de una casa a partir de tres atributos (por ejemplo, los m^2, denotado por x, la antiguedad, denotada por y, y el número de habitaciones, denotado por z). Es decir, es una función:

$$\begin{array}{rccc} f_W\colon & \mathbb{R}^3 & \longrightarrow & \mathbb{R} \\ & (x,y,z) & \longmapsto & f_W(x,y,z) \end{array}$$

Dicha función está parametrizada por unos pesos W, que son definidos a partir de un conjunto de entrenamiento mediante un proceso de optimización.

La cualidad de estos modelos es que se han entrenado para reconocer determinados tipos de patrones. Se puede entrenar un modelo con un conjunto de datos, y que te proporcione un algoritmo que puedas usar para averiguar y obtener información de esos datos. Una vez entrenado el modelo, puedes usarlo para desglosar los datos que no has visto antes y realizar predicciones sobre estos.

Por ejemplo, supongamos que quieres compilar una aplicación que pueda reconocer las emociones de un usuario en función de sus expresiones faciales. Para entrenar este modelo puedes proporcionarle imágenes de caras que estén etiquetadas con una emoción determinada y, a continuación, puedes usar ese modelo en una aplicación que pueda reconocer cualquier emoción del usuario. En la figura 2.1, se ofrece una representación visual del proceso de entrenamiento y predicción utilizando un modelo de aprendizaje automático.

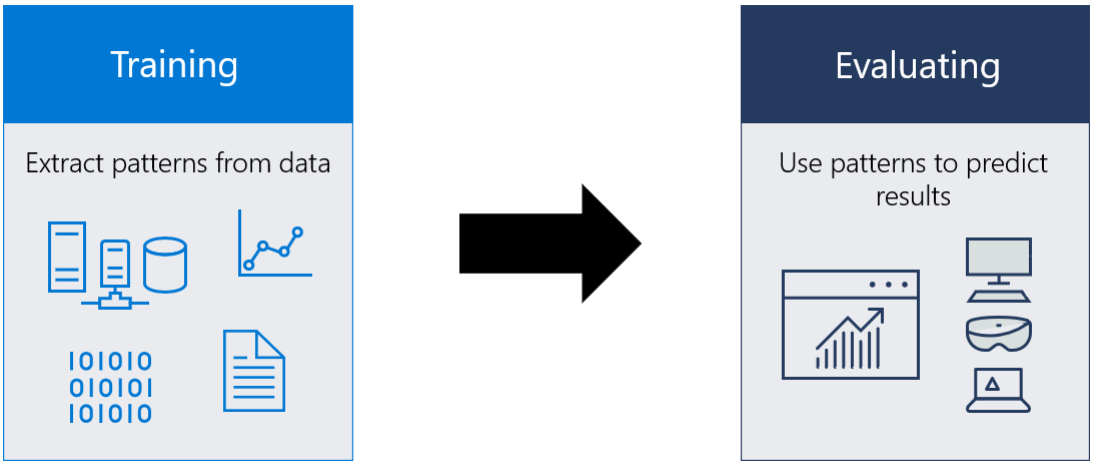

Figura 2.1: Funcionamiento de un modelo de aprendizaje automático [7].

Dentro de los modelos de aprendizaje automático, podemos destacar los modelos de aprendizaje supervisado.

Definición 2.2 (Aprendizaje supervisado)**.** *Consideremos a una pareja* $(x, y) \in X \times Y$*, donde* X *e* Y *son respectivamente espacios de salida y llegada, que representan a un objeto junto a su etiqueta. Normalmente* $X = \mathbb{R}^n$ *con* $n \in \mathbb{N}$*, e* $Y = \mathbb{R}$ *para problemas de regresión, o* Y *es un conjunto discreto de elementos para problemas de clasificación. Para el* ***aprendizaje supervisado*** *contamos con un conjunto de entrenamiento:*

$$\{(x^{(i)}, y^{(i)}), \ldots, (x^{(m)}, y^{(m)})\},$$

con $x^{(i)} \in X$ *e* $y^{(i)} \in Y$*. El conjunto de entrenamiento se utiliza para entrenar los pesos de un algoritmo de aprendizaje mediante un proceso de optimización. Este algoritmo nos otorga una*

aproximación a la función de dependencia desconocida entre objetos y respuestas; es decir, entre las x's y las y's.

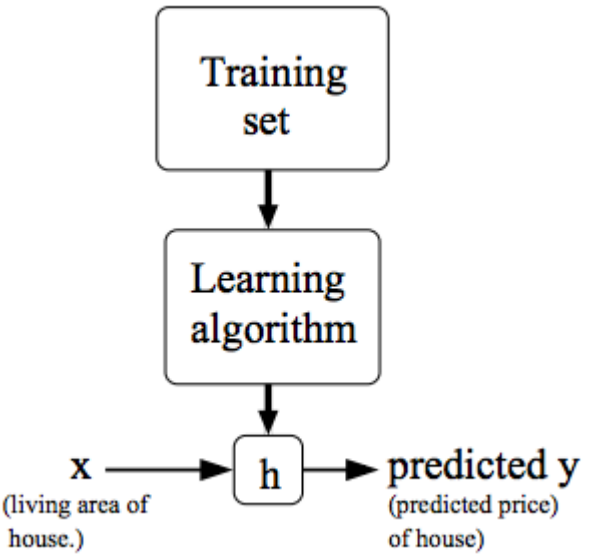

Figura 2.2: Funcionamiento del aprendizaje supervisado [8].

Como se puede ver en el diagrama de la figura 2.2, el aprendizaje supervisado consta de dos pasos esenciales:

1. Entrenamiento, donde usamos el conjunto de entrenamiento como entrada de un algoritmo que nos permita crear un modelo de aprendizaje automático.

2. Prueba, donde utilizamos el modelo obtenido en el paso anterior para realizar nuevas predicciones con objetos nuevos.

Todo este proceso se llama aprendizaje supervisado, ya que al conocer las respuestas de cada ejemplo del conjunto de entrenamiento, podemos corregir el modelo producido por el algoritmo. En concreto, se supervisa el entrenamiento del algoritmo, corrigiendo los parámetros del mismo según los resultados que obtengamos, de forma iterativa.

2.3. Aprendizaje federado

El aprendizaje federado se basa en que los datos permanezcan en sus respectivos dispositivos de origen, también conocidos como clientes, que reciben una copia del modelo global del servidor central. Esta copia del modelo global se entrena localmente con los datos de cada dispositivo. Los pesos del modelo se actualizan a través del entrenamiento local y luego la copia local se envía de vuelta al servidor central. Una vez que el servidor recibe el

modelo actualizado, procede a agregar las actualizaciones, mejorando el modelo global sin revelar ninguno de los datos privados en los que se entrenó. En la figura 2.3, se ofrece una representación visual del proceso de entrenamiento de un modelo de aprendizaje federado.

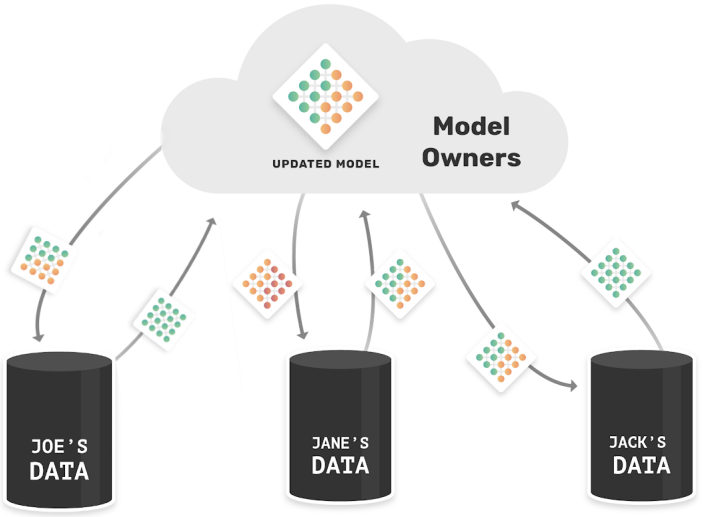

Figura 2.3: Funcionamiento del aprendizaje federado [9].

Algunas de las ventajas del aprendizaje federado son:

- Los investigadores pueden entrenar modelos utilizando datos privados y confidenciales sin tener que preocuparse por el manejo de los datos: los datos permanecen en el dispositivo y solo las actualizaciones del modelo aprendido se transfieren entre el laboratorio y los propietarios de los datos.
- Cumple con las normas de protección de datos como *GDPR* (Reglamento General de Protección de Datos) [10].

Las desventajas más notables del aprendizaje federado son las siguientes:

- El costo de implementar el aprendizaje federado es más alto que recopilar la información y procesarla de forma centralizada, especialmente durante las primeras fases de I+D, cuando el método y el proceso de capacitación aún se están iterando.
- Requiere que los propietarios de los datos realicen cálculos en el dispositivo que contiene los datos; para algunos dispositivos con capacidad de cálculo limitada, esto puede no ser posible o económico.
- La implementación de aprendizaje federado no es suficiente para garantizar la privacidad, ya que las actualizaciones del modelo en un servidor "honesto pero curioso" podrían servir para reconstruir las muestras a partir de las cuales se calcularon las actualizaciones. Estas muestras se pueden usar para inferir información privada y confidencial.

Debido a esta última desventaja, se requiere mezclar la implementación de aprendizaje federado con otras técnicas. Aquí es donde la computación multiparte segura, el cifrado homomórfico y la privacidad diferencial vienen a brindar mayores garantías de seguridad a los propietarios de los datos [11].

El aprendizaje federado, en esencia, es cualquier tipo de aprendizaje automático que ocurre cuando el modelo se lleva a los datos en lugar de los datos al modelo. Hay dos tipos de aprendizaje federado: centrado en modelos y centrado en datos.

2.3.1. Aprendizaje federado centrado en el modelo

En el aprendizaje federado "*centrado en el modelo*", el modelo está preconfigurado (los pesos están definidos) y alojado en la nube. Luego aparecen clientes efímeros, que descargan el modelo, lo mejoran y cargan una nueva versión a la web. Esto suele ocurrir durante un largo período de tiempo (días, semanas, incluso meses). Se encuentra más comúnmente cuando se usan dispositivos como teléfonos inteligentes para mejorar pasivamente un modelo de inteligencia artificial con el tiempo, como un modelo dentro de una aplicación. Un gran ejemplo de aprendizaje federado centrado en el modelo es la aplicación móvil GBoard de Google que aprende las preferencias del usuario y su estilo de escritura a lo largo del tiempo [12].

2.3.2. Aprendizaje federado centrado en los datos

En el aprendizaje federado "*centrado en datos*", el conjunto de datos está preconfigurado (su esquema, atributos y parámetros de seguridad están definidos) y alojado en la nube. Los modelos efímeros aparecen y realizan el entrenamiento localmente de una manera experimental. Si bien esta forma de aprendizaje federado es menos común, es ideal para la exploración científica que el aprendizaje federado centrado en modelos, ya que refleja más fielmente el flujo de trabajo estándar de la ciencia de datos.

Para ver un ejemplo, supongamos que desea entrenar un modelo para detectar nódulos cancerosos en tomografías computarizadas. Si no trabaja en un gran hospital, puede ser muy difícil o simplemente imposible obtener un conjunto de datos que sea suficiente para entrenar un modelo. Afortunadamente, con el aprendizaje federado centrado en los datos, un propietario de datos puede alojar sus conjuntos de datos en un servidor central, y el científico de datos puede enviar solicitudes de entrenamiento e inferencia contra esos datos.

A lo largo de este capítulo hemos visto que el aprendizaje federado permite entrenar modelos de manera descentralizada, donde los datos permanecen en sus ubicaciones originales y solo se comparten actualizaciones de los modelos. Esto es especialmente útil cuando los datos son sensibles o confidenciales y no se pueden compartir libremente entre diferentes entidades. Sin embargo, una preocupación clave en el aprendizaje federado es asegurar que los datos no se revelen durante el proceso de entrenamiento.

Aquí es donde entra en juego el cifrado homomórfico. Este tipo de cifrado permite realizar cálculos en datos cifrados sin necesidad de descifrarlos previamente, lo que garantiza que los datos permanezcan protegidos durante todo el proceso. Sin embargo, implementar esquemas de cifrado homomórfico puede ser complejo y costoso computacionalmente.

Por lo tanto, la combinación del aprendizaje federado con el cifrado homomórfico se convierte en una solución poderosa para abordar los desafíos de privacidad y seguridad en el aprendizaje automático. El cifrado homomórfico permite realizar operaciones criptográficas en datos cifrados compartidos entre múltiples participantes, lo que a su vez protege la privacidad de los datos y minimiza la exposición de información confidencial. La necesidad de esquemas de cifrado en esta combinación radica en asegurar que los datos permanezcan seguros y en garantizar la confidencialidad de la información durante todo el proceso de entrenamiento y predicción.

Es por ello que en el siguiente capítulo trataremos el sistema RSA, uno de los sistemas de cifrado más usados a la hora de implementar un sistema de cifrado. Eso nos servirá para sentar las bases del cifrado y para poder trabajar con un esquema de cifrado homomórfico en el capítulo 4.

Capítulo 3

Sistema RSA

El sistema RSA (Rivest-Shamir-Adleman) [13], es un sistema de criptografía de clave pública, que utiliza el problema de la factorización de números enteros para garantizar la seguridad del cifrado. En particular, la seguridad del RSA se basa en el hecho de que encontrar los factores primos de un número grande es un problema computacionalmente difícil, mientras que multiplicar números grandes es relativamente fácil.

Antes de meternos en materia, enunciaremos una serie de conceptos previos sobre estructuras algebraicas, teoría de números y las bases de cifrado. Por lo tanto, este capítulo se organizará en 3 secciones dedicadas a nociones previas y una última sección en la que haremos uso de esas nociones para presentar el sistema RSA.

3.1. Estructuras algebraicas

Comenzaremos viendo la definición de grupo, necesaria para entender las definiciones que vienen a continuación. Las nociones presentadas en esta sección están extraídas de los apuntes de la asignatura Estructuras Algebraicas de la Universidad de la Rioja.

Definición 3.1 (**Grupo**). *Sea G un conjunto no vacío, y sea $\cdot : G \times G \to G$ una operación binaria interna llamada producto o multiplicación, se dice que $(G, \cdot)$ es un* ***grupo*** *si:*

1. *El producto es asociativo:*

$$(a \cdot b) \cdot c = a \cdot (b \cdot c) \forall a, b, c \in G.$$

2. *Existe $e \in G$ tal que*

$$\forall a \in G,\ e \cdot a = a = a \cdot e.$$

   Diremos que e es el elemento neutro de G.

3. *Todo elemento tiene inverso, es decir, para todo $a \in G$ existe $b \in G$ tal que*

$$a \cdot b = e = b \cdot a$$

Denotaremos al inverso de a mediante a^{-1}.

El orden de un grupo G es su cardinal: $|G|$.

Ahora, veamos la definición de anillo, la cual usaremos más adelante para definir los teoremas que nos proporcionarán los esquemas de cifrado [14].

Definición 3.2 (**Anillo**). *Sea A un conjunto no vacío, y sean $+$ y $\cdot$ dos operaciones binarias internas $A \times A \to A$. Se dice que el conjunto $(A, +, \cdot)$ es un* ***anillo*** *cuando se cumplen las siguientes propiedades:*

1. *La operación $+$ es* asociativa*:*

$$(a + b) + c = a + (b + c)\ \forall a, b, c \in A.$$

2. *La operación $+$ tiene un* elemento neutro *(llamado cero, y denotado 0 ó 0_A)):*

$$0 + a = a = a + 0\ \forall a \in A.$$

3. *Cada $a \in A$ posee un* elemento opuesto *(denotado por $-a$):*

$$a + (-a) = 0 = (-a) + a.$$

4. *La operación $+$ es* conmutativa*:*

$$a + b = b + a\ \forall a, b, c \in A.$$

5. *La operación $\cdot$ es* asociativa*, se cumple*

$$(a \cdot b) \cdot c = a \cdot (b \cdot c)\ \forall a, b, c \in A.$$

6. *La operación $\cdot$ cumple la* propiedad distributiva *respecto de $+$ tanto a la izquierda como a la derecha:*

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c),\ \forall a, b, c \in A.$$
$$(a + b) \cdot c = (a \cdot c) + (b \cdot c),\ \forall a, b, c \in A.$$

Un conjunto de objetos, como los números enteros, se considera un anillo si podemos realizar sobre él una operación "similar a la suma" y otra "similar a la multiplicación".

Definición 3.3 (**Anillo unitario**). *Un* ***anillo es unitario*** *si existe un elemento neutro para $\cdot$ distinto de 0 (se denota 1 ó 1_A y se llama la* ***unidad o identidad de*** *A), es decir, $\forall a \in A$, se cumple que*

$$1 \cdot a = a = a \cdot 1.$$

Definición 3.4 (**Grupo multiplicativo**). *Si* $(A, +, \cdot)$ *es anillo unitario, entonces* $(A^\times, \cdot)$ *es un grupo, llamado* ***grupo multiplicativo*** *o de unidades del anillo A, donde*

$$A^\times = \{a \in A | \exists b \in A \text{ con } a \cdot b = 1 = b \cdot a\}.$$

A los elementos de $A^\times$ *se les llama unidades o elementos invertibles de A.*

Ejemplos de anillos son, el conjunto de todos los números enteros, pero también el conjunto de todos los posibles restos de una división de un número entero por un n determinado, conocido como "todos los números enteros módulo n". Por ejemplo el conjunto de los enteros módulo 5,

$$\mathbb{Z}_5 = \{0, 1, 2, 3, 4\},$$

es un anillo porque tenemos tanto el 0 como el 1 y la operación adición que tras sumar dos números toma como resultado el resto de la división entera por 5 de esa suma. Así que $3 + 4 = 7 \Rightarrow 7 \mod 5 = 2$. La multiplicación funciona de la misma manera, $4 \times 4 = 16 \Rightarrow 16 \mod 5 = 1$. El anillo de los números enteros, es decir, $\mathbb{Z}$ es un anillo unitario, ya que cumple con la definición de anillo con la adición y la multiplicación y tiene como elemento neutro para la multiplicación precisamente el 1.

Otro anillo que usaremos más adelante es $\mathbb{Z}_n$, que es el de los enteros módulo n con la suma, $\mathbb{Z}_n$ también se denota como $\mathbb{Z}/n\mathbb{Z}$, $\mathbb{Z}/(n)$ o $\mathbb{Z}/n$. Además, podemos definir

$$n\mathbb{Z} = \{nk | k \in \mathbb{Z}\}$$

con $n \in \mathbb{N} \cup \{0\}$.

$$\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}$$

es un anillo cociente (con la suma y producto módulo n). Sus elementos son $\overline{k} = k + n\mathbb{Z}$ con $k \in \mathbb{Z}$.

Antes de introducir otro anillo importante en nuestro contexto, vamos a dar las definiciones de número primo y números coprimos o primos dos a dos.

Definición 3.5 (**Número primo**). *Dado* $n \in \mathbb{N}$*, decimos que* n *es* ***primo*** *si y solo si* $\nexists m \in \mathbb{N}$ *con* $1 < m < n$ *tal que* $mcd(n, m) \neq 1$*. A su vez, podemos decir que cuando* $mcd(n, m) = 1$*,* n *y* m *son coprimos.*

Ahora, veamos otro anillo que tiene algunas propiedades que nos serán de gran utilidad a la hora de definir y demostrar teoremas imprescindibles en el ámbito del cifrado homomórfico.

Definición 3.6 ($\mathbb{Z}^*_{n^2}$). *Dado* $n \in \mathbb{N}$*, el conjunto* $\mathbb{Z}^*_{n^2}$ *se define como el conjunto de enteros positivos menores que* n^2 *y que son coprimos con* n^2*. Formalmente, si* n *es un número entero positivo, entonces* $\mathbb{Z}^*_{n^2}$ *se define como:*

$$\mathbb{Z}^*_{n^2} = \{a \in \mathbb{Z} | 1 \leq a < n^2, \ mcd(a, n^2) = 1\}$$

En otras palabras, $\mathbb{Z}^*_{n^2}$ está compuesto por los enteros positivos menores que n^2 que no tienen factores primos en común con n^2. Estos enteros se consideran "coprimos" con n^2 o "relativamente primos" a n^2.

3.2. Teoría de números

En este punto del capítulo, veamos la importancia de la teoría de números mediante la definición de distintos conceptos, el primero que veremos es el de congruencia [15].

Congruencia es un término usado en la teoría de números, para designar que dos números enteros a y b tienen el mismo resto al dividirlos por un número natural $m \neq 0$, llamado módulo. Para definirla, previamente vamos a definir la notación de la relación de divisibilidad denotado por $|$. Para decir que m divide a k, escribiremos $m|k$.

Definición 3.7 (**Congruencia**). *Sea n un entero positivo y sean $a, b \in \mathbb{Z}$, se dice que a es* ***congruente*** *con b, y se denota por*

$$a \equiv b \pmod{n}$$

si n divide exactamente a la diferencia de a y b, es decir,

$$n \mid a - b$$

o lo que es lo mismo, a y b dejan el mismo resto en la división por n.

Además, también se puede afirmar que a se puede escribir como la suma de b y un múltiplo de m, pues si: $m \mid a - b$ entonces $\exists k \in \mathbb{Z}$, tal que $mk = a - b$ y por tanto $a = b + km$.

En otro caso, diremos que a es incongruente *con b.*

El término congruencia se utiliza además con un sentido de identidad matemática, como ejemplo de este uso tenemos el pequeño teorema de Fermat [16] que asegura que para cada primo p y cada entero a no divisible por p tenemos la congruencia:

$$a^{p-1} \equiv 1 \pmod{p}. \tag{3.1}$$

Una vez vista la definición de congruencia, veamos una proposición que nos será muy útil a la hora de trabajar con ellas.

Proposición 3.1. *Sea un entero positivo $a \neq 0$, y otro entero $m \geq 2$ tal que $mcd(a, m) = 1$. Entonces, la ecuación*

$$ax \equiv b \pmod{m}$$

tiene una única solución módulo m.

Demostración. Probemos en primer lugar la existencia de dicha solución. Los m números $a, 2a, 3a, \ldots, ma$ son incongruentes dos a dos módulo m. En efecto, si $ja \equiv ka \pmod{m}$, entonces $m|(j-k)a$. Pero como $\mathrm{mcd}(a, m) = 1$, se deduce que $m|(j-k)$, lo que no puede ser salvo que $j = k$. De este modo, al variar x entre 1 y m, mediante ax se van obteniendo los m restos módulo m que puede haber. En particular, para algún x aparecerá $ax \equiv b \pmod{m}$.

Finalmente, y en cuanto a la unicidad de la solución se refiere, el motivo es similar. Suponemos que r y s son soluciones de la ecuación $ax \equiv b \pmod{m}$, entonces

$$a(r-s) \equiv 0 \pmod{m}$$

Lo cual cual nos permite afirmar que

$$m|(r-s) \Rightarrow r \equiv s \pmod{m}.$$

A todos los efectos, esa ecuación con congruencias sería la misma. Por lo tanto, hemos conseguido probar la unicidad y queda probada la proposición. □

La anterior proposición nos sirve para introducir la definición de inverso de a módulo m. Cuando decimos que a' es el inverso de a módulo m, significa que a' es el número que, al ser multiplicado por a y luego reducido módulo m, produce el residuo de la división de a entre m que es 1. Matemáticamente, si a' es el inverso de a módulo m, entonces se cumple la siguiente relación:

$$aa' \equiv 1 \pmod{m}.$$

En otras palabras, a' es el número tal que su producto con a deja un residuo de 1 cuando se divide por m.

Corolario 3.2. *Si a y m son enteros con $m \geq 2$ y $mcd(a,m) = 1$ entonces $\exists | b \in \mathbb{N}$ tal que b es inverso de a módulo m*

Demostración. Vemos que b es la solución de la ecuación

$$ax \equiv 1 \pmod{m}$$

que sabemos por 3.1 que existe y es única módulo m. □

Ahora, veremos un teorema que nos será de gran utilidad a la hora de demostrar por ejemplo el Teorema de Carmichael o la corrección del sistema RSA.

Teorema 3.3 (**Teorema Chino del Resto**). *Sean $m_1, m_2, \ldots, m_r$ enteros positivos, primos entre si dos a dos y sean cualesquiera enteros arbitrarios $b_1, b_2, \ldots, b_r$. Entonces, el sistema de congruencias*

$$\begin{cases} x \equiv b_1 \pmod{m_1} \\ x \equiv b_2 \pmod{m_2} \\ \ldots \\ x \equiv b_r \pmod{m_r} \end{cases} \tag{3.2}$$

posee exactamente una solución módulo el producto $m_1, m_2, \ldots, m_r$.

Demostración. Sea $M = m_1 m_2 \cdots m_r$ y $M_j = M/m_j$, $j = 1, 2, \ldots, r$. Por ser $\text{mcd}(m_j, m_k) = 1$ para $j \neq k$, se deduce fácilmente que $\text{mcd}(M_j, m_j) = 1$ para todo j. Entonces, el corolario 3.2 prueba que cada M_j tiene un único inverso $M_j^{'}$ módulo m_j (a menudo se usa a^{-1} para denotar el inverso de a módulo m; pero este inverso depende del módulo, y aquí estamos empleando módulos distintos, o sea que la notación $(\cdot)^{-1}$ hubiera resultado confusa). Sea ahora el entero

$$x = b_1 M_1 M_1^{'} + b_2 M_2 M_2^{'} + \cdots + b_r M_r M_r^{'},$$

y analicemos su comportamiento módulo cada uno de los m_j. Es claro que $M_k \equiv 0 \pmod{m_j}$ para $j \neq k$, luego todos los sumandos de x son nulos módulo m_j, salvo quizás el j-ésimo, así que

$$x \equiv b_j M_j M_j^{'} \pmod{m_j} \equiv b_j \pmod{m_j}.$$

En consecuencia, hemos probado que x satisface todas las congruencias del sistema.

Nos falta demostrar que la solución es única módulo M, pero esto es rutinario. En primer lugar, veamos que si x es solución e y cumple que $y \equiv x \pmod{M}$, también y es solución. En efecto, en esas condiciones tendremos $M|(x-y)$ y $m_j|M$, luego $m_j|(x-y)$ y por tanto $y \equiv x \pmod{b_j} \equiv b_j \pmod{b_j}$ para todo j. Comprobemos ahora el recíproco, es decir, que todas las posibles soluciones son de esa manera. Si dos enteros x e y son soluciones del sistema, deberán cumplir $x \equiv y \pmod{m_j}$ para todo j, o sea, $m_j|(x-y)$. Como los m_j son primos entre si dos a dos, tendremos también $M|(x-y)$, o sea $x \equiv y \pmod{M}$. □

Veamos una función que será realmente útil a la hora de trabajar con los conceptos mencionados anteriormente, ya que posee algunas propiedades muy interesantes. La función de Euler, denotada como φ y también conocida como función phi de Euler, indicatriz de Euler o función totiente, desempeña un papel fundamental en la teoría de números [16].

Definición 3.8 (Función φ de Euler). *La **función φ de Euler**, $\varphi : \mathbb{N} \to \mathbb{N}$, se define como*

$$\varphi(n) = |\{k \in \mathbb{Z} | 1 \leq k < n \wedge mcd(k, n) = 1\}|$$

es decir, $\varphi(n)$ propociona la cantidad de números enteros positivos menores que n y coprimos con n.

La función φ es de gran importancia, principalmente porque proporciona información sobre el tamaño del grupo multiplicativo de enteros módulo n. Específicamente, $\varphi(n)$ representa el orden de $\mathbb{Z}_n^*$. De hecho, en combinación con el teorema de Lagrange [16], que establece los posibles tamaños de subgrupos de un grupo, la función φ proporciona una demostración del Teorema de Euler, que veremos más adelante.

Veamos algunas de las propiedades que tiene la función φ de Euler [17].

Proposición 3.4. *La función φ de Euler cumple que:*

1. $\varphi(1) = 1$.

2. *Si p es primo, tenemos*
$$\varphi(p) = p - 1.$$

3. *Si p es primo y k es un número natural, tenemos que*
$$\varphi(p^k) = (p-1)p^{k-1}.$$

4. *Sean m y n dos enteros positivos coprimos, entonces*
$$\varphi(mn) = \varphi(m)\varphi(n).$$

5. *Se cumple que $\forall n \in \mathbb{N}$ y $n \geq 1$,*
$$\varphi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right).$$

Demostración. 1. Es trivial, ya que se sigue de la definición que el elemento 1 solo puede ser coprimo consigo mismo.

2. Se demuestra fácilmente, porque si $n = p$ (con p primo), es claro que $\varphi(p) = p - 1$, pues todos los números menores que p son coprimos con p (es más, es evidente que se verifica la siguiente equivalencia: n es primo si y solo si $\varphi(n) = n - 1$).

3. Debemos observar que si $n = p^k$ (potencia de un primo), los únicos números menores o iguales que p^k que no son primos con él son los que tienen algún factor p, es decir, $1p, 2p, 3p, \ldots, (p^{k-1}-1)p$ y $p^{k-1}p$, que son p^{k-1} números. Los $p^k - p^{k-1}$ números restantes son coprimos con p^k; por tanto
$$\varphi(p^k) = p^k - p^{k-1} = (p-1)p^{k-1}.$$

4. Para demostrar esta propiedad, primero veamos el siguiente lema, que está demostrado en [18].

   Lema 3.5. *Sean n_1 y n_2 dos enteros coprimos. Un entero positivo $x \leq n_1 n_2$ es coprimo con $n_1 n_2$ si y solo si*
$$\begin{aligned} mcd(x \mod n_1, n_1) &= 1 \quad y \\ mcd(x \mod n_2, n_2) &= 1. \end{aligned}$$

Ahora, vamos a demostrar que

$$\varphi(n_1 n_2) = \varphi(n_1)\varphi(n_2).$$

Sea A el conjunto de enteros positivos no mayores que n_1 que son coprimos con n_1, definimos

$$A = \{x \in \mathbb{Z} | 1 \leq x \leq n_1 \text{ y } \text{mcd}(x, n_1) = 1\}$$

Definimos B y C de forma similar

$$B = \{x \in \mathbb{Z} | 1 \leq x \leq n_2 \text{ y } \text{mcd}(x, n_2) = 1\}$$

$$C = \{x \in \mathbb{Z} | 1 \leq x \leq n_1 n_2 \text{ y } \text{mcd}(x, n_1 n_2) = 1\}.$$

Notar que por definición $\varphi(n_1) = |A|, \varphi(n_2) = |B|$ y $\varphi(n_1 n_2) = |C|$. El lema 3.5 nos dice que existe una biyección entre $A \times B$ y C donde $A \times B$ se define como

$$A \times B = \{(a, b) | a \in A \text{ y } b \in B\}.$$

Por tanto, $|A \times B| = |C|$. Sabemos que

$$|A \times B| = |A||B| = \varphi(n_1)\varphi(n_2),$$

por lo que $|C| = \varphi(n_1)\varphi(n_2)$. Utilizando el hecho de que $|C| = \varphi(n_1 n_2)$ obtenemos

$$\varphi(n_1 n_2) = \varphi(n_1)\varphi(n_2).$$

5. Probaremos esta propiedad utilizando la propiedad anterior. Para $n = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$ se tiene

$$\begin{aligned}
\varphi(n) &= \varphi(p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}) \\
&= \varphi(p_1^{a_1})\varphi(p_2^{a_2}) \cdots \varphi(p_k^{a_k}) \\
&= (p_1^{a_1} - p_1^{a_1 - 1})(p_2^{a_2} - p_2^{a_2 - 1}) \cdots (p_k^{a_k} - p_k^{a_k - 1}) \\
&= p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k} \left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right) \cdots \left(1 - \frac{1}{p_k}\right) \\
&= n \prod_{p|n} \left(1 - \frac{1}{p}\right)
\end{aligned}$$

con lo que hemos probado la última propiedad.

□

Teorema 3.6. *Sea $\mathbb{Z}_{n^2}^*$, con $n \in \mathbb{N}$, tenemos que:*

$$|\mathbb{Z}_{n^2}^*| = \varphi(n^2) = n\varphi(n).$$

Demostración. Vamos a tratar de demostrar la igualdad $|\mathbb{Z}_{n^2}^*| = \varphi(n^2) = n\varphi(n)$ paso a paso:

1. Es trivial gracias a la definición de φ de Euler.

2. $\varphi(n^2) = n\varphi(n)$: Es consecuencia de la tercera y cuarta propiedad de la función phi de Euler. Al ser $n = pq$, con p y q primos, tenemos que $\varphi(n^2) = \varphi(p^2q^2) = \varphi(p^2)\varphi(q^2)$ por la propiedad 4. Y, por la propiedad 3, al ser $\varphi(p^2) = (p-1)p^{2-1} = p(p-1) = p\varphi(p)$, si hacemos lo mismo con q, nos queda:

$$\varphi(n^2) = pq\varphi(p)\varphi(q) = pq\varphi(pq) = n\varphi(n).$$

En conclusión, hemos demostrado que $|\mathbb{Z}_{n^2}^*| = \varphi(n^2) = n\varphi(n)$. Esto significa que el grupo multiplicativo de los enteros módulo n^2 tiene la misma cardinalidad que $\varphi(n^2)$, que a su vez es igual a $n\varphi(n)$. □

Ejemplo 3.1. *Podemos ver un ejemplo concreto de lo anterior, tomando los valores: $p = 2$ y $q = 3$, entonces $n = pq = 6$. Calculemos los elementos y las cardinalidades de los conjuntos involucrados:*

$\mathbb{Z}_{n^2}^*$: Este conjunto está compuesto por los enteros positivos menores que n^2 y que son coprimos con n^2. En este caso, $n^2 = 6^2 = 36$. Para determinar los elementos de $\mathbb{Z}_{36}^*$, debemos encontrar los enteros positivos menores que 36 que no tengan factores primos en común con $36 = 2^2 \times 3^2$. Estos números son: 1, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31 y 35. Por lo tanto, $|\mathbb{Z}_{36}^*| = 12$.

$\varphi(n^2)$: La función phi de Euler aplicada a $n^2 = 6^2 = 36$ cuenta la cantidad de enteros positivos menores que 36 y coprimos con 36. En este caso, $\varphi(36) = 12$.

$n\varphi(n)$: Multiplicando $n = 6$ por

$$\varphi(n) = \varphi(6) = \varphi(2 \times 3) = \varphi(2)\varphi(3) = (2-1) \times (3-1) = 1 \times 2 = 2.$$

Con lo que nos queda $n\varphi(n) = 6 \times 2 = 12$.

Podemos observar que efectivamente se cumple la igualdad $|\mathbb{Z}_{n^2}^*| = \varphi(n^2) = n\varphi(n)$ en este ejemplo particular.

Definición 3.9 (**Sistema residual reducido módulo** n)**.** *Un **sistema residual reducido módulo** n es todo conjunto de $\varphi(n)$ enteros, incongruentes módulo n dos a dos, cada uno de ellos primo con n.*

Teorema 3.7. *Si dado $n \in \mathbb{N}$ tal que $\{a_1, a_2, \dots a_{\varphi(n)}\}$ es un sistema residual reducido módulo n y dado $k \in \mathbb{N}$ tal que $mcd(k, n) = 1$, entonces $\{ka_1, ka_2, \dots ka_{\varphi(n)}\}$ es también un sistema residual reducido módulo n.*

Demostración. Ningún par de números ka_i es congruente módulo n, siguiendo la hipótesis del teorema. Además, puesto que $mcd(a_i, n) = mcd(k, n) = 1$, tenemos $\text{mcd}(ka_i, n) = 1$, debido a una propiedad básica del mcd. Luego, cada ka_i es primo con n. □

En teoría de números, el Teorema de Euler [16], también conocido como Teorema de Euler-Fermat, es una generalización del pequeño Teorema de Fermat, y como tal afirma una proposición sobre la divisibilidad de los números enteros.

Teorema 3.8 (**Teorema de Euler**). *Dados $a, n \in \mathbb{Z}$ coprimos, entonces*

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

donde $\varphi(n)$ es la función φ de Euler.

Demostración. Sea $\{b_1, b_2, \ldots b_{\varphi(n)}\}$ un sistema residual reducido módulo n. Entonces $\{ab_1, ab_2, \ldots, ab_{\varphi(n)}\}$ es también un sistema residual reducido módulo n, siguiendo el Teorema 3.7, ya que por hipótesis, al ser a, n coprimos, tenemos que mcdn$(a, n) = 1$. Por lo tanto el producto de todos los enteros del primer conjunto es congruente con el producto de los del segundo conjunto. Por consiguiente

$$b_1 b_2 \cdots b_{\varphi(n)} \equiv a^{\varphi(n)} b_1 \cdots b_{\varphi(n)} \pmod{n}. \tag{3.3}$$

Como $\mathrm{mcd}(b_j, n) = 1$ para cada j, también $\mathrm{mcd}(b_1 b_2 \cdots b_{\varphi(n)}, n)$=1. Esto implica que

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

En efecto, al ser $b_1 b_2 \cdots b_{\varphi(n)}(1 - a^{\varphi(n)})$ múltiplo de m y $\mathrm{mcd}(b_1 b_2 \cdots b_{\varphi(n)}, n)$=1, el factor $1 - a^{\varphi(n)}$ debe ser forzosamente múltiplo de n. (Un argumento alternativo consiste en multiplicar ambos lados de 3.3 por $b_1' b_2' \cdots b_{\varphi(n)}'$, siendo b_j' el inverso de b_j módulo n. Es la proposición 3.1 la que nos asegura la existencia de tales inversos). □

Teorema 3.9. *Si un primo p no divide a a, entonces*

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

Demostración. Es un corolario del teorema anterior, puesto que $\varphi(p) = p - 1$. □

Teorema 3.10. *Para cada entero a y cada primo p tenemos:*

$$a^p \equiv a \pmod{p}.$$

Demostración. Si p no divide a a ($p \nmid a$), estamos ante el teorema anterior. Si $p|a$ entonces tanto a^p como a son congruentes con 0 módulo p. □

El teorema de Euler-Fermat sirve para calcular las soluciones de una congruencia lineal.

Definición 3.10 (**Clase de equivalencia**). *Si $a \in \mathbb{Z}$, definimos*

$$\overline{a} = a + n\mathbb{Z} = \{a + nx | x \in \mathbb{Z}\}.$$

Observamos que $b \in \overline{a}$ si y solo si $b = a + nx$ para algún $x \in \mathbb{Z}$. Como esto ocurre si y solo si $n|(b - a)$, concluimos que

$$b \in \overline{a} \Longleftrightarrow a \equiv b \pmod{n}$$

Es decir, $\overline{a}$ es la ***clase de equivalencia*** *de a.*

Teorema 3.11. *Si $a \in \mathbb{Z}$, entonces $\overline{a}$ es una unidad de $\mathbb{Z}_n$ si y solo si $mcd(a, n) = 1$.*

Demostración. Podemos escribirlo como $mcd(a, n) = 1$ si y solo si $u \in \mathbb{Z}$ tal que $au \equiv 1$ mód n. Supongamos que $mcd(a, n) = 1$. Por la identidad de Bezout [19], existen $u, v \in \mathbb{Z}$ tales que $au + nv = 1$. Entonces $au = 1 - nv \equiv 1 \mod n$. Recíprocamente, si $au \equiv 1 \mod n$, es decir, u es entonces $1 = au + nv$ para cierto v. Si d divide a a y a n, se sigue que d divide a $au + nv = 1$, lo que prueba que $mcd(a, n) = 1$. □

3.3. Bases cifrado

La primera definición que vamos a ver en esta sección es la de clave pública y clave privada, conceptos fundamentales en la criptografía de clave asimétrica, también conocida como criptografía de clave pública [20]. A continuación, se presenta una definición matemática de clave pública y privada:

Definición 3.11 (**Clave pública**). *En criptografía de clave asimétrica, la **clave pública** es un par de números*

$$(e, n),$$

donde e es un número entero llamado exponente de cifrado y n es un número entero llamado módulo.

La clave pública se utiliza para cifrar datos en un proceso de cifrado, y puede ser conocida por cualquier persona. Se utiliza principalmente para cifrar mensajes y garantizar la confidencialidad de la información. Sin embargo, a pesar de conocer la clave pública, es computacionalmente difícil deducir la clave privada correspondiente.

Definición 3.12 (**Clave privada**). *La **clave privada** es un número entero d. La clave privada se mantiene en secreto y solo es conocida por el propietario. Se utiliza para descifrar los datos cifrados con la clave pública correspondiente. La clave privada también se utiliza para firmar digitalmente documentos y garantizar la autenticidad e integridad de la información.*

Ahora, veamos de qué trata un sistema de cifrado de clave pública.

Definición 3.13 (**Sistema de cifrado de clave pública**). *En un **sistema de cifrado de clave pública**, o también denominados **algoritmos de cifrado asimétricos**, cada usuario dispone de una **clave pública** y de una **clave privada**. Las claves públicas*

$$E = \{(e_i, n_i)\}, i \in \mathbb{N}$$

de todos los usuarios están recogidas en un directorio accesible para todo el mundo. Mientras tanto, cada usuario mantiene en secreto su clave privada d.

En resumen, en la criptografía de clave asimétrica, la clave pública (e, n) se utiliza para cifrar datos, mientras que la clave privada (d) se utiliza para descifrar los datos cifrados y firmar digitalmente documentos. Estas claves están matemáticamente relacionadas entre sí, pero la clave privada es mantenida en secreto y es computacionalmente difícil de deducir a partir de la clave pública.

3.4. Sistema RSA

El sistema RSA, denotado por (n, e) (clave pública), se refiere al problema de extraer las raíces e-ésimas módulo n, donde $n = pq$, con p y q dos números primos grandes y desconocidos y e es un entero positivo que sea menor que $\varphi(n) = (p-1)(q-1)$ y a la vez coprimo con $\varphi(n)$. En este sistema, n es un número cuya factorización es desconocida, y las raíces que se desean extraer son las raíces e-ésimas de un número dado módulo n. A lo largo de esta sección, exploraremos los pasos necesarios para aplicar el sistema RSA y utilizaremos un ejemplo para comprender las dificultades y características de este sofisticado sistema.

Antes de adentrarnos en el proceso, estudiaremos los pasos necesarios para el correcto funcionamiento del sistema y revisaremos algunos conceptos que nos ayudarán a comprenderlo mejor.

Uno de los grandes retos de la seguridad en la criptografía y por ende, de la seguridad en el sistema RSA, es la obtención de números primos de gran tamaño. Para ello vamos a introducir el concepto de test de primalidad.

Definición 3.14 (**Test de primalidad**). *La cuestión de la determinación de si un número n dado es primo es conocida como el problema de la primalidad. Un* ***test de primalidad*** *(o chequeo de primalidad) es un algoritmo que, dado un número de entrada n, no consigue verificar de forma concluyente la hipótesis de que n es compuesto.*

Esto es, un test de primalidad solo conjetura que ante la falta de certificación sobre la hipótesis de que n es compuesto podemos tener cierta confianza en que se trata de un número primo. Esta definición supone un grado menor de confianza que lo que se denomina prueba de primalidad (o test verdadero de primalidad), que ofrece una seguridad matemática sobre si un número es primo.

Veamos un ejemplo de un test de primalidad sencillo, es importante destacar que existen otros algoritmos más sofisticados y eficientes para realizar pruebas de primalidad que ofrecen una mayor seguridad matemática sobre si un número es primo o compuesto.

Ejemplo 3.2. *Supongamos que queremos verificar si el número $n = 17$ es primo utilizando el test de primalidad conocido como "Prueba de divisibilidad". Este test consiste en comprobar si n es divisible por algún número primo menor que $\sqrt{n}$.*

En este caso, $\sqrt{n} \approx 4.12$, por lo que debemos verificar si n es divisible por los números primos menores o iguales a 4, es decir, 2 y 3.

Al realizar la división, vemos que n no es divisible por 2 ni por 3. Por lo tanto, el test de primalidad no encontró evidencia concluyente de que n sea compuesto.

Según la definición dada, el test de primalidad solo conjetura que n es primo, ya que no pudo verificar de forma concluyente la hipótesis de que n sea compuesto. Sin embargo, en este caso particular, sabemos que n es realmente primo.

Ahora, veamos como podemos obtener un número primo mediante un algoritmo, ya que para el sistema RSA necesitaremos generar un par de números primos de gran tamaño. Este es un algoritmo basado en el postulado de Bertrand [21] para obtener un número primo aleatorio muy grande. El postulado de Bertrand nos dice que el primo p_{k+1} siempre está en el intervalo abierto $(p_k, 2p_k)$; o en otras palabras, que $p_{k+1} - p_k < p_k$.

Algoritmo 1: Obtención de un número primo aleatorio

Entrada: Un número natural n.
Salida: P (el número primo aleatorio buscado).
$P \leftarrow$ Genera_numero_aleatorio_en_intervalo $[n, 2n]$;
while *P no sea un número primo* **do**
    $P \leftarrow P + 1$;
    if $P = 2n$ **then**
        $P \leftarrow n$;
    end
end
return P;

El algoritmo 1 genera un número primo entre n y $2n$, ya que mientras el número que tenemos no es primo vamos sumándole una unidad a ese número, hasta obtener uno primo o volver al valor n y continuar desde ahí. Gracias al postulado de Bertrand podemos asumir que el anterior algoritmo nos genera un número primo y no entra en un bucle infinito. En el algoritmo anterior se podría incluir una condición para evitar dicho bucle en caso de que fuera necesario.

Ejemplo 3.3. *Tomando $n = 10$ en el algoritmo 1, supongamos que al iniciar* Genera_numero_aleatorio_en_intervalo $[n, 2n]$ *nos devuelve 14, valor entre 10 y 20. Entramos en el bucle al ser $14 = 2 \times 7$ no primo y vemos que $15 = 3 \times 5$ tampoco lo es, seguimos y $16 = 2 \times 8$ tampoco lo es, llegamos 17, el cual sí que es primo por lo que ese será el número que nos devuelva el algoritmo.*

Una vez visto este concepto, veamos los pasos involucrados en el proceso de cifrado RSA:

1. Se hace uso del algoritmo 1, o uno similar, para elegir de forma aleatoria dos números primos grandes p y q, que tienen una longitud en bits similar. La longitud en bits se

refiere al número de bits necesarios para representar cada número primo. Por ejemplo, si se eligen dos números primos de longitud en bits de 1024, significa que cada número primo se representa con 1024 números binarios. Luego, se calcula el producto $n = pq$ que tendrá una longitud en bits igual a la suma de las longitudes en bits de p y q. Este número n se utiliza como el "módulo" para ambas claves, la pública y la privada.

2. Se elige un número entero e positivo que sea menor que $\varphi(n) = (p-1)(q-1)$ y a la vez coprimo con $\varphi(n)$. Este es el llamado "exponente de cifrado" de la clave pública. Dado que la clave pública se comparte abiertamente, la aleatoriedad del número e no es crucial. En la práctica, se suele establecer en 65,537, ya que seleccionar números considerablemente más grandes supondría tener un algoritmo menos eficiente.

3. El mensaje M que se quiere cifrar se transforma en un número entero m tal que $0 \leq m < n$.

4. El mensaje cifrado se calcula como

$$c \equiv m^e \pmod{n}.$$

   Este es el mensaje que se envía.

5. Se determina un d (mediante aritmética modular) que satisfaga la congruencia

$$ed \equiv 1 \pmod{\varphi(n)},$$

   es decir, que d sea el multiplicador modular inverso de $e \mod \varphi(n)$. Expresado de otra manera, $d(e-1)$ es dividido exactamente por $\varphi(n) = (p-1)(q-1)$. d se guarda como el exponente de la clave privada.

6. El mensaje se descifrará calculando

$$m \equiv c^d \pmod{n}$$

La clave pública es (n, e), esto es, el módulo y el exponente de cifrado. La clave privada es (n, d), esto es, el módulo y el exponente de descifrado, que debe mantenerse en secreto.

Ejemplo 3.4. *Para este ejemplo vamos a trabajar con primos pequeños, pero en la realidad se toman dos primos lo suficientemente grandes, tal que $p \neq q$. Sean $p = 11$ y $q = 23$, a partir de estos números se obtiene:*

$$n = p \times q = 11 \times 23 = 253$$

$$\varphi(n) = (p-1) \times (q-1) = 10 \times 22 = 220$$

Buscamos un número e coprimo con $\varphi(n)$*. Para esto se selecciona de forma aleatoria un entero e, tal que* $1 < e < \varphi(n)$ *y* $mcd(\varphi(n), e) = 1$*. Tomamos* $e = 3$*, ya que* $mcd(220, 3) = 1$*.*

Se calcula el exponente privado de RSA, haciendo que se satisfaga la congruencia: $ed \equiv 1 \pmod{\varphi(n)}$*. En la práctica esto puede resolverse buscando un d que sea entero, y que verifique*

$$d = (1 + x \times \varphi(n))/e,$$

para algún valor entero de x. Lo logramos en este caso con $x = 2$*.*

$$3 \times 147 = 441 \equiv 1 \pmod{220} \Rightarrow d = 147$$

Clave pública: $(e, n) = (3, 253)$*.*

Clave privada: $(d, n) = (147, 253)$*.*

Cifrado: $C = M^e \mod n$*.*

Descifrado: $M = C^d \mod n$*, asignemos a cada letra un número usando las tablas 3.1 y 3.2.*

Cuadro 3.1: Primera parte tabla de codificación alfanumérica.

A	B	C	D	E	F	G	H	I	J	K	L	M
00	01	02	03	04	05	06	07	08	09	10	11	12

Cuadro 3.2: Segunda parte tabla de codificación alfanumérica.

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Con las claves del ejemplo vamos a cifrar el mensaje M="SEGURIDAD".

Codificación mensaje M:

$\boldsymbol{S} = 18$ $\boldsymbol{E} = 04$ $\boldsymbol{G} = 06$ $\boldsymbol{U} = 20$ $\boldsymbol{R} = 17$ $\boldsymbol{I} = 08$ $\boldsymbol{D} = 03$ $\boldsymbol{A} = 00$ $\boldsymbol{D} = 03$

Cifrado con Clave Pública: (3,253)

$$18^3 = 5832 \pmod{253} = 13$$
$$3^3 = 27 \pmod{253} = 27$$

$$C = 13 \quad 64 \quad 216 \quad 157 \quad 106 \quad 6 \quad 27 \quad 0 \quad 27$$

Descifrado con Clave Privada: (147,253)

$$13^{147} \equiv 18 \pmod{253}$$

$$27^{147} \equiv 3 \pmod{253}$$

Descodificación mensaje M: $\boldsymbol{S} = 18$ $\boldsymbol{E} = 04$ $\boldsymbol{G} = 06$ $\boldsymbol{U} = 20$ $\boldsymbol{R} = 17$ $\boldsymbol{I} = 08$ $\boldsymbol{D} = 03$ $\boldsymbol{A} = 00$ $\boldsymbol{D} = 03$

Usando las propiedades de la función φ de Euler (definición 3.8), el Teorema de Euler (Teorema 3.8) y el Teorema Chino del Resto (Teorema 3.3) se puede demostrar el siguiente teorema, que es el que nos sirve para demostrar que el algoritmo RSA funciona correctamente y cumple con su objetivo de cifrar y descifrar mensajes de forma segura.

Teorema 3.12. *Sea m un número entero que se quiere cifrar, e un entero positivo llamado "exponente de cifrado", n un entero llamado "módulo" tal que $n = pq$ con p y q primos, y d un entero que cumple $ed \equiv 1 \pmod{\varphi(n)}$ llamado "exponente de la clave privada", tenemos que:*

$$(m^e)^d \equiv m \pmod{n}.$$

Por lo que conociendo la clave privada (d, n), es fácil recuperar el mensaje m que ha sido cifrado mediante m^e.

Demostración. Siendo $m \in \mathbb{Z}_n$ hay solo dos posibles casos que analizar:

1. $\text{mcd}(m, n) = 1$.
2. $\text{mcd}(m, n) \neq 1$.

En el primer caso, se cumple el Teorema de Euler, evaluando:

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

Sabiendo por hipótesis que $ed \equiv 1 \pmod{\varphi(n)}$, podemos escribir:

$$(m^e)^d = m^{ed} = m^{1+k\varphi(n)}, \tag{3.4}$$

siendo k un entero cualquiera que se introduce en la expresión como un coeficiente multiplicativo del exponente. Además,

$$m^{1+k\varphi(n)} = mm^{k\varphi(n)} = m(m^{\varphi(n)})^k,$$

y por el Teorema de Euler

$$m(m^{\varphi(n)})^k \equiv m \pmod{n}$$

Demostrando así que el teorema se cumple en ese primer caso.

En el segundo caso, el Teorema de Euler ya no se cumple, pero podemos aplicar el Teorema Chino del Resto. Es cierto que si $\text{mcd}(p, q) = 1$ entonces:

$$x \equiv y \pmod{p} \wedge x \equiv y \pmod{q} \Rightarrow x \equiv y \pmod{pq} \text{ por el Teorema 3.3.}$$

Entonces, si probamos las siguientes dos afirmaciones, habríamos terminado:

- $(m^e)^d \equiv m \mod p$
- $(m^e)^d \equiv m \mod q$

Al ser $\text{mcd}(m, n) \neq 1$, o bien $\text{mcd}(m, n) = p$ o $\text{mcd}(m, n) = q$ debe ser cierto. Demostraré que las dos afirmaciones anteriores son ciertas en el caso $\text{mcd}(m, n) = p$, siendo absolutamente idénticas (cambiando las letras) para probarlo también para $\text{mcd}(m, n) = q$.

Para el caso

$$(m^e)^d \equiv m \mod p$$

procedemos del siguiente modo. Sea $\text{mcd}(m, n) = p$, esto implica que $m = kp$ para algún $k > 0$, lo que significa que $m \equiv 0 \mod p$. Con respecto a la primera afirmación también tenemos

$$(m^e)^d = ((kp)^e)^d$$

que por lo tanto resulta ser un múltiplo de p, por lo que es igual a cero módulo p. Entonces, la primera declaración se convierte en $0 = 0$ y se demuestra que se cumple.

Probemos ahora que

$$(m^e)^d \equiv m \mod q.$$

Para ello tenemos que el Teorema de Euler resulta demostrarse en $\mathbb{Z}_q$, ya que $\text{mcd}(m, q) = 1$, entonces:

$$m^{\varphi(q)} \equiv 1 \mod q. \tag{3.5}$$

Esto implica que podemos escribir:

$$\begin{aligned}
(m^e)^d &\equiv m^{ed} \mod q \\
&\equiv m^{ed-1}m \mod q \\
&\equiv m^{k(p-1)(q-1)}m \mod q \text{ (por 3.4)} \\
&\equiv m^{(q-1)k(p-1)}m \mod q \\
&\equiv 1^{k(p-1)}m \mod q \text{ (por 3.5)} \\
&\equiv m \mod q
\end{aligned}$$

lo que definitivamente prueba la segunda declaración y el teorema. □

La limitación del cifrado RSA para el aprendizaje federado radica en su incapacidad de realizar cálculos de manera eficiente en datos cifrados. Aunque RSA es un algoritmo criptográfico ampliamente utilizado y efectivo para la seguridad de la información, su naturaleza no permite operaciones homomórficas, que son fundamentales en el aprendizaje federado.

En el caso de RSA, aunque es posible cifrar los datos de cada participante y enviarlos al servidor central, una vez cifrados, los cálculos sobre esos datos cifrados no pueden realizarse de manera eficiente. Esto se debe a que RSA es un sistema de cifrado asimétrico basado en operaciones de exponenciación modular, que son computacionalmente costosas.

Para realizar cálculos en datos cifrados de manera eficiente en el contexto del aprendizaje federado, es necesario utilizar esquemas de cifrado homomórfico más avanzados, como el cifrado homomórfico basado en redes neuronales o el cifrado homomórfico parcial, que es el que desarrollaremos en el siguiente capítulo. Estos esquemas permiten operaciones aritméticas directas en datos cifrados, lo que facilita el entrenamiento de modelos de aprendizaje automático de manera descentralizada y segura.

En resumen, aunque RSA es un algoritmo de cifrado seguro, su limitación en realizar cálculos eficientes en datos cifrados lo hace menos adecuado para el aprendizaje federado, donde la capacidad de realizar operaciones homomórficas es crucial para mantener la privacidad de los datos de los participantes y permitir la colaboración en la construcción de modelos de manera segura.

Capítulo 4

Cifrado homomórfico

En este capítulo vamos a explicar qué es el cifrado homomórfico, cuales son los diferentes tipos de cifrado homomórfico y mostraremos mediante ejemplos su funcionamiento y utilidad en el campo del aprendizaje federado. La privacidad es de entrada uno de los temas más relevantes en el aprendizaje automático privado y el *cifrado homomórfico* (*Homomorphic Encryption*, o *HE*, por sus siglas en inglés), a diferencia de los métodos de cifrado tradicionales, permite realizar cálculos (matemáticos) significativos en datos cifrados o ilegibles. El cifrado homomórfico garantiza que realizar operaciones sobre datos cifrados y descifrar el resultado es equivalente a realizar operaciones análogas sin ningún tipo de cifrado. Los resultados presentados en este capítulo han sido extraidos de [22] y [23].

4.1. Introducción

Cuando se utiliza el cifrado homomórfico, los datos pueden ser cifrados por su propietario y enviados al propietario del modelo para ejecutar el cálculo. Por ejemplo, se aplicaría un modelo de clasificación entrenado a los datos cifrados de un paciente y se devolvería el resultado cifrado (por ejemplo, la predicción de una enfermedad) al paciente. Podemos ver un esquema de cómo sería el esquema de cifrado homomórfico de los datos de un cliente en la figura 4.1. En este caso, no es necesario cifrar los pesos del modelo, ya que el cálculo se realiza en el lado del propietario del modelo. En la actualidad existen restricciones en el tipo de cálculos que se pueden realizar mediante encriptación homomórfica, y el rendimiento de cómputo aún está muy lejos del de las técnicas tradicionales.

El cifrado homomórfico no sólo se utiliza para proteger a los datos de los propietarios de datos; los propietarios de los modelos tienen algunas de las mismas preocupaciones de privacidad en torno a su valiosa propiedad intelectual. Por lo tanto, cuando se utiliza un modelo en un entorno que no es de confianza, es crucial mantener sus parámetros encriptados.

El cifrado homomórfico tiene numerosas aplicaciones que van desde la sanidad hasta las redes eléctricas inteligentes; desde la educación hasta el aprendizaje automático como servicio

(*Machine Learning as a Service*, o *MLaaS*, por sus siglas en inglés). Todos los sectores en los que la privacidad de los datos es primordial y el uso de los mismos suele ser ya complejo debido a: la normativa, la importancia de los datos y los problemas de seguridad. Otros usos notables de esta tecnología tienen que ver con la seguridad no intrusiva y que preserva la privacidad; es decir, sistemas capaces de detectar actividades nefastas a partir de una fuente de datos cifrada y privada. Gracias a la encriptación, estos sistemas no violan la privacidad de nadie, su precisión puede verificarse empíricamente y, como sus parámetros se mantienen privados, no es fácil aplicarles ingeniería inversa.

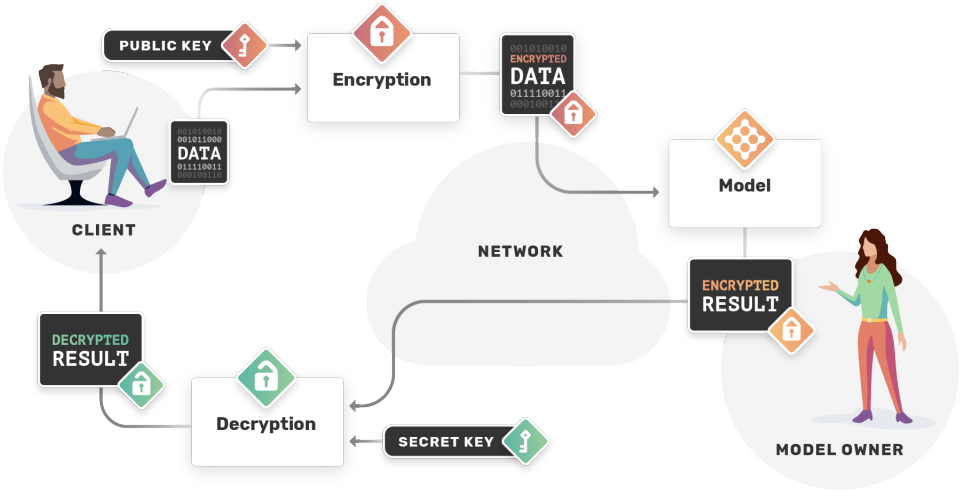

Figura 4.1: Funcionamiento del cifrado homomórfico [24].

Algunas de las ventajas del cifrado homomórfico son:

- Puede realizar la inferencia sobre datos encriptados, de modo que el propietario del modelo nunca ve los datos privados del cliente y, por tanto, no puede filtrarlos ni hacer un mal uso de ellos.
- No requiere interactividad entre los datos y los propietarios del modelo para realizar el cálculo.

Las desventajas más notables del cifrado homomórfico son las siguientes:

- Caros en términos de computación.
- Restringido a ciertos tipos de cálculos.

Existen tres tipos de cifrado homomórfico: completo o *FHE* (*Fully Homomorphic Encryption*), medio o *SHE* (*Somewhat Homomorphic Encryption*), y parcial cuando el tipo de operaciones es limitado, o *PHE* (*Partial Homomorphic Encryption*) [25].

1. **Cifrado Homomórfico Completo (*FHE*):** En el cifrado homomórfico completo o cifrado totalmente homomórfico, se pueden realizar operaciones aritméticas arbitrarias directamente sobre el texto cifrado, lo que significa que se pueden realizar tanto operaciones de suma como de multiplicación sin descifrar los datos. Esto permite realizar cálculos complejos sobre datos cifrados y obtener resultados cifrados que, al ser descifrados, coinciden con el resultado de las operaciones realizadas sobre los datos en claro. El *FHE* es el tipo más poderoso de cifrado homomórfico, pero también es el más computacionalmente intensivo y, por lo tanto, el más lento.

2. **Cifrado Homomórfico Parcial (*PHE*):** El cifrado homomórfico parcial permite realizar solo un conjunto específico de operaciones aritméticas sobre el texto cifrado. Por lo general, se puede realizar una sola operación (ya sea suma o multiplicación) sin necesidad de descifrar los datos. Sin embargo, realizar múltiples operaciones en el texto cifrado no es posible en *PHE*, lo que limita su versatilidad en comparación con *FHE*.

3. **Cifrado Homomórfico Medio (*SHE*):** El cifrado homomórfico medio o cifrado "algo" homomórfico se encuentra entre el *FHE* y el *PHE* en términos de funcionalidad. Permite realizar un conjunto más amplio de operaciones aritméticas que *PHE*, pero no es tan poderoso como *FHE*, lo que significa que no se pueden realizar todas las operaciones aritméticas directamente sobre el texto cifrado. En general, *SHE* es más eficiente en términos computacionales que *FHE* y ofrece un equilibrio entre funcionalidad y eficiencia.

En resumen, cada tipo de cifrado homomórfico tiene sus propias ventajas y limitaciones. *FHE* es el más potente pero más lento, *PHE* es más eficiente pero tiene funcionalidad limitada, y *SHE* ofrece un equilibrio entre funcionalidad y eficiencia. La elección del tipo de cifrado homomórfico depende de las necesidades específicas de la aplicación y los recursos computacionales disponibles. En las siguientes secciones vamos a tratar de explicar cómo funciona cada uno de estos tipos de cifrado homomórfico y cuales son sus características mas destacables.

4.2. Cifrado homomórfico parcial

El cifrado homomórfico es una forma de cifrado que permite realizar operaciones matemáticas o lógicas sobre los datos cifrados. Por ejemplo, supongamos que tenemos dos números m_1 y m_2 y los ciframos utilizando algún esquema de cifrado de clave pública con una clave pública pub y una clave privada $priv$. Obtenemos dos textos cifrados $c_1 = E_{pub}(m_1)$ y $c_2 = E_{pub}(m_2)$. Normalmente, el cifrado pretende que todos los números cifrados sean indistinguibles de los números aleatorios, para cualquiera que no tenga la clave privada necesaria

para el descifrado; y por lo tanto al operar con ellos no es posible recuperar un resultado descifrado.

Sin embargo, con el cifrado homomórfico se conservan algunas relaciones. Por ejemplo, si tenemos un esquema de cifrado homomórfico que permite la adición, habrá una función add_{pub} que cualquiera puede realizar sobre c_1 y c_2 de forma que el resultado, $add_{pub}(c_1, c_2)$, se descifrará a la suma de m_1 y m_2:

$$D_{priv}(add_{pub}(E_{pub}(m_1), E_{pub}(m_2))) = m_1 + m_2$$

Nótese que la función add_{pub} no será necesariamente una adición literal, sino cualquier función que desempeñe el papel descrito anteriormente, según el esquema de cifrado homomórfico correspondiente.

Hace tiempo que existen esquemas de cifrado homomórfico parcial, en los que se puede realizar un número limitado de operaciones con los datos cifrados, por ejemplo, sólo la suma o sólo la multiplicación. En el siguiente apartado se presenta uno de estos esquemas de cifrado: el sistema criptográfico de Paillier. Desarrollaremos el siguiente capítulo siguiendo el libro [23].

4.2.1. Sistema criptográfico Paillier

El criptosistema Paillier, inventado por Pascal Paillier en 1999, es un algoritmo asimétrico probabilístico para la criptografía de clave pública [26]. Este sistema se basa en la creencia de que el problema de calcular las clases del enésimo residuo es computacionalmente difícil. Esto se conoce como Residuidad Compuesta y es la base de este sistema criptográfico. El esquema de cifrado de Paillier se compone de 3 algoritmos: uno de generación de claves, otro de cifrado y un último de descifrado.

Algoritmo de generación de clave

Vamos a proceder a generar la clave, siguiendo los siguientes pasos.

1. Se escogen 2 números primos p y q grandes de forma aleatoria e independiente entre si, de manera que $n = pq$, donde

   $$mcd(n, \lambda) = 1,$$

   con $\lambda = mcm(p-1, q-1)$, siendo mcm el mínimo común múltiplo. Esta propiedad está asegurada si ambos primos son de igual longitud

2. Se selecciona un entero g aleatorio tal que

   $$g \in \mathbb{Z}^*_{n^2}. \tag{4.1}$$

3. Se asegura que n divide al orden de g revisando la existencia del siguiente inverso multiplicativo:

$$\mu = (L(g^{\lambda} \pmod{n^2}))^{-1} \pmod{n}$$

donde la función L se define como

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

Obsérvese que la notación $\frac{a}{b}$ no denota la multiplicación modular de a por la inversa modular multiplicativa de b sino el cociente de a dividido por b, es decir, el mayor valor entero $v \geq 0$ que satisfaga la relación $a \geq vb$.

Como resultado:

1. **La clave pública (de cifrado) es** (n, g)**.**

2. **La clave privada (de descifrado) es** (λ, μ)**.**

Si se utilizan p, q de longitud equivalente, una variante más sencilla de los pasos de generación de claves anteriores sería establecer

$$\begin{aligned} g &= n + 1, \\ \lambda &= \varphi(n) \text{ y} \\ \mu &= \varphi(n)^{'} \pmod{n} \end{aligned}$$

donde $\varphi(n) = (p-1)(q-1)$ y $\varphi(n)^{'}$ es el inverso de $\varphi(n)$ módulo n [27].

Algoritmo de cifrado del mensaje E:

Vamos a proceder al cifrado del mensaje siguiendo los siguientes pasos.

1. Sea m el mensaje a cifrar, tal que

$$m \in \mathbb{Z}_n.$$

2. Se escoge un número aleatorio r, tal que

$$r \in \mathbb{Z}_n^*.$$

3. El mensaje cifrado es:

$$E(m, (n, g)) = g^m \cdot r^n \pmod{n^2}.$$

Algoritmo de descifrado del mensaje D:

Ahora, vamos a proceder a realizar el descifrado del mensaje (el proceso contario), siguiendo los siguientes pasos.

1. Dado el texto cifrado
$$c \in \mathbb{Z}_{n^2}^*.$$
2. El mensaje descifrado es:
$$D(c,(n,g),(\lambda,\mu)) = L(c^\lambda \pmod{n^2}) \cdot \mu \pmod{n}.$$

Como señala el documento original [22], el descifrado es "esencialmente una exponenciación módulo n^2".

El esquema de cifrado Paillier explota el hecho de que ciertos logaritmos discretos se pueden calcular fácilmente. Por ejemplo, por el teorema del binomio [17],

$$\begin{aligned}(1+n)^x &= \sum_{k=0}^{x} \binom{x}{k} n^k \\ &= 1 + nx + \binom{x}{2} n^2 + \textit{mayores potencias de } n^2\end{aligned}$$

Esto indica que:
$$(1+n)^x = 1 + nx \pmod{n^2} \tag{4.2}$$

Por lo tanto, si:
$$y = (1+n)^x \pmod{n^2}.$$

Aplicando la propiedad 4.2 tenemos
$$y = 1 + nx \pmod{n^2}.$$

Restando 1 en ambos lados
$$y - 1 = nx \pmod{n^2}.$$

Despejando x
$$x = \frac{y-1}{n} \pmod{n}.$$

Por lo tanto:
$$L((1+n)^x \pmod{n^2}) = x \pmod{n},$$

para todo $x \in \mathbb{Z}_n$.

Por lo tanto, cuando $g = n+1$, tenemos

$$\begin{aligned}L(c^\lambda \pmod{n^2}) \cdot \mu \pmod{n} &= L((g^m r^n)^\lambda \pmod{n^2}) \cdot \lambda^{-1} \pmod{n} \\ &= L((g^{m\lambda} \pmod{n^2}) \cdot \lambda^{-1} \pmod{n} \\ &= \lambda \cdot m \cdot \lambda^{-1} \pmod{n}.\end{aligned} \tag{4.3}$$

A continuación se muestra un ejemplo del esquema de cifrado Paillier.

Ejemplo 4.1. *Para este ejemplo vamos a trabajar con primos pequeños, pero en la realidad se toman dos primos lo suficientemente grandes, tal que* $p \neq q$. *Sean* $p = 7$ *y* $q = 11$, *a partir de estos números se obtiene:*

$$n = p \times q = 7 \times 11 = 77$$

A continuación, debe seleccionarse un número entero g *de* $\mathbb{Z}^*_{n^2}$, *tal que el orden de* g *sea múltiplo de* n *en* $\mathbb{Z}_{n^2}$. *Si elegimos al azar el número entero*

$$g = 5652,$$

entonces se cumplen todas las propiedades necesarias, incluida la condición de que el orden de g, *que es* $2310 = 30 \times 77$ *esté en* $\mathbb{Z}_{n^2}$, *es decir,* $g \in \mathbb{Z}^*_{n^2}$. *Así, la clave pública del ejemplo será*

$$(n, g) = (77, 5652)$$

Para encriptar un mensaje

$$m = 42,$$

donde $m \in \mathbb{Z}_n$ *elegimos un* r *aleatorio, por ejemplo*

$$r = 23,$$

que es un entero no nulo y $r \in \mathbb{Z}_n$. *Calculamos*

$$\begin{aligned} c &= g^m \cdot r^n \pmod{n^2} \\ &= 5652^{42} \cdot 23^{77} \pmod{5929} \\ &= 4624 \pmod{5929} \end{aligned}$$

Para descifrar el texto cifrado c, *vamos a calcular la clave privada*

$$\lambda = mcm(6, 10) = 30.$$

Por definición $L(u) = \frac{u-1}{n}$, *calculamos entonces*

$$\begin{aligned} k &= L(g^\lambda \pmod{n^2}) \\ &= L(5652^{30} \pmod{5929}) \\ &= L(3928) \\ &= (3928 - 1)/77 \\ &= 51 \end{aligned}$$

Calculamos el inverso de k,

$$\mu = k^{-1} \pmod{n}$$

$$= 51^{-1} \pmod{n}$$
$$= 74 \pmod{n}$$

Por lo tanto, la clave privada es

$$(\lambda, \mu) = (30, 74).$$

Calculamos

$$\begin{aligned} m &= L(c^{\lambda} \pmod{n^2}) \cdot \mu \pmod{n} \\ &= L(4624^{30} \pmod{5929}) \cdot 74 \pmod{77} \\ &= L(4852) \cdot 74 \pmod{77} \\ &= 42. \end{aligned}$$

Por lo que hemos conseguido encriptar y desencriptar con éxito el mensaje $m = 42$ *usando el criptosistema de Paillier.*

Correcto funcionamiento

En esta sección, veremos que se puede demostrar el siguiente teorema, que es el que nos sirve para demostrar que el sistema criptográfico de Paillier funciona correctamente y cumple con su objetivo de cifrar y descifrar mensajes de forma segura.

Teorema 4.1. *Dada la función para cifrar del sistema de Paillier*

$$E(m, (n, g)) = g^m \cdot r^n \pmod{n^2}, \tag{4.4}$$

donde m *es el mensaje a cifrar, tal que* $m \in \mathbb{Z}_n$ *;* (n, g) *es la clave pública (de cifrado) tal que* n *es un entero llamado "módulo" con* $n = pq$ *y* p *y* q *primos; y* g *es un entero aleatorio tal que* $g \in \mathbb{Z}^*_{n^2}$*. Dada la función para descifrar*

$$D(c, (n, g), (\lambda, \mu)) = L(c^{\lambda} \pmod{n^2}) \cdot \mu \pmod{n}, \tag{4.5}$$

donde $c \in \mathbb{Z}^*_{n^2}$ *es el texto cifrado y* (λ, μ) *es la clave privada (de descifrado) tal que* $\lambda = mcm(p-1, q-1)$ *y* $\mu = (L(g^{\lambda} \pmod{n^2}))^{-1} \pmod{n}$*, donde la función* L *se define como* $L(u) = \frac{u-1}{n}$*. Se cumple que:*

$$D(E(m, (n, g)), (n, g), (\lambda, \mu)) = m \pmod{n} \tag{4.6}$$

Demostración. Vamos a manipular 4.6:

$$\begin{aligned} m &= D(E(m, (n, g)), (n, g), (\lambda, \mu)) \pmod{n} \\ &= D(g^m \cdot r^n \pmod{n^2}, (n, g), (\lambda, \mu)) \pmod{n} \text{ (hemos aplicado 4.4)} \\ &= L((g^m \cdot r^n \pmod{n^2})^{\lambda} \pmod{n^2}) \cdot \mu \pmod{n} \text{ (hemos aplicado 4.5)} \end{aligned}$$

$$= L((g^m \cdot r^n \pmod{n^2})^\lambda \pmod{n^2}) \cdot (L(g^\lambda \pmod{n^2}))^{-1} \pmod{n} \text{ (definición de } \mu)$$
$$= \frac{L((g^m \cdot r^n \pmod{n^2})^\lambda \pmod{n^2})}{L(g^\lambda \pmod{n^2})} \pmod{n}$$

Ahora, definimos algunas nociones necesarias para demostrar un lema que usaremos en la demostración.

Definición 4.1 (Función de valor entero). *Sean $n \in \mathbb{N}$ y $g \in \mathbb{Z}^*_{n^2}$, denotamos por E_g a la función de valor entero definida por*

$$E_g : \mathbb{Z}_n \times \mathbb{Z}^*_n \to \mathbb{Z}^*_{n^2}$$
$$(x, y) \mapsto g^x \cdot y^n \mod n^2$$

Sea $n \in \mathbb{N}$, denotamos por $\mathcal{B}_\alpha \subset \mathbb{Z}^*_{n^2}$ el conjunto de elementos de $\mathbb{Z}^*_{n^2}$ orden $n\alpha$ y por $\mathcal{B}$ su unión disjunta para $\alpha = 1, \cdots, \lambda$.

Definición 4.2. *Sean $g \in \mathcal{B}$ y $w \in \mathbb{Z}^*_{n^2}$, llamamos **clase de residuos n-ésima** de w con respecto a g al único número entero $x \in \mathbb{Z}_n$ para el que existe $y \in \mathbb{Z}_n$ tal que $E_g(x, y) = w$.*

Adoptando las notaciones de Benaloh [28], la clase de w se denota por $[w]_g$.

Ahora, enunciamos y demostramos el Lemma 10 de [22].

Lema 4.2. *Para cualquier $w \in \mathbb{Z}^*_{n^2}$, $L(w^\lambda \mod n^2) = \lambda\,[w]_{1+n} \mod n$.*

Demostración. Como $1 + n \in \mathcal{B}$, existe un único par (a, b) en el conjunto $\mathbb{Z}_n \times \mathbb{Z}^*_n$ tal que $w = (1 + n)^a b^n \mod n^2$. Por definición, $a = [w]_{1+n}$. Entonces

$$w^\lambda = (1 + n)^{a\lambda} b^{n\lambda} = (1 + n)^{a\lambda} = 1 + a\lambda n \mod n^2,$$

que nos da el resultado enunciado. □

□

Volviendo a la demostración del teorema, si seguimos manipulando lo que nos había quedado de 4.6, tenemos que:

$$\frac{L((g^m \cdot r^n \pmod{n^2})^\lambda \pmod{n^2})}{(L(g^\lambda \pmod{n^2}))} \pmod{n} = \frac{\lambda\,[g^m \cdot r^n \pmod{n^2}]_{1+n}}{\lambda\,[g]_{1+n}} \pmod{n} \text{ (Lema 4.2)}$$
$$= \frac{[g^m \cdot r^n \pmod{n^2}]_{1+n}}{[g]_{1+n}} \pmod{n}$$

Ahora, sabiendo que en [22] se prueba que

$$[w]_g = \frac{L(w^\alpha \mod n^2)}{L(g^\alpha \mod n^2)} \mod n.$$

nos queda

$$\frac{[g^m \cdot r^n \pmod{n^2}]_{1+n}}{[g]_{1+n}} \pmod{n} = \left[g^m \cdot r^n \pmod{n^2}\right]_g \pmod{n}$$

En [22] también se prueba con $c = g^m \cdot r^n \pmod{n^2}$ que

$$g^m \cdot r^n \pmod{n^2} = g^{\left[g^m \cdot r^n \pmod{n^2}\right]_g} \cdot r^n \pmod{n^2}$$

Por lo tanto, $[g^m \cdot r^n]_g \pmod{n} = m \pmod{n}$ y hemos probado el teorema.

Propiedades homomórficas

Una característica notable del criptosistema de Paillier son sus propiedades homomórficas, junto con su cifrado no determinista. Como la función de cifrado es aditivamente homomórfica, se pueden describir muchas propiedades, vamos a ver algunas de ellas mediante los siguientes dos teoremas.

Teorema 4.3 (**Suma homomórfica de textos planos**). *Dados dos textos cifrados*

$$E(m_1, pk) = g^{m_1} r_1^n \pmod{n^2}$$

y

$$E(m_2, pk) = g^{m_2} r_2^n \pmod{n^2},$$

donde $m_1, m_2 \in \mathbb{Z}_n$ son los dos mensajes a cifrar, pk es la clave pública (de cifrado), es decir, el par (n, g), tal que $n = pq$, con p y q primos, y g es un entero aleatorio tal que $g \in \mathbb{Z}^*_{n^2}$ y r_1 y r_2 se eligen aleatoriamente de $\mathbb{Z}^*_n$. Se cumple que:

1. *El producto de dos textos cifrados se descifrará como la suma de sus correspondientes correspondientes, es decir*

$$D(E(m_1, pk) \cdot E(m_2, pk) \pmod{n^2}) = m_1 + m_2 \pmod{n} \tag{4.7}$$

2. *El producto de un texto cifrado con un texto plano que se eleve a g se descifrará a la suma de los correspondientes textos planos, es decir,*

$$D(E(m_1, pk) \cdot g^{m_2} \pmod{n^2}) = m_1 + m_2 \pmod{n} \tag{4.8}$$

Demostración. Vemos que la equivalencia 4.7 es cierta porque

$$\begin{aligned} E(m_1, pk) \cdot E(m_2, pk) &= (g^{m_1} r_1^n)(g^{m_2} r_2^n) \pmod{n^2} \\ &= g^{m_1+m_2}(r_1 r_2)^n \pmod{n^2} \\ &= E(m_1 + m_2, pk) \end{aligned}$$

Vemos que la equivalencia 4.8 es cierta porque

$$\begin{aligned} E(m_1, pk) \cdot g^{m_2} &= (g^{m_1} r_1^n) g^{m_2} \pmod{n^2} \\ &= g^{m_1+m_2} r_1^n \pmod{n^2} \\ &= E(m_1 + m_2, pk) \end{aligned}$$

□

Teorema 4.4 (**Multiplicación homomórfica de textos planos**). *Dados dos textos cifrados*

$$E(m_1, pk) = g^{m_1} r_1^n \pmod{n^2}$$

y

$$E(m_2, pk) = g^{m_2} r_2^n \pmod{n^2},$$

donde $m_1, m_2 \in \mathbb{Z}_n$ son los dos mensajes a cifrar, pk es la clave pública (de cifrado), es decir, el par (n, g), tal que $n = pq$, con p y q primos, y g es un entero aleatorio tal que $g \in \mathbb{Z}_{n^2}^*$ y r_1 y r_2 se eligen aleatoriamente de $\mathbb{Z}_n^*$. Se cumple que un texto cifrado elevado a la potencia de un texto plano se descifrará con el producto de los dos textos planos, es decir,

$$D(E(m_1, pk)^{m_2}) = m_1 m_2 \pmod{n}, \tag{4.9}$$

Demostración. Vemos que la equivalencia 4.9 es cierta porque

$$\begin{aligned} E(m_1, pk)^{m_2} &= (g^{m_1} r_1^n)^{m_2} \pmod{n^2} \\ &= g^{m_1 m_2} (r_1^{m_2})^n \pmod{n^2} \\ &= E(m_1 m_2, pk) \end{aligned}$$

Más generalmente, un texto cifrado elevado a una constante k descifrará el producto del texto plano y la constante, es decir,

$$D(E(m_1, pk)^k \pmod{n^2}) = k m_1 \pmod{n}.$$

□

Sin embargo, dados los cifrados de Paillier de dos mensajes, no hay forma conocida de calcular un cifrado del producto de estos mensajes sin conocer la clave privada.

Después de haber estudiado el cifrado homomórfico parcial y detallado el sistema criptográfico de Paillier, estudiaremos en la siguiente sección los cifrados algo homomórficos, que son una clase intermedia de esquemas criptográficos que permiten realizar ciertas operaciones aritméticas sobre el texto cifrado, ampliando así su funcionalidad con respecto al cifrado homomórfico parcial. Aunque no alcanzan el nivel de versatilidad del cifrado totalmente homomórfico, los cifrados algo homomórficos ofrecen un equilibrio entre eficiencia y

funcionalidad, lo que los convierte en una opción valiosa en ciertos escenarios donde se requieren operaciones más complejas sin necesidad de desencriptar por completo los datos. En la siguiente sección, exploraremos las capacidades y aplicaciones de los cifrados algo homomórficos, destacando cómo complementan y extienden las funcionalidades del cifrado homomórfico parcial y preparan el terreno para el estudio del cifrado totalmente homomórfico.

4.3. Cifrado algo homomórfico

Los cifrados algo homomórficos suelen basarse en esquemas que son capaces de realizar un cifrado "algo" homomórfico [29]. Estos esquemas sólo pueden realizar un número limitado de operaciones sucesivas de multiplicación y suma sobre el texto cifrado antes de que los resultados se vuelvan poco fiables e imposibles de descifrar. Esta limitación surge directamente de la forma en que estos sistemas garantizan la seguridad al confiar en el ruido o el error para hacer que problemas relativamente simples sean computacionalmente intratables.

Aunque es interesante desde un punto de vista teórico, la construcción basada en retículos es difícil de describir. Por lo tanto, vamos a ver un esquema que es más fácil de entender. Puede ser visto como la versión basada en enteros del esquema basado en retículos. Es decir, podemos incorporar un ideal en un anillo de enteros y, si los parámetros se configuran correctamente, el esquema puede considerarse seguro (contra ataques conocidos). Como ventaja adicional, el procedimiento de *bootstrapping* es más fácil de entender y describir en mayor detalle, ya que no requiere conocimientos previos sobre retículos.

Cifrado algo homomórfico con clave secreta

Comenzamos con la descripción del esquema de cifrado algo homomórfico con clave secreta basado en enteros [30]. El esquema es sorprendentemente sencillo y podemos construir funcionalidades muy complejas a partir de él.

Generación de clave: La clave secreta es un número entero impar, elegido de algún intervalo $p \in \{2^{k-1}, 2^k - 1\}$.

Cifrado: Para cifrar un bit $m \in \{0, 1\}$, establece el texto cifrado como un número entero cuyo residuo módulo p tiene la misma paridad que el texto plano. Es decir, establece

$$c = pq + 2r + m$$

donde los enteros q y r son elegidos al azar dentro de algunos intervalos prescritos, tal que $2r$ es menor que $p/2$ en valor absoluto.

Descifrado: Dado un texto cifrado c y la clave secreta p, el texto de salida es

$$m \equiv (c \pmod{p}) \pmod{2}$$

La ecuación de descifrado se cumple porque

$$(c \pmod p) \pmod 2 \equiv (pq + 2r + m \pmod p) \pmod 2 \equiv 2r + m \pmod 2 \equiv m$$

Por ejemplo, supongamos que $p = 17$; cifremos $m = 1$ de la siguiente manera:

$$c = pq + 2r + m = 17 \cdot 2 + 2 \cdot 0 + 1 = 39$$

donde $q = 2, r = 0$. Es fácil de ver que

$$(c \pmod p) \pmod 2 \equiv (39 \pmod 1)7) \pmod 2 \equiv 1 \pmod 2 \equiv 1$$

Se puede obtener más información interesante con la que seguir investigando sobre ello en [31].

Tras haber estudiado los cifrados algo homomórficos y su capacidad para realizar operaciones parciales sobre datos cifrados, estudiaremos por último el cifrado totalmente homomórfico. A diferencia de los esquemas previos, el *FHE* permite llevar a cabo cualquier operación matemática sobre el texto cifrado, sin necesidad de desencriptar los datos en ningún momento. Este impresionante nivel de versatilidad y seguridad plantea un gran potencial en diversas aplicaciones, desde el procesamiento seguro de datos en la nube hasta la realización de cálculos en entornos distribuidos sin comprometer la privacidad.

4.4. Cifrado totalmente homomórfico

Desde 1978, cuando se formalizó por primera vez la idea del cifrado totalmente homomórfico [32], se han inventado varios criptosistemas para acercarse a la computación de funciones arbitrarias sobre el texto cifrado. Sin embargo, hasta que *Craig Gentry* introdujo la técnica crucial del *bootstrapping* en 2009 [33], todos eran esquemas de cifrado parcialmente o algo homomórficos. Para entender por qué es así, es importante definir con más precisión lo que significa "computar funciones arbitrarias".

A nivel de bits y puertas lógicas, las combinaciones sucesivas de AND y XOR pueden expresar cualquier función booleana, como podemos observar en la figura 4.2. Dado que estas dos puertas se comportan respectivamente como operaciones de multiplicación y adición binaria, podemos concluir que un esquema en el que podamos realizar sumas y multiplicaciones homomórficas sobre el texto cifrado debería ser capaz de lograr cifrado totalmente homomórfico, con algunas salvedades.

A	B	$-$	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}	Y_{11}	Y_{12}	Y_{13}	Y_{14}	Y_{15}
0	0		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1		0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0		0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1		0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

$$Y_0 = A \cdot \bar{A}$$
$$Y_1 = (\bar{A} \cdot \bar{B})$$
$$Y_2 = (\bar{A} \cdot B)$$
$$Y_3 = (\bar{A} \cdot B) + (\bar{A} \cdot \bar{B})$$
$$Y_4 = (A \cdot \bar{B})$$
$$Y_5 = (A \cdot \bar{B}) + (\bar{A} \cdot \bar{B})$$
$$Y_6 = (A \cdot \bar{B}) + (\bar{A} \cdot B)$$
$$Y_7 = (A \cdot \bar{B}) + (\bar{A} \cdot B) + (\bar{A} \cdot \bar{B})$$

$$Y_8 = (A \cdot B)$$
$$Y_9 = (A \cdot B) + (\bar{A} \cdot \bar{B})$$
$$Y_{10} = (A \cdot B) + (\bar{A} \cdot B)$$
$$Y_{11} = (A \cdot B) + (\bar{A} \cdot B) + (\bar{A} \cdot \bar{B})$$
$$Y_{12} = (A \cdot B) + (A \cdot \bar{B})$$
$$Y_{13} = (A \cdot B) + (A \cdot \bar{B}) + (\bar{A} \cdot \bar{B})$$
$$Y_{14} = (A \cdot B) + (A \cdot \bar{B}) + (\bar{A} \cdot B)$$
$$Y_{15} = (A \cdot B) + (A \cdot \bar{B}) + (\bar{A} \cdot B) + (\bar{A} \cdot \bar{B})$$

Figura 4.2: Usando la puerta XOR podemos hacer fácilmente una puerta NOT y con XOR, AND y NOT podemos hacer cualquiera de las 16 funciones posibles que combinan 2 variables binarias [24].

Las operaciones de adición y multiplicación sugieren el uso de un anillo como estructura algebraica subyacente.

4.4.1. Construir un esquema de cifrado totalmente homomórfico

Para entender mejor por qué el error desempeña un papel tan importante, construiremos un esquema de cifrado algo homomórfico sencillo y explicaremos cómo utilizarlo como base para un esquema de cifrado totalmente homomórfico.

Según *Craig Gentry*, tal y como explica en esta serie de charlas [34], una de las formas más sencillas de abordar la construcción de este tipo de esquema es seguir el marco *Polly Cracker*. La idea básica de este marco consiste en utilizar encriptaciones de 0 para disfrazar encriptaciones de cualquier otro mensaje.

En la práctica podemos utilizar esta idea para construir un esquema sencillo que cifre un mensaje de 1 bit $m = \{0, 1\}$ y que opere en el anillo Z_p de enteros módulo p; con p un gran primo que debe permanecer oculto, es de hecho la clave secreta.

Empecemos con el **procedimiento de encriptación**, si queremos encriptar m primero elegimos un término de ruido aleatorio pero pequeño $r < p$ y también un gran q. La encriptación de m será $C(m) = m + 2r + qp$.

Descifrar es tan fácil como tomar el módulo p del texto cifrado y luego el módulo 2 del resultado $D(C(m)) = ((m + 2r + qp) \mod p) \mod 2) = (m + 2r) \mod 2 = m$.

La **adición** funciona de forma intuitiva. Si recibimos dos mensajes encriptados $C(m_1)$ y $C(m_2)$ sin conocer la clave secreta p, podemos devolver $C(m_1) + C(m_2) = m_1 + m_2 + 2(r_1 + r_2) + p(q_1 + q_2)$ que es un cifrado válido de $m_1 + m_2$ con más ruido.

Utilizando la **multiplicación** podemos operar de forma similar porque $C(m_1) * C(m_2) = m_1m_2 + 2(r_2m_1 + r_1m_2 + r_1r_2) + p(m_1q_2 + 2r_1q_2 + q_1m_2 + 2q_1r_2 + q_1q_2)$ es una codificación válida de $m_1 * m_2$. Aquí el ruido crece mucho más rápido que en la suma.

Antes de hablar de por qué un término de ruido creciente es una cuestión tan crucial, centrémonos en por qué lo necesitamos en primer lugar. Si elimin——áramos el ruido de nuestro esquema, éste seguiría siendo capaz de realizar sumas y multiplicaciones homomórficas. Sin embargo, para romper el esquema sin ruido un atacante sólo necesitaría hacerse con dos mensajes encriptados y proceder simplemente calculando el máximo común divisor para el que existe el algoritmo euclidiano que se ejecuta linealmente con el número de dígitos de la entrada más pequeña.

Si añadimos ruido, el problema se vuelve mucho más difícil. En la literatura se conoce como problema del máximo común divisor aproximado o divisor común aproximado y con parámetros razonables se considera difícil de resolver.

4.4.2. Aprendizaje con errores

El problema del máximo común divisor aproximado no es el único problema utilizado para asegurar el cifrado totalmente homomórfico, de hecho numerosos esquemas recientes (como por ejemplo el de la figura 4.3) emplean el problema de aprendizaje con errores que también se conjetura que es difícil de resolver.

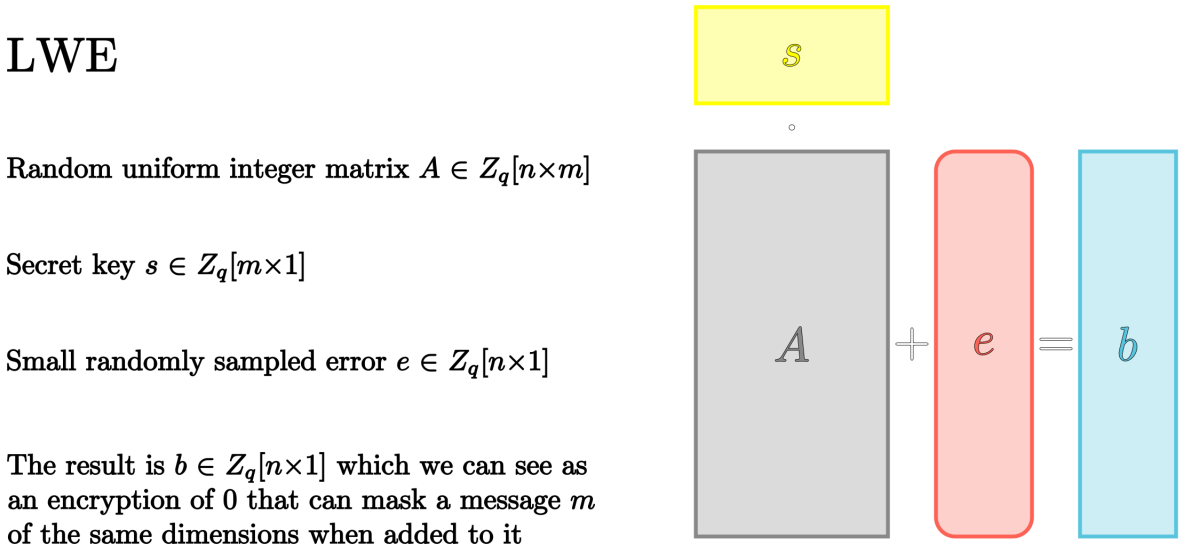

Figura 4.3: Aprendizaje con errores [24].

En su formulación más básica el problema dice que, dada una matriz entera uniforme aleatoria A y un error vectorial entero muy pequeño e, si multiplicamos A por un vector secreto s y luego sumamos e, recuperar s del vector resultante b sin e es computacionalmente intratable incluso conociendo la matriz A. De forma similar al esquema que hemos explicado antes, As + e puede pensarse como un cifrado de 0 y, desde ese punto de partida, se puede desarrollar un esquema diferente.

Capítulo 5

Conclusiones

El desarrollo del cifrado homomórfico, basado en los principios matemáticos del sistema de cifrado RSA, ha sido fundamental para alcanzar la preservación de la privacidad en ciencia de datos. Esta innovadora técnica criptográfica nos ha permitido avanzar hacia el aprendizaje federado, donde múltiples actores pueden compartir y utilizar datos para el entrenamiento de modelos de aprendizaje automático sin comprometer la confidencialidad de la información.

Gracias al cifrado homomórfico, podemos realizar operaciones directamente en datos cifrados sin necesidad de revelar su contenido, manteniendo los datos protegidos en todo momento, incluso durante el proceso de entrenamiento y predicción de modelos. Esta capacidad ha abierto puertas a diversas aplicaciones en sectores como la salud, las finanzas y la inteligencia artificial, permitiéndonos encontrar un equilibrio entre la colaboración y la privacidad.

El aprendizaje federado se ha convertido en una realidad gracias al cifrado homomórfico, lo que nos permite aprovechar el potencial de datos distribuidos en diversas organizaciones y entidades sin comprometer la seguridad y confidencialidad de la información. Ahora, podemos construir modelos de aprendizaje automático avanzados, generando conocimiento y soluciones sin revelar detalles sensibles de los datos individuales.

En conclusión, el uso del cifrado homomórfico, cuyo desarrollo en este trabajo se ha basado en el sistema RSA, nos ha permitido alcanzar la preservación de la privacidad en ciencia de datos y la implementación del aprendizaje federado.

A lo largo del trabajo, hemos explorado diferentes métodos matemáticos para preservar la privacidad en el campo de la ciencia de datos. Esta problemática se ha vuelto significativa en la sociedad actual, donde el avance tecnológico plantea desafíos para proteger el derecho a la privacidad de las personas.

La adquisición de conocimiento sobre cómo lograr un alto grado de privacidad en los datos ha sido una gran experiencia. La preocupación por la masiva recopilación de información personal por parte de empresas con fines lucrativos nos ha llevado a considerar la necesidad de establecer legislación que garantice la privacidad y la aplicación de técnicas como las mencionadas a lo largo del trabajo para protegerla.

Este trabajo ha reafirmado mi pasión por la ciencia de datos y el aprendizaje automático, lo que me motiva a continuar mi formación en el máster ofrecido por esta universidad en el próximo curso.

Desde el momento en que elegí el tema para este trabajo, supe que disfrutaría aprendiendo sobre los métodos y técnicas que garantizan la privacidad, especialmente cuando se aplican en el campo de la ciencia de datos, un área de gran interés personal. El campo de la ciencia de datos y el aprendizaje automático son dos de mis grandes intereses en la actualidad, y este trabajo lo ha reafirmado, es por ello que tengo muchas ganas de comenzar el máster que ofrece esta misma universidad sobre ello el próximo curso.

Bibliografía

[1] Liu, Alex: *Data Science and Data Scientist.* http://www.researchmethods.org/DataScienceDataScientists.pdf, Septiembre 2015. [Último acceso: 19/05/2022].

[2] Provost, Foster y Tom Fawcett: *Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking.* O'Reilly Media, 2013.

[3] Instituto de ingeniería del conocimiento: *Seguridad en Big Data, privacidad y protección de datos.* https://www.iic.uam.es/innovacion/seguridad-big-data/, Julio 2021. [Último acceso: 14/04/2022].

[4] Digital Abogados Área: *Big Data y Protección de Datos.* https://adabogados.net/big-data-y-proteccion-de-datos/, Marzo 2019. [Último acceso: 15/04/2022].

[5] Bousquet, Olivier, Kamalika Chaudhuri y Clément Dombry: *Privacy-Preserving Machine Learning: An Introduction to Recent Advances.* Morgan and Claypool Publishers, 2020.

[6] Bluemke, Emma, Antonio Lopardo y Andrew Trask: *Pivacy-preserving data science, explained.* https://blog.openmined.org/private-machine-learning-explained/, Mayo 2020. [Último acceso: 13/05/2022].

[7] Microsoft, Quinn Radich y Alma Jenks: *¿Qué es un modelo de aprendizaje automático?* https://docs.microsoft.com/es-es/windows/ai/windows-ml/what-is-a-machine-learning-model, Abril 2022. [Último acceso: 15/04/2022].

[8] Cassis, Alejandro: *Aprendizaje Supervisado.* https://inteligenciaartificial101.wordpress.com/2015/10/20/aprendizaje-supervisado/, Octubre 2015. [Último acceso: 21/05/2022].

[9] Lopardo, Antonio: *What is federated learning?* https://blog.openmined.org/what-is-federated-learning, Mayo 2020. [Último acceso: 15/04/2022].

[10] Sellami, Amel: *OpenMined AMA with Anne Lessner and Jason Mancuso.* https://www.youtube.com/watch?v=8YCB8nbTxKY&t=178s, Diciembre 2019. [Último acceso: 21/05/2022].

[11] Fu, Anmin, Kaiping Xue, Hao Wang, Zhiyong Zhang y Jianping Wang: *Secure Federated Learning: A Survey.* IEEE Access, 9:23601–23615, 2021.

[12] Research, Google: *Federated Learning: Collaborative Machine Learning without Centralized Training Data.* `https://ai.googleblog.com/2017/04/federated-learning-collaborative.html`, Abril 2017. [Último acceso: 15/04/2022].

[13] Boneh y Dan: *Twenty Years of attacks on the RSA Cryptosystem.* `https://crypto.stanford.edu/~dabo/pubs/papers/RSA-survey.pdf`, Noviembre 1998. [Último acceso: 05/06/2023].

[14] Huallparimachi, Raul Huillca, Ecler Mamani Vilca y Karla C. Rojas Pedraza: *Esquema de cifrado indistinguible bajo el ataque de texto sin formato elegido (IND-CPA).* `https://revistas.unamba.edu.pe/index.php/riqchary/article/view/81`, Junio 2022. [Último acceso: 21/06/2023].

[15] Gauss, Carl Friedrich: *Disquisitiones Arithmeticae.* `https://web.archive.org/web/20080913204320/http://www.cimm.ucr.ac.cr/da/index.html`, 1995. [Último acceso: 03/06/2023].

[16] Apostol, Tom M.: *Introducción a la teoría analítica de números.* Editorial Reverté, S.A., 1984.

[17] Malumbres, Juan Luis Varona: *Recorridos por la Teoría de Números.* Comité editorial de la Real Sociedad Matemática Española, 2019.

[18] Szymczak, Kamil Kordian: *Euler's phi function, its properties, and how to compute it.* `https://codeforces.com/blog/entry/106851`, Mayo 2023. [Último acceso: 26/06/2023].

[19] Navarro, Gabriel: *Un curso de números.* Universitat de València, 2006.

[20] Francés, José Francisco Vicent: *Propuesta y análisis de criptosistemass de clave pública basados en matrices triangulares superiores por bloques.* `https://rua.ua.es/dspace/bitstream/10045/4081/1/tesis_doctoral_vicent_frances.pdf`, 2007. [Último acceso: 21/06/2023].

[21] Hardy, G. H. y E. M. Wright: *An introduction to the theory of numbers.* `https://t5k.org/glossary/page.php?sort=BertrandsPostulate`, 1979. [Último acceso: 19/06/2023].

[22] Paillier, Pascal: *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes.* `https://link.springer.com/content/pdf/10.1007/3-540-48910-X_16.pdf`, Agosto 2015. [Último acceso: 16/05/2023].

[23] Yi, Xun, Russell Paulet y Elisa Bertino: *Homomorphic Encryption and Applications*. Springer, 2014.

[24] Lopardo, Antonio, Tom Farrand y Adam J Hall: *What is homomorphic encryption?* `https://blog.openmined.org/what-is-homomorphic-encryption/`, Agosto 2020. [Último acceso: 15/04/2022].

[25] AEPD: *Cifrado y Privacidad III: Cifrado Homomórfico*. `https://www.aepd.es/es/prensa-y-comunicacion/blog/cifrado-privacidad-iii-cifrado-homomorfico`, Junio 2020. [Último acceso: 30/05/2022].

[26] Wikipedia: *Sistema criptográfico Paillier*. `https://es.wikipedia.org/wiki/Sistema_criptogr%C3%A1fico_Paillier`, Julio 2019. [Último acceso: 16/04/2022].

[27] Katz, Jonathan y Yehuda Lindell: *Introduction to Modern Cryptography: Principles and Protocols*, volumen 1. Chapman, Hall/CRC, 2007.

[28] Benaloh, J. C.: *Verifiable Secret-Ballot Elections*. PhD Thesis, Yale University, 1988.

[29] Smith, Alice y Bob Johnson: *Some Homomorphic Encryption Schemes*. 2022.

[30] Dijk, M. van, C. Gentry y S. Halevi: *Fully homomorphic encryption over the integers*. Springer, 2010.

[31] Belland, Michael, William Xue y Weilian Chu: *Somewhat Homomorphic Encryption*. `https://courses.csail.mit.edu/6.857/2017/project/22.pdf`, Mayo 2017. [Último acceso: 16/05/2022].

[32] Dijk, M. van, C. Gentry y S. Halevi: *Fully Homomorphic Encryption over the Integers*. Springer, 2010.

[33] Gentry, Craig: *Fully Homomorphic Encryption Using Ideal Lattices*. 2009.

[34] University, Bar Ilan: *Winter School on Cryptography: Fully Homomorphic Encryption - Craig Gentry*. `https://www.youtube.com/watch?v=Y1TxCiOuoYY`, Mayo 2012. [Último acceso: 18/04/2022].