Horizontes matemáticos: un viaje entre la abstracción y la realidad

SECRETOS DEL ÁLGEBRA: MATEMÁTICAS EN LA CRIPTOGRAFÍA MODERNA

Xiana Carrera Alonso

x.carrera@columbia.edu

Julio, 2025

Facultad de Matemáticas

Universidad de Santiago de Compostela

Índice

Estas notas recogen los contenidos impartidos en el minicurso de tres horas "Secretos del álgebra: Matemáticas en la criptografía moderna", que formó parte del curso de verano "Horizontes matemáticos: un viaje entre la abstracción y la realidad" impartido en julio de 2025 en la Universidad de Santiago de Compostela.

El minicurso tenía como objetivo ofrecer una introducción a la criptografía, una de las ramas fundamentales del ámbito de la seguridad de la información, desde un enfoque que permitiese contraponer los aspectos formales de su base matemática con los requisitos prácticos de su implementación.

La necesidad de protección de información ha sido una constante a lo largo de la historia, pero la creciente complejidad de este desafío mantiene ligadas inexorablemente las matemáticas y la criptografía. Lo que en su origen consistía en simples ideas de ocultación, hoy se ha transformado en sofisticados sistemas fundamentados en nociones profundas del álgebra abstracta y/o de la teoría de números. Con el fin de explicar estos últimos, estas notas se dividen en los siguientes capítulos:

1. **Introducción**. Da una visión de alto nivel de la criptografía, presentando la terminología relevante y la notación a utilizar en el desarrollo posterior.

2. **Criptografía clásica**. Repasa algunos de los sistemas criptográficos de mayor relevancia histórica, analizando sus virtudes y problemas. El interés de este apartado radica tanto en la similitud entre los componentes de sistemas modernos e ideas de algoritmos clásicos como en mostrar su evolución hasta dar lugar a los complejos mecanismos actuales.

3. **Criptografía simétrica: AES**. Describe el esquema de *Advanced Encryption Standard* (AES), uno de los algoritmos más utilizados hoy en día por su eficacia y seguridad, y que fue establecido como estándar por el Instituto Nacional de Estándares y Tecnología (NIST) en 2001.

4. **Introducción a la criptografía asimétrica**. Describe las principales ideas de los sistemas de criptografía asimétrica, que comenzaron a desarrollarse durante los años 70. Tras comentar la motivación detrás de uso y sus diferencias con respecto a los sistemas simétricos, se explica uno de los algoritmos más ampliamente usados, RSA, presentado en 1977.

5. **Criptografía de curva elíptica**. Introduce nociones básicas sobre curvas elípticas para después explicar los fundamentos y ventajas de los sistemas de criptografía asimétrica basados en ellas, poniendo como ejemplo el protocolo *Elliptic Curve Diffie-Hellman* (ECDH).

6. **Firmas digitales**. Presenta nociones básicas de firmas digitales, que son mecanismos criptográficos que permiten garantizar que un emisor ha generado un determinado mensaje, de forma análoga a las firmas manuscritas. Finalmente, se presenta el sistema DSA.

1. Introducción

La criptografía nace como respuesta a la necesidad de garantizar la seguridad de la información. Su uso conocido más antiguo se remonta a alrededor del año 1900 a.C., cuando se graban inscripciones cifradas en la tumba de Khnumhotep II, en Egipto. Se sabe del uso de diversas técnicas criptográficas posteriores por parte de civilizaciones como la griega y la romana, que fueron refinadas hacia finales de la Edad Media y a lo largo de la Edad Moderna. El interés por proteger secretos militares, de servicios diplomáticos y gubernamentales incitó su estudio y perfeccionamiento, así como su adaptación a las necesidades tecnológicas de cada época.

En la actualidad, el principal uso de la criptografía tiene lugar a través de medios digitales, y la concepción contemporánea de la seguridad de la información se ha adaptado a estos. Así, la *ciberseguridad* se apoya en tres pilares fundamentales:

- La *seguridad de los equipos*, centrada en controlar dispositivos. Por ejemplo, se deben establecer herramientas para mantener control de acceso (que requiere implementar métodos de autenticación y autorización de los usuarios), detectar y eliminar *malware* y actualizar el sistema operativo y las aplicaciones software regularmente para corregir vulnerabilidades.
- La *seguridad de red*, que supervisa el uso de redes informáticas y protege el intercambio de información entre equipos.
- La *criptografía*, que estudia formas de transformar los datos con el fin de que estos solamente puedan ser comprendidos por usuarios autorizados.

Definición 1. La *criptografía* es el estudio de técnicas matemáticas para imponer principios de la seguridad.

Definición 2. Los *principios de la seguridad* son:

- *Confidencialidad*: la información no puede ser leída por entidades no autorizadas. Hay numerosos enfoques para garantizar confidencialidad, desde el uso de protección física hasta algoritmos matemáticos que transforman los datos de forma que sean ininteligibles.
- *Integridad*: la información no puede ser modificada de forma no autorizada. Para asegurar la integridad de la información, debe ser posible detectar la manipulación de datos (por ejemplo, mediante su inserción, borrado o sustitución) por parte de entidades no autorizadas.
- *Autenticación del origen de los datos*: es posible corroborar que los datos recibidos realmente provienen de la entidad que dice haberlos enviado.

- *Autenticación de entidades*: es posible corroborar que las entidades involucradas son quienes dicen ser en tiempo real. Es decir, se puede asegurar que una entidad está activa en un momento dado de una comunicación. Este concepto es importante a la hora de proporcionar "frescura" y construir protocolos de comunicación.

- *No repudio*: una entidad no puede negar haber realizado una acción o aceptado un compromiso (prueba de autoría). Por ejemplo, si una entidad compra una propiedad a otra entidad y después niega haberlo hecho, es necesario disponer de un procedimiento que permita resolver la disputa.

Observación 1. (a) Es posible considerar otros principios de seguridad adicionales, como el *anonimato* de las entidades involucradas o la *disponibilidad* de los recursos (esto es, que las entidades autorizadas puedan acceder a la información en todo momento).

(b) La confidencialidad, integridad y disponibilidad, en conjunto, reciben el nombre de *tríada CIA* (del inglés *confidentiality*, *integrity*, *availability*).

Existen relaciones a tener en cuenta entre los mencionados principios de seguridad:

Observación 2.

- Autenticación del origen de los datos $\implies$ integridad. Si no se puede asegurar la integridad de los mensajes que envía un emisor, un atacante podría capturarlos y modificarlos de forma que sigan aparentando provenir de su emisor original, cuando en realidad quien envía la información modificada es el atacante.

- No repudio $\implies$ autenticación del origen de los datos. Solamente se pueden dar pruebas de autoría cuando se puede corroborar cuál es la verdadera fuente de la información.

- Autenticación del origen de los datos $\neq$ autenticación de entidades. Para verlo, basta considerar casos de uso donde solo uno de los dos principios es requerido. La autenticación de entidades es necesaria para autorizar el uso a recursos (que, por ejemplo, se puede llevar a cabo mediante un sistema de contraseñas). En cambio, en el contexto en el que una entidad recibe un email, es importante poder determinar de quién proviene, pero no es necesario corroborar si la entidad que lo envió sigue activa o se ha desconectado.

- Confidencialidad $\not\implies$ autenticación del origen de los datos. La confidencialidad (generalmente proporcionada a través de la encriptación de los datos) no garantiza que un mensaje no provenga de una entidad que se está haciendo pasar por otra.

En el ámbito de la criptografía es común emplear la siguiente terminología, que permite distinguir entre diferentes niveles de abstracción:

Definición 3.

- Una *primitiva criptográfica* es una operación matemática con propiedades de seguridad bien definidas que sirve como bloque fundamental para construir algoritmos y protocolos. Algunos ejemplos son la encriptación, las funciones hash y los generadores de números pseudo-aleatorios.

- Un *algoritmo criptográfico* es una especificación particular y precisa de una primitiva. Un algoritmo debe dar suficientes detalles como para que pueda ser implementado. Por ejemplo, AES es un algoritmo que especifica cómo llevar a cabo un proceso de encriptación.

- Un *protocolo criptográfico* es un conjunto estructurado de reglas, interacciones e intercambios de mensajes entre dos o más entidades que permite alcanzar un cierto nivel de seguridad en un entorno específico. Para lograr dicho objetivo, los protocolos pueden emplear diversos algoritmos y primitivas. Algunos protocolos criptográficos son HTTPS (protocolo de transferencia de hipertexto seguro) y Kerberos (un protocolo de autenticación en redes de ordenadores).

Consideraremos un último nivel de abstracción, el de *sistema criptográfico*, que en términos generales hace referencia a un conjunto de algoritmos criptográficos que implementan un principio de seguridad particular (generalmente, confidencialidad). Formalmente, lo podemos definir como sigue:

Definición 4. Un *sistema criptográfico*, *criptosistema* o *sistema de cifrado* es una 5-tupla $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ tal que:

(i) $\mathcal{P}$ es un conjunto finito de posibles *textos planos*.

(ii) $\mathcal{C}$ es un conjunto finito de posibles *textos cifrados*.

(iii) $\mathcal{K}$ es un conjunto finito de posibles *claves*.

(iv) $\mathcal{E} = \{E_k : k \in \mathcal{K}\}$ es un conjunto de funciones de encriptación

$$E_k : \mathcal{P} \longrightarrow \mathcal{C}.$$

(v) $\mathcal{D} = \{D_k : k \in \mathcal{K}\}$ es un conjunto de funciones de desencriptación

$$D_k : \mathcal{C} \longrightarrow \mathcal{P}.$$

(vi) $\forall\, e \in \mathcal{K}\ \ \exists\, d \in \mathcal{K}$ tal que $D_d(E_e(p)) = p\ \ \forall\, p \in \mathcal{P}$.

Por tanto, un sistema criptográfico típicamente incluye tres algoritmos: uno para la generación de claves, uno de encriptación y uno de desencriptación (si bien los dos últimos se pueden ver como partes de un único algoritmo).

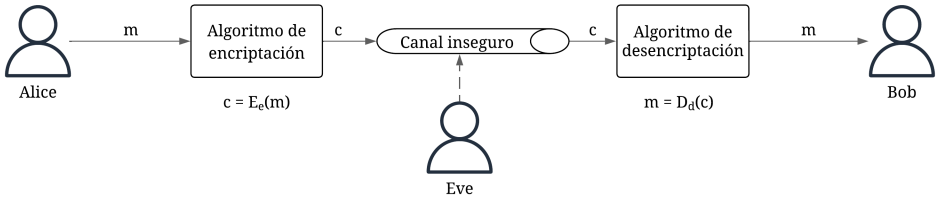

Figura 1: Modelo básico de comunicación entre dos entidades, Alice y Bob, donde Alice quiere enviar un mensaje a Bob y sus textos cifrados están expuestos a ataques de una tercera entidad, Eve.

En la Figura 1 se muestra un sistema criptográfico básico para proporcionar confidencialidad. En él, una usuaria, Alice, desea enviar un mensaje de forma segura a otro usuario, Bob, si bien el canal de transmisión que utilizan está expuesto a ataques de una tercera entidad, Eve. Los elementos que conforman este sistema son:

- Un texto plano m, el mensaje que Alice quiere hacer llegar a Bob.
- Un texto cifrado c, resultado de aplicar el algoritmo de encriptación E_e al texto plano m, siendo e una clave acordada previamente. Si el algoritmo de encriptación tiene buenas propiedades, el texto cifrado debería ser ininteligible y aparentar ser un texto aleatorio. Nótese que puede ser capturado por Eve, de modo que no debería ser considerado un secreto.
- Un algoritmo de encriptación E_e, que determina cómo cifrar un texto plano de entrada en función de una clave de encriptación e. E debe haber sido acordado de antemano por Alice y Bob.
- Un algoritmo de desencriptación D_d, que determina cómo recuperar el texto plano a partir del texto cifrado y de una clave de desencriptación d. Este algoritmo es elegido en conjunción con E_e.

La atacante, Eve, es una entidad externa que trata de determinar el texto plano m. Es posible que Eve consiga hacerse con el texto cifrado y que conozca los algoritmos de encriptación y desencriptación utilizados, pero en ningún caso debe permitirse que conozca la clave de desencriptación d.

Distinguimos dos tipos fundamentales de sistemas criptográficos:

- *Simétricos*, si $e = d$, esto es, si la clave de encriptación y la de desencriptación son la misma.
- *Asimétricos*, si $e \neq d$. En tal caso, se debe asegurar que sea *computacionalmente inviable*[1] determinar d a partir de e. En la mayor parte de sistemas,

[1]El significado intuitivo de que un cálculo sea *computacionalmente inviable* es que su coste en tiempo y/o memoria es demasiado alto para que pueda ser llevado a cabo con la tecnología actual en un margen de tiempo razonable.

d recibe el nombre de *clave privada* y e, *clave pública*. La clave privada debe ser mantenida en secreto, mientras que la clave pública puede ser dada a conocer libremente.

2. Criptografía clásica

En esta sección comentaremos algunas de técnicas de encriptación clásicas, que nos permitirán ilustrar las principales ideas detrás de los algoritmos actuales y los tipos de ataques que se deben esperar frente a ellos.

La criptografía simétrica fue la única empleada hasta el desarrollo de la criptografía asimétrica, en la década de 1970. Todos los sistemas que abarca esta sección son simétricos. Además, distinguiremos entre:

- *Cifrados por sustitución*, en los que se reemplazan símbolos o grupos de símbolos del texto plano por otros símbolos o grupos de símbolos.
- *Cifrados por transposición*, que permutan los símbolos del texto plano.

2.1. Cifrado César

El *cifrado César* es un cifrado por sustitución en el que se reemplaza cada letra del alfabeto del texto plano por una letra "desplazada" un número secreto de posiciones. El nombre de este sistema hace referencia a Julio César, que lo empleaba para proteger mensajes militares usando un desplazamiento de 3 letras.

Si el alfabeto del texto plano es el español, que tiene 27 letras, este se puede representar asignando números del 0 a 26 a cada letra, como se indica en el Cuadro 1.

Cuadro 1: Correspondencia entre las letras en español y los elementos de $\mathbb{Z}/27\mathbb{Z}$.

A	B	C	D	E	F	G	H	I	J	K	L	M	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13

Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26

La operación de desplazamiento se puede ver entonces como una suma empleando aritmética modular con módulo 27 (en inglés, con módulo 26).

Así, sean $n \in \mathbb{Z}/27\mathbb{Z}$ el desplazamiento secreto, $p = p_0\, p_1 \ldots p_r \in (\mathbb{Z}/27\mathbb{Z})^r$ el texto plano y $c = c_0\, c_1 \ldots c_r \in (\mathbb{Z}/27\mathbb{Z})^r$ el texto cifrado. Entonces

$$\begin{aligned} c_i &= E_n(p_i) = p_i + n \pmod{27} \\ p_i &= D_n(c_i) = c_i - n \pmod{27} \end{aligned}$$

para $i = 0, \ldots, r$.

Ejemplo 1. Con un desplazamiento de $n = 3$, tenemos la siguiente correspondencia de alfabetos:

A	B	C	D	E	F	G	H	I	J	K	L	M	N
D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P

Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Por ejemplo, si queremos encriptar `mensaje secreto y extraño`, tenemos que:

texto plano: `mensaje secreto y extraño`
texto cifrado: `ohpvdmh vhfuhwr b hawudqr`

Observación 3. En ocasiones, este sistema criptográfico recibe el nombre de *cifrado por desplazamiento*, y en su lugar se reserva el nombre de *cifrado César* para el caso en el que el desplazamiento es 3.

Para atacar este cifrado, es suficiente emplear un ataque de *fuerza bruta*, en la que se prueban cada una de las posibles claves $n \in \mathbb{Z}/27\mathbb{Z}$ hasta encontrar una con la que, al desencriptar el texto cifrado, se obtenga un mensaje reconocible y coherente[2]. Este ataque es viable porque solamente hay 27 claves posibles.

Observación 4. Suponiendo que hay $|K|$ posibles claves equiprobables en un sistema criptográfico dado, un ataque de fuerza bruta necesita $|K|/2$ intentos de media para encontrar la clave empleada, pues

$$\frac{1}{|K|}\sum_{i=1}^{|K|} i = \frac{1}{|K|}\frac{|K|\,(|K|+1)}{2} = \frac{|K|+1}{2} \approx \frac{|K|}{2}.$$

En particular, si $|K| = 2^k$, entonces se necesitan unos 2^{k-1} intentos de media.

[2]Podría ocurrir que varios desplazamientos den lugar a mensajes reconocibles y coherentes, pero distintos. En ese caso, no se podría determinar cuál es el verdadero mensaje original, a menos que se cuente con información externa sobre su estructura.

2.2. Cifrado afín y cifrado simple de sustitución

El cifrado César puede ser generalizado a un *cifrado afín* como sigue: dado un texto plano $p = p_0\, p_1 \dots p_r \in (\mathbb{Z}/27\mathbb{Z})^r$ y un texto cifrado $c = c_0\, c_1 \dots c_r \in (\mathbb{Z}/27\mathbb{Z})^r$ con una clave (m, n), donde $m \in (\mathbb{Z}/27\mathbb{Z})^\times$ y $n \in \mathbb{Z}/27\mathbb{Z}$, se pone

$$\begin{aligned} c_i &\equiv m\, p_i + n && \pmod{27} \\ p_i &\equiv m^{-1}(c_i - n) && \pmod{27} \end{aligned}$$

para $i = 0, \dots, r$.

El cifrado afín, a su vez, es un caso particular del *cifrado de sustitución simple*, que reemplaza un alfabeto por una permutación cualquiera de este.

Ejemplo 2. La siguiente permutación del alfabeto español define un cifrado de sustitución simple, pero no un cifrado afín:

A	B	C	D	E	F	G	H	I	J	K	L	M	N
H	F	K	D	J	W	V	N	S	I	Q	G	C	R

Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
T	M	B	L	Z	Ñ	Y	X	E	U	P	O	A

En el caso del cifrado de sustitución simple, la clave es la propia permutación, por lo que hay $|K| = 27!$ posibles claves. Nótese que $27! \approx 10^{28} \gg 10^{24} \approx$ el número de estrellas en el universo observable, que es una cantidad demasiado grande como para que sea viable emplear un ataque por fuerza bruta. En su lugar, se puede utilizar un *análisis de frecuencias*. Para ello, se compara la frecuencia media de cada letra en textos en español (véanse las Figuras 2 y 3) con la frecuencia de cada letra en el texto cifrado. A continuación, se buscan emparejamientos entre letras de frecuencias similares, comenzando por las más comunes, intentando deducir palabras a partir del texto cifrado. Si el proceso falla, se realizan nuevas iteraciones modificando ligeramente los emparejamientos hasta revelar un mensaje reconocible y coherente.

El motivo fundamental por el que este sistema es vulnerable a un análisis de frecuencias es que es un *cifrado monoalfabético*.

Definición 5. Se dice que un cifrado de sustitución es *monoalfabético* si cada símbolo del texto plano se reemplaza siempre por un mismo símbolo en el texto cifrado. En caso contrario, se dice que es *polialfabético*.

Este caso demuestra que, al construir una técnica, asegurar que el espacio de claves K es suficientemente grande no es una condición suficiente para asegurar su seguridad (aunque sí es necesaria, ya que de lo contrario sería vulnerable a ataques de fuerza bruta).

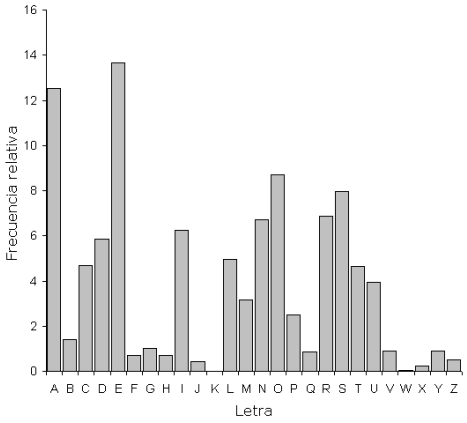

Figura 2: Frecuencia de uso de letras en español. Imagen de Tico, usada con licencia CC BY-SA 3.0.

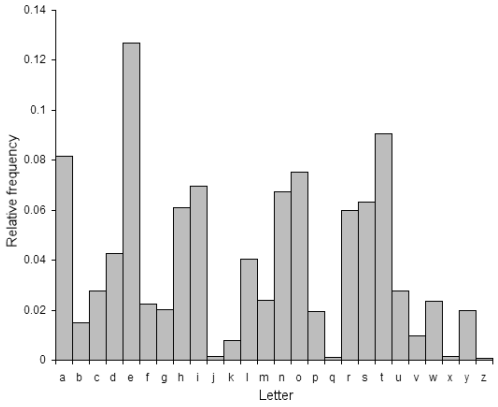

Figura 3: Frecuencia relativa de uso de letras en inglés. Imagen usada con licencia CC BY-SA 3.0.

2.3. Cifrado Hill

El *cifrado Hill* realiza sustituciones sobre bloques de r letras de texto plano. Estas sustituciones están determinadas por r ecuaciones lineales en las que, como antes, consideramos una asignación entre las letras del alfabeto español y los elementos de $\mathbb{Z}/27\mathbb{Z}$.

Este cifrado, desarrollado por el matemático Lester Hill en 1929, fue el primero de tipo *poligráfico*.

Definición 6. Se dice que un cifrado de sustitución es *poligráfico* si opera sobre grupos de símbolos. En caso contrario, se dice que es *simple*.

Formalmente, podemos describir este sistema como sigue: dada una clave $K = (k_{i,j})_{1\leq i,j\leq r} \in \mathcal{K} = \mathcal{M}_{r\times r}(\mathbb{Z}/27\mathbb{Z})$, un texto plano $P = (p_1\, p_2 \ldots p_r) \in \mathcal{P} = \mathcal{M}_{1\times r}(\mathbb{Z}/27\mathbb{Z})$ y el correspondiente texto cifrado $C = (c_1\, c_2 \ldots c_r) \in \mathcal{C} = \mathcal{M}_{1\times r}(\mathbb{Z}/27\mathbb{Z})$, se tiene

$$\begin{aligned} C &= E_K(P) = P\,K \\ P &= D_K(C) = C\,K^{-1}. \end{aligned}$$

Es decir, el texto cifrado se obtiene como

$$(c_1\, c_2 \ldots c_r) = (p_1\, p_2 \ldots p_r) \begin{pmatrix} k_{1,1} & \ldots & k_{1,r} \\ \vdots & \ddots & \vdots \\ k_{r,1} & \ldots & k_{r,r} \end{pmatrix},$$

que es equivalente al siguiente sistema de ecuaciones lineales:

$$\begin{cases} c_1 = (p_1\, k_{1,1} + \cdots + p_r\, k_{r,1}) \pmod{27} \\ \vdots \\ c_i = (p_1\, k_{1,i} + \cdots + p_r\, k_{r,i}) \pmod{27} \\ \vdots \\ c_r = (p_1\, k_{1,r} + \cdots + p_r\, k_{r,r}) \pmod{27}. \end{cases}$$

Para que la matriz K sea invertible, se debe comprobar que $\det(K)$ lo sea en $\mathbb{Z}/27\mathbb{Z}$, esto es, que $\det(K) \in (\mathbb{Z}/27\mathbb{Z})^\times$. Equivalentemente, se debe comprobar que $\det(K)$ sea coprimo con 27. Si es así, su inversa se puede calcular como

$$K^{-1} = (\det(K))^{-1}(\text{adj } K)^t,$$

con $(\text{adj } K)_{i,j} = (-1)^{i+j} D_{i,j}$, donde $D_{i,j}$ es el determinante de la matriz $(r-1) \times (r-1)$ obtenida eliminando la i-ésima fila y la j-ésima columna de K.

El principal beneficio de las técnicas poligráficas es que ocultan las frecuencias de letras individuales. De hecho, el cifrado Hill oculta las frecuencias de secuencias de longitud menor que r: de longitud 2 (denominadas *digramas* o *bigramas*), como "ch" o "de"; de longitud 3 (*trigramas*), como "ion"; y así progresivamente, hasta $r-1$ ($(r-1)$-*gramas*).

No obstante, se puede aprovechar la linealidad del cifrado Hill para construir un ataque haciendo uso de r parejas de texto cifrado-texto plano. Denotando cada texto cifrado como $C_j = (c_{1,j}\, c_{2,j} \ldots c_{r,j})$ y el correspondiente texto plano como $P_j = (p_{1,j}\, p_{2,j} \ldots p_{r,j})$, de forma que se tenga que $C_j = P_j\, K$ para $j = 1, \ldots, r$, se construyen matrices $X = (p_{i,j}) \in \mathcal{M}_{r\times r}(\mathbb{Z}/27\mathbb{Z})$, $Y = (c_{i,j}) \in \mathcal{M}_{r\times r}(\mathbb{Z}/27\mathbb{Z})$. Se tiene entonces la ecuación $Y = X\, K$, de donde se puede dar una expresión para la clave K:

$$K = X^{-1}\, Y.$$

Nótese que, si X no es una matriz invertible, se pueden reemplazar filas de X e Y por nuevas parejas de texto plano y texto cifrado hasta que X sea invertible.

2.4. Cifrado Vigenère

El *cifrado Vigenère* encripta cada letra del texto plano con un cifrado César diferente, esto es, con un desplazamiento distinto. Es, por tanto, un cifrado de sustitución polialfabético.

Para emplearlo, en primer lugar se escoge una clave textual $k = k_0\, k_1 \ldots k_{L-1}$, con $k_i \in (\mathbb{Z}/27\mathbb{Z})$. La encriptación y desencriptación entre un texto plano

$p = p_0\, p_1 \ldots p_r$ y el correspondiente un texto cifrado $c = c_0\, c_1 \ldots c_r$ vienen dadas por

$$\begin{aligned} c_i &= E_k(p_i) = p_i + k_{i \pmod L} && \pmod{27} \\ p_i &= D_k(c_i) = c_i - k_{i \pmod L} && \pmod{27} \end{aligned}$$

para $i = 0, \ldots, r$.

Ejemplo 3. Supongamos que queremos encriptar el mensaje `este es mi mensaje` con la clave `secreto`. Para ello, a cada una de las letras del texto plano y de la clave se le asigna el correspondiente elemento en $(\mathbb{Z}/27\mathbb{Z})$ según el Cuadro 1. La clave se repite hasta que alcanza una longitud igual a la del texto plano. A continuación, se encripta cada letra del mensaje original con un cifrado César cuyo desplazamiento viene dado por la correspondiente letra de la clave:

E	S	T	E	E	S	M	I	M	E	N	S	A	J	E
4	19	20	4	4	19	12	8	12	4	13	19	0	9	4
S	E	C	R	E	T	O	S	E	C	R	E	T	O	S
19	4	2	18	4	20	15	19	4	2	18	4	20	15	19
23	23	22	22	8	12	0	0	16	6	4	23	20	24	23
W	W	V	V	I	M	A	A	P	G	E	W	T	X	W

Este sistema fue propuesto por Giovan Battista Bellaso en 1553, pero en el siglo XIX fue erróneamente atribuido a Blaise de Vigenère, de quien tomó el nombre que ha perdurado hasta hoy.

La estructura polialfabética de este sistema le permite ocultar la información relativa a la frecuencia de las letras del mensaje original. De hecho, durante siglos se creyó que era imposible de descifrar, lo que le valió el apodo de *le chiffre indéchiffrable* ("el cifrado irrompible"). Sin embargo, la información sobre la frecuencia de las letras no se elimina por completo y, basándose en ella, en 1863 Friedrich Kasiski publica un ataque que rompe[3] este sistema y otros construidos de manera similar.

El *método Kasiski* se basa en hacer uso de las repeticiones de la clave k. Estas provocan que, a intervalos regulares, se esté empleando el mismo cifrado César, de forma que se puede emplear un ataque de análisis de frecuencias sobre las posiciones situadas a una distancia igual a la longitud de k, $|k|$. Para averiguar $|k|$, el método Kasiski busca repeticiones de grupos de tres o más caracteres en el texto cifrado y prueba a considerar un $|k|$ tal que $|k|$ divida la distancia entre dichas repeticiones.

[3]Decimos que un sistema criptográfico está *roto* cuando su nivel de seguridad es mucho menor que el intencionado en su diseño, normalmente debido a la construcción de un ataque que es suficientemente rápido, al explotar sus vulnerabilidades.

Ejemplo 4. Si encriptamos `el sabado en el solar` con la clave `perro`, tenemos:

E	**L**	**S**	A	B	A	D	O	E	N	**E**	**L**	**S**	O	L	A	R
4	**11**	**19**	0	1	0	3	15	4	13	**4**	**11**	**19**	15	11	0	18
P	**E**	**R**	R	O	P	E	R	R	O	**P**	**E**	**R**	R	O	P	E
16	**4**	**18**	18	15	16	4	18	18	15	**16**	**4**	**18**	18	15	16	4
20	**15**	**10**	18	16	16	7	6	22	1	**20**	**15**	**10**	6	26	16	22
T	O	K	R	P	P	H	G	V	B	T	O	K	G	Z	P	V

En el mensaje original, la secuencia resaltada `els` aparece repetida. Como esta se alinea con la clave de la misma forma, el resultado de la encriptación es el mismo: `tok`. Entre ambas repeticiones hay 10 posiciones de distancia, por lo que, si quisiéramos realizar un ataque siguiendo el método Kasiski, haríamos análisis de frecuencias considerando longitudes de clave 1, 5 y 10.

Otra posible forma de determinar $|k|$ es mediante el *test de Friedman*, inventado por William F. Friedman en la década de 1920, y que utiliza en el llamado *índice de coincidencia*, que se basa en comparar la distribución de coincidencias de letras en textos.

Observación 5. La encriptación de la máquina Enigma, utilizada para el envío de mensajes confidenciales entre los alemanes durante la Segunda Guerra Mundial, se puede ver como una rotación de alfabetos de sustitución con una clave de gran tamaño, de forma similar al cifrado Vigenère. El alfabeto de sustitución a usar es determinado por la configuración de la máquina y la posición de rotores que se mueven a medida que se introducen letras del mensaje a codificar.

2.5. One-time pad

One-time pad (OTP), que en ocasiones se traduce al español como *libreta de un solo uso*, es un sistema criptográfico con cuatro condiciones fundamentales:

1. La longitud de la clave debe ser mayor o igual que la del texto plano.
2. La clave debe ser aleatoria e independiente del texto plano.
3. Ni la clave ni partes de la clave pueden usarse para encriptar más de un texto plano.
4. Las entidades participantes en la comunicación deben mantener la clave completamente en secreto.

Si estas condiciones se garantizan, OTP es imposible de romper.

OTP se puede definir con diferentes operaciones, como XOR, XNOR o aritmética modular sobre $\mathbb{Z}/27\mathbb{Z}$. Si optamos por XOR, estaremos considerando una operación sobre bits (valores 0 y 1) dada por la tabla de verdad 2.

Cuadro 2: Tabla de verdad de la operación XOR ($\oplus$).

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Sean entonces $\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}/2\mathbb{Z})^r$ y consideremos $p = p_0\, p_1 \dots p_r \in \mathcal{P}$, $c = c_0\, c_1 \dots c_r \in \mathcal{C}$ y $k = k_0\, k_1 \dots k_r \in \mathcal{K}$, con $k_i \sim \mathcal{U}\{0,1\}$ (distribución uniforme discreta en $\{0,1\}$) e independientes entre sí. OTP viene dado por

$$c_i = E_{k_i}(p_i) = p_i + k_i \pmod 2 = p_i \oplus k_i$$
$$p_i = D_{k_i}(c_i) = c_i + k_i \pmod 2 = c_i \oplus k_i.$$

Dados $a, b \in (\mathbb{Z}/2\mathbb{Z})^r$, escribiremos $a \oplus b$ para indicar que se aplica la operación XOR bit a bit.

Para ver por qué es imposible de romper, consideremos un sistema criptográfico $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ cualquiera. Sean $\mathbf{p}$, $\mathbf{c}$ y $\mathbf{k}$ las variables aleatorias que representan elegir un determinado texto plano, texto cifrado y clave de $\mathcal{P}$, $\mathcal{C}$ y $\mathcal{K}$, respectivamente.

Definición 7. Un sistema criptográfico $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ tiene *seguridad perfecta* si

$$\mathbb{P}[\mathbf{p} = p \mid \mathbf{c} = c] = \mathbb{P}[\mathbf{p} = p]$$

para cualesquiera $p \in \mathcal{P}$, $c \in \mathcal{C}$.

Proposición 1. *Las siguientes afirmaciones son equivalentes:*

1. $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ *tiene seguridad perfecta.*
2. *Las variables aleatorias* $\mathbf{p}$ *y* $\mathbf{c}$ *son independientes.*
3. $\mathbb{P}[\mathbf{c} = c \mid \mathbf{p} = p] = \mathbb{P}[\mathbf{c} = c]$ *para cualesquiera* $p \in \mathcal{P}$, $c \in \mathcal{C}$.
4. $\mathbb{P}[\mathbf{c} = c \mid \mathbf{p} = p_1] = \mathbb{P}[\mathbf{c} = c \mid \mathbf{p} = p_2]$ *para cualesquiera* p_1, $p_2 \in \mathcal{P}$.

En 1949, Shannon enuncia y prueba el siguiente teorema:

Teorema 1. *Sea* $S := (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ *un sistema criptográfico tal que* $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}|$.

S tiene seguridad perfecta si y solo si:

1. *Cada clave es usada con igual probabilidad,* $1/|K|$.
2. *Para cualesquiera* $p \in \mathcal{P}$ *y* $c \in \mathcal{C}$, *existe una única clave* $k \in \mathcal{K}$ *tal que* $E_k(p) = c$.

Demostración. " $\Longrightarrow$ ": Supongamos que S tiene seguridad perfecta.

Podemos asumir que $\mathbb{P}[\mathbf{p} = p] > 0 \ \forall \ p \in \mathcal{P}$ y que $\mathbb{P}[\mathbf{c} = c] > 0 \ \forall \ c \in \mathcal{C}$. Fijemos un texto plano $p \in \mathcal{P}$. Para cada $c \in \mathcal{C}$, usando la proposición anterior, tenemos que $\mathbb{P}[\mathbf{c} = c \mid \mathbf{p} = p] = \mathbb{P}[\mathbf{c} = c] > 0$. Por tanto, para todo $c \in \mathcal{C}$ existe $k \in \mathcal{K}$ tal que $E_k(p) = c$.

Por consiguiente, $|C| = |\{E_k \mid k \in \mathcal{K}\}| \leq |\mathcal{K}|$. Como $|C| = |K|$, no existen dos claves k_1, $k_2 \in \mathcal{K}$ distintas tales que $E_{k_1}(p) = E_{k_2}(p) = c$.

Con esto hemos probado que, para cualesquiera $p \in \mathcal{P}$ y $c \in \mathcal{C}$, existe una única clave $k \in \mathcal{K}$ tal que $E_k(p) = c$.

Para ver que cada clave es usada con probabilidad $1/|K|$, fijemos un texto cifrado $c \in \mathcal{C}$. Escribiendo $\mathcal{P} = \{p_1, \ldots, p_n\}$, sea $k_i \in \mathcal{K}$ la clave tal que $E_{k_i}(p_i) = c$ para todo $i = 1, \ldots, n$. Por el teorema de Bayes se tiene que

$$\mathbb{P}[\mathbf{p} = p_i \mid \mathbf{c} = c] = \frac{\mathbb{P}[\mathbf{c} = c \mid \mathbf{p} = p_i] \, \mathbb{P}[\mathbf{p} = p_i]}{\mathbb{P}[\mathbf{c} = c]} = \frac{\mathbb{P}[\mathbf{k} = k_i] \, \mathbb{P}[\mathbf{p} = p_i]}{\mathbb{P}[\mathbf{c} = c]}.$$

Que S tenga seguridad perfecta implica que $\mathbb{P}[\mathbf{p} = p_i \mid \mathbf{c} = c] = \mathbb{P}[\mathbf{p} = p_i]$. Deducimos entonces que $\mathbb{P}[\mathbf{c} = c] = \mathbb{P}[\mathbf{k} = k_i$.

Por consiguiente, $\mathbb{P}[\mathbf{k} = k_i = \mathbb{P}[\mathbf{c} = c] = \mathbb{P}[\mathbf{k} = k_j$ para cualesquiera i, j, de modo que las claves son equiprobables. Puesto que hay $|K|$ claves, se tiene que la probabilidad de cada una de ellas es $1/|K|$.

" $\Longleftarrow$ ": Supongamos que $\mathbb{P}[\mathbf{k} = k] = 1/|K|$ para toda clave $k \in \mathcal{K}$, y que para todo $p \in \mathcal{P}$ y $c \in \mathcal{C}$ existe una única clave $k \in \mathcal{K}$ tal que $E_k(p) = c$. Sigue entonces que

$$\mathbb{P}[\mathbf{c} = c \mid \mathbf{p} = p] = \frac{1}{|K|}$$

para todo $p \in \mathcal{P}$ y $c \in \mathcal{C}$. Por tanto, dados $c \in \mathcal{C}$ y p_1, $p_2 \in \mathcal{P}$, se tiene que

$$\mathbb{P}[\mathbf{c} = c \mid \mathbf{p} = p_1] = \frac{1}{|K|} = \mathbb{P}[\mathbf{c} = c \mid \mathbf{p} = p_2]$$

y, empleando la proposición anterior, S tiene seguridad perfecta. □

Corolario 1. *One-time pad tiene seguridad perfecta.*

Demostración. En primer lugar, comprobemos que OTP cumple la condición (vi) de la definición de sistema criptográfico 4 para el caso simétrico. Esto es, veamos que para todo $p \in \mathcal{P} = (\mathbb{Z}/2\mathbb{Z})^r$ y para todo $k \in \mathcal{K} = (\mathbb{Z}/2\mathbb{Z})^r$, tenemos que $D_k(E_k(p)) = p$. Para ello, basta observar que

$$\begin{aligned} D_k(E_k(p)) &= D_k(p \oplus k) && \text{(definición de encriptación)} \\ &= (p \oplus k) \oplus k && \text{(definición de desencriptación)} \\ &= p \oplus (k \oplus k) && \text{(asociatividad de XOR)} \\ &= p && (k \oplus k = 0). \end{aligned}$$

Puesto que los bits de las claves son variables aleatorias independientes e idénticamente distribuidas con distribución uniforme discreta $\mathcal{U}\{0,1\}$, tenemos que

$$\mathbb{P}[\mathbf{k} = k] = \frac{1}{|\{0,1\}|^r} = \frac{1}{2^r} = \frac{1}{|K|}$$

para todo $k \in \mathcal{K}$.

Además, dados un texto plano $p \in \mathcal{P}$ y un texto cifrado $c \in \mathcal{C} = (\mathbb{Z}/2\mathbb{Z})^r$, existe una única clave $k \in \mathcal{K}$ tal que $p \oplus k = c$, que es

$$k = c \oplus p.$$

□

Una consecuencia importante de la seguridad perfecta es que garantiza la *seguridad incondicional*.

Definición 8. Un sistema es *incondicionalmente seguro* si no es posible romperlo incluso si se dispone de recursos computacionales ilimitados.

Proposición 2. *Un sistema con seguridad perfecta es incondicionalmente seguro frente a ataques donde el atacante solo dispone de texto cifrado.*

Sin embargo, OTP tiene importantes desventajas:

- Las claves deben ser tan largas como los mensajes, lo que dificulta asegurar el almacenamiento seguro de las claves y su distribución a las entidades que se quieren comunicar.
- Se requiere un mecanismo que permita generar grandes cantidades de bits de forma completamente aleatoria.
- Es necesario establecer protocolos de seguridad robustos para asegurar que ningún adversario tiene acceso a las claves y que cada una de ellas se utiliza a lo sumo una vez.

Debido a estos requerimientos, OTP es poco usado en aplicaciones modernas.

Observación 6. Es importante que cada clave no se reutilice. Supongamos que dos mensajes $p_1, p_2 \in \mathcal{P}$ se encriptan con la misma clave $k \in \mathcal{K}$. Si un adversario se hace con los textos cifrados $c_1 = p_1 \oplus k$ y $c_2 = p_2 \oplus k$, puede calcular

$$\begin{aligned} c_1 \oplus c_2 &= (p_1 \oplus k) \oplus (p_2 \oplus k) \\ &= p_1 \oplus (k \oplus k) \oplus p_2 \qquad \text{(conmutatividad y asociatividad de XOR)} \\ &= p_1 \oplus p_2 \qquad\qquad\qquad\qquad\qquad\qquad (k \oplus k = 0). \end{aligned}$$

El resultado, $p_1 \oplus p_2$, tiene la suficiente estructura como para que en muchos casos sea posible recuperar los mensajes originales.

2.6. Cifrados por transposición

Los *cifrados por transposición* son aquellos que realizan una permutación de los símbolos del texto plano.

Un ejemplo sencillo es el *cifrado rail fence* (valla de tablones), en el que el texto plano se escribe en diagonales, alternando la dirección (hacia abajo y hacia arriba, alternando cada vez que se alcanza el extremo inferior y superior, respectivamente), y luego se lee el resultado fila a fila (tablón a tablón).

Ejemplo 5. Si queremos encriptar el mensaje `espia capturado` con 3 filas (3 tablones), tendríamos:

E A T D
S I C P U A O
P A R

y, por tanto, el texto cifrado sería `eatd sicpuao par`.

No obstante, este cifrado se puede romper con un ataque de fuerza bruta, dado que la clave (el número de filas/tablones) no puede ser mayor o igual que la longitud del texto plano, o este quedaría inalterado.

Otro ejemplo es el *cifrado por transposición columnar simple*, en el que se escribe el mensaje sobre un rectángulo, fila a fila, y luego se lee columna a columna, pero donde el orden de las columnas está dado por la clave. Si el rectángulo es de tamaño $m \times n$, entonces la clave es una permutación de n elementos, de forma que el tamaño del espacio de claves es $n!$.

Ejemplo 6. Supongamos que queremos encriptar el mensaje `entrada libre a las tres` en un rectángulo 4×5 con clave $(3, 2, 1, 4, 5)$. Tendremos

Clave:	3	2	1	4	5
Texto cifrado:	e	n	t	r	a
	d	a	l	i	b
	r	e	a	l	a
	s	t	r	e	s

y, por tanto, el texto cifrado será `tlarnaetedrsrileabas`.

Para fortalecer el cifrado por transposición columnar simple, se puede sencillamente aplicar dos veces.

Los cifrados de transposición pueden ser detectados fácilmente, ya que mantiene la frecuencia de las letras del alfabeto del texto plano. Por otro lado, claves similares a la auténtica producirán permutaciones similares, de modo que es posible construir ataques empleando algoritmos de optimización como *hill-climbing* o algoritmos genéticos.

3. Criptografía simétrica: AES

En 1997, el Instituto Nacional de Normas y Tecnología (NIST) anunció un concurso destinado a un nuevo algoritmo de cifrado que reemplazase al antiguo estándar, DES (*Data Encryption Standard*). Las principales condiciones para los algoritmos presentados eran las siguientes:

- Ser de cifrado simétrico y operar sobre bloques de 128 bits.
- Admitir claves de longitud variable (128, 192 y 256 bits).
- Ser más rápido que el algoritmo Triple DES al ejecutarse en diferentes plataformas.

Se propusieron quince algoritmos, que fueron sometidos a un intenso escrutinio. En agosto de 1999 se anunciaron los cinco finalistas: MARS, RC6, Rijndael, Serpent y Twofish. Finalmente, en 2000, Rijndael (diseñado por Vincent Rijmen y Joan Daemen) fue seleccionado como el algoritmo ganador, principalmente por motivos de rendimiento. En 2001, el NIST publica una versión ligeramente modificada de Rijndael, que pasa a conocerse como AES (*Advanced Encryption Standard*) y que actualmente es un estándar del gobierno de Estados Unidos para la encriptación.

AES es una *red de sustitución-permutación*, en la que se alternan operaciones de sustitución y de transposición, con el objetivo de aprovechar las ventajas que ofrecen ambos enfoques.

Parte de las operaciones de AES utilizan la aritmética del cuerpo finito de $2^8 = 256$ elementos, que se introduce a continuación.

3.1. Preliminares sobre cuerpos finitos

Los cuerpos finitos (los cuerpos con un número finito de elementos) tienen una gran importancia en criptografía, ya que el hecho de que la memoria de un ordenador sea finita implica que solamente puede manejar números con un tamaño finito. En contraposición, no todos los elementos de cuerpos infinitos como $\mathbb{Q}$ o $\mathbb{R}$ se pueden representar de forma exacta, lo cual introduce errores de redondeo y truncamiento que pueden dar lugar a errores durante la ejecución de protocolos o crear vulnerabilidades.

Adicionalmente, querremos trabajar con cuerpos para poder utilizar operaciones de suma y multiplicación para las que esté garantizada la existencia de elementos opuestos e inversos, fundamental para invertir operaciones (por ejemplo, para que sea posible desencriptar).

Los siguientes resultados caracterizan los cuerpos finitos:

Teorema 2. *Consideremos un primo p y un entero $m \geq 1$.*

(i) Existe un cuerpo finito de p^m elementos.

(ii) Dados dos cuerpos finitos de p^m elementos, estos son isomorfos.

Observación 7. Habitualmente, el cuerpo finito de p^m elementos se denota por $\mathbb{F}_{p^m}$ o $\mathrm{GF}(p^m)$, donde GF simboliza *Galois field*, cuerpo de Galois, en honor a Évariste Galois, el primer matemático que estudió los cuerpos finitos.

Proposición 3. *No existen cuerpos finitos de q elementos si q no es una potencia de un primo.*

Ejemplo 7. El cuerpo finito de 7 elementos es

$$\mathbb{F}_7 \cong \mathbb{Z}/7\mathbb{Z} = \{0, 1, 2, 3, 4, 5, 6\}.$$

Nos serán especialmente útiles los cuerpos finitos de 2^m elementos, ya que esto permite trabajar cómodamente con bits y bytes, así como optimizar el almacenamiento en memoria.

Para construir explícitamente un cuerpo $\mathbb{F}_{p^m}$ elementos, con $m > 1$, consideremos un polinomio $f \in \mathbb{F}_p[X]$ irreducible, mónico y de grado m, que ha de ser de la forma

$$f(X) = X^m + a_{m-1}X^{m-1} + \cdots + a_1X + a_0, \quad \text{con } a_i \in \mathbb{F}_p.$$

Como f es irreducible y $\mathbb{F}_p$ es un cuerpo (lo que implica que $\mathbb{F}_p[X]$ es un dominio de ideales principales), tenemos que el ideal generado por f, (f), es maximal. Por tanto,

$$\mathbb{F}_{p^m} := {}^{\mathbb{F}_p[X]}\!\diagup_{(f)}$$

es un cuerpo, y se puede demostrar que tiene p^m elementos.

Por consiguiente, $\mathbb{F}_{p^m}$ está formado por polinomios con coeficientes en $\mathbb{F}_p$ módulo f. Tenemos entonces que sus elementos son de la forma

$$g(X) = \sum_{i=0}^{m-1} b_i X^i$$

con $b_i \in \mathbb{F}_p$.

Ejemplo 8.

1. $\mathbb{F}_{2^3} \cong {}^{\mathbb{F}_2[X]}\!\diagup_{(X^3+X+1)} \cong {}^{\mathbb{F}_2[X]}\!\diagup_{(X^3+X^2+1)}$. Este ejemplo muestra que un cuerpo finito puede estar definido por diferentes polinomios.

2. AES trabaja con el cuerpo de 2^8 elementos y el polinomio $m(X) := X^8 + X^4 + X^3 + X + 1$, es decir, ${}^{\mathbb{F}_2[X]}\!\diagup_{(X^8+X^4+X^3+X+1)} \cong \mathbb{F}_{2^8}$.

Las propiedades del cuerpo $\mathbb{F}_2[X]/(X^8+X^4+X^3+X+1)$ facilitan el cálculo de la suma y multiplicación de polinomios. Sean, por ejemplo, los polinomios $g(X)=X^6+X^4+X^2+X+1$ y $h(X)=X^7+X+1$. Puesto que sus coeficientes están en $\mathbb{F}_2$, su suma será $g(X)+h(X)=X^7+X^6+X^4+X^2$.

$$\begin{array}{cccccccc} & & X^6 & & X^4 & & X & 1 \\ + & X^7 & & & & & X & 1 \\ \hline & X^7 & X^6 & & X^4 & & & \end{array}$$

Ahora bien, esta operación es equivalente a considerar el XOR de 010101011 y 100000011: $010101011 \oplus 100000011 = 11010100$. Por tanto, es fácilmente implementable en un ordenador.

$$\begin{array}{cccccccc} & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ \oplus & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ \hline & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{array}$$

Multiplicar $f(X)$ y $g(X)$ con el método habitual (multiplicando monomio a monomio y después aplicando un módulo por m si el grado del resultado es mayor o igual que 8) es una operación bastante costosa. No obstante, es posible acelerarla usando que

$$X^8 \bmod m(X) = \overline{m(X)-X^8} = X^4+X^3+X+1. \tag{1}$$

Por consiguiente, dado un polinomio genérico de $\mathbb{F}_2[X]/(X^8+X^4+X^3+X+1)$, $\sum_{i=0}^{7} a_i X^i$, tenemos que

$$X\sum_{i=0}^{7} a_i X^i = a_7X^8 + a_6X^7 + \cdots + a_1X^2 + a_0X.$$

Si $a_7=0$, no es necesario aplicar el módulo por m. Si $a_7=1$, sí, de modo que aplicando (1) obtendríamos

$$X\sum_{i=0}^{7} a_i X^i = a_6X^7 + \cdots + a_1X^2 + a_0X + (X^4+X^3+X+1).$$

Representando los polinomios por sus coeficientes (esto es, en binario), tenemos que

$$X\cdot(a_7\,a_6\ldots a_1\,a_0) = \begin{cases} (a_6\ldots a_1\,a_0\,0) & \text{si } a_7=0 \\ (a_6\ldots a_1\,a_0\,0)\ \oplus\ (00011011) & \text{si } a_7=1. \end{cases}$$

Para multiplicar por potencias de X, basta aplicar repetidas veces el procedimiento anterior. Y, para multiplicar por polinomios, basta aplicar la propiedad distributiva con respecto a la suma (que recordemos que es una operación XOR). Por ejemplo,

$$\begin{aligned} g(X) \cdot (00000110) &= g(X) \cdot (00000100 \oplus 00000010) \\ &= [g(X) \cdot (00000100)] \oplus [g(X) \cdot (00000010)]. \end{aligned}$$

3.2. Encriptación con AES

AES opera sobre una entrada de 16 bytes (128 bits), que se disponen por columnas en una matriz 4×4:

$$\begin{pmatrix} b_{0,0} & \cdots & b_{0,3} \\ \vdots & \ddots & \vdots \\ b_{3,0} & \cdots & b_{3,3} \end{pmatrix}.$$

Sobre esta matriz "estado" se van realizando sucesivas operaciones hasta obtener el resultado cifrado final. Estas operaciones se organizan en rondas, cuyo número depende del tamaño de la clave. Para claves de 128 bits, se realizan 10 rondas; para claves de 192 bits, 12 rondas; y para claves de 256 bits, 14 rondas.

Cada ronda aplica las siguientes cuatro operaciones, en orden:

1. *SubBytes*: Sustitución de bytes. Cada uno de los 16 bytes de la matriz estado es sustituido por otro byte según una tabla fija y cuyos detalles forman parte del algoritmo y son públicos.

2. *ShiftRows*. Cada una de las filas de la matriz estado es desplazada a la izquierda un cierto número de posiciones. Aquellas entradas que quedan fuera del margen izquierdo se re-insertan por la derecha. La primera fila queda intacta, la segunda se desplaza una posición, la tercera se desplaza dos posiciones y la cuarta se desplaza tres posiciones.

3. *MixColumns*. Cada una de las columnas de la matriz estado es multiplicada por una matriz fija y cuyos detalles forman parte del algoritmo, empleando la aritmética de $\mathbb{F}_2[X]/(X^8 + X^4 + X^3 + X + 1)$:

$$\begin{pmatrix} b'_{0,j} \\ b'_{1,j} \\ b'_{2,j} \\ b'_{3,j} \end{pmatrix} := \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} b_{0,j} \\ b_{1,j} \\ b_{2,j} \\ b_{3,j} \end{pmatrix} \quad \text{para } 0 \leq j \leq 3,$$

   donde hemos representado los polinomios de $\mathbb{F}_2[X]/(X^8+X^4+X^3+X+1)$ por su valor numérico en base 10. Por ejemplo, 1 sería 00000001, 2 sería 00000010, etc.

4. *AddRoundKey*. Se calcula un XOR de uno de los 16 bytes de la matriz estado con cada uno de los 16 bytes de otra matriz correspondiente a la clave (ligeramente alterada y expandida con respecto a la clave original).

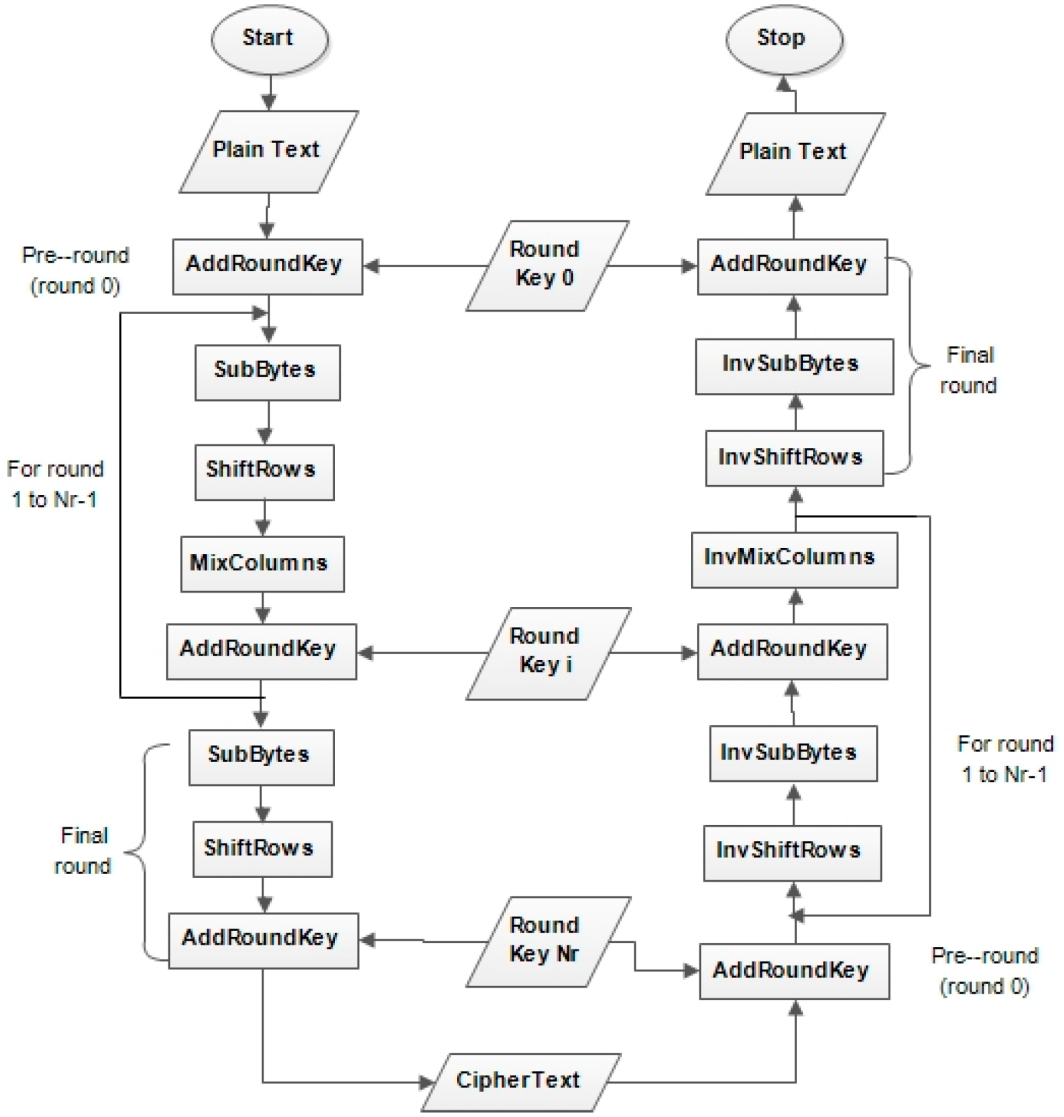

Figura 4: Operaciones de las rondas de AES en la encriptación (izquierda) y en la desencriptación (derecha). Imagen de [**abiyoke**].

La primera y última ronda de cada encriptación son diferentes: en la primera, solo se aplica *AddRoundKey*, y en la última, se aplica *SubBytes*, *ShiftRows* y *AddRoundKey*, sin *MixColumns*. La Figura 4 muestra un diagrama con el esquema de rondas completo.

3.3. Desencriptación con AES

En AES, la desencriptación consiste en invertir cada una de las operaciones de la encriptación y aplicarlas en orden contrario.

Por consiguiente, cada ronda consiste de las operaciones $ShiftRows^{-1}$, $SubBytes^{-1}$, $AddRoundKey^{-1}$ y $MixColumns^{-1}$. De nuevo, en la primera ronda solo se aplica $AddRoundKey^{-1}$, y en la última no se aplica $MixColumns^{-1}$.

AddRoundKey, al consistir en una operación XOR, es su propia inversa: $AddRoundKey = AddRoundKey^{-1}$. Para invertir *ShiftRows*, basta mover las filas las correspondientes posiciones a la derecha y, para invertir *SubBytes*, basta utilizar una tabla de sustitución inversa. En el caso de *MixColumns*, se multiplica con la inversa de la matriz empleada en la encriptación.

4. Introducción a la criptografía asimétrica

4.1. Motivación

La criptografía asimétrica fue desarrollada a lo largo de la década de 1970 con el objetivo de superar algunos de los problemas más restrictivos de la criptografía simétrica, como la necesidad de establecer una clave secreta entre las entidades participantes a través de un canal seguro. Las dificultades logísticas de este proceso no son triviales y, además, se puede volver muy costoso rápidamente, ya que para permitir la comunicación de N participantes dos a dos, se necesitan establecer claves distintas para cada pareja de entidades, que supone un total de $N(N-1)/2$ claves (es decir, del orden de N^2).

La criptografía asimétrica resuelve este problema considerando dos claves para cada entidad: una privada, que se mantiene en secreto, y una pública, que puede ser dada a conocer libremente, establecida con el requisito de que sea computacionalmente inviable determinar la clave privada a partir de la clave pública. Que no sea necesario establecer canales seguros para transmitir las claves públicas simplifica enormemente su distribución. Las Figuras 5 y 6 comparan los modelos de encriptación y desencriptación simétricos y asimétricos.

No obstante, los algoritmos de encriptación asimétricos son considerablemente más lentos que los simétricos, de forma que, en la práctica, se suelen utilizar para comunicar datos de poco tamaño, como PINs, números de tarjetas de crédito o incluso claves que después pueden ser empleadas con algoritmos simétricos.

En este capítulo nos centraremos en describir RSA, un caso particular de los sistemas de encriptación asimétricos, cuyo principal objetivo es proporcionar confidencialidad, pero que no aseguran la autenticación del origen de los datos ni la integridad de los datos. No obstante, estos pueden ser proporcionados a través de firmas digitales, otro tipo de técnica de criptografía asimétrica.

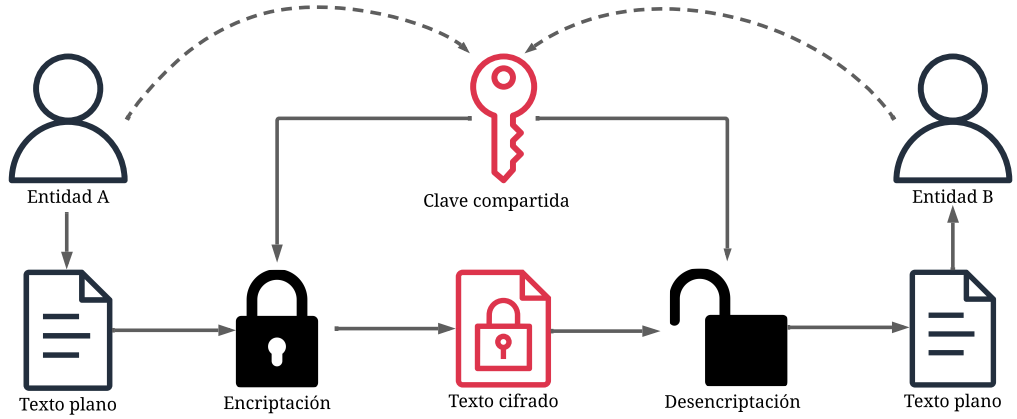

Figura 5: Modelo de encriptación y desencriptación simétrica.

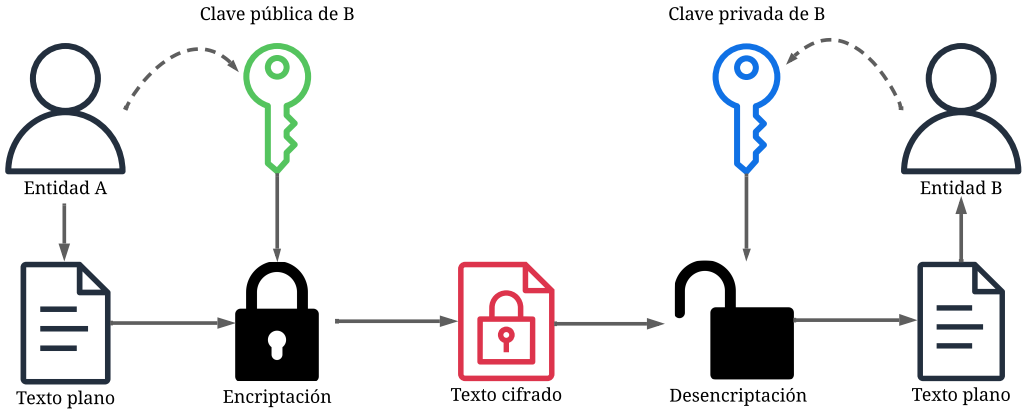

Figura 6: Modelo de encriptación y desencriptación asimétrica.

4.2. RSA

El sistema criptográfico RSA, publicado en 1977, debe su nombre a sus desarrolladores, R. Rivest, A. Shamir y L. Adleman. Aunque fue uno de los primeros cifrados asimétricos, sigue siendo uno de los más populares y habitualmente usados. Su seguridad se fundamenta en la intratabilidad[4] de la factorización de enteros en primos.

Para poder describir el sistema RSA, recordamos a continuación algunas ideas básicas sobre la función φ de Euler:

[4]Se dice que un problema es *intratable* cuando no existe un algoritmo que lo resuelva en un tiempo razonable (polinomial) en función del tamaño de entrada.

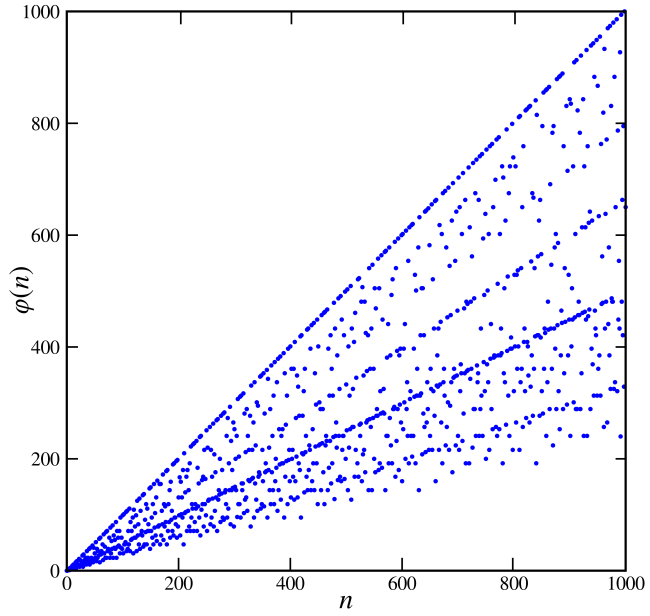

Figura 7: Primeros mil valores de la función φ de Euler.

Definición 9. La función φ de Euler (o función totiente de Euler) viene dada por

$$\begin{aligned}\varphi : \mathbb{N} &\longrightarrow \mathbb{N} \\ n &\longmapsto \varphi(n) = |\{k \in \mathbb{N} \mid 1 \leq k \leq n \text{ y } \gcd(k,n) = 1\}|\,.\end{aligned}$$

Es decir, dado un entero positivo n, $\varphi(n)$ es la cantidad de enteros positivos menores que n y coprimos con él.

Cuadro 3: Primeros diez valores de la función φ de Euler.

n	1	2	3	4	5	6	7	8	9	10
$\varphi(n)$	1	1	2	2	4	2	6	4	6	4

Los primeros mil valores de la función φ de Euler aparecen representados en la Figura 7. Los primeros diez se muestran en el Cuadro 3. Estos últimos se pueden calcular fácilmente a partir de las siguientes propiedades:

Proposición 4. *La función φ de Euler cumple que:*

(a) Si p es un número primo, se tiene que $\varphi(p) = p - 1$.

(b) Si p es un número primo y k es un entero positivo, entonces $\varphi(p^k) = (p-1)p^{k-1}$.

(c) Si m, n son enteros positivos y $\gcd(m,n) = 1$, se tiene que $\varphi(mn) = \varphi(m)\,\varphi(n)$.

(d) *Si p y q son primos y $p \neq q$, $\varphi(p\,q) = (p-1)\,(q-1)$.*

Demostración. (a) Como p es primo, todos los enteros positivos estrictamente menores que p son coprimos con p.

(b) Como p es primo, dado $m \in \mathbb{N}$ tenemos que $\gcd(p^k, m) \in \{1, p, p^2, \ldots, p^k\}$. Por tanto, $\gcd(p^k, m) > 1$ si y solo si m es múltiplo de p. Hay exactamente p^{k-1} enteros positivos que sean múltiplos de p y menores o iguales que p^k: $\{p, 2p, 3p, \ldots, p^{k-1}\,p = p^k\}$. Por tanto, los otros $p^k - p^{k-1} = p^{k-1}(p-1)$ números entre 1 y p^k son coprimos con p^k.

(c) Si $m = 1$ o $n = 1$, el enunciado se cumple.

Supongamos que $m > 1$ y $n > 1$. Sean

$$\begin{aligned} A &:= \{a \in \mathbb{N} \mid 1 \leq a < m,\ \gcd(a, m) = 1\} \\ B &:= \{b \in \mathbb{N} \mid 1 \leq b < n,\ \gcd(b, n) = 1\} \\ C &:= \{c \in \mathbb{N} \mid 1 \leq c < mn,\ \gcd(b, mn) = 1\}. \end{aligned}$$

Entonces, $|A| = \varphi(m)$, $|B| = \varphi(n)$ y $|C| = \varphi(mn)$. Por tanto, nos basta demostrar que $|C|$ y $|A \times B|$ tienen el mismo número de elementos.

Por el teorema chino de los restos, como $\gcd(m, n) = 1$, la aplicación

$$\begin{aligned} \pi : \mathbb{Z}/_{mn\mathbb{Z}} &\longrightarrow \mathbb{Z}/_{m\mathbb{Z}} \times \mathbb{Z}/_{n\mathbb{Z}} \\ x &\longmapsto (x \bmod m, x \bmod n) \end{aligned}$$

es biyectiva. Ahora bien, $A \subset \mathbb{Z}/m\mathbb{Z}$, $B \subset \mathbb{Z}/n\mathbb{Z}$ y $C \subset \mathbb{Z}/mn\mathbb{Z}$, y tenemos que $x \in C$ si y solo si $\pi(x) \in A \times B$, ya que

$$\begin{aligned} \gcd(x, mn) = 1 &\iff \gcd(x, m) = 1 \text{ y } \gcd(x, n) = 1 \\ &\iff \gcd(x \bmod m, m) = 1 \text{ y } \gcd(x \bmod n, n) = 1. \end{aligned}$$

□

Teorema 3 (Teorema de Euler)**.** *Sean a y n enteros positivos. Si* $\gcd(a, n) = 1$, *entonces*

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

Observación 8. El caso particular del teorema de Euler en el que n es primo se corresponde con el pequeño teorema de Fermat.

Con esto, podemos definir el sistema RSA como sigue:

Definición 10 (RSA)**.** Consideremos $n = pq$, con p y q primos. Sean $\mathcal{P} = \mathcal{C} = \mathbb{Z}/n\mathbb{Z}$ y $e \in \mathbb{Z}$ tal que e es coprimo con $\varphi(n) = (p-1)(q-1)$ y $e > 1$.

Sea $d \in \mathbb{Z}$ tal que $d\,e \equiv 1 \pmod{\varphi(n)}$, es decir, $d = e^{-1} \pmod{\varphi(n)}$.

Tomando e como clave pública y d como clave privada de una entidad, sus funciones de encriptación y desencriptación con RSA son:

$$E_e(x) = x^e \pmod{n}$$
$$D_d(c) = c^d \pmod{n},$$

donde $x \in \mathcal{P}$ y $c \in \mathcal{C}$.

Proposición 5. *RSA es un sistema criptográfico bien definido, esto es, se cumple que $D_d(E_e(x)) = x$ para todo $p \in \mathcal{P}$.*

Demostración. Sea $n = pq$, con p y q primos. Sean $d, e \in \mathbb{Z}$ tales que $de \equiv 1 \mod \varphi(n)$. Equivalentemente, $de = t\varphi(n) + 1$ para un oportuno $t \in \mathbb{Z}$.

Caso 1. Consideremos un texto plano $x \in (\mathbb{Z}/n\mathbb{Z})^\times$, esto es, x coprimo con n. Entonces,

$$D_d(E_e(x)) \equiv (x^e)^d = x^{t\varphi(n)+1} = (x^{\varphi(n)})^t\, x \equiv 1^t\, x = x \pmod{n},$$

donde hemos usado que, por ser x y n coprimos, $x^{\varphi(n)} \equiv 1 \pmod{n}$, por el teorema de Euler.

Caso 2. Supongamos ahora que $x \notin (\mathbb{Z}/n\mathbb{Z})^\times$. Por tanto, $x \equiv 0 \pmod{p}$ o $x \equiv 0 \pmod{q}$.

Sin pérdida de generalidad, supongamos que $x \equiv 0 \pmod{p}$. Entonces, $(x^e)^d \equiv 0 \equiv x \pmod{p}$.

Si, además, $x \equiv 0 \pmod{q}$, basta aplicar el teorema chino de los restos.

Si $x \not\equiv 0 \pmod{q}$, por el pequeño teorema de Fermat tenemos que

$$(x^e)^d = x^{ed-1} = x^{t(p-1)(q-1)} = (x^{q-1})^{t(p-1)}\, x \equiv 1^{t(p-1)}\, x = x \pmod{q},$$

donde en la segunda igualdad hemos usado que $ed = t\varphi(n)+1 = t(p-1)(q-1)+1$. Para finalizar, basta aplicar el teorema chino de los restos. □

Observación 9. En realidad, la clave pública de RSA es el par (n, e), ya que para encriptar también hace falta conocer el módulo n. Puesto que también es necesario conocerlo para desencriptar, se dice que la clave privada es (p, q, d) o (n, d) (son opciones son iguales en la práctica, pero la primera refuerza que p y q deben permanecer en secreto, mientras que la segunda hace patente que se puede prescindir de p y q una vez se ha calculado n). Hasta ahora habíamos tomado solo e y d con el propósito de simplificar la notación y aligerar la explicación del funcionamiento del sistema.

Las claves de RSA se pueden generar siguiendo el algoritmo 1.

Algoritmo 1 Generación de claves en RSA.

1: Escoger dos primos p y q grandes y distintos.
2: Calcular $n = pq$ y $\varphi(n) = (p-1)(q-1)$.
3: Escoger aleatoriamente $e \in \mathbb{Z}$ tal que $1 < e < \varphi(n)$ y $\gcd(e, \varphi(n)) = 1$.
4: Calcular $d = e^{-1} \pmod{\varphi(n)}$ usando el algoritmo de Euclides extendido.
5: Tomar (n, e) como clave pública y (n, d) como clave privada.

El siguiente ejemplo muestra cómo se puede emplear el algoritmo de Euclides extendido para calcular d.

Ejemplo 9.

1. Sean $p = 3$ y $q = 11$ (que, en aplicaciones reales, resultarían excesivamente pequeños, pero que nos sirven a efectos ilustrativos).

2. Tenemos entonces que $n = 33$ y que $\varphi(n) = 20$.

3. Sea $e = 7$.

4. Planteemos la ecuación $20x + 7y = 1$. Observamos que $20x + 7y = 1 \pmod{20}$ es equivalente a $7y = 1 \pmod{20}$, de modo que y es el elemento inverso de e buscado.

   La ecuación $20x + 7y = 1$ tiene solución por la identidad de Bézout, y puesto que

   $$20 = 7 \cdot 2 + 6$$
   $$7 = 6 \cdot 1 + 1,$$

   llegamos a que $1 = 7 - 6 \cdot 1 = 7 - (20 - 2 \cdot 7) \cdot 1 = 3 \cdot 7 - 1 \cdot 20$. Por tanto, $d = y = 3$.

5. Tomamos la clave pública $(33, 7)$ y la clave privada $(33, 3)$.

Se pueden distinguir tres enfoques para atacar los principios matemáticos en los que se fundamenta RSA:

(i) Factorizar n en sus factores primos p y q, lo que permitiría calcular $\varphi(n) = (p-1)(q-1)$ y, una vez conocido este, calcular $d = e^{-1} \pmod{\varphi(n)}$.

(ii) Determinar $\phi(n)$ conociendo tan solo n y e, sin calcular p y q.

(iii) Determinar d directamente, sin calcular $\phi(n)$.

Los análisis de la seguridad de RSA han tendido a centrarse en el enfoque (i), ya que (i) y (ii) son problemas equivalentes. Además, determinar d a partir de n y e parece ser un problema tan difícil como (i) (de hecho, se ha probado

que esto es cierto, salvo por una diferencia de tiempo polinomial, si se asume la hipótesis extendida de Riemann).

En la actualidad, el mejor algoritmo conocido para factorizar enteros es la *criba general del cuerpo de números* (GNFS, del inglés *general number field sieve*), que es subexponencial. No se han encontrado algoritmos polinomiales para este problema en ordenadores clásicos, pero sí se ha desarrollado uno en ordenadores cuánticos: el algoritmo de Shor, descubierto por Peter Shor en 1994. No obstante, los ordenadores cuánticos actuales no son suficientemente potentes como para romper RSA en la práctica y, aunque esta es una preocupación de cara a su uso futuro, hoy en día RSA sigue siendo uno de los sistemas criptográficos más utilizados.

5. Criptografía de curva elíptica

La criptografía de curva elíptica (ECC, por el inglés *elliptic curve cryptography*) es un enfoque de criptografía asimétrica basado en la estructura de grupo de las curvas elípticas. En los últimos años ha adquirido una creciente popularidad, debido a que ninguno de los ataques conocidos contra ella es de tiempo subexponencial. Esto permite alcanzar un nivel de seguridad equivalente al de sistemas como RSA con claves de tamaño significativamente menor, como se puede ver en el Cuadro 4, lo que a su vez proporciona importantes mejoras en velocidad de computación y uso de memoria, ancho de banda y energía.

Las curvas elípticas son curvas proyectivas, no afines, por lo que una descripción general de las mismas requiere introducir conceptos relacionados con los espacios proyectivos. Para evitar un formalismo excesivo, obviaremos este tipo de detalles, de los que damos una visión simplificada.

Cuadro 4: Comparación entre el número de bits requeridos en la clave de algoritmos simétricos, en las claves públicas (L) y privadas (N) de Diffie-Hellman y DSA (*Digital Signature Algorithm*, algoritmo de firma digital), en el módulo de RSA y en el orden n de ECC para obtener un nivel de seguridad equivalente. Se indica también el factor entre los tamaños de RSA y ECC.

Algoritmos de clave simétrica	**Diffie-Hellman, DSA**	**RSA**	**ECC**	**Factor RSA/ECC**
80	L = 1024, N = 160	1024	160	6.4
112	L = 2048, N = 224	2048	224	$64/7 \approx 9.143$
128	L = 3072, N = 256	3072	256	12
196	L = 7680, N = 384	7680	384	20
256	L = 15360, N = 512	15360	512	30

5.1. Curvas elípticas

Definición 11. Sea K un cuerpo con char $K \neq 2, 3$. Una curva elíptica E sobre K está definida por una ecuación de la forma

$$y^2 = x^3 + ax + b, \tag{2}$$

con $a, b \in K$ tales que $4a^3 + 27b^2 \neq 0$. E contiene los puntos $(x, y) \in K^2$ que cumplen esta ecuación y un punto extra, denotado por O y denominado *punto del infinito*. Por tanto, E es el conjunto

$$E = \{(x, y) \in K^2 \mid y^2 = x^3 + ax + b\} \cup \{O\}.$$

Observación 10.

- E es simétrica con respecto a $y = 0$, como se puede ver en la Figuras 8. El simétrico de O es el propio O.
- La condición $4a^3 + 27b^2 \neq 0$ garantiza que E no tiene *puntos singulares*. Es equivalente a que el polinomio $x^3 + ax + b$ no tenga raíces repetidas. De existir algún punto singular, como es el caso de la curva representada en la Figura 9, no podemos hablar de curva elíptica.

Las curvas elípticas tienen la peculiaridad de que poseen una estructura de grupo abeliano con la siguiente operación:

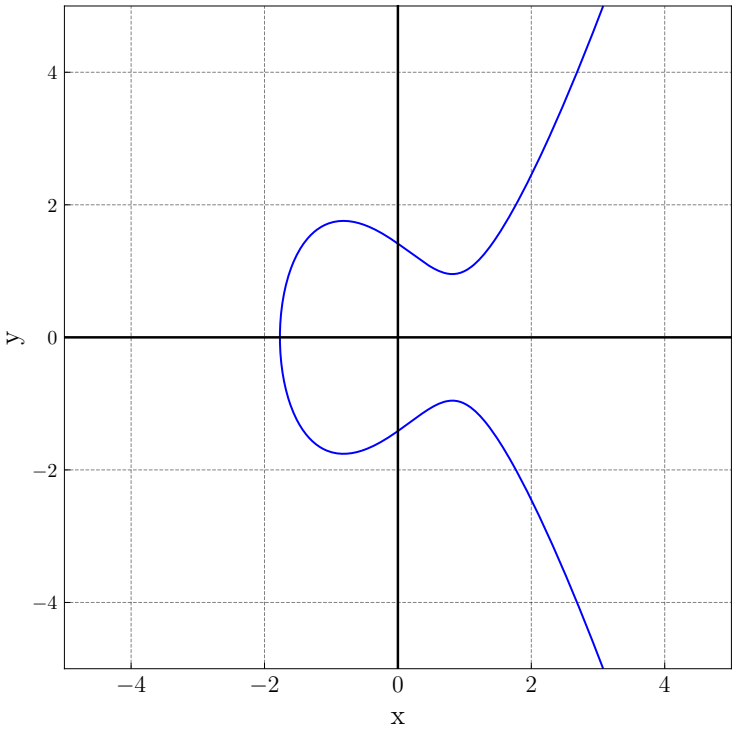

Figura 8: Curva elíptica definida sobre $\mathbb{R}$ y dada por $y^2 = x^3 - 2x + 2$, un caso particular de la forma (2).

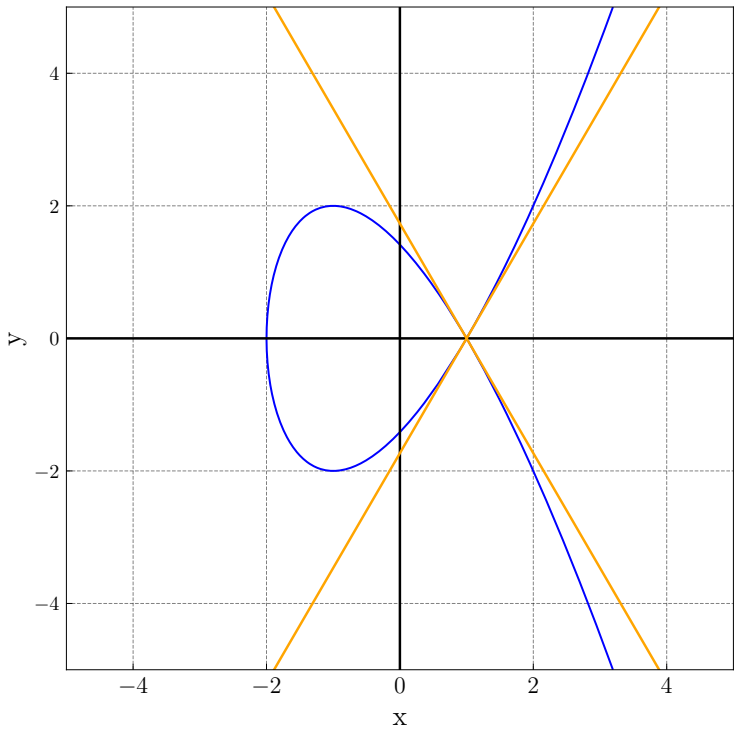

Figura 9: En azul, la curva definida sobre $\mathbb{R}$ dada por la ecuación $y^2 = x^3 - 3x + 2$. No es una curva elíptica, por tener un punto singular, el $(1, 0)$. En naranja se muestran las rectas tangentes a dicho punto.

Definición 12. Sea E una curva elíptica sobre K.

Sean $P, Q \in E$. Sea L la recta que pasa por P y Q si $P \neq Q$, o la recta tangente a P si $P \neq Q$. Denotamos por $P * Q$ el tercer punto de intersección de L y E.

Definimos como $P + Q$ el punto simétrico de $P * Q$ con respecto a la recta $y = 0$.

La construcción de esta operación está representada en la Figura 10.

Observación 11. La existencia del punto del infinito O (más precisamente, el carácter proyectivo de las curvas elípticas), nos permite garantizar que una curva elíptica y una recta tienen siempre tres puntos de intersección, contados con multiplicidad. Cualquier recta vertical pasa por O.

Teorema 4. *$(E, +)$ tiene estructura de grupo abeliano, donde O es el elemento neutro y donde el opuesto de un punto $P = (x, y) \in E$ es su simétrico $-P = (x, -y)$.*

El algoritmo 2 recoge fórmulas para calcular las coordenadas de $P + Q$ en función de las coordenadas de P y Q.

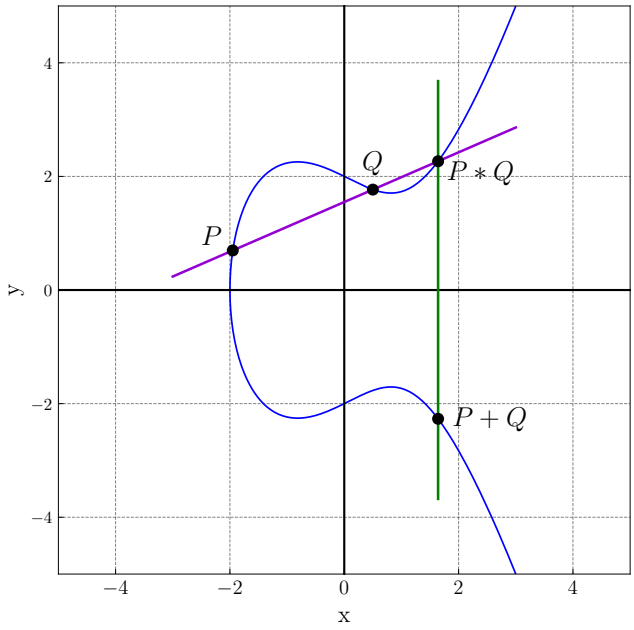

Figura 10: Suma de dos puntos sobre la curva elíptica dada por la ecuación $y^2 = x^3 - 2x + 4$, tomando $O = (0 : 1 : 0)$.

Algoritmo 2 Suma de puntos de una curva elíptica.

Entrada: Una curva elíptica $E : y^2 = x^3 + ax + b$ sobre un cuerpo K y puntos $P, Q \in E$.
Salida: $P + Q$.

if $P = O$ o $Q = O$ **then**
  return $P + Q = O$.
else
  Escribimos $P = (p_1, q_1)$, $Q = (q_1, q_2)$.
  if $p_1 = q_1$ y $p_2 \neq q_2$ **then**
    return $P + Q = O$.
  else if $P = Q$ y $p_2 = 0$ **then**
    return $P + Q = O$.
  else if $P = Q$ y $p_2 \neq 0$ **then**
    $m \leftarrow \frac{3p_1^2 + a}{2p_2}$.
  else
    $m \leftarrow \frac{q_2 - p_2}{q_1 - p_1}$.
  end if
  $r_1 \leftarrow m^2 - p_1 - p_2$.
  $r_2 \leftarrow m(p_1 - r_1) - p_2$.
  return $P + Q = (r_1, r_2)$.
end if

5.2. El problema del logaritmo discreto en curvas elípticas

De manera análoga a cómo RSA se fundamenta en el problema de la factorización de enteros, la ECC se basa en el problema del logaritmo discreto en curvas elípticas, que es un caso particular del problema del logaritmo discreto.

Definición 13. Sea $G = \langle g \rangle$ un grupo cíclico finito de orden n. El *problema del logaritmo discreto* (DLP, *discrete logarithm problem*) en G consiste en, dado $y \in G$, determinar $x \in \{0, 1, \ldots, n-1\}$ tal que $y = g^x$.

Con la notación de la definición anterior, un sistema criptográfico basado en el problema del logaritmo discreto sobre un grupo G debe ser tal que encontrar x sea computacionalmente inviable, pero que calcular la operación del grupo sea sencillo. La elección de G es importante: por ejemplo, en el grupo aditivo $\mathbb{Z}/p\mathbb{Z}$, con p primo, se puede utilizar el algoritmo de Euclides para hallar el valor de m en la ecuación lineal $b = ma \bmod p$, para $a, b \in \mathbb{Z}/p\mathbb{Z}$ dados. Dicho algoritmo requiere a lo sumo $2 \log n$ pasos, por lo que $\mathbb{Z}/p\mathbb{Z}$ no da lugar a un problema suficientemente robusto.

El caso en el que G es un subgrupo del grupo multiplicativo de un cuerpo finito, $G < \mathbb{F}_p^\times$ (que necesariamente implica que G es cíclico), es la versión utilizada en algunos algoritmos de criptografía asimétrica, como Diffie-Hellman, ElGamal y DSA.

En el caso de que G sea un subgrupo cíclico del grupo de puntos de una curva elíptica, hablamos del *problema del logaritmo discreto en curvas elípticas*.

Definición 14. Sea E una curva elíptica definida sobre un cuerpo finito $\mathbb{F}_q$, con $q = p^k$ para un cierto p primo y un entero $k \geq 1$. Consideremos un punto $P \in E$ de orden n, y un punto $Q \in \langle P \rangle$. El *problema del logaritmo discreto en curvas elípticas* (ECDLP, *elliptic curve discrete logarithm problem*) consiste en determinar $l \in \{0, 1, \ldots, n-1\}$ tal que

$$Q = lP = \underbrace{P + \cdots + P}_{l \text{ veces}}$$

l se denomina *logaritmo discreto de Q en base P*.

En la criptografía de curva elíptica, Q es la clave pública y l, la clave privada. Los elementos de la tupla (q, a, b, P, n) reciben el nombre de *parámetros de dominio*.

Los parámetros de dominio deben cumplir ciertas condiciones para asegurar que no se dan una serie de casos particulares para los que sí hay algoritmos subexponenciales. Por la dificultad de comprobar estas condiciones, es común emplear parámetros de dominio publicados por autoridades reconocidas como el Instituto Nacional de Normas y Tecnología (NIST, *National Institute of*

Standards and Technology), el Grupo de Estándares para la Criptografía Eficiente (SECG, *Standards for Efficient Cryptography Group*) o *ECC Brainpool*.

Numerosos algoritmos de criptografía asimétrica clásica han sido adaptados a la ECC, como Diffie-Hellman, ElGamal o DSA. A continuación, vemos la adaptación de Diffie-Hellman.

5.3. Elliptic-curve Diffie-Hellman (ECDH)

Supongamos que dos entidades, Alice y Bob, desean establecer una clave secreta compartida. Tras fijar los parámetros de dominio, Alice elige un entero $n_A \in \{1, \ldots, n-1\}$, que será su clave privada, y calcula $P_A = n_A P \in E$, su clave pública. B hace lo análogo, con un entero $n_B \in \{1, \ldots, n-1\}$ y $P_B = n_B P \in E$ sus claves privada y pública, respectivamente.

Finalmente, A envía P_A a B, y B envía P_B a A. Ambos calculan la clave secreta compartida $K = n_A P_B = n_B P_A = n_A n_B P \in E$.

La seguridad de este proceso, que se recoge en el algoritmo 3, recae en la dificultad para un atacante de obtener K, para lo cual tendría que calcular n_A o n_B a partir de P_A o P_B, que es un caso particular del ECDLP.

Algoritmo 3 Intercambio de claves con Elliptic Curve Diffie-Hellman (ECDH)

Entrada: Un punto base P de orden n en una curva elíptica E sobre un cuerpo finito $\mathbb{F}_q$, con $q = p^k$ para un primo p y un entero $k \geq 1$.
Salida: Clave secreta compartida K.

1: Alice elige su clave privada $n_A \in \{1, 2, \ldots, n-1\}$ y Bob elige su clave privada $n_B \in \{1, 2, \ldots, n-1\}$.
2: Alice calcula su clave pública $P_A = n_A P$ y Bob calcula su clave pública $P_B = n_B P$.
3: Alice envía P_A a Bob y Bob envía P_B a Alice.
4: Alice calcula $K = n_A P_B = n_A n_B P$ y Bob calcula $K = n_B P_A = n_B n_A P$.
5: **Resultado:** Alice y Bob comparten la clave secreta $K = n_A n_B P$.

La clave secreta K se puede utilizar para generar una clave de cifrado simétrico para algoritmos como AES. En tal caso, es común tomar una de las coordenadas de K o una función sencilla de K.

Tanto ECDH como el algoritmo clásico de Diffie-Hellman son vulnerables a *ataques de intermediario* (*man-in-the-middle*). No obstante, la adición de mecanismos para la autenticación de las claves de ambas partes permite evitar este problema.

6. Firmas digitales

Las firmas digitales son primitivas criptográficas que permiten la integridad, la autenticación del origen de los datos y el no repudio. De forma análoga a las firmas escritas, su principal propósito es ligar una entidad a un conjunto de datos de forma que la existencia de una relación entre ambos pueda ser verificada por una entidad externa e independiente. La entidad que es ligada a los datos recibe el nombre de *firmante* o *signatario*. La Figura 11 muestra un modelo básico de uso de firmas digitales.

Definición 15. Un *esquema de firma digital* es una 5-tupla $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ tal que:

(i) $\mathcal{P}$ es un conjunto finito de posibles *mensajes*.

(ii) $\mathcal{A}$ es un conjunto finito de posibles *firmas*.

(iii) $\mathcal{K}$ es un conjunto finito de posibles *claves*.

(iv) $\mathcal{S} = \{\text{sig}_k : k \in \mathcal{K}\}$ es un conjunto de algoritmos de firmado polinomiales $\text{sig}_k : \mathcal{P} \longrightarrow \mathcal{A}$.

(v) $\mathcal{V} = \{\text{ver}_k : k \in \mathcal{K}\}$ es un conjunto de algoritmos de verificación polinomiales $\text{ver}_k : \mathcal{P} \times \mathcal{A} \longrightarrow \{\text{true},\ \text{false}\}$.

(vi) Para cualesquiera $k \in \mathcal{K}$, $x \in \mathcal{P}$ e $y \in \mathcal{A}$ se tiene que

$$\text{ver}_k(x, y) = \begin{cases} \text{true}, & \text{si } y = \text{sig}_k(x) \\ \text{false}, & \text{en otro caso.} \end{cases}$$

Dada una clave $k \in \mathcal{K}$ y un mensaje $x \in \mathcal{P}$, decimos que un par $(x, \text{sig}_k(x))$ es un *mensaje firmado*.

Observación 12. Puede haber más de un $y \in \mathcal{A}$ tal que $\text{ver}_k(x, y) = \text{true}$, dependiendo de la definición de ver_k.

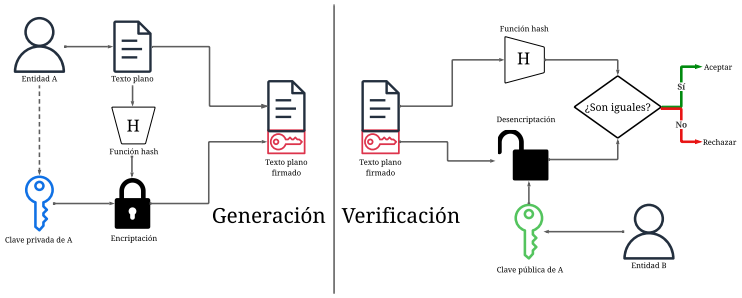

Figura 11: Modelo para la generación y verificación de firmas digitales.

En un sistema de firma digital, dada una clave $k \in \mathcal{K}$, el algoritmo de firmado sig_k es privado, mientras que el algoritmo de verificación ver_k es público.

Para que un sistema de firma digital sea seguro, dado un mensaje $x \in \mathcal{P}$, requerimos que sea computacionalmente inviable para cualquier entidad distinta de Alice generar una firma $y \in \mathcal{A}$ tal que $\text{ver}_k(x, y) = \text{true}$.

Observación 13. Aunque los sistemas de firmas digitales son sistemas de criptografía asimétrica, es importante tener en cuenta sus diferencias con los sistemas de encriptación asimétricos. En estos últimos, la encriptación usa la clave pública, mientras que la desencriptación usa la clave privada. En los sistemas de firmas digitales, el algoritmo de firmado es privado, mientras que la verificación es pública. Se puede ver una comparativa en el Cuadro 5.

Cuadro 5: Comparación entre los esquemas de firmas digitales y los esquemas de encriptación asimétrica.

Esquemas de firmas digitales	**Esquemas de encriptación asimétrica**
Solo el poseedor del secreto puede generar una firma	"Cualquiera" puede encriptar datos
"Cualquiera" puede verificar que una firma es válida	Solo el poseedor del secreto puede desencriptar datos encriptados

6.1. DSA

DSA (*digital signature algorithm*, algoritmo de firma digital) es un estándar del gobierno de EE.UU. para firmas digitales. Fue propuesto por el NIST en 1991, y se convirtió en el primer esquema de firmas digitales reconocido oficialmente por un gobierno. Su seguridad está basada en el problema del logaritmo discreto.

DSA utiliza *funciones hash*, cuyo propósito es transformar datos de entrada en una salida corta que sirva a modo de identificador de esos datos, y que tenga una baja probabilidad de ser la salida también para datos de entrada distintos.

Definición 16. Una *función hash* H es una aplicación que lleva una cadena de texto de longitud arbitraria en una cadena de texto de longitud fija. Su nivel de seguridad depende de su cumplimiento de las siguientes tres propiedades:

1. *Resistencia a la preimagen*: dada una imagen h de H, es computacionalmente inviable encontrar una preimagen x tal que $H(x) = h$.

2. *Segunda resistencia a la preimagen*: dada una cadena de entrada x, es computacionalmente inviable encontrar otra cadena de entrada x', con $x \neq x'$, tal que $H(x) = H(x')$.

3. *Resistencia a la colisión*: es computacionalmente inviable encontrar cadenas de entrada x y x', con $x \neq x'$, tales que $H(x) = H(x')$.

La propiedad de resistencia a la colisión implica la segunda resistencia a la preimagen, pero no la resistencia a la preimagen. Una función hash que no satisface alguna de estas tres propiedades es vulnerable a ataques maliciosos.

En lo que sigue, H denotará una función hash con a lo sumo n bits de salida (si la salida fuese mayor, basta truncarla a n bits). Cabe destacar que si H no cumple las tres propiedades de seguridad de las funciones hash, DSA es vulnerable a que un atacante que consiga firmas de mensajes generadas por una entidad A pueda forjar firmas válidas para nuevos mensajes en nombre de A, por lo que es recomendable usar algoritmos estándar como SHA-2 o SHA-3.

Los pasos de los procesos de generación de claves, generación de firmas y verificación de firmas se indican en los algoritmos 4, 5 y 6, respectivamente.

Algoritmo 4 Generación de claves con ECDSA para una entidad A.

Salida: Una clave privada y una clave pública para una entidad A.

1: A elige un número primo p de L bits (es decir, tal que $2^{L-1} < p < 2^L$) para un L múltiplo de 64 y tal que $512 \leq L \leq 1024$.
2: A elige un número primo q de N bits tal que q divide a $p-1$.
3: A elige un entero h de forma aleatoria del conjunto $\{2, \ldots, p-2\}$.
4: A calcula $g \leftarrow h^{(p-1)/q} \pmod{p}$.
5: **if** $g = 1$ **then**
6: Se vuelve al paso 3 y se elige otro h.
7: **end if**
8: A elige un entero x de forma aleatoria del conjunto $\{1, \ldots, q-1\}$.
9: A calcula $y \leftarrow g^x \pmod{p}$.
10: La clave privada de A será x. Su clave pública será y. Los parámetros de dominio del algoritmo serán (p, q, g), que se comparten con las entidades que emplearán este sistema.

Demostración de que la verificación de firmas con DSA es correcta. Queremos ver que si una entidad A genera una firma (r, s) con el algoritmo 5, entonces el algoritmo 6, ejecutado por otra entidad B, devuelve el resultado correcto.

Puesto que $g = h^{(p-1)/q}$ y dado que h es coprimo con p, por el pequeño teorema de Fermat tenemos que $g^q \equiv h^{p-1} \equiv 1 \pmod{p}$. Ahora bien, como $g > 0$ y q es primo, g debe tener orden q.

De $s = k^{-1}(e + xr) \pmod{q}$ obtenemos que $k = es^{-1} + xrs^{-1} \equiv ew + xrw \pmod{q}$.

Como g tiene orden q, sigue que

$$g^k \equiv g^{ew} g^{xrw} \equiv g^{ew} y^{rw} \equiv g^{u_1} y^{u_2} \pmod{p}.$$

Y concluimos que

$$r = (g^k \bmod p) \bmod q = (g^{u_1} y^{u_2} \bmod p) \bmod q = v.$$

Algoritmo 5 Algoritmo de generación de firmas de DSA

Entrada: La clave privada x de una entidad A, los parámetros de dominio del sistema, (p, q, g), y un mensaje m.
Salida: Una firma (r, s) para el mensaje m.

1: Se elige un entero $k \in \{1, \dots q-1\}$ aleatoriamente.
2: Se calcula $r \leftarrow (g^k \bmod p) \bmod q$.
3: **if** $r = 0$ **then**
4: Se vuelve al paso 1 y se elige otro k.
5: **end if**
6: Se calcula $e \leftarrow H(m)$.
7: Se calcula $s \leftarrow k^{-1}(e + xr) \pmod q$.
8: **if** $s = 0$ **then**
9: Se vuelve al paso 1 y se elige otro k.
10: **end if**
11: **return** (r, s).

Algoritmo 6 Algoritmo de verificación de firmas de DSA

Entrada: Un mensaje m, una firma (r, s) de m, la clave pública y de la entidad que ha generado (r, s).
Salida: La aceptación o el rechazo de la firma.

1: Si $r \notin \{1, \dots, q-1\}$ o $s \notin \{1, \dots, n-1\}$, la firma se rechaza.
2: Se calcula $e = H(m)$.
3: Se calcula $w \leftarrow s^{-1} \pmod q$.
4: Se calcula $u_1 \leftarrow ew \pmod q$ y $u_2 = rw \pmod q$.
5: Se calcula $v \leftarrow (g^{u_1} y^{u_2} \bmod p) \bmod q$.
6: Si $v = r$, la firma se acepta. En otro caso, se rechaza.

□

Las comprobaciones de que r y s estén en el intervalo $[1, n-1]$ en la verificación permiten evitar ataques que focalizan este punto.

En un futuro próximo, se espera que DSA sea reemplazado por su variante en curvas elípticas, ECDSA, que mantiene una estructura muy similar pero tiene una mayor seguridad, gracias a la mayor dificultad del ECDLP.

Referencias

[1] Keith, M. (2017). *Everyday Cryptography: Fundamental Principles and Applications*, 2nd ed., Oxford University Press.

[2] Menezes, A. J., Vanstone, S. A. y Van Oorschot, P. C. (1996). *Handbook of Applied Cryptography*, 1st ed., CRC Press, Inc.

[3] Abikoye, O. C., Haruna, A. D., Abubakar, A., Akande, N. O., y Asani, E. O. (2019). Modified Advanced Encryption Standard Algorithm for Information Security. *Symmetry*, 11(12), 1484.

[4] Arzhantseva, G. y Finn-Sell, M. (2019). *Topics in Algebra: Cryptography WS2019* [Notas], Universidad de Viena. Disponible en https://www.mat.univie.ac.at/~gagt/crypto2019/. Consultado por última vez el 31 de julio de 2025.

[5] Stallings, W. (2016). *Cryptography and Network Security: Principles and Practice*, 7nd ed., Pearson, Esssex.

[6] Carrera, X. (2025). *Curvas elípticas y aplicaciones en criptografía* [Trabajo de fin de grado]. Universidad de Santiago de Compostela.

[7] Silverman, J. H. (2009). *The arithmetic of elliptic curves*, 2nd ed., Graduate Texts in Mathematics, 106, Springer-Verlag, New York.