Manual de Criptografía

Fundamentos matemáticos de la Criptografía para un estudiante de Grado

FRANCISCO JOSÉ PLAZA MARTÍN

Ediciones Universidad Salamanca

MANUAL DE CRIPTOGRAFÍA

Fundamentos matemáticos de la Criptografía para un estudiante de Grado

FRANCISCO JOSÉ PLAZA MARTÍN

MANUAL DE CRIPTOGRAFÍA

Fundamentos matemáticos de la Criptografía para un estudiante de Grado

Ediciones Universidad Salamanca

Documentos Didácticos
169

Ediciones Universidad de Salamanca
y Francisco José Plaza Martín

Motivo de cubierta: "Detalle del arca conservada en la sala de manuscritos
de la Biblioteca General Histórica. Universidad de Salamanca"

1.ª edición: enero, 2021
ISBN: 978-84-1311-463-7 (PDF)
ISBN: 978-84-1311-464-4 (POD)

DOI: https://doi.org/10.14201/0DD0169

Ediciones Universidad de Salamanca
Plaza San Benito s/n
E-37002 Salamanca (España)
http://www.eusal.es
eus@usal.es

Realizado por:
Nueva Graficesa S.L.
Teléfono: 923 26 01 11
Salamanca (España)

Hecho en UE-Made in EU

Ediciones Universidad de Salamanca es miembro de la UNE
Unión de Editoriales Universitarias Españolas
www.une.es

♠

Catalogación de editor en ONIX accesible en https://www.dilve.es/ CEP

Dedicado mis padres, Francisco y Ángela,
y a mis profesores, José María y Ángel.

Índice general

I. Presentación

El libro que tienes en tus manos pretende ser un manual para un curso universitario de introducción a la criptografía desde una perspectiva matemática. Está orientado a proporcionar el conocimiento básico para entender los fundamentos matemáticos de las técnicas criptográficas modernas tanto a estudiantes de matemáticas y física como de informática e ingeniería.

En primer lugar, y para situar al lector en el panorama que juega la criptografía en la sociedad actual, enumeraremos algunas situaciones cotidianas en las que la criptografía aparece en nuestras vidas. Así, recordamos la comunicación cifrada de WhatsApp, los códigos CVV y CV2 de las tarjetas de crédito para dificultar su uso fraudulento, el cifrado PGP de los correos electrónicos, los protocolos de navegación `https://`, los criptosistemas en los mandos de apertura de coches o en la televisión de pago, el comercio electrónico (p. ej. la pasarelas de pago Redsys) o cómo los nuevos discos duros admiten cifrar la información guardada (p. ej. con AES). También tenemos noticia de otros usos, aunque solamente los conozcamos por las películas y noticias de espionaje, que engloban la esteganografía (el caso de la espía rusa que escondía la información en sus fotografías de Facebook), las comunicaciones entre servicios secretos (la máquina Enigma es quizás la más famosa de todas) o los casos de Wikileaks o E. Snowden.

No resulta ahora extraño afirmar que la criptografía moderna es una combinación de diferentes temas de diversas especialidades. Hagamos un breve repaso a las disciplinas involucradas para obtener una percepción de las íntimas conexiones entre ellas. Para empezar, en la parte más teórica de la criptografía podemos citar la teoría de números (desde la aritmética modular hasta la teoría de símbolos pasando por el estudio de la distribución de los números primos), la probabilidad (para la generación pseudoaleatoria

de números). En su parte más práctica, que cae de lleno en la informática y en la ingeniería, tenemos desde el diseño de dispositivos específicos (microprocesadores con instrucciones criptográficas nativas), protección física de la comunicación (sistemas de apantallamiento que dificultan la intercepción), o protocolos para implementar sistemas criptográficos (p. ej. `https://`, `sftp://`,...). A caballo entre las dos partes anteriores se sitúa la teoría de la computabilidad así como la algoritmia (p. ej. máquinas de Turing, algoritmos, funciones resumen o hash). Finalmente, no podemos obviar los aspectos legales relacionados con las comunicaciones y con la privacidad.

A lo hora de abordar el estudio de la criptografía podemos elegir centrarnos en alguna de las materias sobre las que se apoya (matemáticas, informática, ingeniería,...) o bien en el aprendizaje de casos de uso, por ejemplo, cifrado de comunicaciones, aplicaciones al comercio electrónico, aspectos legales,... Todos estos enfoques son necesarios y se complementan en esta disciplina interdisciplinar.

Una vez descrito este panorama general, podemos entrar en una descripción más específica de este manual. Su hilo conductor será la matemática (principalmente, el álgebra) que constituye los cimientos sobre los que se levanta la criptografía. Este texto orientado a un estudiante universitario de grados en matemáticas, física o informática con unos conocimientos básicos de álgebra (aritmética modular en los números enteros, aspectos básicos de teoría de grupos y de álgebra lineal) que desea conocer la matemática sobre la que se apoya la criptografía moderna. No obstante, se ha buscado un equilibrio entre la parte más abstracta y la más práctica, tendiendo puentes entre ambas. Por un lado se centra en la parte más teórica, priorizando la parte algebraica y postponiendo la parte de probabilidad o computabilidad para otros cursos. Por otro, la parte anterior se complementa con problemas y, además, se describen explícitamente muchos de los algoritmos usados actualmente, dejando al lector más inquieto la opción de programarlos por si mismo.

En cuanto a la presentación y estructura, cada capítulo comienza con las referencias en las que se basan los contenidos, ya que este libro es esencialmente un manual para clase y no un libro para expertos en la materia. Aunque se incluyen resultados teóricos enunciados como **Definiciones** y **Teoremas**, también está presente el lado más práctico e incluimos **Problemas** y **Soluciones** así como Algoritmos (enmarcardos en unas cajas de texto). Cada capítulo concluye con unos Temas para ampliar, en los que se exhiben las conexiones de los contenidos con otras aplicaciones o disciplinas. Finalmente, esperamos que los apéndices (Aritmética Modular, Bibliografía e Índice Temático) faciliten al lector el uso de este libro.

Quiero concluir esta presentación con un emotivo recuerdo a José María Muñoz Porras, quien no solo me introdujo en esta disciplina, sino que me formó como matemático.

Salamanca, diciembre de 2020.

II. Introducción

II.1 ¿Qué es la Criptografía?

La criptografía moderna es una materia intrínsecamente interdisciplinar, fruto de la cooperación de diversas disciplinas. Estas especialidades abarcan desde la teoría de números y la probabilidad hasta el diseño de *hardware* pasado por la teoría de la computabilidad y la algoritmia. Los usos actuales de la criptografía han sobrepasado ampliamente la concepción más tradicional, *el cifrado de las comunicaciones*, y posibilitan una panoplia de aplicaciones como son las compras *online*, las subastas por internet, las votaciones electrónicas, la firma digital, las sedes electrónicas de las instituciones, etc.

Desde el punto de vista matemático, la Criptografía es una rama de *Teoría de la Información* conjuntamente con la Teoría de Códigos y la Compresión de la información. Todas ellas tienen en común que estudian la transmisión de la información a través de un canal y que siempre tenemos dos agentes: un emisor y un receptor. Estos agentes se comunican enviando datos a través de un canal. No podemos actuar sobre el canal ni modificarlo, pero sí podemos conocer propiedades del canal. Describamos sucintamente estas ramas:

- Teoría de Códigos: estudia cómo preparar la información (*codificar*) para que, aunque sufra errores en la transmisión, el receptor pueda recuperar (*decodificar*) la información original. Matemáticamente necesita técnicas de espacios vectoriales sobre cuerpos finitos aunque los más sofisticados usan curvas algebraicas, teoría de módulos. Entre los códigos comúnmente usados están los siguientes: BCH, RS, Goppa,... y entre sus aplicaciones cotidianas cabe citar: TDT, CD, DVD, ADSL, fibra, WIFI, etc.

- Criptografía: se ocupa de estructurar la información (*encriptar*, *cifrar*) para que el receptor legítimo sea el único capaz de recuperar esa información (*desencriptar*, *descifrar*). Esto ha de ser así incluso con terceros que intercepten la transmisión. Matemáticamente necesita técnicas que van desde los grupos finitos hasta las curvas elípticas y espacios de Hilbert (en criptografía cuántica). Los siguientes sistemas resultarán conocidos al lector: RSA, DES, Triple DES, AES, Diffie-Helman,... y se aplican en los siguientes protocolos: https, firma digital, autenticación, votación electrónica, DNI electrónico, etc.
- Compresión: trata cómo enviar la información (*comprimir*) para que el receptor pueda recuperarla (*descomprimir*), quizás con alguna pérdida asumible, pero minimizando el volumen de datos que transitan por el canal. Matemáticamente se basa en espacios de Hilbert, wavelets,... y los usos son por todos conocidos: MP3, MP4, DivX, H265.

Lo cierto es que una transmisión en el mundo real requiere la combinación de los tres aspectos, en ese sentido, debemos destacar los hallazgos y planteamientos de C. Shannon. Pocas veces se puede decir que el nacimiento de un área, la Teoría de la Información, se debe a un único investigador.

En una primera aproximación conviene tratar separadamente cada rama, y centrarnos en la esencia de cada uno. Así, por ejemplo, mientras que en Teoría de Códigos no nos importa que alguien vea la información transmitida, en Criptografía confiaremos que el canal no ha introducido errores.

Una primera división de los métodos criptográficos es la siguiente:

- Criptografía simétrica o de clave secreta: emisor y receptor comparten una misma clave que sirve tanto para cifrar como para descifrar. Cabe destacar que cada pareja (emisor, receptor) necesitan una clave para su comunicación privada (y, por tanto, si se quiere un sistema criptográfico para n usuarios, se necesitarían $\binom{n}{2}$ claves en total, cada usuario debería memorizar $n-1$ claves). Además, se necesita un canal alternativo seguro para establecer por primera vez dicha clave (por ejemplo, la carta que envía el banco con el PIN de una tarjeta de crédito asume que el correo postal es un canal alternativo seguro). Hay situaciones en las que no se dispone de dicho canal, o bien, no es práctico (p. ej. para comprar online la entrada de cine de esta tarde, no puedo esperar a recibir una carta con mi clave). Estos sistemas criptográficos son los usados en tarjetas de crédito (PIN secreto) y otras comunicaciones. Computacionalmente suelen ser más sencillos y necesitan menos operaciones. Matemáticamente usa operaciones algebraicas sencillas (sumas y productos, aritmética modular) y su fortaleza radica en usar muchas iteraciones y tamaños de clave muy grandes. Desde el punto de vista histórico, la criptografía simétrica es la más antigua y se ha usado desde el cifrado de César hasta la máquina Enigma.
- Criptografía asimétrica o de clave pública: emisor y receptor tienen cada uno de ellos una clave con dos partes, una parte que es pública (clave pública y que es conocida por todo el mundo) y una parte que es privada (clave privada, que solamente conoce su dueño y que ha de ser mantenida en secreto). Son sistemas que no asumen la disponibilidad de un canal alternativo seguro y, en consecuencia, muy prácticos para comunicaciones online o con usuarios a los que no se conocían previamente. Si aparece un nuevo usuario del sistema de comunicación criptográfico, no es necesario dar nuevas claves a los usuarios

existentes, simplemente se asignan una clave pública y otra privada al nuevo usuario. Computacionalmente son complejos y requieren potencia de cálculo. Matemáticamente la criptografía asimétrica usa operaciones algebraicas complejas (exponenciales y logaritmos discretos, curvas elípticas) y su fortaleza se basa en problemas matemáticos computacionalmente complejos (p. ej. factorizar enteros, calcular logaritmos discretos). Son los sistemas más modernos por su buena adaptación a las necesidades actuales derivadas de la sociedad digital.

Las situaciones reales combinan ambos métodos del siguiente modo. Con los métodos de la criptografía asimétrica se crea un canal seguro que se utiliza para enviar una clave de un sistema de criptografía simétrica. A partir de ese momento, se utiliza la criptografía simétrica con dicha clave. De este modo, se necesita potencia de cálculo solamente en la primera fase de la comunicación, es decir, al usar criptografía asimétrica. En la segunda parte, al usar criptografía simétrica, se gana en velocidad de la comunicación y no se requiere de máquinas tan potentes.

II.2 Fortaleza de los Sistemas Criptográficos

El criptoanálisis estudia y desarrolla técnicas por las que un tercero podría acceder a información no destinada a él. A estas técnicas las llamamos comúnmente ataques. Los clasificaremos en:

- Ataques parciales: son las técnicas para desvelar **una** determinada comunicación.
- Ataque general: son las técnicas usadas para conocer las claves usadas y, por tanto, desvelar **todas** las comunicaciones.

Cada ataque es *una receta, un truco*, que explota una debilidad del sistema. Puede ser por fuerza bruta (p. ej. probando claves al azar), o pueden basarse en cierta información (p. ej. todos los mensajes comienzan igual o se firman igual, como el caso de la máquina Enigma), o porque conocemos unos pocos dígitos de la clave, etc. La mejor forma de entenderlo es pensar en las películas de espías. No es una ciencia exacta en el sentido que son ideas que a veces funcionan o a veces no, siendo necesario pasar sucesivamente de un tipo de ataque a otro pero que, en todo caso, compromete la seguridad del sistema.

La fortaleza de un sistema criptográfico, es decir, la resistencia a ataques (métodos por los que un tercero podría acceder a información no destinada a él) se basa en:

- diseñar sistemas cuyo ataque por fuerza bruta sea equivalente a resolver un problema matemáticamente muy complejo (p. ej. factorizar un número entero).
- mantener en secreto la mayor información posible (incluso el método usado, o el alfabeto etc) reduce automáticamente la posibilidad de éxito del posible ataque. Por ejemplo, se suele decir que el sistema RSA, diseñado y dado a conocer por Rivest, Shamir y Adleman en 1979, venía siendo utilizado años antes por la NSA. También se piensa que ha sucedido lo mismo con la criptografía basada en curvas elípticas.
- modificar el sistema en función de los ataques conocidos. Por ejemplo, el método de factorización de Fermat es efectivo cuando el número dado es producto de dos números de tamaño similar, por lo que para defenderse de un posible ataque basado en él, se toma un número con factores de tamaño distinto.

En todo caso, es muy importante destacar que todo lo anterior (complejidad de sistemas criptográficos, criptoanálisis, ataques, fortaleza, etc.) hacen referencia a la complejidad computacional de un determinado algoritmo.

Por ejemplo, el problema «encontrar el m.c.d. de dos números» puede ser:

- muy complejo, si el algoritmo consiste en factorizar ambos números y determinar los factores comunes;
- muy fácil, si el algoritmo consiste en aplicar el algoritmo de Euclides.

Como decíamos antes, la fortaleza de un sistema se basa en primer lugar en la complejidad de resolver un problema matemático; dicho con más precisión, se basa en un determinado problema matemático tal que todos los algoritmos conocidos para su resolución sean muy complejos. Es decir, que si se descubre un nuevo algoritmo, hay que reevaluar la seguridad del sistema.

Precisamente, hoy en día estamos en esta situación. La próxima irrupción de los ordenadores cuánticos hará posible implementar nuevos algoritmos para viejos problemas y alguno de ellos (p. ej. factorización) dejarán de ser problemas difíciles o complejos (p. ej. algoritmo de factorización de Shor). Por tanto, habría que estudiar si los sistemas actuales se pueden modificar para resistir este ataque o bien hay que diseñar nuevos sistemas criptográficos.

II.3 Esquema General de un Sistema Criptográfico

En el sentido amplio de Teoría de la Información (como se expuso anteriormente), un sistema criptográfico toma una información (llamada texto claro o texto plano) y la encripta mediante una función C de cifrado obteniendo un texto cifrado. Dicho texto es enviado por el emisor al receptor, que mediante la aplicación de una función de descifrado D recupera el texto claro.

Es habitual asumir los siguientes convenios para aplicar el proceso anterior:

- se fija un conjunto de símbolos, que denominamos alfabeto. Pueden ser un conjunto de números $0,1,\ldots,N-1$ como elementos de $\mathbb{Z}/N$, puede ser un conjunto de caracteres (A, B,..., Z, por simplificar 26 letras mayúsculas), toda la tabla ASCII, etc.
- se suele establecer una biyección del alfabeto usado para el texto claro (p. ej. caracteres del español) con un conjunto de números. De este modo, dada la información, se traduce símbolo a símbolo en números para poder hacer operaciones algebraicas y una vez hechas, se deshace el cambio.
- se tiene una aplicación de cifrado

$$C : \{\text{textos claros}\} \to \{\text{textos cifrados}\}$$

- se tiene una aplicación de descifrado

$$D : \mathrm{Im}(C) \to \{\text{textos claros}\}$$

- se cumple $D \circ C = \mathrm{Id}$.

II.4 La Actualidad de la Criptografía

La idea de la criptografía asimétrica (clave pública) fue presentada en Stanford en 1976 por Martin Hellman, Ralph Merkle y Whitfield Diffie y dio lugar poco después al sistema que se conoce actualmente como el criptosistema de las mochilas (véase §III.4.C). Desde entonces, y particularmente con la popularización de Internet, este tipo de criptografía se ha extendido ampliamente.

Actualmente nos encontramos protocolos criptográficos al navegar (p. ej. protocolos https, ssl, tls), al enviar correos (PGP, GPG), al guardar información en un disco duro (AES), sistemas de autenticación al conectarse a un router, etc. Otra serie de aplicaciones novedosas abarcan la firma digital, el no repudio, la integridad de documentos, las votaciones electrónicas, los juegos online, las plataformas de pago seguro, etc. Pero no debemos asumir que este tipo de criptografía es exclusivo de internet, al contrario, la criptografía invade ámbitos en los que habitualmente no reparamos, por ejemplo, los códigos CVV y CV2 de las tarjetas de crédito o los sistemas criptográficos empleados en la comunicación entre la llave de un vehículo y el propio vehículo.

Dos retos a los que se está enfrentando la criptografía, y lo seguirá haciendo en el futuro próximo, son la aparición del ordenador cuántico y la relación con la privacidad. Respecto del primero de ellos, es un hecho que la computación cuántica se podría usar para desvelar los mensajes cifrados con la criptografía asimétrica (p. ej. RSA). Frente a este panorama, ha surgido desde hace unos años un nuevo tipo de criptografía, la criptografía cuántica, que ya está preparada para esta nueva realidad (algoritmos de Schor, etc.). En cuanto al segundo reto, la relación con la privacidad, es cierto que empieza a ser común oír hablar del cifrado de extremo a extremo de Whatsapp u otras aplicaciones similares, si bien no es un protocolo abierto y auditable (por tanto, no es verificable).

II.5 Temas para ampliar

Una de las estrategias para ocultar la información es colocar esa información donde nadie se la espera o disimulada entre otros datos. Es la denominada *esteganografía*.

- `http://www.criptored.upm.es/crypt4you/temas/privacidad-proteccion/leccion7/leccion7.html`
- `http://www.jjtc.com/stegdoc/steg1995.html`

A todos nosotros nos han pedido alguna vez los códigos CVV y CV2 que se encuentran en el reverso de las tarjetas de crédito pero ¿qué valor, qué uso tienen estos números? Estos números los obtiene la empresa emisora de la tarjeta (Visa, 4B, etc.) a partir de los datos de la tarjeta (número de 16 dígitos, fecha de caducidad, nombre, etc.) mediante un algoritmo secreto. Así, cuando nos lo solicitan, lo que se quiere es comprobar que los datos de la tarjeta son válidos y no han sido falsificados.

- `http://www.darkcoding.net/credit-card/cvv-numbers/`
- `https://www.mejorestarjetasdecredito.es/codigo-de-seguridad-de-las-tarjetas-cvv/`

La criptografía aparece en los lugares más insospechados, por ejemplo, en los mandos de los coches. Así, para evitar la captura de la señal enviada por el mando para la apertura o cierre del vehículo, se cifra la comunicación y se envían cierto tipo de datos (como un contador del número de veces que se ha abierto) que hacen que mando y vehículo estén «sincronizados» y evite que el coche responda a otro mando aunque sea de la misma frecuencia.

- Garcia, F. D., Oswald, D., Kasper, T. y Pavlidès, P., «*Lock It and Still Lose It. On the (In)Security of Automotive Remote Keyless Entry Systems*», `https://www.n0secure.org/wp-content/uploads/2017/06/sec16_paper_garcia.pdf`

- Glocker, T., Mantere, T. y Elmusrati, M., «*A Protocol for a Secure Remote Keyless Entry System Applicable in Vehicles using Symmetric-Key Cryptography*», `https://arxiv.org/pdf/1612.00993.pdf`

En cuanto a la intimidad y la privacidad, y a pesar de las comunicaciones cifradas de las aplicaciones de mensajería, lo cierto es que la sociedad aún no percibe la privacidad como un derecho o como buena en sí misma. No hace falta más que pensar que no ciframos los correos electrónicos (aunque es muy sencillo), ni exigimos que los programas informáticos o redes sociales que usamos tengan implementadas estas medidas de protección. En esta dirección es muy recomendable la lectura del libro *Vigilancia Permanente* (*Permanent Record*) de Edward Snowden. Y para muestra, véase la siguiente cita de este libro:

> En última instancia, decir que no te importa la privacidad porque no tienes nada que esconder no es diferente a afirmar que no te importa la libertad de expresión porque no tienes nada que decir; o que no te importa la libertad de prensa porque no te gusta leer; o que no te importa la libertad de religión porque no crees en Dios; o que no te importa la libertad de reunión pacífica porque eres un agorafóbico perezoso y antisocial. El hecho de que esta o aquella libertad no tenga importancia para ti ahora mismo no quiere decir que la tenga o que no la vaya a tener mañana, para ti o para tu vecino…

La referencia completa del libro es la siguiente:

- Snowden, E., «*Vigilancia Permanente (Permanent Record)*», Editorial Planeta (2019), ISBN: 9788408215561, 448 páginas.

Referencias para este Capítulo

- [1] Capítulos 1 y 2
- [5] Capítulo 1
- [7] Capítulo 2

III. Criptografía Simétrica

III.1 Sistemas Clásicos de Sustitución

III.1.A César

Recibe este nombre por haber sido empleada por el legendario militar romano Julio César. Se trata de un código de sustitución muy sencillo, pero ilustra perfectamente las partes básicas de un sistema criptográfico así como las técnicas de criptoanálisis.

Como alfabeto se tomarán las letras mayúsculas A, B,..., Z (sin Ñ) junto con una biyección con $\mathbb{Z}/26$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*

Fijar una clave de cifrado es elegir un $n \in \mathbb{Z}/26$ y la aplicación de cifrado es

$$C : \mathbb{Z}/26 \to \mathbb{Z}/26$$

dada por $C(a) := a + n$ y la de descifrado es, por tanto, $D(a) := a - n$.

Ejercicio III.1.1 Codificar el mensaje

MOVER LAS TROPAS AL OTRO LADO DEL RIO

usando el cifrado de César con la clave $3 \in \mathbb{Z}/26$. Deducir cuál es la aplicación de descifrado.

Solución: En primer lugar hemos de traducir el mensaje a números según la tabla anterior. Obtenemos los números $12, 14, 21, 4, 17, \ldots$; seguidamente aplicamos la aplicación

de cifrado C (que consiste en sumar 3 módulo 26) y obtenemos $15, 17, 24, 7, 20, \ldots$ Deshaciendo el cambio resulta PRYHU... que será el mensaje cifrado. La aplicación de descifrado consistirá en el mismo método sumando 23 módulo 26.

Figura III.1: Imágenes de ruedas de cifrados de César. A la izquierda, la usada por el Ministro de Interior danés en 1910. A la derecha, la usada por el ejército Confederado en la Guerra Civil norteamericana (1865). Imágenes del National Cryptologic Museum (Wikimedia Commons).

Criptoanálisis de César

Siempre que consideremos un protocolo criptográfico debemos analizar las posibles vías de conseguir descifrarlo, bien sea descifrando mensajes concretos, bien obteniendo las claves de cifrado/descifrado. Esto se conoce como criptoanálisis.

Los tres modos más directos de romper el criptosistema de César son los siguientes:

- El ataque por fuerza bruta consistiría en probar sucesivamente con todas las claves posibles, es decir, con claves $0, 1, 2, \ldots, 25$ hasta dar con la clave de descifrado. Esto es, un atacante prueba a descifrar el mensaje interceptado utilizando la clave i, si el resultado tiene sentido entonces ha tenido éxito y para; en caso contrario, pasa a $i+1$ y vuelve a empezar. La fortaleza frente a este tipo de ataque se basa en el tamaño del espacio de claves: si dicho espacio de claves es muy grande, entonces un atacante tardará mucho tiempo en recorrerlo hasta dar con la clave de descifrado.
- Ataque basado en un conocimiento adicional: sabiendo que un mensaje interceptado termina con el símbolo K y que está firmado por PEDRO, podríamos deducir la clave de descifrado pues es la traslación de $\mathbb{Z}/26$ que lleva K a O.
- Ataque basado en el análisis de frecuencias. Obsérvese que una vez codificado un mensaje mediante el código de Julio César las frecuencias de las letras se mantienen. Es decir, si sabemos que el texto claro está en español (cuya letra más frecuente es la E) y tenemos una cantidad suficientemente grande de texto cifrado interceptado entonces parece razonable probar como aplicación de descifrado la aplicación que asigna el símbolo más frecuente de los textos cifrados a la letra E.

Ejercicio III.1.2 ¿Se obtendría alguna ventaja frente a ataques aplicando a un mensaje una doble codificación por César?

Observación III.1.3 — Código de sustitución. Dado un alfabeto de N letras, un código de sustitución consiste en fijar una biyección de $\mathbb{Z}/N$. Por ejemplo, si fijamos como tabla de sustitución (*clave*) la siguiente:

QWERTYUIOPASDFGHJKLZXCVBNM

quiere decir que para cifrar el texto se sustituye la A por Q, la B por W, la C por E, y así sucesivamente. El código de Julio César es un código de sustitución.

Obsérvese que las frecuencias de las letras se mantienen y aunque descifrar una letra no implica descifrarlas todas (como en el código del César), si se tienen suficientes mensajes cifrados (con la misma clave) sería posible deducir las frecuencias de todos los símbolos y entonces será muy fácil descrifarlos.

El análisis de frecuencias en criptoanálisis se debe al sabio árabe Al-Kindi (801-873 d.C.).

III.1.B Vigenère

Podemos pensar en el cifrado de Vigenère como una estrategia para aumentar el espacio de claves del cifrado de César, de forma que un ataque por fuerza bruta requiera más tiempo.

En este sistema, partimos el texto claro en bloques de longitud n y la clave consiste en n símbolos (o n números vía la biyección establecida). El cifrado toma un bloque y le suma (pensando en la suma de vectores) la clave. Por ejemplo, para cifrar el mensaje del Ejercicio III.1.1 con la clave AVE se procedería así:

- se toman los 3 primeros símbolos del texto claro (puesto que AVE tiene longitud 3);
- se suma MOV y AVE, es decir $(12,14,21)+(0,21,4)=(12,9,25)$ en $\mathbb{Z}/26\times\mathbb{Z}/26\times\mathbb{Z}/26$. Es decir, MJZ.
- se continúa con los siguientes 3 caracteres hasta finalizar.

Respecto del criptoanálisis de este tipo de cifrado, observemos en primer lugar que si el alfabeto tiene N símbolos y se elige una clave de longitud n, entonces tenemos N^n claves. Por contra, en el cifrado de César solo teníamos N claves. Hemos aumentado el espacio de claves por lo que será más robusto frente a un ataque por fuerza bruta.

Sin embargo, sigue habiendo otras formas de proceder que pueden terminar rompiendo este sistema criptográfico. Veremos dos de ellos. Ambos tienen la misma estructura

- establecer la longitud de la clave n;
- una vez establecida, se construyen n cadenas del siguiente modo: se toman los símbolos en las posiciones $1, 1+n, 1+2n, \ldots$, la segunda consiste de los símbolos en las posiciones $2, 2+n, 2+2n, \ldots$, etc.
- para cada una de estas cadenas (que ahora estarían cifradas por una misma clave, como si fuera César), digamos la i-ésima, se realiza una análisis de frecuencias, obteniendo el símbolo i-ésimo de la clave de descifrado.

Realmente, la etapa que queda por explicar es cómo determinar la longitud de la clave. Veamos cómo.

Índice de Coincidencia

Sea p la probabilidad de tomar al azar dos letras iguales en un texto original. Sea I la probabilidad de tomar al azar dos letras iguales en los mensajes interceptados (es

Figura III.2: Mecanismo de cifrado por Vigenère usado por el ejército Confederado en la Guerra Civil norteamericana (1865). Imágenes del National Cryptologic Museum (Wikimedia Commons).

decir, en los textos cifrados). Sea q la probabilidad de tomar al azar dos letras iguales entre las del alfabeto.

Entonces, si I está muy cerca de p significa que la longitud de la clave es pequeña, y si I está cerca de q significa que la longitud de la clave es grande.

Es más, si k es la longitud de la clave y n la longitud del texto cifrado, se verifica la siguiente aproximación

$$k \approx \frac{(p-q)n}{(p-I)+n(I-q)} \tag{III.1}$$

Evidentemente, la fracción de la derecha da un número fraccionario y la longitud de la clave es un número entero, por lo que probaremos con los enteros más cercanos a dicha fracción. Cuanto más grande sea el texto interceptado más preciso será el resultado.

Las probabilidades anteriores han de ser calculadas en función del alfabeto usado, por ejemplo, para los caracteres del alfabeto en español $p = 0{,}0775$ mientras que en inglés es $p = 0{,}0656$, en ambos casos $q = 0{,}0385$. Pero si consideramos espacios, mayúsculas y minúsculas, hay que volverlas a calcular.

Intuitivamente p grande significa que hay más orden en el texto claro, mientras que p más cerca de q significa que hay más desorden en el texto claro. En el primer caso, es que pocas cadenas de símbolos al azar tienen sentido en ese lenguaje (p. ej. en español no encontramos las cadenas xx, yy, bz,...).

Test de Kasiski

Se trata de buscar cadenas *largas* que se repitan en los mensajes interceptados. Una vez detectadas, se calculan las distancias entre los primeros símbolos de cada cadena repetida $d_1, \ldots, d_r$. Entonces, la longitud de la clave es divisor del máximo común divisor de las diferencias $d_2 - d_1, d_3 - d_2, \ldots, d_r - d_{r-1}$.

Para tener éxito, necesitamos interceptar gran cantidad de texto cifrado, y hacer varios intentos. Por ejemplo, buscamos cadenas de longitud 3 que se repitan al menos 4 veces (menos veces podría ser una casualidad, pero esto va en función de la longitud del texto), calculamos el m.c.d. de las distancias entre ellas, y hacemos análisis de frecuencias. Si no tiene sentido, volvemos a empezar con otros números, por ejemplo, cadenas de 2 que se repitan al menos 5 veces, etc.

III.2 Otros Criptosistemas

Desde el punto de vista matemático, C. Shannon puso de manifiesto las características que debe tener un sistema criptográfico: *confusión y difusión.*

- confusión: mezclar la información de modo que el resultado resista ataques estadísticos (p. ej. análisis de frecuencias, estudio de relaciones entre símbolos; en general, criptoanálisis diferencial).
- difusión: diseminar las información por bloques (p. ej. que un cambio en un 1 bit afecte a todo el resultado final).

La idea es cifrar grandes bloques de texto de longitud prefijada. Pero no se hace como Vigenère, sino que se entremezclan los símbolos de todo el bloque en el cifrado. Los casos de r-grafos y cifrados afines son modelos matemáticos sencillos que incorporan estas dos propiedades, mientras que DES, Triple DES, AES ya son modelos completamente operativos y usados comercialmente.

III.2.A r-grafos

Siguiendo con la meta incrementar la fortaleza de estos sistemas, veamos a continuación otro método de ampliar el espacio de claves. Podemos pensar que estamos ampliando el alfabeto; de hecho en lugar de trabajar con N símbolos, trabajaremos con N^r del siguiente modo. Cada símbolo del alfabeto nuevo se corresponde con r-símbolos (de ahí r-grafo) del antiguo mediante la siguiente biyección

$$\begin{aligned} \mathbb{Z}/N \times \ldots \mathbb{Z}/N &\xrightarrow{\sim} \mathbb{Z}/N^r \\ (a_{r-1}, \ldots, a_0) &\longmapsto a_{r-1}N^{r-1} + \ldots + a_1 N + a_0 \end{aligned}$$

Obsérvese que es solamente una biyección (no es morfismo de grupos ni de anillos). A la hora de operar, solamente nos importan las operaciones del lado derecho, es decir, la aritmética de $\mathbb{Z}/N^r$. Esta biyección se usa para la traducción de los caracteres del texto plano en números, por tanto, dado un texto plano se dividirá en bloques de longitud r cada uno, y cada uno de estos bloques se le asigna un número por la biyección anterior.

La aplicación de cifrado es entonces una biyección de conjuntos

$$C : \mathbb{Z}/N^r \xrightarrow{\sim} \mathbb{Z}/N^r$$

Ha de ser biyectiva para que exista inversa y pueda descifrarse.

Proposición III.2.1 En la situación anterior, si $C(x) = ax + b$ para ciertos $a, b \in \mathbb{Z}/N^r$, entonces C es biyectiva si y solo si $(a, N^r) = 1$. En ese caso, la inversa $D = C^{-1}$ es $D(x) = a^{-1}x - a^{-1}b$.

Demostración. Sea $d = (a, N^r)$. Si $d \neq 1$, entonces $C(0) = C(\frac{1}{d}N^r)$, por lo que no es inyectiva. Si $d = 1$, entonces a tiene inverso, y se comprueba fácilmente que $D(x) = a^{-1}(x-b) = a^{-1}x - a^{-1}b$ es la inversa.

Ejercicio III.2.2 Encriptar el mensaje «BUENOS DÍAS» mediante un 2-grafo sobre $\mathbb{Z}/26$ y con la aplicación de cifrado $C(x) = 17x + 32$. Calcular la aplicación de descifrado.

Solución: En primer lugar, dividimos el texto claro en bloques de 2 símbolos y a cada uno le asignamos un número. El primero es BU $= 1 * 26 + 20 = 46$. Como

$C(46) = 17*46+32 = 814 = 138$ módulo $26^2 = 676$. Finalmente, $138 = 5*26+6$. Luego BU se cifra como FG. Se prosigue de este modo hasta terminar el ejercicio.

Proposición III.2.3 Si consideramos las aplicaciones $C(x) = ax+b$ con $a,b \in \mathbb{Z}/N^r$ y $(a,N^r) = 1$, entonces el espacio de claves tiene $\phi(N^r)\cdot N^r$ elementos (siendo ϕ el indicador de Euler).

III.2.B Transformaciones Afines

Como continuación de las ideas anteriores, surge la consideración del uso de aplicaciones afines para cifrar, esto es las aplicaciones

$$\begin{aligned} \mathbb{Z}/N\times\ldots\mathbb{Z}/N &\xrightarrow{C} \mathbb{Z}/N\times\ldots\mathbb{Z}/N \\ x &\longmapsto Ax+b \end{aligned}$$

siendo A una matriz con entradas en $\mathbb{Z}/N$ y b un vector de $\mathbb{Z}/N\times\ldots\mathbb{Z}/N$.

Proposición III.2.4 En la situación anterior, si $C(x) = Ax+b$, entonces C es biyectiva si y solo si $(\det(A),N) = 1$. En ese caso, la inversa $D = C^{-1}$ es $D(x) = A^{-1}x - A^{-1}b$. En particular, la inversa es una aplicación afín.

Ejercicio III.2.5 Documentarse sobre la *escítala espartana*. Tomar un ejemplo concreto (de longitud de la cinta y de radio del cilindro) y modelarlo en términos de un cifrado afín.

E	n		u	n		l	u	g	a
r		d	e		l	a		M	a
n	c	h	a	,		d	e		c
u	y	o		n	o	m	b	r	e
	n	o		q	u	i	e	r	o
	a	c	o	r	d	a	r	m	e

Figura III.3: En la izquierda el cifrado consiste en escribir el texto claro sobre una cinta y el texto cifrado se obtiene leyendo el resultado después de enrollar la cinta en un cilindro. De modo equivalente, en la derecha, el texto claro se escribe por filas de izquierda a derecha y el texto cifrado se obtiene leyendo por columnas de arriba a abajo. (Wikimedia Commons).

Observación III.2.6 Obviamente, podríamos combinar los dos sistemas anteriores, r-grafos y transformaciones afines, para conseguir cifrados por medio de transformaciones afines de $\mathbb{Z}/N^r\times\ldots\times\mathbb{Z}/N^r$.

Criptoanálisis

Respecto del criptoanálisis, hay que decir que no se pueden generalizar las ideas anteriormente vistas (tipo Kasiski, etc.) aunque siempre hay que considerar en primer lugar la fortaleza del sistema frente al ataque por fuerza bruta. Para calcularla esta resistencia hay que determinar el tamaño del espacio de claves, esto es, el cálculo exacto del número de aplicaciones afines (biyectivas).

Es una cuenta laboriosa pero sin complejidad. Por ejemplo, si tomamos $N = p$ un número primo, vamos a calcular el número de matrices $n\times n$ con entradas en $\mathbb{Z}/p$

que son invertibles (en $\mathbb{Z}/p$). Que una matriz sea invertible (que siempre equivale a que su determinante sea invertible) se traduce, por ser $\mathbb{Z}/p$ un cuerpo, a que tenga determinante no nulo y, por tanto, a que las vectores dados por sus columnas sean linealmente independientes. Como primera columna, tomamos un vector no nulo, hay tantos como $p^n - 1$. Dadas i columnas, por ser linealmente independientes, cada vector del subespacio que generan se escribe de forma única como combinación lineal de ellas. Por tanto, el subespacio generado por las i columnas (vectores l.i.) tiene exactamente p^i elementos. La columna $i+1$-ésima es un vector cualquiera que no yace en dicho subespacio, por lo que hay $p^n - p^i$ elecciones posibles. En conclusión, el número de elementos es $\prod_{i=0}^{n-1}(p^n - p^i)$.

$$\#\{A \in \mathrm{Mat}_{n\times n}(\mathbb{Z}/p) | A \text{ es invertible } \} = \prod_{i=0}^{n-1}(p^n - p^i)\,. \tag{III.2}$$

A partir de este cálculo se puede generalizar al caso $\mathbb{Z}/p^r$, considerando el paso al cociente $\mathbb{Z}/p^r \to \mathbb{Z}/p^{r-1}$, y de ahí al caso general $\mathbb{Z}/n$ utilizando el Teorema chino de los restos.

Por otro lado, la aplicación tanto del test de Kasiski como del índice de coincidencia (con el objetivo de determinar r en los r-grafos, o el rango del $\mathbb{Z}/N$-módulo en las transformaciones afines) no arrojaría resultados válidos. La razón es que estos crifrados trabajan con grupos de símbolos (r en el primer caso, tantos como el rango en el segundo) por los que al mezclarlos entre sí desaparecen las coincidencias buscadas o se hacen extremadamente raras. Son las consecuencias de las características de confusión y difusión.

Como comentarios finales, observemos que:

- El cifrado de César corresponde a la transformación afín $\mathbb{Z}/N \to \mathbb{Z}/N$ dada por $C(x) = x + b$ con $b \in \mathbb{Z}/N$.
- El cifrado de Vigenère corresponde a $(\mathbb{Z}/N)^n \to (\mathbb{Z}/N)^n$ dada por $C(x) = x + b$ siendo $b \in (\mathbb{Z}/N)^n$.
- Si en este cifrado por transformaciones afines se toma $b = 0$, es el conocido como cifrado de Hill.

Observación III.2.7 Obsérvese que las transformaciones afines de dimensión 1 de una letra son un caso particular de códigos de sustitución. En general, si ϕ es una transformación afín de $\mathbb{Z}/N$, entonces la frecuencia relativa de un símbolo $x \in \mathbb{Z}/N$ en el texto original y la frecuencia relativa de $\phi(x)$ en el texto cifrado coinciden.

Observación III.2.8 Obsérvese que si encriptamos *ab* mediante una transformación afín de dimensión 1 a pares de letras (*digraph transformations*) y obtenemos *xy*, entonces *y* solo depende de *b*, por lo que se puede realizar un análisis de frecuencias para intentar descrifarlo.

III.2.C DES, Triple DES y AES

DES: Data Encryption Standard

Describamos el funcionamiento. Se parte de una clave de 64 bits de longitud y de un texto claro de 64 bits.

- Creación de subclaves: A partir de la clave original, se crean 16 subclaves de 48 bits cada una. Tomando la clave inicial K, se extraen y reordenan 56 de sus

bits (según una tabla), denotemos el resultado por K'. Sea ahora I_0, D_0 sus dos mitades izquierda y derecha respectivamente (por lo que tienen 28 bits cada). Definimos I_i (resp. D_i) como la rotación cíclica a la izquierda I_{i-1} (resp. D_{i-1}) para $1 \leq i \leq 16$. Definimos la clave K_i como una determinada extracción de 48 bits de la concatenación $I_i D_i$ (de nuevo, según otra tabla).

- La función Feistel: Se trata de una función F que depende de una entrada de 32 bits y de una subclave de 48 bits, y devuelve 32 bits. Se procede del siguiente modo, dados los 32 bits, se reordenan y expanden (según una tabla) a 48 bits. Se hace un XOR con dicha expansión y la subclave, y el resultado se ordena como 8 bloques de 6 bits. A cada bloque de 6 bits se le aplica una transformación (las llamadas S-boxes) que los mezcla de forma no lineal y da una salida de 4 bits. Los 32 bits resultantes se vuelven a reordenar según una permutación.
- Cifrado de un bloque de 64 bits: Se aplica una permutación P a los bits del bloque y sean $\mathscr{I}_0$ y $\mathscr{D}_0$ las mitades izquierda y derecha respectivamente de dicho resultado (de 32 bits cada una). Se hace ahora 16 rondas de la siguiente iteración

$$\begin{aligned} \mathscr{I}_i &= \mathscr{D}_{i-1} \\ \mathscr{D}_i &= \mathscr{I}_{i-1} \operatorname{XOR} F(\mathscr{D}_{i-1}, K_i) \end{aligned}$$

  El texto cifrado es el resultado de aplicar P^{-1} a $\mathscr{I}_{16}\mathscr{D}_{16}$.

En el caso del DES, las propiedades requeridas por Shannon, *confusión y difusión* son consecuencia de la combinación entre las permutaciones y las cajas S.

Observación III.2.9

- La clave original de 64 bits consta de 56 bits y los 8 restantes son bits de paridad.
- El proceso de descifrado usa la misma estructura pero utilizando las claves en orden inverso (la razón práctica es que se simplifica el *software* o *hardware* necesario).
- Es muy interesante leer la historia del DES, por ejemplo, cómo la NSA presionó a IBM para rebajar la longitud de la clave, o como las cajas S (S boxes) permanecieron en secreto hasta hace poco. Todo ello alimentó la creencia de que la NSA tenía una *puerta trasera* o un método de romper el sistema.
  - `https://en.wikipedia.org/wiki/Data_Encryption_Standard`
  - `http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm`
- Ha sido un sistema muy utilizado (p. ej. en los cajeros automáticos). El sistema tiene 2^{56} claves, y desde 2012 hay sistemas que pueden examinar completamente dicho espacio de claves (i. e. ataque de fuerza bruta).
- La pieza fundamental en este sistema es la consideración de funciones que mezclan (confunden y difunden) los bits. En general, serían expresiones polinómicas del tipo $(\mathbb{Z}/N)^n \to (\mathbb{Z}/N)^r$ que no sean lineales. Puede consultarse la teoría de polinomios de permutaciones *(permutation polynomials)* en `https://en.wikipedia.org/wiki/Permutation_polynomial`. El caso más sencillo, son las aplicaciones afines vistas anteriormente, es decir, los cifrados afines son un *modelo de juguete* de estos otros sistemas.

Triple DES

Esta basado en tres aplicaciones sucesivas del DES con tres claves distintas K_1, K_2, K_3. Si E_i (resp. D_i) es la aplicación de encriptado con la clave K_i (resp. descifra-

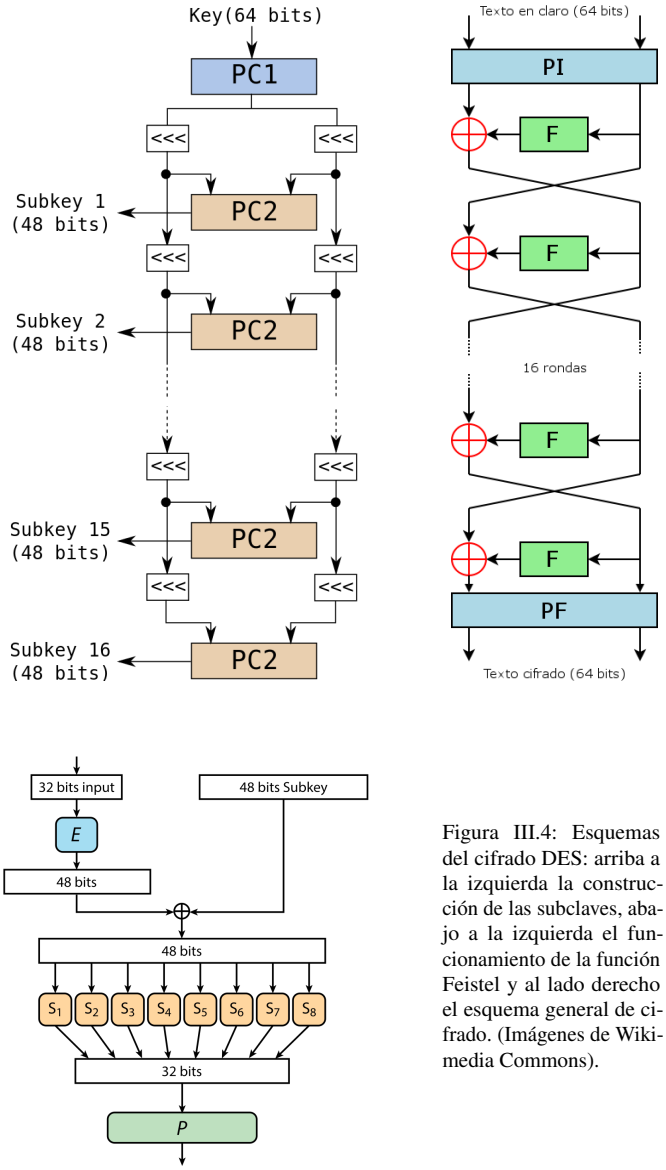

Figura III.4: Esquemas del cifrado DES: arriba a la izquierda la construcción de las subclaves, abajo a la izquierda el funcionamiento de la función Feistel y al lado derecho el esquema general de cifrado. (Imágenes de Wikimedia Commons).

do), entonces el texto cifrado es el resultado de aplicar $E_3 \circ D_2 \circ E_1$. El procedimiento de descifrado consiste en aplicar $D_1 \circ E_2 \circ D_3$ al texto cifrado.

De este modo se obtendría teóricamente una longitud de clave de 168 bits (un espacio de claves con 2^{168} elementos) pero se prueba (por el diseño del DES) que la seguridad efectiva proviene únicamente de 112 bits. Se han descrito algoritmos para conseguir *colisiones*, es decir, construir mensajes que arrojen el mismo resultado (nos referimos a cuando se usa Triple DES como firma, i. e. como función hash, lo veremos más adelante). Por esta y otras razones, el NIST lo declaró obsoleto en 2017.

AES

El AES (*advanced encryption standard*) surgió a partir de la necesidad de disponer de nuevos sistemas frente al obsoleto DES que había sido roto. Tiene similitudes de diseño con el DES y, aunque no usa las funciones Feistel, sigue garantizando los principios de confusión y difusión mediante una red de sustituciones y permutaciones.

Se usa la misma clave para cifrar y descifrar y actualmente no se conocen ataques (que puedan ser llevados a la práctica). El AES utiliza una clave de 256 bits o, equivalentemente, su espacio de clave tiene $2^{256} \sim 1,15 \times 10^{77}$ elementos, que garantiza su seguridad frente a los ataques estándar de fuerza bruta.

Actualmente en uso, lo encontramos en el cifrado de discos duros y en comunicaciones por internet, entre otras aplicaciones.

Observación III.2.10 Según Snowden, la NSA está trabajando en romper el AES. Los procesadores más modernos incorporan instrucciones propias para el AES, que proporcionan velocidad en el cifrado/descifrado así como una mayor seguridad frente a ataques (hoy en día ataques teóricos, pero quizás haya en el futuro otros ataques).

III.3 Ejercicios

III.3.A César y Vigenère

Problema III.3.1 Se pide descifrar el siguiente mensaje sabiendo que está en encriptado al modo de Julio César (aunque no se conoce el idioma).

PXPXKXENVDRUXVTNLXHYMXGMAXYKXJN
XGVRFXMAHWGXXWLEHGZXKVBIAXKMXQM

Solución: Puesto que no disponemos de un texto interceptado suficientemente largo, ni conocemos la lengua del texto claro, hemos de ir haciendo pruebas y examinando si los resultados obtenidos son consistentes.

Utilizaremos para ello que las letras más frecuentes en español son

E	A	O	S	R	N
13,67 %	12,53 %	8,68 %	7,98 %	6,87 %	6,71 %

mientras que en inglés son

E	T	A	O	N	I
13,0 %	9.1 %	8.2 %	7.5 %	7.0 %	6.7 %

Probaremos con el idioma inglés, cuya letra más común es la E. En el texto interceptado la letra más frecuente es la X, con una frecuencia del 23 % . Si así fuera, la aplicación de cifrado tendría que asignar E$\mapsto$X, es decir, sumar 19 mod 26. La aplicación de descifrado será entonces sumar 7 mod 26. De este modo P$\mapsto$W, X$\mapsto$E, etc. y el resultado final (incluyendo espacios a posteriori) reza así

WE WERE LUCKY BECAUSE OFTEN THE FREQUENCY
METHOD NEEDS LONGER CIPHER TEXT

Problema III.3.2 Encriptar el mensaje

HOY SALIMOS A LAS OCHO

(eliminando los espacios) mediante el crifado de Vigenère con la clave ROSA en $\mathbb{Z}/26$.

Problema III.3.3 El siguiente texto está cifrado por el método de Vigenère. Descifradlo.

```
ÑÑEXTLKIPWWPI,SLKQYYZLKZHYXWPIWLDLIGYIASZLIOPLCCZV.YSAELWDZJRMUAIG
PW.D,LEEDTPDQHGDXÑEDKWCQLMDLDNMMOUL,QZEKBEAC.PXOULPHÑAYLMÑATBUAILO
VUE,.DQDL.EYDLOWNIZCWZIEHEEDOLEYE.LDAWTLOQEKLYEXTTQUWT-IJHLWDEWFDE
IRTKEPWKWHZD,LVYQGPOOGRAWOHRXW-D,TCODY.HYSHLYYVHKW.SZNIÑGTYIYYZMWY
EG.TAXGPEZDSAAYYHMKXXSLYHQHHTÑOGDVZJRMHYQJAEYEOLO,PVLSYVTTMLGTPXYD
R.SLHGPDLD,YEBGROHDFHOHYVTTMYD,NIYILJPOELMDAZVPDHGS.DQQVWDOOVDDAWS
PXYIYNIYDHKUATY,GYIÑÑIZHPAOYSLIHYPYBCUJL-IHDO.EHDL.MYYG.IZESODZERA
DYYFÑQLPGLIYIJNOAMFYDLMGTH.E.XVZPZÑQJQ,LEZOYK-MI.LYYSMPEYEOLIHIHLV
ZDRYEXGSLSADHMOÑCZYIMIHPDHIJOCDWH-IMVZLWQVVWDZELLWLDLBPAXZWDEMYYXP
FHLQODY,WPCSBDQMT.POWG.YZUL.VYEVLOQEGATODLSGÑIHXV,FRPS-OWTILULCTOC
HTMZMTODPSGDLOULMMHDOTVJDLDDHLIMOESLLSYMYBUAXRWDAMTLQHDLWWZPZNE,WG
PW.DO,EHWKLEAEGYXJIKLYZCXPOYYZYPOULDDÑMOK-,QLCDAUOCDYVZRMYYJXQXWTW
IZD..UAE,WUAWGMEHD,AEYVCYSNILLUAOG.WOMGMJAAGPOOE
```

donde «-» indica un espacio.

Solución: Es un texto largo para que podamos usar las distintas herramientas de índice de coincidencia y análisis de frecuencias. En primer lugar, examinamos todo el texto y encontramos que hay 30 símbolos distintos. De este modo, el alfabeto usado será «ABCDEFGHIJKLMNÑOPQRSTUVWXYZ.,-» donde el último guión representa un espacio. En primer lugar, hemos de determinar la longitud de la clave para lo cual usaremos el índice de coincidencia.

La lista de frecuencias de estos símbolos en el texto cifrado es

$$\{32,7,16,63,46,6,28,40,38,13,15,68,33,8,14,46,\\ 34,22,12,23,28,17,22,40,21,66,33,7,24,18,12\}$$

Si f_i denota la frecuencia del i-ésimo símbolo, entonces tenemos

$$I=\sum_{i=1}^{30}\frac{f_i(f_i-1)}{n(n-1)}=0{,}0432493$$

siendo $n=852$ la longitud del texto. En español, con esta lista de 30 símbolos, el examen de frecuencias sobre textos planos largos, arroja un valor $p=0{,}0803979$. Y la elección de caracteres al azar entre esta colección de 30 símbolos, permite calcular $q=0{,}0319458$. Entonces, la fórmula (III.1) es

$$\frac{(p-q)n}{(p-I)+n(I-q)}=\\ =\frac{(0{,}0803979-0{,}0319458)852}{(0{,}0803979-0{,}0432493)+852(0{,}0432493-0{,}0319458)}\approx 4{,}2699$$

Con lo cual, el primer candidato a longitud de clave sería 4. Tras probar, no se obtiene resultados, así que probamos con longitud 5. En este caso, se ha de dividir el mensaje

en 5 subcadenas. Sea M_i la subcadena del mensaje interceptado formado por los caracteres en las posiciones j con $j = i \bmod 5$. Es decir, M_1 =ÑLWLZYWGZL... consta los caracteres en las posiciones $1, 6, 11, 16, \ldots$ Haciendo análisis de frecuencias para M_1 comprobamos que L es el más frecuente. Puesto que la posición de la L en el alfabeto es la 11 y la de la E es la 4 (comenzando por la A en la posición 0) y asignando el carácter más frecuente con la letra E (el carácter más frecuente en español), la forma de descifrar es sumar 23 módulo 30 y, de este modo, hemos descifrado M_1. Se repite el proceso para $M_2, \ldots, M_5$ y se vuelve a recomponer el mensaje a partir de las subcadenas. Las claves de descrifrado para cada una de estas cinco subcadenas son $\{23, 18, 26, 4, 26\}$.

El texto recuperado es un fragmento de *El Quijote* que comienza así

> Hechas, pues, estas prevenciones, no quiso aguardar mas tiempo a poner en efeto su pensamiento, apretandole a ello la falta que el pensaba que hacia en el mundo su tardanza,

III.3.B Transformaciones Afines (dimensión 1)

Problema III.3.4 Se intercepta el mensaje:

KCVGUWIQLZCVLIWTDVGZDGZWODTVWCT
GETCGUCTVDQJLTGIQDUEDTUWLVGZDGKD QLV

y se sabe que el texto claro está en español, que utiliza 28 símbolos (alfabeto con Ñ y espacio como carácter 28) y que ha sido encriptado mediante una transformación afín de $\mathbb{Z}/28$ (unidades de una letra). Además, por mensajes anteriores, se sabe que las letras más repetidas son D, L y C por este orden.

Descrifar el mensaje.

Solución: Una transformación afín de $\mathbb{Z}/28$ es del tipo $x \mapsto ax + b$. Para descifrar tendremos que determinar a, b tales que $D \mapsto E$, $L \mapsto A$ y $K \mapsto O$, de donde resulta el sistema $E = aD + b, A = aL + b, O = aC + b$ o, traduciendo a sus equivalentes numéricos, $4 = a3 + b, 0 = a11 + b, 15 = a2 + b$, que resulta en $a = 17, b = 9$. Es decir, la expresión de la aplicación de descrifado es $x \mapsto 17x + 9 \bmod 28$. De ese modo, resulta el texto claro

LOS CIFRADOSAFINES DE DIMENSION UNO
CONSERVAN FRECUENCIAS DE LETRAS

Problema III.3.5 Se intercepta el mensaje (en inglés):

SZOOTWHQ

y se sabe que ha sido encriptado mediante una transformación afín (de dimensión 1) de pares de letras (*digraph transformation*).

Descrifad el mensaje sabiendo que los pares que más se repiten en los mensajes interceptados son IX y TQ y que los que más se repiten en el inglés habitual son TH y HE (en este orden).

Suplantad la identidad y mandar GOODWORK usando la misma clave de encriptación.

Solución: Una transformación afín de dimensión 1 sobre pares de letras (con un alfabeto de 26 símbolos) es $T : (\mathbb{Z}/26^2) \to (\mathbb{Z}/26^2)$ de la forma $T(x) = Ax + b$ siendo

$A \in (\mathbb{Z}/26^2)^*$ y $b \in (\mathbb{Z}/26^2)$. Las hipótesis se escriben como $T(\mathrm{TH}) = \mathrm{IX}$ y $T(\mathrm{HE}) = \mathrm{TQ}$ y, sustituyendo cada letra por su valor resulta un sistema en $\mathbb{Z}/26^2$:

$$\begin{aligned} T(19*26+7) &= A(19*26+7)+b = 8*26+23 = 231 \\ T(7*26+4) &= A(7*26+4)+b = 19*26+16 = 510 \end{aligned}$$

La matriz de coeficientes de este sistema es $\begin{pmatrix} 19*26+7 & 1 \\ 7*26+4 & 1 \end{pmatrix} = \begin{pmatrix} 501 & 1 \\ 186 & 1 \end{pmatrix}$ cuyo determinante es 315 que es primo con 26^2 y, siendo invertible, el sistema tiene solución y es única. La inversa de la matriz de coeficientes es $\frac{1}{315}\begin{pmatrix} 1 & -1 \\ -186 & 501 \end{pmatrix} = \begin{pmatrix} 191 & 485 \\ 302 & 375 \end{pmatrix}$. El vector columna (A,b) se obtiene multiplicando la inversa por el vector columna $(231,510)$ y, operando, resulta que $A = 115, b = 76$. En resumen, $T(x) = 115x+76$ es la aplicación de cifrado. Calculando la transformación afín inversa, que es la aplicación de descifrado resulta $D(x) = 115^{-1}x - 115^{-1}*76 = 435x+64$. Para descifrar tenemos que ir operando por dígrafos

$$\begin{aligned} D(\mathrm{SZ}) &= D(18*26+25) = 435(18*26+25)+64 = 8*26+19 = IT \\ D(\mathrm{OO}) &= D(14*26+14) = 435(14*26+14)+64 = 8*26+18 = IS \\ D(\mathrm{TW}) &= D(19*26+22) = 435(19*26+22)+64 = 3*26+14 = DO \\ D(\mathrm{HQ}) &= D(7*26+16) = 435(7*26+16)+64 = 13*26+4 = NE \end{aligned}$$

El cifrado se haría análogamente, $T(\mathrm{GO}) = 115(6*26+14)+76 = 0*26+22 = AW$, etc.

Problema III.3.6 Se intercepta el mensaje (en inglés):

DXM SCE DCCUVGX

y se sabe que ha sido encriptado mediante una transformación afín de pares de símbolos en el alfabeto con 30 letras siguiente: $A = 0, \ldots, Z = 25$, $\_ = 26$ (el espacio), $? = 27$, $! = 28$, $' = 29$. Además, se observa que los pares que más se repiten en los mensajes interceptados son $M\_$, $U\_$ y IH y los que más se repiten en el inglés habitual son $E\_$, $S\_$ y $\_T$ (en este orden).

Descrifad el mensaje.

Suplantad la identidad y mandar *YES I'M JOKING!* usando la misma clave de encriptación.

Problema III.3.7 Se intercepta el siguiente texto:

```
NZ ZYÑCKDY. KJÑKNGYRVS.YJKNSÑXT YRWYL,ÑKRXÑ ZIKNSEGÑÑYSKD
. J .KKSÑ .YÑPIXWUDXKEHSYÑCP.HÑ UKV XSSGJKIXCKK YNL,ÑKKSÑ
 .YÑPN X EJJPYSULÑ,NGÑ  ADDYEEHSEGCKYGXSYS KNGÑ K!'ÑKUHÑKK
 YNTGHPH NKRXEH MS  KDY.ZAZEMEGÑKKXPKNSEGÑKWSEEHSYS KNGK
 Ñ .,L Ñ KDCKNGÑ . Ñ,CÑVSSDÑK.SÑ S BZE  HPYODYÑKNZKTYDLYÑ
!'VKCH.!'CKW!',SJZYNVS SÑ Ñ!'VKKSKDCKRXYNVÑLZ
KRST  !'AZAÑVNXÑ PCZYO,,ISÑ,,
YUHIKNZ KCÑCKRXNFJKCPEEHSÑ N X YSUKRZJ IL
```

Y se sabe que se ha cifrado por una transformación afín de pares de letras (es decir, de $\mathbb{Z}/32^2$), que el alfabeto es: « *ABCDEFGHIJKLMNÑOPQRSTUVWXYZ¡!.,* » y que la primera palabra es COMO.

Descifradlo.

Si no supiéramos la primera palabra, podríamos intentarlo por análisis de frecuencias de pares de letras. Intentarlo de esta forma.

III.3.C Transformaciones Afines (dimensión 2)

Problema III.3.8 Compárense las transformaciones afines de dimensión 1 de pares de letras con las transformaciones afines de dimensión 2 de una letra.

Problema III.3.9 Se intercepta el mensaje (en inglés):

!IWGVIEX!ZRADRYD

y se sabe que ha sido encriptado mediante una transformación afín a pares de símbolos en el alfabeto con 29 letras siguiente: $A = 0,\ldots, Z = 25$, $\_ = 26$ (el espacio), $? = 27$ y $! = 28$. Además, se sabe que está firmado por el remitente que es *MARIA*.

Descrifar el mensaje.

Problema III.3.10 Se intercepta el mensaje (en inglés):

WUXHURWZNQR XVUEXU!JHALGQGJ?

y se sabe que ha sido encriptado mediante una transformación afín de vectores (x,y) en un alfabeto de 841-letras. Aquí se entiende que $x = 29x_1 + x_2 \in \mathbb{Z}/841 = \mathbb{Z}/29^2$ y $\mathbb{Z}/29$ se corresponde con las letras del ejercicio anterior. Por tanto cada 4 letras del mensaje interceptado dan un vector; por ejemplo, *abcd* daría $(29a+b, 29c+d)$. Se sabe que el mensaje lleva la firma HEADQUARTERS.

Descrifar el mensaje.

Problema III.3.11 Se intercepta el mensaje (en inglés):

FBRTLWUGAJQINZTHHXTEPHBNXSW

y se sabe que ha sido encriptado por el cifrado de Hill de dimensión 3 (i. e. una transformación lineal) en un alfabeto de 26 letras; es decir, de $(\mathbb{Z}/26)\times(\mathbb{Z}/26)\times(\mathbb{Z}/26)$. Se sabe que el mensaje lleva la firma JAMESBOND.

Descrifar el mensaje.

Solución: Si $\mathscr{A}$ es la matriz de la aplicación lineal $(\mathbb{Z}/26)\times(\mathbb{Z}/26)\times(\mathbb{Z}/26) \to (\mathbb{Z}/26)\times(\mathbb{Z}/26)\times(\mathbb{Z}/26)$ que descifra, entonces tomando los valores asociados a cada letra, tendremos

$$\mathscr{A}\begin{pmatrix} T \\ E \\ P \end{pmatrix} = \begin{pmatrix} J \\ A \\ M \end{pmatrix} \qquad \mathscr{A}\begin{pmatrix} H \\ B \\ N \end{pmatrix} = \begin{pmatrix} E \\ S \\ B \end{pmatrix} \qquad \mathscr{A}\begin{pmatrix} X \\ S \\ W \end{pmatrix} = \begin{pmatrix} O \\ N \\ D \end{pmatrix}$$

con

$$\mathscr{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Resulta un sistema de 9 ecuaciones con 9 incógnitas, $a_{11},\dots,a_{33}$. Para simplificarlo, veamos cómo se puede transformar en tres sistemas de tres ecuaciones cada uno. Tomando las primeras ecuaciones de cada uno de los tres anteriores se obtiene un sistema de tres ecuaciones para las incógnitas a_{11},a_{12},a_{13}, es decir, tenemos tres sistemas

$$\mathscr{B}\begin{pmatrix}a_{11}\\a_{12}\\a_{13}\end{pmatrix}=\begin{pmatrix}J\\E\\O\end{pmatrix}\qquad \mathscr{B}\begin{pmatrix}a_{21}\\a_{22}\\a_{23}\end{pmatrix}=\begin{pmatrix}A\\S\\N\end{pmatrix}\qquad \mathscr{B}\begin{pmatrix}a_{31}\\a_{32}\\a_{33}\end{pmatrix}=\begin{pmatrix}M\\B\\D\end{pmatrix}$$

con

$$\mathscr{B}=\begin{pmatrix}T&E&P\\H&B&N\\X&S&W\end{pmatrix}$$

De donde resulta

$$\mathscr{A}=\begin{pmatrix}18&13&3\\21&18&19\\19&3&11\end{pmatrix}$$

III.3.D Criptoanálisis

Problema III.3.12 Nótese que si el número de posibles claves en un criptosistema basado en una transformación afín es *pequeño* entonces se puede intentar descifrar por fuerza bruta.

Calcular el número de claves, de una transformación afín de dimensión n, en los siguientes casos:

- en $\mathbb{Z}/26$;
- en $(\mathbb{Z}/26)^2$;
- en $(\mathbb{Z}/29^2)\times(\mathbb{Z}/29^2)$.

Solución: Una transformación afín definida sobre $\mathbb{Z}/26$ es de la forma $T(x)=Ax+b$ con $A\in(\mathbb{Z}/26)^*$ y $b\in\mathbb{Z}/26$. Por tanto hay $\phi(26)\cdot 26$ claves posibles, siendo ϕ el indicador de Euler. Esto es, $12\cdot 26=312$ claves. En segundo caso, $\mathbb{Z}/26\times\mathbb{Z}/26\to\mathbb{Z}/26\times\mathbb{Z}/26$, vendrá dado por $T(x)=Ax+b$ con A una matriz invertible con entradas en $(\mathbb{Z}/26)$ y $b\in\mathbb{Z}/26\times\mathbb{Z}/26$. Por el Teorema Chino de los Restos, se observa que dar A es equivalente a dar dos matrices, que denotemos A_2 y A_{13}, tal que A_i tiene entradas en $\mathbb{Z}/i$ y es invertible. Por el Ejercicio III.2 hay $(2^2-1)(2^2-2)$ elecciones para A_2 y $(13^2-1)(13^2-13)$ para A_{13}. En total, 2 044 224 claves. El resto del problema se resuelve de modo similar.

Problema III.3.13 Supongamos que se encripta mediante una transformación afín ϕ de $\mathbb{Z}/N$ y conocemos $\phi(x)$ para un x. ¿Cuantas ϕ posibles hay?, ¿y si conocemos $\phi(x_1)$ y $\phi(x_2)$ para dos letras distintas x_1, x_2?

III.4 Temas para ampliar

III.4.A La Máquina Enigma

Es uno de los sistemas criptográficos más populares por su papel en la Segunda Guerra Mundial. Fue diseñada por Alemania a partir de unos modelos previos. La

idea principal es que la clave de encriptado (que no es especialmente larga) para el cifrado del carácter n-ésimo depende de la clave utilizada en el instante anterior y del carácter $(n-1)$-ésimo. Fue un sistema muy robusto durante los primeros años, hasta que Alan Turing, al servicio del MI5, consiguió romperlo. El enfoque teórico usado por A. Turing constituyó el nacimiento de la Teoría de la Computabilidad (dando lugar, por ejemplo, a lo que conocemos hoy en día como máquinas de Turing) y, en ese sentido, construyó *la máquina universal*, más conocida como la *bomba de Turing*.

El éxito se basó en una larga lista de factores entre los que citamos los siguientes: los primeros estudios realizados por matemáticos polacos, la intercepción de un libro de claves, el hecho de que muchos mensajes cifrados comenzaran con los mismos caracteres (del tipo lugar, fecha) y que se cifraban demasiados mensajes (proporcionando una amplia colección de mensajes cifrados para su estudio), una deficiencia en el diseño de la máquina (el cableado y engranajes limitaban el espacio de claves, siendo este bastante menor que el espacio de claves teóricos), etc.

Aún con la guerra terminada, el MI5 tardó años en desvelar que había roto el sistema, dejando durante ese tiempo que empresas y gobiernos pensaran que el cifrado con Enigma seguía siendo seguro.

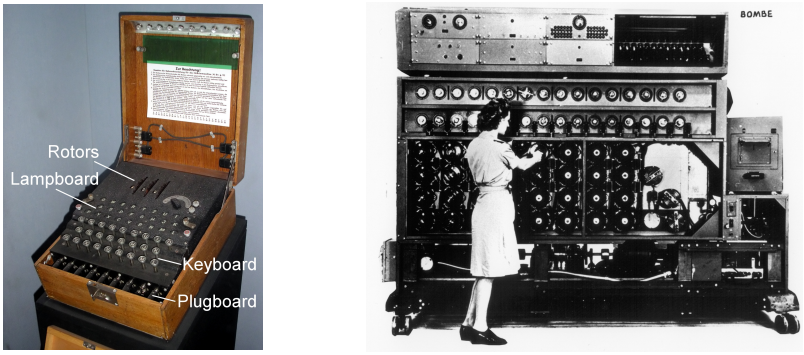

Figura III.5: En la izquierda la máquina Enigma (una de sus versiones) con indicación de sus partes principales. A la derecha, la máquina construida por A. Turing para romper el cifrado en Bletchley Park. Es conocida popularmente como *la bomba de Turing*. (Wikimedia Commons).

Referencias:

- `https://en.wikipedia.org/wiki/Enigma_machine`
- `https://es.wikipedia.org/wiki/Enigma_(maquina)`
- `http://www.kriptopolis.org/enigma`
- `http://russells.freeshell.org/enigma/`
- `http://www.codesandciphers.org.uk/enigma/index.htm`

III.4.B Libreta de un Solo Uso (One-time pad)

Es un sistema criptográfico que no puede ser roto. Su inconveniente es que *para cada comunicación necesita una clave de la misma longitud que el mensaje* y que estas claves han de ser compartidas entre emisor y receptor con anterioridad. En este sentido es un cifrado de Vigenère con longitud de clave infinita y que se usa una única vez. En el lenguaje de Shannon es un *cifrado perfecto*, pues su intercepción no proporciona

ninguna información sobre el texto claro. No obstante, su implementación requiere de un sistema de generación aleatoria de números y pequeños fallos de diseño pueden aportar pistas sobre el criptoanálisis.

- `https://es.wikipedia.org/wiki/Libreta_de_un_solo_uso`

III.4.C Criptosistema de las Mochilas (Knapsack)

Fue uno de los primeros criptosistemas de llave pública y fue inventado por Ralph Merkle y Martin Hellman en 1978. Su seguridad esta basada en un problema NP-completo (véase § §IV.4.A), concretamente el problema de la mochila de Teoría de códigos, y el receptor legítimo conoce una *puerta trasera* que le permite descifrar el mensaje. Fue roto en 1984 y no ofrece funcionalidades para firmar.

- W. Diffie and M. Hellman, «*New directions in cryptography*», en IEEE Transactions on Information Theory, vol. 22, no. 6, págs. 644-654, noviembre 1976.
- R. Merkle and M. Hellman, «*Hiding information and signatures in trapdoor knapsacks*», en IEEE Transactions on Information Theory, vol. 24, no. 5, págs. 525-530, septiembre 1978.
- A. Shamir, «*A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem*», 23rd Annual Symposium on Foundations of Computer Science (1982), Chicago, IL, USA, 1982, págs. 145-152.
- De la Bibliografía, la referencia [7, Capítulo 8, sección 6].
- Fuster, A., «*Técnicas criptográficas de protección de datos*», Editorial Ra-ma.
- Gebbie, S., «*A survey of the mathematics of Cryptology*», Master Thesis, `https://www.gethos.net/thesis/Stewart_Gebbie-Mathematics_of_Cryptography.pdf`

Referencias para este Capítulo

- [1] Capítulos 3 y 5
- [2] Capítulo 2
- [4] Capítulo 3
- [5] Capítulo 3
- [7] Capítulo 7

IV. Complejidad, Factorización y Logaritmo Discreto

En los Capítulos anteriores hemos manejado los conceptos de fortaleza, debilidad, difícil, fácil y otros similares de manera intuitiva y sin el rigor necesario. No obstante, nos ha servido para demostrar la necesidad y la conveniencia de tener una interpretación matemáticamente rigurosa de dichos términos. También hemos visto que siempre que nos aparece un problema o un cálculo en criptografía, hemos de determinar qué algoritmo se va a usar para resolverlo o computarlo.

En este Capítulo se aborda el problema matemático de medir la complejidad de los algoritmos. Es importante destacar que

- la complejidad depende el algoritmo, de la receta que hay que seguir para resolver el problema (p. ej. el algoritmo de Euclides). La complejidad es una característica del algoritmo y no del problema.
- nos ceñimos a una teoría matemática de la complejidad. Llevar a la práctica estos algoritmos (por ejemplo, implementados en *software* o *hardware*, puede añadir complejidad adicional).

El diseño de un sistema criptográfico, teniendo en cuenta las secciones anteriores, suele consistir en un procedimiento tal que los posibles ataques sean equivalentes a problemas muy complejos. Con más precisión, hay que cerciorarse de que cualquier algoritmo capaz de romper el sistema (bien obteniendo sus claves, bien descifrando mensajes) tenga complejidad mayor que cualquier polinomio.

Los dos problemas complejos en los que se basan muchos protocolos criptográficos son la factorización de números y el cálculo del logaritmos discretos.

Observación IV.0.1 Este estudio de la complejidad se podría realizar en varios niveles. El presentado aquí es el más básico. El siguiente paso sería abordar la Teoría

de la Computabilidad, que se ocupa de determinar qué problemas matemáticos son computables (resolubles mediante un algoritmo o, equivalentemente, una máquina de Turing). Por contra, existen problemas que no pueden ser resueltos con algoritmos aún disponiendo de recursos y tiempo ilimitados. Son los problemas irresolubles: problema de decisión (en el ámbito de la lógica de primer orden), problema de parada (si un programa se parará o correrá indefinidamente), etc. Entre los padres de esta teoría cabe citar a Alonzo Church, Kurt Gödel, Stephen Kleene, Emil Leon Post, y Alan Turing. Tras este estudio teórico, se podría considerar un nivel adicional, de carácter práctico, que se ocuparía de cuestiones relacionadas con el *hardware* (tiempos de lectura escritura en memoria, ciclos de reloj de cada instrucción, etc.). No obstante, la complejidad matemática medida en términos de operaciones básicas (véase Definición IV.1.1) es un indicador que estima con gran precisión la complejidad final de un algoritmo.

IV.1 Operaciones y Tiempos de Cómputo

IV.1.A Operación Básica

Definición IV.1.1 Llamaremos operación u operación básica a la consistente en sumar dos bits teniendo en cuenta un posible acarreo.

Por ejemplo, para sumar los números en binario 1100 y 10101

$$\begin{array}{r} {\scriptstyle 1\ 1\ 0\ 0} \\ 1\,1\,0\,0 \\ +\,1\,0\,1\,0\,1 \\ \hline 1\,0\,0\,0\,0\,1 \end{array}$$

hemos realizado 5 operaciones básicas.

Proposición IV.1.2 La suma de dos números de k y l bits necesita menos de $\max\{k,l\}$ operaciones.

Ejercicio IV.1.3 Realizar la resta:

$$110010 - 10110$$

Demostrar que el número de operaciones necesarias para restar un número de k bits otro de l bits ($l \leq k$) es menor o igual que k.

Indicación: Observar que $10110 + (10110 \text{ XOR } 11111) = 100000 - 1$. Por tanto:

$$110010 - 10110 = \big(110010 + (10110 \text{ XOR } 11111) + 1\big) - 100000$$

Si asumimos que la operación lógica XOR, y que la resta de 100000 no consumen tiempo (la última resta siempre es de esa forma, por lo que es sencillamente el cambio del valor de un bit), tendremos que el número de operaciones es el de la suma de 110010 y $(10110 \text{ XOR } 11111)$ que tienen 6 y 5 bits respectivamente.

Veamos un ejemplo de multiplicación. Para multiplicar los números en binario 1010 y 101 hacemos

$$\begin{array}{r} 1\,0\,1\,0 \\ \times\quad 1\,0\,1 \\ \hline 1\,0\,1\,0 \\ 0\,0\,0\,0 \\ 1\,0\,1\,0 \\ \hline 1\,1\,1\,0\,1\,0 \end{array}$$

Por tanto, hacer la multiplicación es hacer la suma de los tres números de la parte inferior. Pero una operación básica no puede sumar tres a la vez (con acarreo), solamente dos (con acarreo). Por ello, hay que sumar primeros los números de las dos primeras filas, y al resultado sumarle la tercera fila.

Veamos también un ejemplo de división. Para dividir 11101 entre 101, desplazamos el menor de los números dados, 101, tantas posiciones a la izquierda como sea posible mientras podamos hacer la diferencia con el número mayor, 11101, esto es

$$\begin{array}{r} 1\,1\,1\,0\,1 \\ -\,1\,0\,1 \\ \hline 1\,0\,0\,1 \end{array}$$

y repetimos hasta que el número resultante sea menor que 101. En este caso, basta con una vez más

$$\begin{array}{r} 1\,0\,0\,1 \\ -\quad 1\,0\,1 \\ \hline 1\,0\,0 \end{array}$$

siendo este último número el resto de la división. El cociente tiene 1 en las posiciones que corresponden a los desplazamientos que hemos podido hacer para las diferencias (en nuestro caso, en la tercera y primera posición), y 0 en las que no se han podido (en el ejemplo, en la segunda). En nuestro caso, hemos concluido que

$$11101 = 101 \cdot 101 + 100$$

A la hora de determinar la complejidad hemos convenido que solamente tendremos en cuenta las operaciones básicas por lo que siempre se supondrá que ni los desplazamientos, ni las comparaciones (dados dos números, determinar si el primero es mayor o igual que el segundo) no consumen ni tiempo ni operaciones.

Proposición IV.1.4 La multiplicación (y la división) de dos números de k y l bits requiere como máximo de $k \cdot l$ operaciones.

Definición IV.1.5 Sean $f, g : \mathbb{N} \to \mathbb{Z}$ dos funciones que toman valores positivos para todo n suficientemente grande. Se dice que $f(n) = O(g(n))$ cuando existe una constante c tal que $f(n) < c \cdot g(n)$ para todo n suficientemente grande.

Dado un algoritmo, diremos que tiene complejidad $O(g(n))$ si $f(n) = O(g(n))$ siendo $f(n)$ el número de operaciones básicas necesarias para completarlo sobre el entero n.

Ejercicio IV.1.6 Probar los siguientes enunciados:

- si $f(n)$ es un polinomio de grado d con el coeficiente de n^d positivo, entonces $f(n) = O(n^d)$;
- más general, si $f_1(n) = O(g_1(n))$ y $f_2(n) = O(g_2(n))$, entonces $f_1(n)f_2(n) = O(g_1(n)g_2(n))$;
- la complejidad de multiplicar dos números de k bits es $O(k^2)$.

Observación IV.1.7 Como principio se asume que aquellos algoritmos con complejidad polinómica son realmente computables (con suficiente tiempo y recursos), aquellos que tengan complejidad $O(2^n)$ (exponencial), $O(n^n)$ y similares, se consideran no computables (en el sentido que, por mucho que podamos incrementar los medios disponibles para su ejecución, basta incrementar ligeramente n para evitar que sea computable en un tiempo razonable). Ver la sección §IV.4.A.

Observación IV.1.8 Para hacerse idea de cómo de grandes son los números, pongamos unos ejemplos:

- $3 \cdot 10^{80}$ es el número de partículas elementales en el universo observable.
- $4 \cdot 10^{17}$ segundos desde el Big Bang. Si lo medimos en la unidad mínima de tiempo (Planck) el número es del orden de 10^{61}.
- el superordenador SUMMIT de IBM alcanzó en 2019 el record de $148 \cdot 10^{15}$ FLOPS (floating point operations per second) de velocidad. Ver §IV.4.B.

IV.1.B Otras Operaciones. Exponenciación Rápida

Hasta ahora hemos asumido que los números están representados en binario. Pero, ¿qué sucede si manejamos números independientemente de su representación? Es decir, nos podemos preguntar, ¿dados dos números (en abstracto) cuántas operaciones se necesitarían para sumarlos, multiplicarlos, etc.?

Proposición IV.1.9 Dado un número natural n, su expresión binaria tiene $[\log_2(n)] + 1$ dígitos.

Demostración. Observar que 2^a se escribe en binario como $10.^a.0$, un 1 seguido de a ceros, que son $a+1$ dígitos. Por tanto hay que buscar el menor a tal que $n < 2^a$, y darse cuenta que esa condición equivale a que n tenga a dígitos binarios. Tomando logaritmos en base 2, $\log_2(n) < a$, y parte entera, podemos escribir $[\log_2(n)] + 1 \leq a$. Recordando que a es el mínimo, se obtiene el resultado.

Corolario IV.1.10 Para realizar la suma de dos números m y n se requiere como máximo máx$\{[\log_2(m)] + 1, [\log_2(n)] + 1\}$ operaciones. La multiplicación de m y n requiere como máximo de $([\log_2(m)] + 1)([\cdot \log_2(n)] + 1)$ operaciones.

Proposición IV.1.11 Dados dos números m y n con $m \geq n$, probar que el cálculo de su m.c.d. por el algoritmo de Euclides tiene complejidad acotada por

$$O\Big(m(\log_2(m))^2\Big).$$

Demostración. Si denotamos $r_0 := m$, $r_1 := n$, entonces hemos de calcular recursivamente las divisiones

$$r_i = r_{i+1}c_i + r_{i+2} \qquad \text{con } 0 \leq r_{i+2} < r_{i+1} \text{ para cada } i \geq 1$$

hasta que terminemos con $r_{i+2} = 0$ y $r_{i+1} \neq 0$. En ese caso, r_{i+1} es el m.c.d.

La complejidad de realizar todas estas divisiones es $\sum \log_2(r_i)\log_2(r_{i+1})$, donde $i \geq 0$ recorre los índices tales que $r_{i+1} \neq 0$. Puesto que $r_{i+1} < r_i$, tenemos $r_{i+1} \leq r_i - 1)$ y, por tanto, acotamos

$$\sum_{i \text{ t.q. } r_{i+1}\neq 0} \log_2(r_i)\log_2(r_{i+1} \leq \sum_{i=0}^{m-2} \log_2(m-i)\log_2(m-(i+1)) \leq m(\log_2(m))^2 .$$

Observación IV.1.12 El tiempo efectivo de cómputo del algoritmo de Euclides depende fuertemente de los números de partida. Probar que el máximo se alcanza cuando los números m, n son términos consecutivos de la sucesión de Fibonacci (que comienza por los términos 0, d con $d \in \mathbb{N}$, continúa con $d, 2d, 3d, 5d, 8d, \ldots$ donde cada término se define como la suma de los dos anteriores). Esta dependencia de los datos iniciales sucede, en general, en todos los algoritmos. Por ello, los algoritmos tendrán siempre una complejidad del peor caso, una complejidad esperada, etc.

Algoritmo de Exponenciación Rápida

Más adelante veremos que en criptografía asimétrica (p. ej. clave pública Diffie-Hellman y RSA) es necesario calcular potencias del tipo g^e en $\mathbb{Z}/m$ siendo g y m números muy grandes. Por ello, es conveniente estudiar la complejidad de este tipo de operaciones y disponer de algoritmos para realizarlas rápidamente. En este sentido, el algoritmo de exponenciación rápida proporciona un método de cálculo muy eficiente.

Para calcular g^e en $\mathbb{Z}/m$, se expresa e en binario $\sum_i e_i 2^i$ (con $0 \leq e_i \leq 1$). Entonces

$$g^e = g^{\sum_i e_i 2^i} = \prod_i (g^{2^i})^{e_i} = \prod_{i|e_i=1} g^{2^i}$$

Observar además que $g^{2^{i+1}} = (g^{2^i})^2$. Es decir, cada término es el cuadrado del anterior. Puesto que se trabaja en $\mathbb{Z}/m$ se considera $g^{2^i} \bmod m$ para cada $i \geq 0$. El siguiente algoritmo se basa en esta estrategia para realizar el cálculo.

Algoritmo de Exponenciación Rápida

Para el cálculo de g^e en $\mathbb{Z}$, $e \geq 0$, con una variable auxiliar $r = 1$:

1. si $e = 0$, entonces devuelve r, FIN.
2. hacemos $r = r \cdot g$ y $g = g^2$.
3. si e es impar, entonces hacemos $e = (e-1)/2$, en otro caso $e = e/2$, y volvemos al paso 1.

Ejercicio IV.1.13 Calcular 6^{73} en $\mathbb{Z}/100$ con el algoritmo de exponenciación rápida.

Solución: En primer lugar, hacemos $g = 6$, $e = 73$ y $r = 1$. Hacemos $r = 1 \cdot 6 = 6$ y $g = 6^2 = 36$. Como e es impar, su nuevo valor es $(73-1)/2 = 36$ y volvemos a empezar. Hacemos $r = r \cdot g = 6 \cdot 36 = 16 \bmod 100$ y $g = 36^2 = 96 \bmod 100$. Como e es par, su nuevo valor es $36/2 = 18$. Hacemos $r = r \cdot g = 16 \cdot 96 = 36 \bmod 100$ y $g = 96^2 = 16 \bmod 100$. Como e es impar, su nuevo valor es $18/2 = 9$. Se continúa sucesivamente y el resultado final es 16. Por otro lado, observar que $73 = 64+8+1$ por lo que $6^{73} = 6^{64} \cdot 6^8 \cdot 6$, que podemos calcular fácilmente computando los cuadrados sucesivos 6, 6^2, $(6^2)^2 = 6^4$, $((6^2)^2)^2 = 6^8, \ldots$

IV.1.C Ejercicios

Problema IV.1.14 Determinar el número de operaciones para realizar los siguientes cálculos:

- las sumas $1100+10101$ y $1100101+1101$;
- los productos 1100×1001 y 110001×1101;
- las divisiones 1100×101 y 110001×101;

Problema IV.1.15 Determinar una cota superior del número de operaciones necesarias para calcular n^m (m un número constante dado).

Solución: El número de bits de la expresión binaria de n^{m-1} es $[\log_2(n^{m-1})]+1 = [(m-1)\log_2(n)]+1$, por lo que el número de operaciones necesarias para multiplicar n y n^{m-1} será menor que $([\log_2(n)]+1)([(m-1)\log_2(n)]+1)$ cuya complejidad es, consecuentemente, $O(\log_2(n)^2)$.

Problema IV.1.16 Determinar una cota superior del número de operaciones necesarias para calcular n^n.

Solución: Lo calcularemos por recursivamente. La primera parte es calcular n^2, que requiere de $([\log_2(n)]+1)^2$ operaciones. Del mismo modo, el cálculo de n^{m-1} por n necesita $([\log_2(n)]+1)([(m-1)\log_2(n)]+1)$ operaciones. El número total de operaciones se obtiene como la suma

$$\sum_{i=1}^{n}\big([\log_2(n)]+1\big)\big([(i-1)\log_2(n)]+1\big)$$

Por tanto su complejidad es

$$O\Big(\sum_{i=1}^{n}\big([\log_2(n)]+1\big)\big([(i-1)\log_2(n)]+1\big)\Big) =$$
$$= O\Big(\binom{n+1}{2}\big(\log_2(n)\big)^2\Big) = O\Big(n^2\big(\log_2(n)\big)^2\Big)$$

Problema IV.1.17 Probar que el número de operaciones necesarias para calcular $n!$ es menor o igual que $n(n-2)([\log_2 n]+1)^2$.

Problema IV.1.18 Determinar una cota superior del número de operaciones necesarias para calcular la expresión decimal de un número binario n.

Problema IV.1.19 Demostrar que una cota superior del número de operaciones necesarias para calcular el número combinatorio $\binom{m}{n}$ es $2m(m-1)([\log_2 n]+1)^2$.

Problema IV.1.20 Determinar una cota superior del número de operaciones necesarias para calcular el producto de dos polinomios de grados m y n.

Problema IV.1.21 Modificar el algoritmo de Euclides del siguiente modo. Dados a,b en vez de escribir $a=bc+r$ donde c es el cociente y $0\le r<b$ es el resto, escribir $a=bc'+r'$ donde $-\frac{b}{2}\le r'<\frac{b}{2}$ (es decir, si $r<\frac{b}{2}$ entonces $c'=c$ y $r'=r$; en otro caso, entonces $a=bc+r=bc+b-b+r=b(c+1)+(r-b)$ y por tanto $c'=c+1$ y $r'=r-b$). Observar que entonces $|r'|\le\frac{b}{2}$.

Estimar el número máximo de divisiones para llevar a cabo el algoritmo de Euclides.

Determinar una cota superior del número de operaciones necesarias para calcular el m.c.d. de dos números usando el algoritmo de Euclides anterior.

Problema IV.1.22 Dar un algoritmo (basado en el algoritmo de Euclides) para factorizar un número en potencias de números primos. Calcular una cota superior para el número de operaciones necesarias.

Problema IV.1.23 Determinar cotas superiores para las operaciones suma, resta, multiplicación y división de dos números en $\mathbb{Z}/N$.

Problema IV.1.24 Determinar una cota superior para el número de operaciones necesarias en el algoritmo de exponenciación rápida.

Problema IV.1.25 Determinar una cota superior para el número de operaciones necesarias para calcular una antimagen de un par (a,b) por la biyección del Teorema Chino de los restos $\mathbb{Z}/mn \xrightarrow{\sim} \mathbb{Z}/m \times \mathbb{Z}/n$.

Problema IV.1.26 Determinar una cota superior para el número de operaciones necesarias para invertir una matriz $n \times n$.

Idem para resolver un sistema de ecuaciones de n ecuaciones con n incognitas.

Idem si todas las operaciones se realizan en $\mathbb{Z}/N$.

IV.2 Factorización

A continuación veremos que obtener la factorización de un número es muy complejo pero que, sin que suponga ninguna contradicción, determinar si un número es primo o compuesto es se puede hacer en tiempo polinómico.

Asumiremos el siguiente resultado sobre la distribución de los números primos.

Teorema IV.2.1 Para cada $n \in \mathbb{N}$, sea $\pi(n)$ el número de números primos menores que n, es decir,

$$\pi(n) := \#\{x \in \mathbb{N} \text{ t.q. } x < n \text{ y } x \text{ primo}\}.$$

Entonces se cumple que

$$\lim_{n\to\infty} \frac{\pi(n)}{n/\ln n} = 1.$$

En particular, el n-ésimo número primo es *aproximadamente* $n \cdot \ln n$ (podemos pensar que está en un entorno de este número). Aquí, ln denota el logaritmo neperiano.

Ejercicio IV.2.2 Calcular la probabilidad de que un número de k bits sea primo ($k \gg 0$).

Solución: Observemos que hay $2^k - 2^{k-1}$ números con exactamente k bits. Para k muy grande, el número de primos entre 2^{k-1} y $2^k - 1$ tiende a $2^k \frac{1}{\ln 2^k} - 2^{k-1} \frac{1}{\ln 2^{k-1}}$. Por tanto, la probabilidad buscada es

$$\frac{1}{\ln 2} \frac{\frac{1}{k}2^k - \frac{1}{k-1}2^{k-1}}{2^k - 2^{k-1}} = \frac{k-2}{k(k-1)\ln 2}.$$

IV.2.A Factorización (fuerza bruta)

Imaginemos que queremos factorizar un número n, es decir, no nos basta con saber si es primo o compuesto, queremos expresar n como producto de potencias de números primos. Para ello, no hay más remedio que ir tomando los primos de una lista y viendo si dividen o no a n. Acotemos *grosso modo* cuántas operaciones lleva únicamente esa comprobación. Si p_i es el i-ésimo número primo, al menos tendremos que hacer $\sum_i([\log_2(n)]+1)([\log_2(p_i)]+1)$ operaciones donde la suma recorre los primos p_i con $p_i \leq \sqrt{n}$. Por el Teorema anterior, el número de operaciones será mayor que

$$\log_2(n)\sum_{i=1}^{n/\ln n}[\log_2(p_i)] \sim \log_2(n)\sum_{i=1}^{n/\ln n}[\log_2(i\ln i)] = \log_2(n)\sum_{i=1}^{n/\ln n}[\log_2(i)+\log_2(\ln i)]$$

y, observando que para cada $j \in \mathbb{N}$ existen exactamente 2^j números naturales a verificando $[\log_2(a)] = j$ (i. e. números a con exactamente $j+1$ dígitos binarios en base 2). Entonces la suma anterior es mayor que

$$\log_2(n)\cdot\sum_{j=1}^{[\log_2(n/\ln n)]-1} 2^j = \log_2(n)\cdot\left(2^{[\log_2(n/\ln n)]}-1\right) \sim n$$

Intuitivamente podemos hablar del tamaño de un número como su número de bits, es decir la parte entera de su logaritmo en base 2. Por tanto, hemos demostrado

Teorema IV.2.3 La complejidad de factorizar un número de k bits por el algoritmo de fuerza bruta (recorriendo todos los primos) es $O(2^k)$.

Para ilustrar este resultado, si la complejidad de factorizar un número con $k = 1024$ bits (tamaño estándar en el cifrado RSA) necesitaríamos una tabla con los $[2^k/\ln(2^k)]+1 \sim 1{,}7\cdot 10^{305}$ primeros números primos, y realizar unas $2^{1024} \sim 1{,}8\cdot 10^{308}$ operaciones básicas. Son números que desbordan absolutamente nuestra capacidad (ver Observación IV.1.8).

Observación IV.2.4 Recordemos que si tenemos una factorización de n, $n = p_1^{n_1}\cdot\ldots\cdot p_r^{n_r}$, entonces se tiene una expresión (aunque no sea una factorización) de $\phi(n)$ pues conocemos la fórmula $\phi(n) = \prod_j (p_j-1)p_j^{n_j-1}$. Por otro lado, si $\phi(n) = q_1^{n_1}\cdot\ldots\cdot q_s^{n_s}$, entonces probando sucesivamente a dividir n entre $q_1,\ldots,q_s$, obtendríamos una descomposición de n.

IV.2.B Factorización de Fermat

Proposición IV.2.5 Sea n un número natural impar. Hay una correspondencia biyectiva entre modos de expresar n como diferencia de cuadrados y modos de expresar n como producto de dos números.

Demostración. Sea n el número impar dado. Si tenemos una expresión $n = x^2-y^2$, resulta una factorización $n = (x+y)(x-y)$. Recíprocamente, si tenemos $n = a\cdot b$, entonces planteamos el sistema $x+y = a, x-y = b$, cuya solución es $x = \frac{1}{2}(a+b)$, $y = \frac{1}{2}(a-b)$.

Esta proposición, da lugar al algoritmo de factorización de Fermat, que describimos a continuación. El algoritmo concluye con una descomposición como producto de dos

factores, no necesariamente primos. La notación $\lceil x \rceil$ representa el menor entero mayor o igual a x.

Algoritmo de factorización de Fermat

Dado un número natural n, se procede como sigue:

1. tomamos una variable auxiliar $i = 0$;
2. si $\sqrt{n}$ es entero, ya está factorizado, se termina;
3. se incrementa i y se calcula $\sqrt{(\lceil \sqrt{n} \rceil + i)^2 - n}$;
4. si es un entero m, se tendrá $n = (\lceil \sqrt{n} \rceil + i)^2 - m^2$ y por tanto factoriza (ver Proposición IV.2.5);
5. si no, se vuelve al punto 3.

El algoritmo concluye con una descomposición como producto de dos factores, no necesariamente primos.

Proposición IV.2.6 Este método es efectivo cuando los factores son de tamaño similar.

Demostración. Tras aplicar el método de Fermat, tenemos m, i verificando $n = (\lceil \sqrt{n} \rceil + i)^2 - m^2$. Entonces la factorización obtenida, $n = a \cdot b$, es

$$a = \lceil \sqrt{n} \rceil + i + m \qquad b = \lceil \sqrt{n} \rceil + i - m$$

Teniendo en cuenta $i = 0, 1, 2, \ldots$, $\log_2 m \approx \frac{1}{4} \log_2(n)$ y $\log_2(\lceil \sqrt{n} \rceil) \approx \frac{1}{2} \log_2 n$, se concluye que a, b tienen un tamaño similar entre sí (de hecho, similar a $\frac{1}{2} \log_2 n$. El recíproco, es análogo por este argumento junto con la Proposición IV.2.5.

Supongamos que hemos aplicado la factorización de Fermat infructuosamente. Entonces podemos modificar ligeramente el proceso de Fermat e intentarlo de nuevo. Veamos cómo.

Algoritmo de factorización de Fermat generalizado

Dado un número grande n. Se elige un k pequeño, $k = 1, 2, \ldots$, y se procede así:

1. se aplica la factorización de Fermat a $k \cdot n$;
2. si no se consigue una factorización, entonces se elige otro k y se vuelve al punto 1;
3. supongamos que se ha factorizado $k \cdot n = x^2 - y^2$. Sabemos que los factores son $x + y$, $x - y$ con tamaño similar a $\sqrt{k \cdot n}$ que es mucho mayor que k y que k es pequeño;
4. calculamos $\mathrm{mcd}(n, x + y)$ y $\mathrm{mcd}(n, x - y)$ por el algoritmo de Euclides, obteniendo divisores de n.

Discutamos brevemente el último paso. Cuando se llega a este paso, se ha obtenido una expresión del tipo $k \cdot n = x^2 - y^2 = (x + y)(x - y)$. Si se tuviera $\mathrm{mcd}(n, x + y) = 1$, entonces $(x + y) | k$, lo cual es imposible por los tamaños de los números. De donde se concluye que necesariamente $\mathrm{mcd}(n, x + y) \neq 1$ y, por tanto, es un divisor de n.

IV.2.C Factorización ρ de Pollard

Paradoja del cumpleaños

Si tenemos un grupo de 23 o más personas, entonces la probabilidad de que al menos 2 cumplan años el mismo día es mayor del 50 % .

Se llama paradoja porque 23 parece pequeño para la intuición. Pero no hay ninguna contradicción a nivel matemático, y el hecho se basa en que no fijamos a priori la fecha de cumpleaños, solamente afirmamos que hay dos personas con la misma fecha.

La paradoja anterior es consecuencia del siguiente

Teorema IV.2.7 Si de un conjunto con n elementos se toman d elementos (con repetición, cada vez que se toma uno, se repone antes de tomar el siguiente), entonces la probabilidad de tomar dos veces el mismo es aproximadamente $1-\exp(-\frac{d(d-1)}{2n})$. En particular, para tener una probabilidad mayor que $\frac{1}{2}$, basta tomar al menos

$$\left\lceil \frac{1}{2}(1+\sqrt{1+8n\operatorname{Ln}(2)}\right\rceil \approx 1{,}17741\sqrt{n} \tag{IV.1}$$

elementos (el lado derecho es una aproximación para $n \gg 0$).

Demostración. La probabilidad de que en cada elección (con repetición) elijamos elementos distintos es $\prod_{i=1}^{d}(1-\frac{i-1}{n})$. Teniendo en cuenta $\exp(\alpha) > 1+\alpha$ (observar que para $n \gg d$ es una buena aproximación), resulta que la probabilidad de elegir elementos distintos es menor que $\prod_{i=1}^{d}\exp(-\frac{i-1}{n}) = \exp(-\frac{d(d-1)}{2n})$.

Igualando este número a $1/2$, tomando logaritmos y despejando d en función de n, se obtiene la segunda parte.

Observación IV.2.8 En general si tenemos una función $H : X \to Y$ y se toman k elementos en X tales que los valores de H estén distribuidos uniformemente, entonces la probabilidad de que dos valores coincidan es mayor que $1/2$ si se cumple que k es mayor que (IV.1).
Por ejemplo, si $Y = \{0,1\}^n$, entonces con $2^{n/2}$ evaluaciones de la función H se encontrará una pareja x, x' con $H(x) = H(x')$ con probabilidad mayor que $1/2$. En la actualidad y para evitar estos ataques, se suele tomar $n \geq 160$.

La moraleja es que las coincidencias son «fáciles» de conseguir: obtener una coincidencia (*colisión*) tiene que ver con la raíz cuadrada del número. Por ejemplo, dado n, buscar una colisión en este contexto consiste en determinar dos números con el mismo resto módulo n (misma clase en $\mathbb{Z}/n$) o, equivalentemente, a haber elegido dos veces la misma clase de equivalencia en $\mathbb{Z}/n$. En este principio se basan muchos algoritmos, como el de Pollard para factorizar números enteros así como otros para atacar criptosistemas.

ALGORITMO DE FACTORIZACIÓN ρ DE POLLARD

Supongamos que queremos factorizar n. Elegimos una función cualquiera $f(x)$ y la iteramos mod n, obteniendo una sucesión

$$a_1 = 1, a_2 = f(1), a_3 = f(f(1)), \ldots, a_i = f(a_{i-1}), \ldots$$

y se procede así

1. se calculan a_1, a_2. Si $(a_2 - a_1, n) \neq 1$, entonces se acaba. En caso contrario, guardamos a_1 y a_2 y proseguimos.
2. en general, en la iteración i-éxima, se calculan a_i, a_{2i-1}, a_{2i}.
3. si $(a_{2i} - a_i, n) \neq 1$, entonces se acaba. Si no, se guardan a_i, a_{2i} en memoria. Se borra a_{i-1} de la memoria.

Observar que su complejidad depende de la complejidad de la función f elegida, en el caso anterior es $O(\sqrt[4]{n})$ o, expresando en términos del tamaño del número (número de bits), $O(2^{k/4})$. Este es el mejor método para algunos números pequeños.

Observación IV.2.9 El tiempo esperado para encontrar el menor factor p de un número compuesto n es del orden de $(\log(n))^2\sqrt{p}$. El algoritmo original estudiaba el máximo común divisor de $a_i - a_j$ y n, mientras que la variación que se presenta aquí se debe a Floyd. Pollard también describió otro método de factorización conocido como $p-1$ así como un algoritmo de cálculo del logaritmo discreto.

Ejercicio IV.2.10 Factorizar $n = 1357$ con el algoritmo de Pollard.

Solución: Tomamos una función que sea fácil de evaluar, por ejemplo, $f(x) = x^2 + 1$ mod n y $a_1 = 1$. Empezamos a calcular $a_i = f(a_{i-1})$. En primer lugar, La siguiente etapa consiste en

1. $a_1 = 1, a_2 = 2$, luego $(a_2 - a_1, n) = 1$ y tenemos que continuar.
2. $a_3 = f(a_2) = 4, a_4 = f(a_3) = 26$. Como $(a_4 - a_2, n) = 1$, seguimos.
3. continuamos hasta $a_6 = 1021, a_{12} = 906$. En este caso, $(a_{12} - a_6, n) = 23$.

Un modo de entender cómo funciona este algoritmo aplicado a este ejemplo, es el siguiente. Ahora que sabemos que $1357 = 23 \times 59$, entonces $\mathbb{Z}/1357 \simeq \mathbb{Z}/23 \times \mathbb{Z}/59$ por lo que, por el Teorema Chino de los Restos, podemos hacer los cálculos en $\mathbb{Z}/23 \times \mathbb{Z}/59$. Escribe la tabla de restos de $a_1, a_2, \ldots$ en $\mathbb{Z}/23 \times \mathbb{Z}/59$ y observa que $a_6 = a_{12} = 9$ en $\mathbb{Z}/23$ y $a_6 = 18$, $a_{12} = 21$ en $\mathbb{Z}/59$. Por tanto, $a_{12} - a_6 = (0,3) \in \mathbb{Z}/23 \times \mathbb{Z}/59$, y esta es la razón por la que podemos calcular un divisor.

IV.2.D Base de Factores: Algoritmo de Dixon

Hemos visto en el Teorema IV.2.3 que disponer de una lista de números primos suficiente para factorizar números de cierto tamaño es inviable. No obstante, con una lista «pequeña» es posible diseñar un algoritmo de factorización: el algoritmo de Dixon de factorización por base de factores. Veamos cómo funciona.

Algoritmo de Factorización de Dixon

- Consideramos una lista de primos $\{p_1,\dots,p_k\}$ (por ejemplo, los cien primeros números primos). Sea n el número que queremos factorizar. Sea una variable auxiliar $i=0$.
- Calculamos $\left(\lceil\sqrt{n}\rceil+i\right)^2$. Si los únicos factores primos de ese número están en la lista, calculamos los exponentes $\alpha_{i,j}$ tal que

$$\left(\lceil\sqrt{n}\rceil+i\right)^2 = p_1^{\alpha_{i,1}}\cdot\ldots\cdot p_k^{\alpha i,k} \mod n. \tag{IV.2}$$

- Incrementamos i y volvemos al punto anterior (hasta que i alcance un número predeterminado).
- Para cada i en el que hayamos hecho la descomposición (IV.2), consideremos el vector $\alpha_i := (\overline{\alpha}_{i,1},\dots,\overline{\alpha}_{i,k}) \in (\mathbb{Z}/2)^k$ formados por las clases de los exponentes módulo 2.
- Buscamos una combinación lineal $\sum_{i\in I}\lambda_i\alpha_i = 0 \mod 2$. Donde I es el conjuntos de i tales que hemos considerado la descomposición (IV.2). Esto se puede hacer, por ejemplo, por el método de eliminación de Gauss.
- Sea x el resto de dividir $\prod_{\substack{i\in I\\ \lambda_i=1}}\left(\lceil\sqrt{n}\rceil+i\right)$ entre n. Sea y el resto de dividir $x^2=\prod_{j=1}^k p_j^{\sum_i \lambda_i\alpha_{i,j}}$ entre n que, como todos los exponentes son pares, es un cuadrado. Sea z el resto de dividir $\prod_{j=1}^k p_j^{\frac{1}{2}\sum_i \lambda_i\alpha_{i,j}}$ entre n. Tenemos la identidad

$$x^2 = y = z^2 \mod n.$$

- Calculando $(x\pm z,n)$, por el algoritmo de Euclides, podemos obtener divisores de n.

Observación IV.2.11

- Tiene una complejidad $O\left(\exp\left(2\sqrt{2}\sqrt{\log n\cdot\log(\log(n))}\right)\right)$.
- Además, por Dickman-de Bruijn, sabemos que la probabilidad de que un número tenga todos sus factores menores que $n^{1/a}$ es aproximadamente a^{-a}, de donde resulta que la tamaño óptimo de la base de factores es una potencia de $\exp\left(\sqrt{\log(n)\log(\log(n))}\right)$.
- La criba cuadrática (*quadratic sieve*) es una optimización del algoritmo de Dixon.

Ejercicio IV.2.12 Factorizar $n=6157$ con el algoritmo de Dixon para la base de factores $2,3,5,7,11$.

Solución: Tenemos $\lceil\sqrt{6157}\rceil = 79$. Además, $79^2 = 84 \mod n$ y $84 = 2^2\cdot 3\cdot 7$; $80^2 = 243 \mod n$ y $243 = 3^5$; $81^2 = 404 \mod n$ y 404 no descompone como producto de potencias de los primos de la base; $82^2 = 567 \mod n$ y $567 = 3^4\cdot 7$. Observamos que una combinación lineal nula es

$$1\cdot(\overline{0},\overline{1},\overline{0},\overline{1},\overline{0}) + 1\cdot(\overline{0},\overline{1},\overline{0},\overline{0},\overline{0}) + 1\cdot(\overline{0},\overline{0},\overline{0},\overline{1},\overline{0})$$

(los vectores corresponden a 79^2, 80^2 y 82^4 y las componentes a $2,3,5,7,11$). El resto de dividir $79\cdot 80\cdot 82$ entre n es $x=1052$. El resto de dividir $\prod_{j=1}^k p_j^{\frac{1}{2}\sum_i \lambda_i\alpha_{i,j}} = \left((2^2\cdot 3\cdot\right.$

$7)\cdot 3^5\cdot(3^4\cdot 7))^{1/2} = 2\cdot 3^5\cdot 7$ entre n es $z = 3402$. Finalmente, $(3402-1052,n) = 47$, resulta que 47 es un divisor de n. Efectivamente $6157 = 47\cdot 131$.

IV.3 Logaritmo Discreto

IV.3.A Definición

Definición IV.3.1 — grupo cíclico y generadores. Sea G un grupo (notación multiplicativa). Un elemento $g \in G$ se dice que es un generador cuando $G = \langle g\rangle = \{\ldots, g^{-2}, g^{-1}, g^0 = 1, g^1 = g, g^2, \ldots, g^i, \ldots\}$.

Un grupo que admita elemento generadores se denomina grupo cíclico.

Ser cíclico equivale a que existe una epimorfismo $\mathbb{Z} \to G$. Todos los grupos aditivos $\mathbb{Z}/n$ son cíclicos, pero de los grupos multiplicativos $(\mathbb{Z}/n)^*$, ¿cuáles son cíclicos?

Teorema IV.3.2 Dado un número natural n, el grupo $(\mathbb{Z}/n)^* = \{b \in \mathbb{Z}/n \text{ t.q. } (b,n) = 1\}$ no es cíclico en general. Probar que es cíclico si y solo si $n = 1, 2, 4, p^k, 2p^k$ donde p es un primo impar y $k \geq 1$.

Definición IV.3.3 Sea p un número primo y sea g un generador del grupo cíclico $(\mathbb{Z}/p)^*$ (que sabemos que es cíclico).

Sea $b \in (\mathbb{Z}/p)^*$. El logaritmo (discreto) de b se define como el menor número natural i tal que $b = g^i \bmod p$.

En particular, se puede generalizar la noción de logaritmo discreto a $(\mathbb{Z}/n)^*$ con $n = 1, 2, 4, p^k, 2p^k$.

Ejercicio IV.3.4 Calcular generadores de $(\mathbb{Z}/p)^*$ para $p = 5, 7$.

Solución: Hagamos el caso de $(\mathbb{Z}/5)^*$ y procederemos por fuerza bruta: tomamos elementos y calculamos qué subgrupo generan. Puesto que el indicador de Euler es $\phi(5) = 5-1 = 4$, calcularemos las potencias $0, 1, \ldots, \phi(5)-1$ (recuérdese también el pequeño Teorema de Fermat, Corolario V.1.8). Por ejemplo, $\langle 1\rangle = \{1\}$, $\langle 2\rangle = \{2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 3\}$, $\langle 3\rangle = \{3^0 = 1, 3^1 = 3, 3^2 = 4, 3^3 = 2\}$, $\langle 4\rangle = \{4^0 = 1, 4^1 = 4, 4^2 = 1, 4^3 = 2\}$. Por lo que los únicos generadores son 2 y 3. Nótese que el inverso de 2 es 3.

IV.3.B Algoritmos

Una forma simple de calcular el logaritmo discreto es por fuerza bruta. Se calculan sucesivamente las potencias del generador $g, g^2, g^3, \ldots$ hasta que consigamos un i con $g^i = b$. La complejidad de este algoritmo (usando exponenciación rápida) es polinómica en p y, por tanto, exponencial en el número de bits de p. Por ello, se asume que un criptosistema cuya seguridad radique en el problema del logaritmo discreto es seguro.

Debemos matizar la afirmación anterior. El sistema de fuerza bruta sería un algoritmo válido para calcular cualquier logaritmo y es, por tanto, un algoritmo de propósito general. No obstante, hay algoritmos que, si bien no son eficientes para cualquier entrada, sí lo son para cierto tipo de números concretos, son los denominados algoritmos de propósito especial. Entonces, tendremos que elegir las claves de nuestro sistema de forma que se eviten este tipo de números. Discutiremos a continuación alguno de ellos. La computación cuántica podría cambiar este panorama.

Consideremos n con $(\mathbb{Z}/n)^*$ cíclico (recuérdese el Teorema IV.3.2). Por ejemplo, si n puede ser factorizado por algún método, digamos $n = p_1^{n_1} \cdot \ldots \cdot p_r^{n_r}$ con p_i primos distintos dos a dos, entonces el Teorema Chino de los Restos nos da un isomorfismo de anillos

$$\mathbb{Z}/n \simeq \mathbb{Z}/p_1^{n_1} \times \ldots \times \mathbb{Z}/p_r^{n_r}$$

(solo necesitamos el algoritmo de Euclides para tener una expresión explícita del isomorfismo) y, por tanto, un isomorfismo entre los grupos de invertibles

$$\left(\mathbb{Z}/n\right)^* = \mathbb{Z}/\phi(n) \simeq \left(\mathbb{Z}/p_1^{n_1}\right)^* \times \ldots \times \left(\mathbb{Z}/p_r^{n_r}\right)^*$$

En el caso en que todos los factores $(\mathbb{Z}/p_1^{n_1})^*$ sean cíclicos, el generador g da lugar a generadores en cada factor, g_j, al igual que el elemento $b \mapsto (b_1, \ldots, b_r)$ y el problema es equivalente a calcular los logaritmos discretos en cada factor. El siguiente algoritmo es un algoritmo de propósito general que se basa en estos hechos.

Algoritmo «Paso de bebé-paso de gigante» de Shanks

Sea g un generador de $(\mathbb{Z}/n)^*$ y sea $a \in (\mathbb{Z}/n)^*$. Queremos calcular $\log_g a$.

1. Sea $m = [\sqrt{n}]$
2. Construimos una tabla con los pares $(j, a \cdot g^{-j})$ para $j = 0, 1, \ldots, m$.
3. Consideramos una variable auxiliar i con valor inicial $i = 0$ (un contador).
4. Definimos $c = g^{im}$.
5. Si c aparece en la tabla, es decir, $c = ag^{-j}$ para un j, entonces el logaritmo es $i \cdot m + j$.
6. Si c no aparece, entonces incrementamos i y volvemos al punto 4.

Para el caso en que el orden del grupo cíclico sea un número compuesto, el siguiente algoritmo es más eficiente. Si los divisores primos del orden del grupo son números pequeños, este algoritmo es muy eficiente. Es, por tanto, un algoritmo de propósito especial y se debe a Pohlig-Hellman.

Algoritmo de Pohlig-Hellman (I)

Sea g un generador de un grupo G de orden $n = p^e$ con p primo. Sea $h \in G$ y queremos calcular $\log_g h$.

1. Tomamos una variable auxiliar $x_0 := 0$.
2. Calculamos $\gamma := g^{p^{e-1}}$ que tiene orden p.
3. Para cada $k \in \{1, \ldots, e-1\}$, hacemos lo siguiente
   - calculamos $h_k := (g^{-x_k} h)^{p^{e-1-k}}$, observemos que el orden de este elemento divide a p, por lo que $h_k \in \langle \gamma \rangle$.
   - utilizando el algoritmo de Shanks, determinamos $d_k \in \{0, \ldots, p-1\}$ tal que $\gamma^{d_k} = h_k$.
   - hacemos $x_{k+1} := x_k + p^k d_k$.
4. Resolvemos el sistema de ecuaciones $x = x_i \bmod p_i^{e_i}$ para $i \in \{1, \ldots, r\}$. El Teorema Chino de los restos asegura que la solución x es única módulo n.
5. Se cumple que $x_e = \log_g h$.

Problema IV.3.5 Demostrar que si g es un generador e i verifica $(i, p-1) = 1$, entonces g^i es otro generador.

Problema IV.3.6 Comprobar que $\log_2 5 = 24 \bmod 101$, esto es, $2^{24} = 5 \bmod 101$.

Algoritmo de Pohlig-Hellman (II)

Sea g un generador de un grupo G de orden n, supongamos que n tiene una descomposición conocida $n = \prod_{i=1}^{r} p_i^{e_i}$. Sea $h \in G$ y queremos calcular $\log_g h$.

1. Para cada $i \in \{1, \ldots, r\}$, hacemos lo siguiente
   - calculamos $g_i := g^{n/p_i^{e_i}}$.
   - calculamos $h_i := h^{n/p_i^{e_i}}$, observemos que $h_i \in \langle g_i \rangle$.
   - utilizando el algoritmo anterior sobre el grupo $\langle g_i \rangle$, determinamos $x_i \in \{0, \ldots, p_i^{e_i} - 1\}$ tal que $g_i^{x_i} = h_i$.
2. Resolvemos el sistema de ecuaciones $x = x_i \bmod p_i^{e_i}$ para $i \in \{1, \ldots, r\}$. El Teorema Chino de los restos asegura que la solución x es única módulo n.
3. Se cumple que $x = \log_g h$.

¿Qué podemos hacer para evitar esta situación? Tendremos que tomar un n que sea difícil factorizar, por ejemplo, que tenga pocos factores, muy grandes y de tamaños distintos. Como también sabemos que una factorización de $\phi(n)$ puede dar lugar a una factorización de n, hay que poner las mismas condiciones en $\phi(n)$. En el caso $n = p$ primo, la elección pasa por tomarlo tal que $\frac{p-1}{2}$ sea primo. Hay que indicar que no es difícil conseguir números primos p (construidos aleatoriamente) tales que $\frac{p-1}{2}$ sea primo.

La complejidad de calcular el logaritmo discreto en un grupo de orden n por fuerza bruta es $O(n)$ (es decir, exponencial en el número de cifras de n), mientras que el algoritmo de Shanks la reduce a $O(\sqrt{n})$. El peor caso del algoritmo de Pohlig-Hellman tiene complejidad igual a la del algoritmo de Shanks, pero en general, para $n = \prod_{i=1}^{r} p_i^{e_i}$ la complejidad es $O(\sum_i e_i(\log n + \sqrt{p_i}))$ por lo que es muy eficiente si los factores primos de n son pequeños.

Por ejemplo: por fuerza bruta enumerando todas las potencias, el cálculo del logaritmo de 3 en base 5 en $\mathbb{Z}/2017$ implica la realización de 1029 multiplicaciones módulo 2017. Con el algoritmo de Shanks basta con 66 (aunque se requiere mayor uso de memoria). Ver `https://maths-people.anu.edu.au/~brent/pd/rpb231.pdf`.

Para concluir, podemos citar el algoritmo de Pollard para el PLD (problema del logaritmo discreto) basado en encontrar colisiones; concretamente para calcular el logaritmo de $h \in (\mathbb{Z}/n)^*$ con base g, buscamos pares (a,b) y (a',b') tales que $g^a h^b = g^{a'} h^{b'}$. De ser así, tendremos $a - a' = x(b' - b) \bmod \phi(n)$ donde x denota el logaritmo buscado. Si $b' - b$ es coprimo con $\phi(n)$, entonces despejamos x. La complejidad es $O(\sqrt{n})$.

Algoritmo ρ de Pollard para el PLD

Sea g un generador de un grupo G de orden n. Sea $h \in G$ y queremos calcular $x := \log_g h$.

1. Se expresa G como una unión disjunta de tres subconjuntos aproximadamente del mismo tamaño $G = S_1 \cup S_2 \cup S_3$.
2. para $k = 0, 1, 2, \ldots$ se calculan las ternas $(x_k, a_k, b_k) \in G \times \mathbb{Z}/n \times \mathbb{Z}/n$ del siguiente modo $(x_0, a_0, b_0) := (e_G, 0, 0)$ y en general

$$(x_{k+1}, a_{k+1}, b_{k+1}) := \begin{cases} (hx_k, a_k, b_k + 1) & \text{si } x_k \in S_1 \\ (x_k^2, 2a_k, 2b_k) & \text{si } x_k \in S_2 \\ (gx_k, a_k + 1, b_k) & \text{si } x_k \in S_3 \end{cases}$$

   y observar que se cumple $x_k = g^{a_k} h^{b_k}$.
3. utilizando la misma idea que en el Algoritmo IV.2.C, buscamos una colisión entre las x_k; esto es dos enteros $i \neq j$ tales que $x_i = x_j$. Lo hacemos comparando x_{2k} con x_k en la k-ésima iteración.
4. si encontramos esa colisión, $x_i = x_j$, y además $b_j - b_i$ es coprimo con n, entonces $g^{a_i} h^{b_i} = g^{a_j} h^{b_j}$, de donde $a_i - a_j = x(b_j - b_i) \bmod n$ y podemos determinar x.
5. en otro caso, incrementamos k y seguimos buscando colisiones.

IV.4 Temas para ampliar

IV.4.A P y NP

Hemos visto que la teoría de la complejidad computacional es el estudio de los recursos mínimos necesarios para resolver problemas computacionales y que el recurso más básico es el tiempo de ejecución de un modelo basado en máquinas de Turing, aunque la forma más simple se basa en las operaciones básicas (ver Definición IV.1.1).

Un algoritmo se considera eficiente si su tiempo de ejecución crece de forma polinómica con la longitud de entrada y, en caso contrario, se considera que el algoritmo es inviable. Se dice que un problema está en la clase **P** si admite un algoritmo eficiente. Se cree que el problema de factorización de enteros no está en la clase **P** y tanto el cifrado RSA como la firma RSA así como muchos otros criptosistemas de clave pública se basan en esta conjetura.

Otra clase de complejidad importante es la denominada **NP**. Digamos que es la clase de todos los problemas computacionales cuyas soluciones se pueden verificar en tiempo polinomial. Por ejemplo, la verificación de que una factorización es correcta es un problema en **P** puesto que se trata de hacer una multiplicación, por tanto factorizar un número es un problema en **NP**. En cierto modo, uno desea considerar problemas para los cuales pueda reconocer cuándo ha obtenido una solución. Es evidente que $\mathbf{P} \subseteq \mathbf{NP}$ y se cree que $\mathbf{P} \neq \mathbf{NP}$. Esta última creencia es necesaria en criptografía, aunque no suficiente, ya que $\mathbf{P} \neq \mathbf{NP}$ se refiere a la complejidad del peor de los casos.

En cualquier caso, a la luz de las secciones anteriores ha quedado clara la importancia de estudiar y diseñar algoritmos para los problemas de factorización y del logaritmo discreto. Con cada nuevo algoritmo, bien de propósito general o bien de

propósito especial, se debe estudiar su complejidad así como las situaciones en las que se vuelve especialmente eficiente. De este modo, aparecen ciertas condiciones sobre los parámetros de partida (orden del grupo, etc) que se deben evitar. Como moraleja, hay que mantenerse al día respecto de la publicación de nuevos algoritmos. Otros algoritmos complementarios a los vistos anteriormente son: la factorización por fracciones continuas, factorización con curvas elípticas, criba con cuerpos de números, etc.

- Capítulo 2, Sección 4 de [6]
- Vadhan, S., «*Computational Complexity*», en Encyclopedia of Cryptography and Security (2011), págs. 235-240.
- Schaefer, E., «*An introduction to cryptography*», `http://cgi.di.uoa.gr/~halatsis/Crypto/Bibliografia/Crypto_Lectures/Schaefer_intro_crypto.pdf`
- `https://en.wikipedia.org/wiki/Time_complexity`.

IV.4.B Superordenadores y Potencia de Cálculo

El superordenador Summit fue presentado por el Laboratorio Nacional de Oak Ridge del Departamento de Energía de EE.UU. el 8 de junio de 2018. Con un rendimiento máximo de 200.000 billones de cálculos por segundo (esto es 10^{17}), o 200 petaflops, Summit fue el más potente del momento (ocho veces más potente que el sistema anterior de ORNL, Titan). Summit realiza más de tres mil millones de millones de cálculos de precisión mixta por segundo, o 3,3 exaops. En otras palabras, el IBM Summit tiene una potencia de cálculo equivalente a 100.000 consolas PS4. El peso total del Summit alcanza los 340.000 kilos, ocupa 520 metros cuadrados y su sistema de refrigeración bombea 15.000 litros de agua por minuto. Ver también `https://computerhoy.com/reportajes/tecnologia/ibm-summit-superordenador-mas-potente-del-mundo-627963`

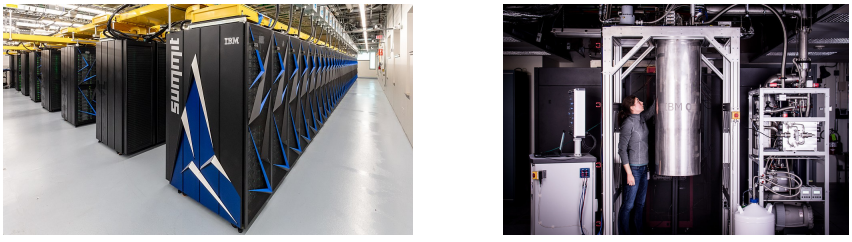

Figura IV.1: En la fotografía de la izquierda, el superordenador Summit (Wikimedia Commons). Al lado derecho, Katie Pooley, integradora de procesos en el equipo Q de IBM, examina un criostato con el nuevo prototipo de un procesador cuántico comercial en su interior. (Crédito: Andy Aaron, IBM).

IV.4.C Criptografía Cuántica y Ordenadores Cuánticos

Los algoritmos clásicos, como los que hemos visto en este libro, están pensados para operar bajo un determinado tipo de reglas, son las reglas que responden a la computación clásica, a los ordenadores actuales o, desde otra perspectiva, a las máquinas de Turing. No obstante, basándose en las matemáticas propias de la física cuántica, como pueden ser los espacios de Hilbert y la transformada de Fourier, se pueden

plantear otro tipo de algoritmos que requerirían para su ejecución de un nuevo tipo de computación. Es la denominada computación cuántica y serían los ordenadores cuánticos los adecuados para este tipo de programas.

El algoritmo clásico conocido más eficiente para la factorización de números es el criba de cuerpos de números cuya complejidad es subexponencial, concretamente

$$O(\exp(1{,}9(\log N)^{1/3}(\log\log N)^{2/3}))\,.$$

Por contra, el algoritmo cuántico de Shor para la factorización se ejecuta en tiempo polinomial en $\log N$. Actualmente y si bien es cierto que apenas hay computadores cuánticos, es una realidad que la aparición de la computación cuántica ha desbaratado la seguridad de los sistemas criptográficos basados en la factorización, en el logaritmo discreto y en el logaritmo discreto para curvas elípticas.

Los protocolos criptográficos diseñados para resistir ataques basados en computación cuántica se agrupan bajo el nombre genérico de criptografía post-cuántica y son de gran interés hoy en día. Si bien estos protocolos se vienen estudiando desde el punto de vista teórico en las últimas décadas, no ha sido hasta muy recientemente cuando los ordenadores cuánticos empiezan a ser una realidad. Las empresas IBM y Google lideran actualmente esta tecnología.

- Rieffel, E. y Polak, W., «*Quantum Computing - A Gentle Introduction*», MIT Press, 2011.
- Laforest, M., «*The Mathematics of Quantum Mechanics*», Quantum Cryptography School for Young Students, University of Waterloo, `https://uwaterloo.ca/institute-for-quantum-computing/sites/ca.institute-for-quantum-computing/files/uploads/files/mathematics_qm_v21.pdf`
- Rué, J. y Xambó, S., «*Mathematical Essentials Of Quantum Computing*», `https://web.mat.upc.edu/sebastia.xambo/QC/qc.pdf`
- Shor, P., «*Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*», SIAM J. Comput., 26(5), págs. 1484-1509.
- «*What is quantum computing?*», `https://www.ibm.com/quantum-computing/learn/what-is-quantum-computing/`
- Ghose, S., «*A beginner's guide to quantum computing*», `https://www.ted.com/talks/shohini_ghose_a_beginner_s_guide_to_quantum_computing`

Referencias para este Capítulo

- [1] Capítulos 1, 2, 8 y 9
- [3] Capítulo 6
- [5] Capítulos I y V
- [6] Capítulo 2
- [7] Capítulos 2 y 14

V. Números Primos

V.1 Pruebas de Primalidad

En el capítulo anterior hemos visto que *es imposible tener un algoritmo de factorización determinista y computable* (i. e. complejidad polinómica) para *cualquier* número. Así pues, la factorización (eligiendo bien el número para evitar las situaciones en las que los algoritmos conocidos son eficientes) es un buen problema matemático sobre el que construir un sistema criptográfico ya que, básicamente, un atacante/espía debería factorizar un número grande n para poder romper el sistema. En consecuencia, tanto emisor como receptor necesitan construir números grandes n difíciles de factorizar.

¿Qué números n son difíciles de factorizar? Si fijamos el número de bits de n, entonces el caso n primo sería difícil (los algoritmos anteriores no darían ninguna respuesta). Igualmente, el caso en que los factores primos de n son grandes también sería difícil (los anteriores nos devuelven factores pequeños, p. ej. Pollard) pero los factores tendrían que tener tamaños distintos (en otro caso, Fermat podría tener éxito). Finalmente, observemos que los cálculos en el algoritmo de Dixon se llevan a cabo con números primos con n, es decir, en $(\mathbb{Z}/n)^*$, por lo que si este el orden grupo tiene divisores pequeños, también tendremos opciones de factorizar n. Por ello al tomar n se debe evitar que $\phi(n)$, el indicador de Euler de n, tenga factores pequeños.

Por tanto, este enfoque pasa necesariamente por la necesidad de construir números primos grandes para obtener n como su producto. Llegamos a una aparente contradicción ya que para saber si un número es primo, tenemos que probar a dividirlo por todos primos menores. Entonces, ¿cómo construimos números primos grandes si no hay proceso determinista para factorizar un número? En realidad, debemos diferenciar que se tratan de dos problemas distintos, uno es determinar la primalidad y otro conseguir una

factorización. *Es necesario encontrar propiedades que distingan los números primos de los números compuestos* sin necesidad de recurrir a su factorización. Estas propiedades existen y, además, dan lugar a *pruebas probabilísticas de primalidad*. No son procesos deterministas pero pueden hacerse tan fiables como deseemos.

Figura V.1: A la izquierda, Pierre de Fermat (1601-1665) matemático francés, y a la derecha Carl Gustav Jacob Jacobi (1804-1851) matemático alemán (Wikimedia Commons).

V.1.A Fermat

Teorema V.1.1 — Pequeño Teorema de Fermat. Si n es primo, entonces $a^{n-1} = 1 \mod n$ para todo $a \in \mathbb{Z}$ con $(a,n) = 1$.

El recíproco del Pequeño Teorema de Fermat no es cierto en general, sin embargo sí podemos reinterpretarlo en forma de un test de primalidad.

Test de Fermat

Dado n, si encontramos a con $2 \leq a \leq n-1$ tal que $a^{n-1} \neq 1 \mod n$, entonces n no es primo.

Recordemos que $a^{n-1} \mod n$ se puede calcular eficientemente con el algoritmo de exponenciación rápida.

Ejercicio V.1.2 Decidir si 341 es primo o no.

Solución: Utilizando la exponenciación rápida, vamos a calcular expresiones del tipo $a^{340} \mod 341$. De este modo, tenemos $2^{340} = 1 \mod 341$ y $3^{340} = 56 \mod 341$ por lo que no es primo. De hecho, $341 = 11 \cdot 31$.

Ejercicio V.1.3 Utilizar el test de Fermat para probar que 1111 no es primo.

Este test no es completamente satisfactorio ya que hay números compuestos que verifican el Teorema de Fermat.

Definición V.1.4 Un número de Carmichael es un número no primo n tal que $a^{n-1} = 1 \mod n$ para todo $a \in \mathbb{Z}$ con $(a,n) = 1$.

Problema V.1.5 Calcular 2^{560}, 5^{560} y $7^{560} \mod 561$.

Observación V.1.6

- Un número impar no primo n es de Carmichael si y solo si para cada divisor primo p de n, p^2 no divide a n y $p-1$ divide a $n-1$.
- Un número de Carmichael tiene al menos tres factores.
- Los números de Carmichael más pequeños son: $561 = 3\cdot 11\cdot 17$, $1105 = 5\cdot 13\cdot 17$, $1729 = 7\cdot 13\cdot 19$, ...
- Alford-Granville-Pomerance probaron que existen infinitos números de Carmichael. Ver Alford, W. R.; Granville, Andrew; Pomerance, Carl, «*There are infinitely many Carmichael numbers*», Ann. of Math. (2) 139 (1994), no. 3, págs. 703-722.

V.1.B Solovay-Strassen

Se trata de un test probabilístico sobre la primalidad de un número y ilustra perfectamente la conexión con la *Teoría de Números* y, más específicamente, con los símbolos de Legengre y Jacobi.

Se define el *símbolo de Legendre* de un entero $a \in \mathbb{Z}$ y un primo $p \in \mathbb{Z}$ por

$$\left(\frac{a}{p}\right) := \begin{cases} 1 & \text{si } a \text{ es un cuadrado en } (\mathbb{Z}/p)^* \\ -1 & \text{si } a \text{ no es un cuadrado en } (\mathbb{Z}/p)^* \\ 0 & \text{si } a = 0 \text{ en } \mathbb{Z}/p \end{cases}$$

Teorema V.1.7 Para un primo p y un entero a coprimo con p, se cumple que

$$(p-1)! = -\left(\frac{a}{p}\right) a^{\frac{p-1}{2}} \qquad \mod p\,.$$

Demostración. Consideramos el subconjunto de $(\mathbb{Z}/p)^*$ dado por las clases cuyo cuadrado es distinto de a módulo p. En este subconjunto, podemos agrupar sus elementos por parejas (x,y) cuando se cumpla que $x\cdot y = a$ en $\mathbb{Z}/p$ (ya que, dado x que es invertible en $\mathbb{Z}/p$, determinamos y por $y = x^{-1}a$). Si a no es un cuadrado, estas parejas (x,y) verifican $x \neq y$ por lo que hay exactamente $\frac{1}{2}(p-1)$ parejas. Multiplicando todos los elementos de $(\mathbb{Z}/p)^*$, directamente por un lado y agrupados en estas parejas por otro, resulta

$$1\cdot 2\cdot \ldots \cdot (p-1) \equiv a^{\frac{p-1}{2}} \qquad \mod p\,.$$

Si a es un residuo cuadrático, sea $a = b^2$ en $\mathbb{Z}/p$, entonces hay $\frac{1}{2}(p-1)-1$ parejas (x,y) con $x \neq y$, quedando precisamente los elementos b y $-b$ desparejados. Como $b\cdot(-b) = -a$, multplicando todos los elementos de $(\mathbb{Z}/p)^*$, tenemos

$$1\cdot 2\cdot \ldots \cdot (p-1) \equiv (-a)a^{\frac{p-1}{2}-1} \equiv -a^{\frac{p-1}{2}} \qquad \mod p\,.$$

Recordando la definición del símbolo de Legendre, se concluye.

Este resultado nos permite recuperar varios resultados clásicos de forma sencilla.

Corolario V.1.8

- Teorema de Wilson: $n \in \mathbb{N}$ es primo si y solo si $(n-1)! = -1$ módulo n.
- Criterio de Euler: para un primo p y un entero a coprimo con p, se cumple que

$$\left(\frac{a}{p}\right) = a^{\frac{p-1}{2}} \qquad \text{mod } p$$

  (esta expresión explícita fue la definición original de Legendre en 1798).
- Pequeño Teorema de Fermat: para un primo p y un entero a, se cumple

$$a^p \equiv a \qquad \text{mod } p$$

Demostración. Respecto del directo del Teorema de Wilson basta tomar $a = 1$ y observar que $\left(\frac{a}{p}\right) = 1$ ya que 1 es un residuo cuadrático $!1^2 = 1$. Para el recíproco, la identidad implica que $(a,n) = 1$ para todo $1 \leq a \leq n-1$, de donde el resultado.

El Criterio de Euler resulta de sustituir la identidad del Teorema de Wilson en el lado izquierdo de la fórmula del Teorema V.1.7.

Para el último resultado, observar que si $(a,p) \neq 1$, entonces $p|a$ y $a^p = a = 0$ módulo p. Si $(a,p) = 1$, el enunciado es equivalente a $a^{p-1} = 1$ módulo p. Para lo cual basta con elevar al cuadrado la identidad del Criterio de Euler.

A partir del símbolo de Legendre se puede introducir el símbolo de Jacobi, que es una generalización de aquel.

Definición V.1.9 Para $n, a \in \mathbb{N}$, con n impar definimos el símbolo de Jacobi por

$$\left(\frac{a}{n}\right) := \left(\frac{a}{p_1}\right)^{s_1} \cdot \ldots \cdot \left(\frac{a}{p_r}\right)^{s_r}.$$

donde $p_1^{s_1} \cdot \ldots \cdot p_r^{s_r}$ descomposición en producto de potencias de primos de n.

Proposición V.1.10 El símbolo de Jacobi verifica las siguientes propiedades:

1. si n es un primo impar, entonces el símbolo de Jacobi es igual al símbolo de Legendre.
2. si $a = b$ en $\mathbb{Z}/n$, entonces $\left(\frac{a}{n}\right) = \left(\frac{b}{n}\right)$.
3. si $(a,n) \neq 1$ en $\mathbb{Z}$, entonces $\left(\frac{a}{n}\right) = 0$.
4. $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right)$.
5. Ley de Reciprocidad Cuadrática: si m, n son números naturales impares y coprimos, entonces

$$\left(\frac{m}{n}\right)\left(\frac{n}{m}\right) = (-1)^{\frac{m-1}{2}\frac{n-1}{2}} = \begin{cases} -1 & \text{si } m = n = 3 \text{ módulo } 4 \\ 1 & \text{en otro caso} \end{cases}$$

6. y, a partir de los anteriores, tenemos

$$\left(\frac{-1}{n}\right) = (-1)^{\frac{n-1}{2}} = \begin{cases} -1 & \text{si } n = 3 \text{ módulo } 4 \\ 1 & \text{si } n = 1 \text{ módulo } 4 \end{cases}$$

$$\left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}} = \begin{cases} -1 & \text{si } n = 3,5 \text{ módulo } 8 \\ 1 & \text{si } n = 1,7 \text{ módulo } 8 \end{cases}$$

La aplicación reiterada de estas propiedades nos permiten calcular el símbolo de Jacobi en tiempo polinomial (sin recurrir a la factorización explícita de n).

Ejercicio V.1.11 Calcular el símbolo de Jacobi $\left(\frac{936}{6035}\right)$

Solución:

$$\begin{aligned}\left(\frac{936}{6035}\right) &= \left(\frac{8\cdot 117}{6035}\right) = \left(\frac{2}{6035}\right)^3\left(\frac{117}{6035}\right) = (-1)\left(\frac{6035}{117}\right) = \\ &= (-1)\left(\frac{68}{117}\right) = -\left(\frac{2}{117}\right)^2\left(\frac{17}{117}\right) = -\left(\frac{117}{17}\right) = -\left(\frac{15}{17}\right) = \\ &= -\left(\frac{3}{17}\right)\left(\frac{5}{17}\right) = -\left(\frac{17}{3}\right)\left(\frac{17}{5}\right) = -\left(\frac{2}{3}\right)\left(\frac{2}{5}\right) = (-1)(-1)(-1) = -1\end{aligned}$$

Todos estos resultados anteriores constituyen la base del test de primalidad de Solovay-Strassen que vemos a continuación. Dado $n \in \mathbb{N}$ impar y una *base* a, decimos que a es un *testigo de Euler para* n si

$$\left(\frac{a}{n}\right) \neq a^{\frac{n-1}{2}} \mod n$$

y un *mentiroso de Euler para* n en caso de que n sea compuesto pero se cumpla $\left(\frac{a}{n}\right) = a^{\frac{n-1}{2}} \mod n$ (Criterio de Euler). Es decir, que si existe un testigo entonces n es compuesto.

Proposición V.1.12 — test de primalidad de Solovay-Strassen. Dado $n \in \mathbb{N}$ impar, se tiene

1. si n es primo, se cumple $\left(\frac{a}{n}\right) = a^{\frac{n-1}{2}} \mod n$ para toda base a;
2. si n es compuesto, menos de la mitad de las bases a con $(a,n) = 1$ son mentirosos de Euler, es decir, $\left(\frac{a}{n}\right) = a^{\frac{n-1}{2}} \mod n$.

En particular, dados n impar y a, si se cumple $\left(\frac{a}{n}\right) = a^{\frac{n-1}{2}} \mod n$, entonces n es compuesto con probabilidad menor que $\frac{1}{2}$ o, equivalentemente, n es primo con probabilidad mayor o igual que $\frac{1}{2}$. En general, dados n impar y $a_1, \ldots, a_k$, si se cumple $\left(\frac{a_i}{n}\right) = a_i^{\frac{n-1}{2}} \mod n$ para todo i, entonces n es primo con probabilidad mayor o igual que $1 - (\frac{1}{2})^k$.

Demostración. (2) Es fácil probar, por las propiedades del símbolo de Jacobi, que si a es un mentiroso de Euler con $(a,n) = 1$ y b es un testigo de Euler, entonces ab es un testigo de Euler. Como 1 es un mentiroso, resulta que el número de testigos a es siempre mayor o igual que el número de mentirosos.

Ejercicio V.1.13 Determinar el conjunto de mentirosos de Euler para $n = 65$.

Solución: Tengamos en cuenta que $65 = 5 \cdot 13$. La lista de cuadrados en $\mathbb{Z}/5$ es $1^2 = 1, 2^2 = 4, 3^2 = 4, 4^2 = 1$. Por tanto

$$\left(\frac{a}{5}\right) := \begin{cases} 1 & \text{si } a = 1,4 \text{ en } \mathbb{Z}/5 \\ -1 & \text{si } a = 2,3 \text{ en } \mathbb{Z}/5 \\ 0 & \text{si } a = 0 \text{ en } \mathbb{Z}/5 \end{cases}$$

Para el caso de $n = 13$ utilizaremos la Proposición V.1.10 para algunos casos. Tenemos $\left(\frac{1}{13}\right) = 1$, $\left(\frac{2}{13}\right) = -1$ pues $13 = 5 \bmod 8$, $\left(\frac{3}{13}\right) = (-1)^{6\cdot 2}\left(\frac{13}{3}\right) = \left(\frac{1}{3}\right) = 1$ por la ley de reciprocidad, $\left(\frac{4}{13}\right) = 1$ pues es un cuadrado, $\left(\frac{5}{13}\right) = (-1)^{6\cdot 4}\left(\frac{13}{5}\right) = \left(\frac{3}{5}\right) = -1$ por la primera parte, $\left(\frac{6}{13}\right) = \left(\frac{2}{13}\right)\left(\frac{3}{13}\right) = -1$ por los casos anteriores, etc.

Los mentirosos son los números compuestos a tales que $\left(\frac{a}{65}\right) = a^{32} \bmod 65$. La lista de los que verifican esta igualdad es $\{1, 8, 14, 18, 47, 51, 57, 64\}$ de los cuales los compuestos son todos excepto el 1 y el 47.

Ejercicio V.1.14 Dado $n = 221$, aplicar el test de Solovay-Strassen a las bases 47 y 2.

Solución: En el primer caso, tenemos $\left(\frac{47}{221}\right) = (-1)^{110\cdot 23}\left(\frac{221}{47}\right) = \left(\frac{33}{47}\right) = \ldots = -1$. Por otro lado, $47^{\frac{221-1}{2}} = 47^{110}$ y $110 = 64 + 32 + 8 + 4 + 2$, por lo que aplicando la exponenciación rápida, tenemos $47^{110} = 220 = -1 \bmod 221$. En el segundo, tenemos $\left(\frac{2}{221}\right) = -1$ pues $221 = 5 \bmod 8$. Por otro lado, $2^{\frac{221-1}{2}} = 2^{110}$ y, por la exponenciación rápida, tenemos $2^{110} = 30 \bmod 221$.

Observación V.1.15 Para implementar este test se usa el algoritmo de exponenciación rápida. La complejidad del algoritmo viene dada por la complejidad del cálculo del símbolo de Jacobi, el de la exponenciación rápida, y del número de bases k que examinemos. Se puede determinar la complejidad media y máxima. Así mismo, la probabilidad de $\frac{1}{2}$ es cierta para un única base, pero se alcanza pocas veces, por lo que para k bases elegidas mediante una distribución uniforme se espera una probabilidad bastante superior a $1 - (\frac{1}{2})^k$.

V.1.C Miller-Rabin

Consideremos la siguiente construcción. Dado un número natural impar, $n > 1$, se puede escribir como $n = 1 + 2^e k$ con $E \geq 1$ y k impar. Por tanto, tenemos

$$\begin{aligned} x^{n-1} - 1 &= x^{2^e k} - 1 = (x^{2^{e-1}k})^2 - 1 = (x^{2^{e-1}k} - 1)(x^{2^{e-1}k} + 1) = \\ &= \ldots = (x^k - 1)(x^k + 1)(x^{2k} + 1) \cdot \ldots \cdot (x^{2^{e-1}k} + 1) \end{aligned}$$

Si ahora se toma n primo impar, por el pequeño teorema de Fermat, el lado izquierdo es 0 módulo n y, teniendo en cuenta que $\mathbb{Z}/n$ es íntegro, entonces uno de los factores del lado derecho ha de ser 0 módulo n y, por ello,

$$\text{existe un } i \in \{0, \ldots, e-1\} \text{ tal que } a^k = 1 \bmod n \text{ o } a^{2^i k} = -1 \bmod n \qquad \text{(V.1)}$$

Puesto que estas relaciones tienen sentido para todo n, damos la siguiente noción.

Definición V.1.16 Dado $n > 1$ impar con $n = 1 + 2^e k$ y k impar, se dice que a es un mentiroso de Miller-Rabin si n es compuesto y se cumple (V.1). El número a se llama testigo de Miller-Rabin, cuando cumple

$$\text{para todo } i \in \{0, \ldots, e-1\} \text{ se tiene que } a^k \neq 1 \bmod n \text{ y } a^{2^i k} \neq -1 \bmod n$$

esto es, la negación de (V.1).

Observación V.1.17

- La existencia de un testigo de Miller-Rabin implica que n es compuesto.
- Además, se cumple que si $a^{2^i k} = \pm 1 \bmod n$, entonces $a^{2^j k} = 1 \bmod n$ para todo $j \in \{i+1, \ldots, e\}$.
- Un testigo de Euler es testigo de Miller-Rabin.

Teorema V.1.18 Sea $n > 1$ impar y compuesto. El número de testigos de Miller-Rabin para n en el conjunto $\{2, \ldots, n-2\}$ es mayor que $\frac{3}{4}(n-4)$; es decir, la probabilidad de que un número a con $2 \leq a \leq n-2$ sea mentiroso de Miller-Rabin es menor que $\frac{1}{4}$.

TEST DE MILLER-RABIN

Sea n es primo y $n = 1 + 2^e k$ con k impar. Fijemos un número de rondas t.

1. tomamos a entre 2 y $n-2$.
2. si $(a,n) > 1$ (por algoritmo de Euclides), entonces terminamos y concluimos (con certeza) que n es compuesto.
3. se calcula $a^k \bmod n$.
4. si $a^k = \pm 1 \bmod n$, entonces a pasa el test. Si no, se calcula $(a^k)^2 \bmod n$, si es -1 a pasa el test. Si no, se calcula $(a^k)^4$, etc.
5. Si se alcanza $(a^k)^{2^e}$ y es $\neq -1 \bmod n$, entonces a es un testigo de Miller-Rabin. Terminamos y concluimos (con certeza) que n es compuesto.
6. Si no hemos alcanzado el número de rondas t, volvemos al primer punto.
7. Si hemos alcanzado t rondas (ningún a ha sido testigo), entonces la probabilidad de que n sea compuesto es menor que $(\frac{1}{4})^t$. O, equivalentemente, n es primo con probabilidad mayor que $1 - (\frac{1}{4})^t$.

La complejidad de realizar t veces el test sobre n es $O(t \log n)$; es decir, tiempo polinómico en el número de bits de n. Por otro lado, basándose en estas ideas, se puede obtener un método de factorización.

Problema V.1.19 Sea $n = 561$. Aplicar el test de Miller-Rabin con $a = 2$ y concluir que 561 es compuesto.

Sea $n = 15$. Determinar todos los testigos y mentirosos de Miller-Rabin.

Problema V.1.20 Aplicar los test de Fermat y de Miller-Rabin para probar que $2^{2^5} + 1$ no es primo. Comparar la eficiencia de los métodos.

Problema V.1.21 ¿Cuántos mentirosos de Miller-Rabin existen para para 221?

V.2 Construcción de Números Primos

Los métodos anteriores permiten dar algoritmos para construir números primos, mejor dicho, números cuya probabilidad de ser primo sea tan alta como previamente se haya fijado. La estrategia se basa en construir un número impar al azar de tamaño dado y en comprobar que satisface distintos criterios (deterministas y probabilísticos) que aseguren el resultado. La estimación de la complejidad (o tiempo necesario) de esta construcción se basa en dos hechos, por un lado la probabilidad de que un número sea primo (Teorema IV.2.1) y en la complejidad de los criterios aplicados.

CONSTRUCCIÓN DE UN NÚMEROS PRIMOS DE k BITS

- Se toman el primer y último bit iguales a 1 y los $k-2$ centrales se escogen al azar entre 0 y 1 mediante un generador aleatorio que siga una distribución uniforme.
- Se comprueba que no sea múltiplo de los números primos menores que una cota dada, por ejemplo, 10^6.
- El segundo criterio es aplicar el test probabilístico de Miller-Rabin un número t de veces, por ejemplo, $t = 3$.
- Si el número pasa ambos filtros, se considera primo (¡aunque pudiera resultar no serlo!).

Las elecciones anteriores aseguran que para $k \geq 1000$, más de 300 dígitos decimales, se tiene que la probabilidad de que un número compuesto supere ambos criterios es menor de $1/2^{80} \sim 10^{-24}$).

Observación V.2.1 Hay otros muchos criterios para determinar la primalidad de modo probabilístico. Por ejemplo, el test probabilítico de Lucas, basado en el símbolo de Jacobi, es casi tan efectivo como el de Miller-Rabin y conceptualmente similar al de Solovay-Strassen con base de factores. Por otro otro lado, el test de primalidad de Baillie-PSW consiste en una combinación del test de Lucas con una versión del de Fermat y es especialmente efectiva; de hecho es determinista para números inferiores a 2^{64} y, aunque se conjetura que hay números compuestos que pasan el test, no se conoce ninguno. PSW son las iniciales de Carl Pomerance, John Selfridge y Samuel Wagstaff. A modo de ejemplo, la función `PrimeQ` de *Mathematica* usa estos criterios.

- Pomerance, C., «Are There Counter-Examples to the Baillie-PSW Primality Test?» (1984), `www.pseudoprime.com/dopo.pdf`
- Pomerance, C., Selfridge, J., & Wagstaff, S., «The Pseudoprimes to $25 \cdot 10^9$», Mathematics of Computation (1980), 35(151), 1003-1026, doi:10.2307/2006210
- «Baillie-PSW primality test», Wikipedia, `https://en.wikipedia.org/wiki/Baillie-PSW_primality_test`.

V.3 Temas para ampliar

V.3.A Generación de Números Pseudoaleatorios

Como hemos visto, y quedará aún más patente en el Capítulo VI sobre protocolos de cifrado, muchas veces será necesario comenzar con un número aleatorio. Pero si nuestro generador de números no es aleatorio y resulta predecible, entonces todo el sistema criptográfico estará en entredicho. Es más preciso hablar de generador de números *pseudoaleatorios*.

Un generador de números pseudoaleatorios (PRNG, *pseudorandom number generator*) es un algoritmo que, partiendo de un valor inicial (la semilla), genera una secuencia que parece aleatoria, en el sentido que un observador que no conoce el valor de la semilla no puede distinguir la salida de una verdadera sucesión aleatoria. Rigurosamente, un generador de números pseudoaleatorios ha de pasar pruebas estadísticas que muestren que la distribución de los números generados es uniforme. Entre las aplicaciones de los PRNG se encuentran las simulaciones y la criptografía. Adicionalmente, las aplicaciones criptográficas requieren que la salida no sea predecible a partir

de salidas anteriores, y se necesitan algoritmos más elaborados. Por tanto, un PRNG en criptografía debe resistir a atacantes que dispongan de una descripción del generador, que observen una gran cantidad de salida, y que utilicen métodos de criptoanálisis. Dado que la secuencia generada por un PRNG está completamente determinada por su semilla, la entropía no puede ser mayor que el de la semilla, independientemente de la longitud de la secuencia generada. Esto distingue a un PRNG de un verdadero generador aleatorio.

- De la Bibliografía [7, Capítulo 5].
- Goldreich, O., «*Modern cryptography, probabilistic proofs and pseudorandomness*», Algorithms and Combinatorics, 17, Springer-Verlag, Berlin, 1999. ISBN: 3-540-64766-X
- Gu, Q., Paillier, P., Lange, T., Teske, E., Hankerson, D., Menezes, A., Quisquater, J.-J., «*Pseudorandom Number Generator*», en Encyclopedia of Cryptography and Security (2011), págs. 995-996.
- Blum, L., Blum, M. y Shub, M., «*A simple unpredictable random number generator*», SIAM J. Comput. 15 (1986), págs. 364-383.

V.3.B Sobre las pruebas de primalidad

Como hemos visto las pruebas de primalidad se pueden dividir en pruebas deterministas y pruebas probabilísticas. Mientras las primeras confirman de forma indudable si un número es primo, las segundas solo lo hacen con una probabilidad tan alta como queramos. Típicamente una prueba probabilística acierta plenamente en caso de anunciar que un número es compuesto y tiene una probabilidad de fallar al declarar que un número es primo.

Una de las principales características de una prueba de este tipo es el tiempo de ejecución. De hecho, la meta es encontrar pruebas con complejidad polinomial de modo que su utilización sea eficiente. No obstante, aunque hoy en día se conocen ya algoritmos deterministas que corren en tiempo polinomial, lo cierto es que las pruebas probabilísticas siguen siendo mucho más eficientes. Por ejemplo, basta dos rondas de Miller-Rabin y una de Lucas para generar un número primo (probable) con probabilidad de error menor que 2^{-100} y, de hecho, no se conocen un número compuesto que pase una ronda de Miller-Rabin seguida de una de Lucas.

Entre los test probabilísticos de complejidad polinomial hemos visto los de Soloway-Strassen y Miller-Rabin pero también podemos citar el de Frobenius-Grantham basado en polinomios cuadráticos y el automorfismo de Frobenius. La teoría de curvas elípticas ha permitido probar otros test probabilísticos como Goldwasser-Kilian o Elliptic Curve Primality (también conocido como ECPP, Atkin o Atkin-Morain) cuyos tiempos de ejecución se espera que sean polinomiales.

Entre los algoritmos deterministas están el de Adleman-Pomerance-Rumely (también conocido como Cohen-Lenstra-Bosma, Jacobi Sum Test o método ciclotómico) de complejidad casi polinomial o el de Agrawal-Kayal-Saxena que es polinomial.

Respecto del test de Miller-Rabin, hemos de indicar que trabajo original de Miller establecía un criterio determinista asumiendo que la Hipótesis de Riemann Generalizada. Esta hipótesis afirma que los ceros de una función, concretamente la función L de Dirichlet, yacen en la parte negativa de la recta real o bien en la recta compleja de números con parte real $1/2$, una consecuencia de esta hipótesis es que si a y d son coprimos, entonces la sucesión $a, a+d, a+2d, \ldots$ contiene infinitos números primos.

Posteriormente, Rabin modificó esta idea para obtener una prueba probabilística que no estaba condicionada a la validad de la Hipótesis de Riemann Generalizada, el resultado es el test de Miller-Rabin que hemos visto.

Para una descripción más detallada, consúltese

- Adleman, L.M, C. Pomerance, and R.S. Rumely, «*On distinguishing prime numbers from composite numbers*», Annals of Mathematics (1983), 117, págs. 173-206.
- Agrawal, M., N. Kayal, y N. Saxena, «*PRIMES is in P*» (2002).
- Liskov M., «*Miller-Rabin Probabilistic Primality Test*», en Encyclopedia of Cryptography and Security, van Tilborg, H. C. A. y Jajodia, S. (eds), Springer (2011), Boston, MA.
- Miller, G. L., «*Riemann's hypothesis and tests for primality*», J. Computer and System Sci- ences (1976), 13, págs. 300-317.
- Conrad, K., «*The Miller-Rabin test*», `https://kconrad.math.uconn.edu/blurbs/ugradnumthy/millerrabin.pdf`

REFERENCIAS PARA ESTE CAPÍTULO

- [1] Capítulo 6
- [3] Capítulo 5
- [4] Capítulo 5
- [5] Capítulo II
- [7] Capítulos 4 y 5

VI. Criptografía Asimétrica

En el Capítulo III vimos los cifrados simétricos, que usan la misma clave para cifrar y descifrar, y mencionamos alguna de sus ventajas (son rápidos) y desventajas (requieren de un canal alternativo seguro para compartir la clave).

En este Capítulo, expondremos sistemas que permiten la comunicación cifrada sin necesidad de disponer un canal alternativo seguro. Es una gran ventaja que permite una comunicación cifrada entre usuarios que no se conozcan previamente y que se adapta muy bien a la comunicación por internet. Un ejemplo típico, por la inmediatez que deseamos, sería la compra de una entrada cuando ya estamos en la misma cola del cine. El peaje que hay que pagar es que estos sistemas son computacionalmente más complejos. Su seguridad reside en la complejidad de los algoritmos involucrados y cuyo estudio se ha abordado en el Capítulo anterior. Tienen como estrategia común que cada usuario tiene una clave que consta de dos partes, una parte pública y que se da a conocer a todo el mundo, y otra parte privada que ha de mantenerse en secreto.

Más adelante ampliaremos estos sistemas para su utilización en otras aplicaciones y protocolos, como firmas digitales, integridad de documentos, subastas electrónicas, pruebas de conocimiento cero, etc.

VI.1 ElGamal

VI.1.A Comunicación Cifrada

La primera parte de este sistema de cifrado asimétrico, previa al establecimiento de una comunicación, consiste en la construcción y distribución de las claves y otros elementos necesarios para su funcionamiento.

Preparación ElGamal

- Se fija un primo p mucho mayor que el número de usuarios. Se fija un generador g de $(\mathbb{Z}/p)^*$. Sean $A, B, C, \ldots$ los usuarios.
- Cada usuario elige un número (el usuario A elige el número $1 < a_A < p-1$, B el $1 < a_B < p-1$, etc) denominado **clave privada**. Cada usuario conserva su clave en secreto.
- Se publican p, g y los resultados de las operaciones $g^{a_A}, g^{a_B}, \ldots$ mod p. Sería como un listín telefónico del tipo

$$\begin{array}{cccc} A & B & C & \ldots \\ g^{a_A} & g^{a_B} & g^{a_C} & \ldots \end{array}$$

  (siempre mod p) que se denominan **claves públicas** de los usuarios.
- Todos los usuarios conocen p, g y el listado con las claves públicas.

Una vez hecho todo esto, ya podemos describir el algoritmo detalladamente.

Comunicación ElGamal

- El usuario A desea mandar un mensaje a B. Asumimos, tras una correspondencia entre alfabeto y números, que el mensaje es un número $n \in \mathbb{Z}/p$.
- A elige un número al azar k y calcula g^k. Además, A consulta el catálogo y toma g^{a_B} y calcula $n \cdot (g^{a_B})^k$. Finalmente, A envía el par $(g^k, n \cdot (g^{a_B})^k)$.
- Cuando B recibe dos números (x, y), B calcula x^{a_B} (el primer número del mensaje recibido elevado a su clave privada que solo él conoce). Entonces calcula $y \cdot (x^{a_B})^{-1}$ que resulta ser el mensaje original.

Observación VI.1.1

- Las potencias se calculan módulo p y, por tanto, los exponentes se calculan módulo $\phi(p)$. En cuanto a la implementación, las potencias se calculan con la exponenciación rápida y los inversos con el algoritmo de Euclides.
- Además, es mejor si A elige un número k al azar primo con $\phi(p)$. En efecto, si $k|\phi(p)$, entonces g^k tiene orden $\frac{\phi(p)}{(k,\phi(p))} = \frac{p-1}{(k,p-1)}$. En consecuencia, cuanto mayor sea (k, p), más fácil será encontrar colisiones entre las potencias de g^k, esto es un i tal que $(g^k)^i = g^k$ mod$p$ y, eventualmente, esto podría determinar k. Entonces es claro que conociendo k, puesto que g, p, g^{a_A} son públicos, se obtendría el mensaje n.

Proposición VI.1.2 La seguridad de ElGamal radica en la dificultad del problema del logaritmo discreto (PLD). Se debe tomar p primo muy grande tal que $\frac{p-1}{2}$ sea primo.

Demostración. A la hora de examinar la seguridad, hemos de considerar todos los ataques y estrategias conocidas. Consideremos las dos más obvias, aunque la argumentación para las demás es similar.

La primera consiste en intentar recuperar la clave privada de un usuario mediante análisis de su clave pública, es decir, calcular a_A a partir de g^{a_A} mod p. Es claro que pasa por calcular el logaritmo discreto (ver §IV.3).

La segunda se trataría del *hombre en medio* (*man in the middle*) . Si C es capaz de interceptar las comunicaciones entre A y B y conoce todos los datos públicos: p, g, claves públicas, etc., entonces ¿podría descifrar los mensajes? En ese caso, C, de modo similar a cómo haría el destinatario legítimo B, tendría que calcular a_B, que le permite calcular n. De nuevo, se trata del problema del logaritmo discreto. Otra opción sería que C calculara k, el logaritmo discreto en base g del primer número interceptado, y con k, calcular $(g^{A_B})^k$ (pues g^{A_B} es la clave pública de B y por tanto, conocida). Pero esto es de nuevo inviable.

La elección de p se deduce ahora de la sección IV.3.

Problema VI.1.3 Hacer los cálculos para encriptar y desencriptar el mensaje $n = 7$ en la siguiente situación $p = 97$, $g = 5$, $a_A = 36$ (secreta), $g^{a_A} = 50$ (pública), $a_B = 58$ (secreta) y $g^{a_B} = 44$ (pública).

Solución: A elige un número k al azar (primo con $\phi(p) = \phi(97) = 96$), digamos $k = 17$. Calcula $5^{17} = 83 \bmod 97$. Calcula $7 \cdot 44^{17} = 67 \bmod 97$. Envía $(83, 67)$. Ahora B recibe $(83, 67)$. Calcula $83^{58} = 65 \bmod 97$. Entonces $67 \cdot 65^{-1} = 67 \cdot 3 = 7 \bmod 97$, que era el mensaje original.

Problema VI.1.4 Sea el número primo $p = 65537$ y $g = 5$ un generador de $\mathbb{Z}/p$. Nuestra clave secreta es $a_A = 13908$.

Hemos acordado con B que los mensajes los interpretaremos como trigrafos en un alfabeto de 31 símbolos, que numerados correlativamente, son A-Z, espacio, punto, interrogante, admiración y apóstrofe.

Recibimos el mensaje $(29095, 23846)$ encriptado por el sistema de ElGamal. Descifradlo.

Solución: En primer lugar, calculamos $29095^{13908} \bmod 65537$ y obtenemos 18637, cuyo inverso es 31543. Seguidamente, tenemos

$$31543 \cdot 23846 = 6229 = 6 \cdot 31^2 + 14 \cdot 31 + 29 \mod 65537$$

que, utilizando el alfabeto del enunciado, corresponde a los caracteres «GO!».

VI.1.B Firma Digital

El sistema anterior no certifica la identidad del emisor. De hecho, otro usuario C puede suplantar la identidad de A y mandar el mensaje $(g^k, n \cdot (g^{a_B})^k)$ a B.

No obstante, se puede conseguir que B pueda validar a A como emisor de dicho mensaje, es decir, es posible diseñar un sistema de firma o de autenticación. El protocolo que presentamos a continuación es la base para el DSS (digital signature standard).

Partimos de los mismos datos que para una comunicación (ver §VI.1.A). La firma de A es un mensaje que consta del nombre o algún dato identificativo de A pero escrita como una secuencia de números en $\mathbb{Z}/p$). Pero la prueba de la identidad de A no radica en este número, sino en que A es el único usuario capaz de resolver una ecuación en la que interviene dicho número.

SISTEMA DE FIRMA (BASE DE DSS)

- Si A quiere mandar su firma a B, entonces elige k al azar con $(k, p-1) = 1$ y y calcula $r = g^k \bmod p$ que lo considera como un entero. Seguidamente, A toma un dígito de la firma, sea s, resuelve la ecuación

$$s = a_A \cdot r + kt \qquad \bmod p-1$$

  y envía la terna $(g^k, t, s) \bmod p-1$ a B. Se repite para cada dígito de la firma.
- Cuando B recibe (x, y, s), entonces calcula $(g^{a_A})^x \cdot x^y \pmod p$ y $g^s \pmod p$ y comprueba si son iguales. En caso afirmativo, asume la autoría de A.
- Si C quisiera suplantar a A, tendría que ser capaz de mandar la terna (x, y, s) tal que $(g^{a_A})^x \cdot x^y = g^s \pmod p$, es decir, tiene que ser capaz de resolver $y \equiv k^{-1}(s - a_A r) \bmod p-1$. Pero esto solo lo podrá hacer si conoce a_A.

Ejercicio VI.1.5 Hacer los cálculos para enviar la firma $s = 7$ en la siguiente situación $p = 97$, $g = 5$, $a_A = 36$ (secreta) y $g^{a_A} = 50$ (pública).

Solución: El usuario A elige un k al azar, por ejemplo $k = 11$ y escribe la ecuación $7 = 36 \cdot 5^{11} + 11 \cdot t \bmod 96$. Resolviéndola, obtiene $t = 89$. Por lo tanto A envía a B la terna $(5^{11}, 89, 7) = (29, 89, 7) \bmod 96$. Estos cálculos son todos módulo 96 pues corresponden a exponentes en $\mathbb{Z}/97$.

Al recibir B dicha terna, calcula $50^{29} \cdot 71^{89} = 40 \bmod 97$ así como $5^7 = 40 \bmod 97$. Como son iguales, acepta la firma.

VI.2 Otros Criptosistemas Asimétricos

VI.2.A Diffie-Hellman

Supongamos que disponemos de unos datos (primo, generador, claves) como en ElGamal (ver §VI.1.A).

La propuesta que hicieron Diffie-Hellman permite que dos usuarios compartan una clave secreta (por ejemplo, para un sistema de encriptado de clave privada) sin necesidad de recurrir a más datos o elecciones. Casi simultáneamente, Malcolm Williamson diseñó un sistema similar para el *Government Communications Headquarters* (GCHQ) británico, pero esto tardó años en conocerse.

DIFFIE-HELLMAN: INTERCAMBIO DE CLAVE

La clave que pueden usar dos usuarios A y B es $g^{a_A a_B}$. De hecho, A puede calcular esta clave ya que g^{a_B} está en el directorio y $(g^{a_B})^{a_A} = g^{a_A a_B}$. De modo similar, B también puede calcularla. Ningún otro usuario puede calcular esa clave a partir de g^{a_B} y g^{a_A} ya que requeriría saber calcular el logaritmo discreto.

En resumen, hemos visto que los datos del directorio de ElGamal permiten

- criptografía asimétrica (ElGamal),

- firma digital (DSS),
- criptografía simétrica (Diffie-Hellman),

por lo que es un sistema muy versátil y completo.

Ejercicio VI.2.1 Supongamos que C sabe que A y B se están comunicando con un sistema de Diffie-Hellman y que también conoce p, g, el directorio de claves públicas, etc. Probar que poder determinar la clave compartida de A y B equivale a resolver el problema del logaritmo discreto. ¿Supondría una ventaja conocer mensajes interceptados?

VI.2.B Protocolos sin Clave

Sobre 1980, Shamir propuso un método de transmisión cifrada sin tener ni clave secreta ni clave pública. Se denomina el protocolo de tres pasos de Shamir o protocolo sin claves de Shamir.

Protocolo de tres pasos de Shamir

- Supongamos que A quiere mandar un mensaje a B. Se fija un primo grande p.
- Entonces A elige un número e_A y calcula su inverso d_A módulo $\phi(p) = p-1$.
- El usuario B elige otro número e_B y también calcula su inverso.
- Observemos que e_A, e_B son únicamente ¡para esta transmisión!
- Si el mensaje es M, $1 < M < p$, entonces A calcula M^{e_A} y se lo transmite a B.
- Este lo recibe (y no puede hacer «nada» con ello), calcula $(M^{e_A})^{e_B}$ y se lo devuelve a A.
- De nuevo, A calcula $((M^{e_A})^{e_B})^{d_A}$ y lo manda a B.
- Como el valor de esta última expresión es $((M^{e_A})^{e_B})^{d_A} = M^{e_B}$, B lo eleva a d_B y obtiene M.

Poco después, James Massey y Jim K. Omura hicieron alguna modificación a este sistema y, de hecho, derivó en una patente. Si el protocolo anterior se enuncia sobre el cuerpo finito $\mathbb{Z}/p$ (p un número primo), la adaptación de Massey-Omura lo sustituye por el cuerpo $\mathbb{F}_{2^n}$, cada interlocutor elige d, e tales que $de = 1$ módulo $2^n - 1 = \#\mathbb{F}_{2^n}^*$ y se realizan determinadas adaptaciones que lo hacen computacionalmente más eficiente.

A la hora de estudiar la fortaleza de este sistema, podemos comenzar estudiar los siguientes ataques llevados a cabo a partir de una intercepción de la transmisión. Imaginemos que hemos interceptado el número primo p usado así como alguna de las transmisiones.

- si disponemos de dos transmisiones entre A y B, por ejemplo, $x = M^{e_A}$ y $y = (M^{e_A})^{e_B}$, entonces sabríamos que la clave e_B utilizada por B verificaría $x^{e_B} = y$ y, por tanto, la calcularíamos $e_B = \log_x y$.
- si solamente interceptamos una transmisión x, podríamos intentar calcular potencias de x, digamos $\{x, x^2, x^3, \ldots\}$ buscando una colisión; esto es, $x^i = x^j \bmod p$. De aquí podríamos determinar divisores de $\#\mathbb{F}_p - 1$.

La defensa en ambos casos pasa por tomar p muy grande de modo que $p-1$ tenga factores grandes, p.ej., $\frac{p-1}{2}$ primo. Estos argumentos habría que adaptarlos para el caso de Massey-Omura.

Un gran desventaja de este sistema es que no proporciona un sistema de autenticación.

VI.3 RSA

VI.3.A Criptosistema

Este criptosistema fue descrito públicamente por Ronald Rivest, Adi Shamir y Leonard Adleman en 1977. La tradición matemática de poner los autores en orden alfabético se rompió debido a la insistencia de Adleman que consideraba que su aportación fue menor que la de Shamir y Rivest ya que *solamente* fue una observación aritmética. El sistema estuvo protegido por una patente hasta el año 2000. No obstante, la desclasificación en 1997 de documentos *top secret* de la agencia *Government Communications Headquarters (GCHQ)* desveló que Clifford Cocks había desarrollado un sistema equivalente en 1973.

Presentamos a continuación el resultado matemático sobre el que se asiente el RSA y que, además, asegura el funcionamiento del algoritmo incluso si M no es invertible módulo n.

Proposición VI.3.1 Para todo par de números naturales n, M, se cumple

$$M^{\phi(n)+1} \equiv M \mod n$$

En particular, si $a = 1 \bmod \phi(n)$, entonces $M^a = M \bmod n$.

Demostración. Recordemos que el Teorema Chino de los Restos implica que se tiene un isomorfismo de anillos $\psi : \mathbb{Z}/n \simeq \prod_i \mathbb{Z}/p_i^{r_i}$ siendo $n = \prod p_i^{r_i}$ y que la fórmula explícita de la función indicador de Euler, $\phi(n) = \prod(p_i - 1)p_i^{r_i-1} = \prod \phi(p_i^{r_i})$ nos permite escribir $\phi(n) = \phi(p_i^{r_i}) \cdot m_i$ para cierto número natural m_i. De todo lo anterior resulta la relación

$$\psi(M^{\phi(n)+1}) = \left((M^{\phi(p_1^{r_1})})^{m_1} \cdot M, \dots\right)$$

Observemos que si $(M, p_1) = 1$, entonces $(M^{\phi(p_1^{r_1})})^{m_1} \cdot M = 1^{m_1} \cdot M = M \bmod p_1^{r_1}$. Por otro lado si $(M, p_1) \neq 1$, entonces $p_1 | M$ y entonces $M^{\phi(p_1^{r_1})} = 0$ ya que $\phi(p_1^{r_1}) \geq p_1^{r_1-1} \geq r_1$ para todo primo $p_1 \geq 2$ y $M^{r_1} = 0 \bmod p_1^{r_1}$. Como $\phi(n) \geq \phi(p_1^{r_1})$ el lado derecho también es $0 \bmod p_1^{r_1}$.

ALGORITMO RSA: RIVES, SHAMIR, ADLEMAN

- *Generación de claves*: se toman dos números primos muy grandes p, q, por ejemplo, mayores que 10^{100}. Se define $n = pq$ y, por tanto, $\phi(n) = (p-1)(q-1)$. Se toma un número al azar e tal que $(e, \phi(n)) = 1$ y se calcula su inverso $d = e^{-1} \bmod \phi(n)$. Se da a conocer la clave pública (n, e) y se deja oculto el resto de datos, en particular, la *clave privada* d.

- *Directorio*: un sistema de este tipo, implica que los usuarios $A, B, \ldots$ hacen las elecciones anteriores y que se publica un directorio del tipo (A, n_A, e_A), $(B, n_B, e_B), \ldots$
- *Cifrado*: si un usuario quiere enviar un mensaje M, $0 \leq M < n$ a otro que tiene clave pública (n, e), entonces el emisor envía el resultado del cálculo $N = M^e \bmod n$ (observar que $0 \leq N < n$).
- *Descifrado*: el receptor, usando su clave privada d, ha de calcular $N^d = (M^e)^d = M \bmod n$ para obtener el mensaje.

Observación VI.3.2

- En el proceso anterior, se utiliza el algoritmo de Euclídes para los inversos y la exponenciación rápida para las potencias (ver §IV.1.B).
- Un aspecto clave es la construcción de números primos grandes (ver §V.2).
- Aunque realmente un usuario es capaz de generar sus claves por si mismo, lo cierto es que se suele *delegar* en una autoridad o institución de confianza la generación y distribución de claves (p. ej. la Fábrica Nacional de Moneda y Timbre, FNMT).
- Las necesidades computacionales del RSA son altas por lo que, en función del grado de seguridad deseado, a veces se utiliza RSA para la creación de un canal seguro por el que se comparte una clave de un sistema de clave simétrica.
- Otro uso común es la firma digital (se cifra poca cantidad de información, pero que requiere alta seguridad).

Problema VI.3.3 En un sistema RSA, mis claves son $(319, 33)$. Un amigo me quiere enviar el mensaje 104. ¿Cómo ha de hacer? ¿Cómo descifro el mensaje recibido?

Solución: En primer lugar, $319 = 11 \cdot 29$, $\phi(319) = 10 \cdot 28$, $d = 33^{-1} = 17 \bmod 280$. Mi amigo ha de computar $104^{33} \bmod 319$, que da como resultado 191 y me lo envía. Yo recibo 191, entonces $191^{17} = 104 \bmod 319$ es el mensaje descifrado.

Definición VI.3.4 Se define la función de Carmichael de un número natural n como el menor entero positivo $\lambda(n)$ tal que

$$a^{\lambda(n)} \equiv 1 \mod n \text{ para todo } a \text{ con } (a, n) = 1.$$

Ejercicio VI.3.5 Probar las siguientes propiedades:

- Si $n = \prod p_i^{r_i}$, entonces $\lambda(n) = \operatorname{mcm}(\lambda(p_1^{r_1}), \ldots, \lambda(p_s^{r_s}))$.
- Si $p = 2$ y $r > 2$, entonces $\lambda(p^r) = \frac{1}{2}\phi(p^r)$. En otro caso, $\lambda(p^r) = \phi(p^r)$.
- $\lambda(n) | \phi(n)$ para todo n.
- Puesto que en el RSA se toman $p, q > 2$, en dicho algoritmo se puede reemplazar $\phi(n)$ por $\lambda(n)$. Observar que para p, q coprimos, esto es $(p, q) = 1$, se tiene

$$\lambda(n) = \lambda(\operatorname{mcm}(p, q)) = \operatorname{mcm}(\lambda(p), \lambda(q)) = \operatorname{mcm}(p - 1, q - 1).$$

VI.3.B Criptoanálisis

Ataque a las Claves

El primer ataque que se nos ocurre es intentar calcular la clave privada d a partir de la clave pública (n, e). Para ello tiene que calcular $\phi(n)$ y después resolver la ecuación

$d = e^{-1} \bmod \phi(n)$. Lo segundo es rápido, pero el cálculo de $\phi(n)$ es tan costoso como factorizar n como probamos a continuación.

Proposición VI.3.6 Dado n, conocer $\phi(n)$ equivale a conocer p y q.

Demostración. En efecto, si conocemos p,q, entonces $\phi(n) = (p-1)(q-1)$. Recíprocamente, si conocemos n y $\phi(n)$, entonces $p+q = pq+1-(p-1)(q-1) = n+1-\phi(n)$ por lo que p,q son las raíces de la ecuación $x^2-(n+1-\phi(n))x+n=0$.

Por tanto, hay que elegir p,q tal que tanto $n = p\cdot q$ como $\phi(n)$ sean difíciles de factorizar. A la vista de las situaciones en las que los algoritmos de factorización (ver § §IV.2) tienen éxito, deberemos elegir p y q del siguiente modo

- p, q son números primos grandes, p. ej. mayores que 10^{100} (más de 300 bits cada uno),
- p, q tienen distinto tamaño (p. ej. uno de 300 bits y otro de 400 bits), en otro caso, Fermat y sus variantes podrían factorizar n.
- $p-1$ y $q-1$ no han de tener factores pequeños (salvo el 2), así que se puede imponer que $\frac{p-1}{2}$ y que $\frac{q-1}{2}$ sean primos.

Birthday attack o ataque del cumpleaños

Observemos que

$$\lambda(n) = \operatorname{mcm}(p-1,q-1) = \frac{(p-1)(q-1)}{(p-1,q-1)} = \frac{\phi(n)}{(p-1,q-1)}$$

por lo que si $p-1$ y $q-1$ tienen muchos factores comunes, entonces $\operatorname{mcm}(p-1,q-1)$ será «pequeño» y, por tanto, un ataque que busque factorizar $\phi(n)$ puede triunfar fácilmente.

Veamos cómo se implementa este ataque al RSA. La esperanza de encontrar una colisión se fundamenta en la paradoja del cumpleaños (ver Teorema IV.2.7).

Birthday attack o ataque del cumpleaños

Supongamos que las claves públicas de un usuario son (n,e) y que interceptamos muchos mensajes cifrados dirigidos a él. Fijemos un número de iteraciones N. Para cada mensaje interceptado c, empezamos a calcular c^i para $i = 1,2,\ldots,N$. Si resulta que $c^i = c^j \bmod n$, entonces $i-j$ divide a $\phi(n)$. Repitiendo esto para varios mensajes, obtendremos unos números $k_1,\ldots,k_r$ que dividen a $\phi(n)$, por lo que $\phi(n)$ es múltiplo del m.c.m. de $k_1,\ldots,k_r$. Por tanto, si el usuario ha elegido n con $\phi(n)$ igual a producto de factores «pequeños» (con relación al N elegido), entonces podremos probar a tomar

$$d = e^{-1} \mod \operatorname{mcm}(k_1,\ldots,k_r)$$

y probar d como clave de descifrado, es decir, si c^d tiene sentido. Si no lo tiene, entonces hay que seguir calculando más k's. También se puede probar tomando $\phi(n)$ igual a múltiplos de m.c.m.$(k_1,\ldots,k_r)$.

Para evitar esta situación, habrá que tomar p y q de modo que $(p-1,q-1)$ sea pequeño, equivalentemente, $\phi(n)$ con factores grandes, por ejemplo, tomar $n = p\cdot q$ con $(p-1)/2$, $(q-1)/2$ primos.

Problema VI.3.7 Simular un programa que realice el ataque basado en la «paradoja del cumpleaños». Como parte preliminar a esta práctica se ha de construir una función que simule los sucesivos mensaje cifrados que se van interceptando.

Exponentes públicos pequeños

Si se envía el mismo mensaje a distintos usuarios en cuya clave pública coincida el exponente de encriptación, un atacante puede utilizar las distintas encriptaciones para intentar recuperar el mensaje. Obsérvese que, *aunque no se desvelan las claves privadas, sí se accede al texto claro del mensaje.*

Problema VI.3.8 Una empresa envía a sus clientes VIP el mismo mensaje cifrado con RSA. Consultando el listado de las claves públicas de los destinatarios, observamos que hay varios clientes con el mismo exponente de encriptación. Descifrad el mensaje suponiendo que las claves de tres receptores son $(391,3)$, $(55,3)$ y $(87,3)$ y que los textos cifrados recibidos por ellos, que son el mismo mensaje original, son 208, 38 y 32.

Solución: El Teorema Chino de los Restos nos da un isomorfismo de anillos $\mathbb{Z}/(391\cdot 55\cdot 87)\simeq \mathbb{Z}/391\times\mathbb{Z}/55\times\mathbb{Z}/87$. Usando dos veces el algortimo de Euclides/Bezout, se consigue una antiimagen $\bar{n}=103823$ de $(208,38,32)$. Como el producto $391\cdot 55\cdot 87$ es mucho más grande que todos los números involucrados, es razonable pensar si la raíz cúbica (puesto que el exponente común es 3) de $n\in\mathbb{Z}$ con clase $\bar{n}$, en caso de ser un número entero, es el mensaje. Si no lo es, se puede probar con n más un múltiplo de $391\cdot 55\cdot 87$. En nuestro caso, $103823^{\frac{1}{3}}=47$, por lo que 47 es el mensaje.

La elección de un exponente de encriptación pequeño es común ya que puede acelerar los procesos de cifrado y de comprobación de firma. Sin embargo, como acabamos de probar, puede no ser seguro sobre todo si se envía el mismo mensaje o si el mensaje es muy corto. A la vista de la estructura del algoritmo de exponenciación rápida, se pueden realizar otras elecciones, más seguras y relativamente rápidas como $e=2^16+1=65537$ (que además es primo).

Ataque del módulo común

Veamos un ataque en el que si varios usuarios de un sistema RSA comparten el mismo módulo, entonces cualquiera de ellos puede obtener las diferentes claves privadas de estos usuarios. El proceso sería así.

ATAQUE DEL MÓDULO COMÚN (I)

- Supongamos que dos usuarios A, y B de un sistema RSA tienen claves son (n,e_A) y (n,e_B) donde $(e_A,e_B)=1$ y que hemos interceptado el mismo mensaje que se les ha enviado a ambos, sean c_A y c_B.
- Puesto que $(e_A,e_B)=1$, el algoritmo de Euclides nos permite calcular a, b tales que $ae_A+be_B=1$.
- Entonces

$$(c_A)^a\cdot(c_B)^b = m^{ae_A}\cdot m^{be_B} = m^{ae_A+be_B} = m$$

ya que $c_A=m^{e_A} \bmod n$ y $c_B=m^{e_B} \bmod n$.

Problema VI.3.9 Aplicar el ataque del módulo común al caso de claves $(493,3)$ y $(493,5)$ y mensajes cifrados 293 y 421.

Por el protocolo RSA, es obvio que desvelar nuestra clave privada d deja expuesta todos nuestros mensajes. Pero no solamente eso, conocer la clave privada permite factorizar n, tal y como muestra el siguiente resultado (conectado con la Proposición VI.3.6.

Proposición VI.3.10 Un usuario del RSA con claves públicas (n,e) y privada d, puede factorizar n de modo eficiente (complejidad polinomial).

Demostración. Definimos $k = d \cdot e - 1$ y observemos que k ha de ser un múltiplo de $\phi(n)$. Como $\phi(n)$ es par, entonces $k = 2^t r$ con $t \geq 1$ y r impar. Además, para todo $g \in (\mathbb{Z}/n)^*$ se tiene que $g^k \equiv 1 \bmod n$ y que, por tanto, $g^{k/2}$ es una raíz cuadrada de la unidad módulo n. Sabemos que n es producto de dos primos (que queremos calcular) por lo que 1 tiene 4 raíces cuadradas en $\mathbb{Z}/n$ (por el Teorema Chino de los Restos). Dos de estas raíces son ± 1, mientras que las otras dos, digamos $\pm x$, nos permitirían determinar los factores de n mediante el cálculo de $(x-1,n)$. Para conseguir esta raíz x, tomamos $g \in \mathbb{Z}/n$ al azar, si $(g,n) \neq 1$, ya tenemos un factor de n. Si $(g,n) = 1$, entonces calculamos las potencias $g^r, g^{2r}, \ldots, g^{2^t r} \bmod n$. Puesto que el último de la serie es $g^{2^t r} = g^k = 1 \bmod n$, si encontramos i tal que $g^{2^{i-1} r} \neq 1$ y $g^{2^i r} = 1 \bmod n$, entonces ya tenemos $x = g^{2^{i-1} r}$. Todos los cálculos se llevan a cabo con el algoritmo de exponenciación rápida y con el algoritmo de Euclides. La última cuestión trata sobre cómo de fácil es encontrar g verificando lo anterior. Observemos que si g no nos vale para argumentar, entonces $g^r = 1 \bmod n$. Por el Teorema de Cauchy, existe un elemento h de orden 2 en $(\mathbb{Z}/n)^*$. Entonces si g cumple $g^r = 1 \bmod n$, entonces $(hg)^r \neq 1 \bmod n$ pues r es impar. Por tanto, al menos la mitad de los elementos de $(\mathbb{Z}/n)^*$ nos valen para argumentar.

Y por tanto, da lugar al siguiente ataque, cuya defensa pasa por que cada usuario debe tener un módulo *único*, es decir, el n de su clave pública (n,e).

Ataque del módulo común (II)

Supongamos que dos usuarios A, y B de un sistema RSA comparten el mismo módulo, es decir, el mismo entero $n = pq$ (sin necesidad de conocer sus factores). Entonces, A es capaz de factorizar n y, por tanto, de calcular $\phi(n)$. Con ello, y con la clave pública de B, puede calcular la clave privada de este último.

VI.4 Temas para ampliar

VI.4.A McEliece (Teoría de códigos)

La teoría de códigos añade información redundante para permitir la lectura correcta del mensaje (en caso de errores en la transmisión). La criptografía trata de impedir la lectura a los receptores no legítimos. Sin embargo, McEliece presentó en 1978 uno de los primeros cifrados asimétricos combinando estos aspectos. Sus ventajas principales son el cifrado y descifrado computacionalmente eficientes así como su seguridad aunque, por contra, el tamaño de su clave pública es enorme y el texto cifrado es más largo que el texto plano.

Supongamos que tenemos una familia de códigos que corrige d errores y para el que conocemos un algoritmo de corrección eficiente.

Un usuario de este sistema ha de elegir matrices arbitrarias S del tipo $k \times k$ y P de permutación $n \times n$ junto con un código de la familia y una matriz codificadora S del tipo:

$$(\mathbb{F}_{2^l})^k \xrightarrow{M} (\mathbb{F}_{2^l})^n$$

Entonces $M' := PMS$ es un código lineal arbitrario y, por tanto, *difícil* de decodificar. El código dado por M' es del mismo tipo que el dado por M (y corrige el mismo número de errores). El usuario da a conocer M' como clave pública o clave de encriptación. Las matrices M, P, S son la clave privada y quedan en secreto.

Si queremos mandarle un mensaje $m \in (\mathbb{F}_{2^l})^k$ a este usuario, tomaremos sus clave pública M' y un vector arbitrario b (de modo que el número de bits iguales a 1 sea menor que d) entonces se transmite lo siguiente:

$$c = M' \cdot m + b$$

(se elige un b para cada m).

Al recibir c, se calcula $P^{-1}c = MSc + P^{-1}b$. Como P es de permutación, $P^{-1}b$ tiene menos de d bits por lo que nuestro código corrector de errores es capaz de recuperar m. Obtenemos así Sc. Como conocemos S^{-1} recuperaremos c.

Observemos que si 2^l y d son grandes, es difícil atacar por fuerza bruta.

- [7, Capítulo 8, sección 5].
- Sendrier N., «*McEliece Public Key Cryptosystem*», en Encyclopedia of Cryptography and Security, van Tilborg, H. C. A. y Jajodia, S. (editores), Springer, Boston.

VI.4.B Complementos sobre Criptoanálisis

Ataque con Criptotexto Elegido *(Chosen Ciphertext Attack)*

Para situar el contexto en que se desarrolla el estudio del criptoanálisis hemos de recordar el siguiente «axioma» de la criptografía.

Principio de Kerckhoffs

El sistema criptográfico tiene que ser seguro frente a un adversario que conoce todo sobre él con excepción de la clave privada. Es decir, la seguridad del sistema ha de ser intrínseca y no puede radicar en mantener en secreto el diseño.

Además, el concepto de seguridad es mucho más ambicioso que impedir que un atacante pueda recuperar textos claros a partir de textos cifrado, de hecho, se pretende que el atacante, partiendo de textos cifrados, no pueda[1] obtener ninguna información adicional sobre el texto claro aparte de la obvia. Esta es la denominada seguridad semántica.

[1] Rigurosamente, queremos decir que con probabilidad 1 no se pueda obtener ninguna información adicional mediante algoritmos de complejidad polinomial. La información contenida en un mensaje se puede medir con la función de entropía.

Por esta razón, es habitual ponerse en el siguiente escenario. El atacante puede acceder a la aplicación de cifrado y de descifrado en los siguientes términos:

- el atacante elige dos mensajes, se toma uno de ellos al azar y se cifra y el resultado se entrega al adversario, que debe tratar de distinguir a cuál de los dos mensajes corresponde;
- el atacante puede obtener el texto claro correspondiente a uno de dos textos cifrados que él haya elegido, pero no puede preguntar por el texto cifrado correspondiente a uno de los dos mensajes que trata de distinguir;
- el atacante puede realizar su ataque de forma adaptativa, es decir, puede elegir los textos claros y los textos cifrados de los apartados anteriores en función de las respuestas que ha ido obteniendo y también en función del texto cifrado que trata de distinguir.

Un ataque en el que se permite las acciones anteriores se denomina ataque con criptotexto elegido (*adaptative chosen ciphertext attack*, CCA). Existen protocolos criptográficos que resisten los ataques CCA aunque no todas las aplicaciones requieren este nivel de seguridad.

Otros Ataques al RSA

El criptosistema RSA se ha convertido en uno de los más populares y, consecuentemente, en uno de los más atacados. En la sección §VI.3.B hemos visto una serie de ataques basadas en propiedades aritméticas o en la elección de las claves.

Sin ánimo de ser exhaustivos, mencionaremos algunos otros ataques al RSA. El primer grupo se basa bien en propiedades teóricas, al igual que los anteriores, bien en un conocimiento adicional (desvelado parcial de la clave); entre los primeros citaremos los ataques propuestos por Franklin-Reiter, Coppersmith, Weiner, etc.

El segundo grupo de ataques explota las propiedades físicas de la implementación (tiempos de ejecución, consumo energético del dispositivo, fallos aleatorios, etc.).

- Bleichenbacher, B., «*Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1*», en Advances in Cryptology - CRYPTO'98, Lecture Notes in Computer Science 1462, pp. 1-12, Springer, 1998. `http://www.simovits.com/archive/pkcs.pdf`
- Boneh, D., «*Twenty Years of Attacks on the RSA Cryptosystem*», `http://www.ams.org/notices/199902/boneh.pdf`
- Everstine, E.W., «*Partial Key Exposure Attack On Low-Exponent RSA*», `http://www.cs.umd.edu/Honors/reports/Everstine/390ResearchPaper.pdf`
- Wiener, M., «*Cryptanalysis of Short RSA Secret Exponents*», `http://www3.sympatico.ca/wienerfamily/Michael/MichaelPapers/ShortSecretExponents.pdf`

VI.4.C Juegos de cartas

En el Capítulo VIII veremos varios protocolos y aplicaciones modernas de la criptografía. A modo de pequeño anticipo, veamos como las ideas de la criptografía asimétrica permitiría simular el reparto de cartas de un mazo entre dos jugadores. Evidentemente, un esquema completo para jugar a las cartas requeriría la adición de otros protocolos.

Supongamos que A y B desean jugar una partida de cartas. Tenemos una baraja con 40 cartas y se deben repartir 3 cartas a cada uno para comenzar.

- Los jugadores A y B se ponen de acuerdo en un número primo grande p. Además eligen e_A y e_B respectivamente y los mantienen en secreto.
- Entonces A baraja las cartas, obtiene un orden $C_1,\dots,C_{40}$ y encripta estos « mensajes»:

  $$C_1^{e_A}, C_2^{e_A}, \dots, C_{40}^{e_A}$$

  y lo envía a B.
- B elige tres de estos valores $C_{a_1}^{e_A}, C_{a_2}^{e_A}, C_{a_3}^{e_A}$ que serán las cartas de A, y los envía a A.
- B elige de nuevo otros tres, los eleva a e_B obteniendo $(C_{b_1}^{e_A})^{e_B}, (C_{b_2}^{e_A})^{e_B}, (C_{b_3}^{e_A})^{e_B}$, que serán las cartas de B, y los envía a A. Obsérvese que B desconoce en todo momento que cartas está eligiendo.
- A recibe las tres primeras, que serán las suyas, y puede saber cuales son elevando a d_A ya que $(C_{a_1}^{e_A})^{d_A} = C_{a_1}$, etc.
- A recibe las tres segundas, que serán las de B, y no puede saber cuales son. Entonces las eleva a d_A y se lo devuelve a B:

  $$((C_{b_1}^{e_A})^{e_B})^{d_A} = C_{b_1}^{e_B}, \dots$$

- B recibe estos datos, entonces las eleva a d_B y ya conoce sus cartas $(C_{b_1}^{e_B})^{d_B} = C_{b_1}, \dots$

VI.4.D Criptografía Homomórfica

Cada vez es más frecuente que tanto particulares como empresas deslocalicen el almacenamiento de datos a través de servicios tipo Dropbox, Drive, Azure,..., es decir, guardándolos en la nube. Podemos argumentar que su seguridad está garantizada si enviamos esos datos cifrados, pero el problema surge si queremos operar con estos datos sin necesidad de descargarlos y descifrarlos, pues en ese caso perderíamos muchas de las ventajas de tenerlos en la nube.

Entre las situaciones que podría ser interesante operar de este modo, estarían las siguientes:

- realizar un recuento de una votación electrónica secreta (cada voto afirmativo/negativo se cifra, se opera los votos cifrados para «sumarlos» sin posibilidad de conocer el sentido de cada voto, y finalmente se descrifa el resultado obteniendo el número de votos afirmativos;
- un empleado de una gran empresa puede trabajar con datos personales de los clientes, calculando medias, perfiles, etc sin posibilidad de acceder a ningún dato de un cliente particular.

La idea sería poder operar con los datos cifrados de modo que si el resultado obtenido se descifra, entonces se obtiene el mismo resultado que si hubiéramos operado con los datos en claro. Es decir, estamos aludiendo a que las aplicaciones de cifrado y descifrado tengan ciertas características de los morfismos de grupos. Por ejemplo, las aplicaciones de cifrado y descifrado del RSA son morfismos de grupos y permitirían llevar a la práctica esta idea. Sin embargo, esta idea básica es susceptible de ser atacada y necesita ser modificada (p. ej. esquema de Paillier, esquema de cifrado completamente homomórfico de Gentry).

- https://www.aepd.es/es/prensa-y-comunicacion/blog/cifrado-privacidad-iii-cifrado-homomorfico
- Gómez Pardo, J.L., «*Cifrado homomórfico: ejemplos y aplicaciones*», La Gaceta de la RSME, vol. 15 (2012), https://gaceta.rsme.es/abrir.php?id=1111
- Damgård, I., Groth, J. y Salomonsen, G., «*The theory and implementation of an electronic voting system*», en Secure Electronic Voting, págs. 77-99, Kluwer Academic Publishers, 2002.
- Gentry, C. y Halevi, S., «*Implementing Gentry's Fully Homomorphic Encryption Scheme*», en Advances in Cryptology - EUROCRYPT 2011, Lecture Notes in Computer Science 6632, págs. 129-148, Springer, 2011.

Referencias para este Capítulo

- [1] Capítulo 7
- [2] Capítulo 3
- [3] Capítulos 1, 2 y 4
- [4] Capítulo 7
- [5] Capítulo IV
- [7] Capítulos 3 y 8

VII. Criptografía con Curvas Elípticas

VII.1 Introducción

La criptografía con curvas elípticas (*elliptic curve cryptography* o ECC) fue introducida por Koblitz y Miller y puede ser aplicada para los mismos usos que otros sistemas de criptografía asimétrica (p. ej. RSA).

Matemáticamente está basada en el problema del logaritmo discreto para curvas elípticas (ECDLP, por sus siglas en inglés) que es aún más difícil que el logaritmo discreto (DLP) visto en la sección §IV.3. De hecho, la complejidad de los mejores algoritmos para el DLP es subexponencial mientras que para el ECDLP es exponencial. En la práctica, esto se traduce en que una clave de 160 bits para ECC proporciona tanta seguridad como una de 1024 para el RSA.

Antes de entrar en su descripción, es conveniente volver a recordar el esquema general de un sistema criptográfico (ver §II.3). En dicho esquema se tiene una aplicación de cifrado, C, y una aplicación de descifrado, D, y el cálculo de D es «fácil» para el receptor legítimo mientras que es «difícil» para el ilegítimo. En el caso de la criptografía simétrica esto se consigue con un espacio de claves de gran tamaño (p. ej. AES, ver §III.2.C). En el caso del RSA, la dificultad radica en el problema de factorización, mientras que otros sistemas como Diffie-Helman se basan en el problema del logaritmo discreto.

La esencia en el problema del logaritmo discreto es la siguiente: dado un grupo cíclico G y un elemento $g \in G$, es fácil calcular $g, g^2, \ldots$ pero dado a es difícil calcular $x \in \mathbb{N}$ tal que $g^x = a$. Si anteriormente hemos usado como grupo G el grupo de invertibles de un cuerpo finito ahora tomaremos G un grupo construido a partir de una curva elíptica, que es un objeto muy estudiado en geometría algebraica. De modo

intuitivo podemos afirmar que, al ser curvas elípticas sobre cuerpos finitos, estamos añadiendo a la aritmética del cuerpo la aritmética propia de la curva, y esta es la razón por la que la complejidad aumenta considerablemente.

VII.2 Curvas Elípticas

Definición VII.2.1 Una curva elíptica E sobre un cuerpo $\Bbbk$ será una cúbica plana irreducible y no singular, es decir, un polinomio irreducible $p(x,y) \in \Bbbk[x,y]$ de grado menor o igual que tres tal que el sistema $p(x,y) = 0, \partial_x p(x,y) = 0, \partial_y p(x,y) = 0$, no tenga soluciones.

Decimos que E está en forma forma de Weierstrass si el polinomio tiene la expresión

- $y^2 = x^3 + ax + b$ si $\mathrm{char}\,\Bbbk \neq 2,3$;
- $y^2 + xy = x^3 + ax^2 + b$ si $\mathrm{char}\,\Bbbk = 2$;
- $y^2 = 4x^3 + b_2x^2 + 2b_4x + b_6$ si $\mathrm{char}\,\Bbbk = 3$.

Proposición VII.2.2 Sea E una curva elíptica de ecuación $p(x,y) = 0$. Probar que, existe una única transformación afín de $\mathbb{A}_2$ (un cambio de coordenadas del tipo $x' = ax + by + c, y' = dx + ey + f$) que la lleva a forma de Weierstrass.

La demostración de la Proposición se basa en unas sencillas operaciones algebraicas. La forma de Weierstraß resulta práctica tanto para determinar si una cúbica es o no es singular así como para abordar la clasificación de curvas elípticas. La clasificación propiamente dicha, es decir determinar cuando dos curvas elípticas son isomorfas, se enunciará más adelante en términos del invariante j.

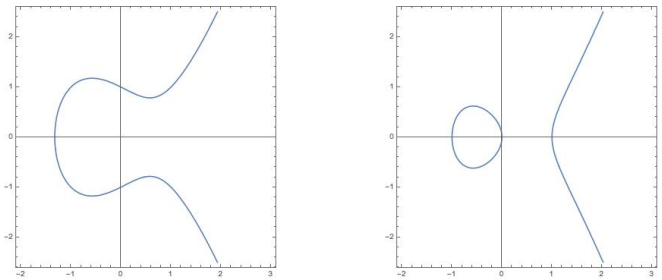

Figura VII.1: A la derecha, la gráfica de la curva elíptica $y^2 = x^3 - x + 1$ sobre $\Bbbk = \mathbb{R}$. A la izquierda, la de $y^2 = x^3 - x$.

Definición VII.2.3 Dada una cúbica plana de ecuación afín $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$, se define su discriminante como

$$\Delta = -b_2^2b_8 - 8b_4^3 - 27b_6^2 + 9b_2b_4b_6$$

siendo $b_2 = a_1^2 + 4a_4$, $b_4 = 2a_4 + a_1a_3$, $b_6 = a_3^2 + 4a_6$, $b_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2$.

Proposición VII.2.4

- El discriminante no depende de los cambios lineales de coordenadas.
- Una cúbica plana es no singular si y solamente si $\Delta \neq 0$.

- Si $\mathrm{char}\,\Bbbk \neq 2,3$, entonces el discriminante de $y^2 = x^3 + ax + b$ es

$$\Delta = -16(4a^3 + 27b^2)\,.$$

Observación VII.2.5 Si $\mathrm{char}\,\Bbbk \neq 2,3$, se define el invariante j por $j := -1728\frac{4a^3}{\Delta}$. Se cumple que dos curvas elípticas E, E' son isomorfas (cuando se permiten coeficientes con valores en el cierre algebraico, $\bar{\Bbbk}$) si y solo si tienen el mismo invariante j.

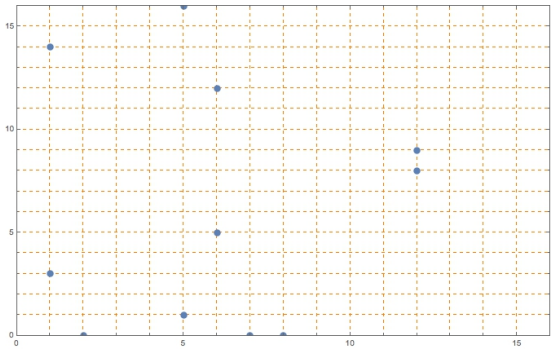

Figura VII.2: Gráfica de la curva elíptica $y^2 = x^3 + x + 7$ sobre $\mathbb{Z}/17$. Sus puntos sobre el plano afín tienen coordenadas $(1,3)$, $(1,14)$, $(2,0)$, $(5,1)$, $(5,16)$, $(6,5)$, $(6,12)$, $(7,0)$, $(8,0)$, $(12,8)$, $(12,9)$.

VII.3 Estructura de Grupo

En esta sección introduciremos el conjunto de puntos de una curva elíptica y veremos que tiene una estructura de grupo. Aunque la geometría proyectiva y la geometría algebraica permiten exprimir al máximo la estructura de estos objetos para sus aplicaciones en criptografía (ver §VII.5), podemos hacer una aproximación en términos más sencillos. Para ello es fundamental la siguiente propiedad geométrica.

Proposición VII.3.1 Sea $p(x,y) = 0$ una curva elíptica E sobre $\Bbbk$ en forma de Weierstraß y $\alpha x + \beta y + \gamma = 0$ una recta.

Si $\beta = 0$ y la intersección de la curva y la recta tiene un punto, entonces la curva y la recta se cortan exactamente en dos puntos.

Si $\beta \neq 0$ y la intersección de la curva y la recta tiene dos puntos, entonces la curva y la recta se cortan exactamente en tres puntos.

Para la demostración, basta sustituir la ecuación de la recta en la cúbica. Obtenemos una ecuación cuadrática en el primer caso, por lo que si tiene una raíz en $\Bbbk$, entonces tiene las dos (pueden ser iguales, en cuyo caso hablaremos de un punto de corte con multiplicidad dos). En el segundo, resulta una cúbica y si tiene dos soluciones, entonces ha de tener sus tres raíces en $\Bbbk$.

Observación VII.3.2 El Teorema de Bezout en geometría proyectiva supera esta dicotomía, pues en el plano proyectivo sobre un cuerpo algebraicamente cerrado, dos curvas de grado de grados m, n se cortan exactamente en m, n puntos. Inspirados por

esto, haremos el artificio de considerar un punto adicional, que llamaremos punto del infinito, para evitar recurrir la geometría proyectiva (ver la sección §VII.5.A).

La proposición anterior justifica la siguiente construcción geométrica sobre una curva elíptica E con ecuación de Weierstraß $p(x,y)$. Sean $P,Q \in \mathbb{A}_2(\Bbbk)$ dos puntos del plano afín cuyas coordenadas satisfacen $p(x,y)=0$ y con distinta coordenada x, definimos $P+Q$ como el punto de E construido así

1. se traza la recta que pasa por P y Q y se corta con E, obteniendo tres puntos $\overline{PQ} \cap E = \{P,Q,R'\}$;
2. se traza la recta vertical que pasa por R' y se corta con E, obteniendo un segundo punto de corte, sea R;
3. el punto $P+Q$ se define igual a R.

Definición VII.3.3 Sea $\Bbbk \hookrightarrow \Bbbk'$ es una extensión de cuerpos. Si E un curva elíptica sobre $\Bbbk$ en forma de Weierstrass, con ecuación $p(x,y)=0$, entonces definimos los puntos de E con valores en $\Bbbk'$, denotado por $E(\Bbbk')$, como

$$E(\Bbbk') = \{O\} \cup \{(\alpha,\beta) \in \mathbb{A}_2(\Bbbk') \text{ t. q. } p(\alpha,\beta)=0\}$$

donde el punto $O \in E(\Bbbk')$ recibe el nombre de punto del infinito y los demás puntos de $E(\Bbbk')$ se llamarán puntos afines.

Tomaremos el siguiente convenio

- la recta que une un punto afín P con el punto del infinito, es la recta vertical que pasa por P.;
- los puntos de corte de E con la recta vertical $x=\gamma$, son el punto del infinito junto con las soluciones del sistema $p(x,y)=0, x=\gamma$.

De este modo, cualquier recta que corte a E en dos puntos al menos, entonces corta exactamente en tres puntos.

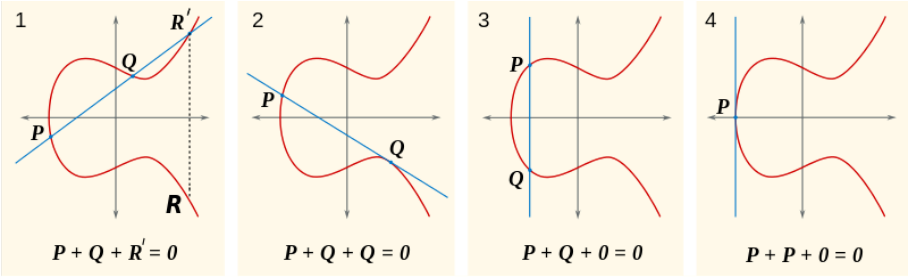

Figura VII.3: Ejemplos de la ley de grupo a partir de construcciones geométricas.

Teorema VII.3.4 Sea $\Bbbk \hookrightarrow \Bbbk'$ es una extensión de cuerpos y E un curva elíptica sobre $\Bbbk$ en forma de Weierstrass. Consideramos la siguiente construcción geométrica, dados dos puntos $P,Q \in E(\mathbb{R})$ definimos $P+Q$ por

1. se traza la recta que pasa por P y Q y se corta con E, obteniendo tres puntos $\overline{PQ} \cap E = \{P,Q,R'\}$ (si $P=Q$, se toma la tangente a P);
2. se traza la recta que pasa por O y R' (la recta vertical que pasa por R') y se corta con E, es decir, $\overline{OR'} \cap E = \{O,R,R'\}$;
3. el punto $P+Q$ se define igual a R.

La ley de composición anterior dota a el conjunto $E(\Bbbk')$ de una estructura de grupo abeliano donde O es el elemento neutro. Además, si $\Bbbk' \subseteq \Bbbk >>$, entonces $E(\Bbbk')$ es subgrupo de $E(\Bbbk >>)$.

Ejercicio VII.3.5 Probar las propiedades de grupo a partir de la construcción geométrica. (No es fácil, pero por ejemplo, $P+Q=Q+P$ y $P+O=P$, o la existencia de elemento opuesto, sí son fáciles). Se puede abordar también con técnicas de geometría algebraica.

Hasta aquí, hemos introducido un nuevo grupo $E(\Bbbk)$ con la intención de reemplazar a $\mathbb{Z}/n$ en las construcciones criptográficas vistas hasta ahora. En primer lugar, nos tenemos que asegurar que tiene suficientes puntos y que su aritmética es aún más compleja que la de $\mathbb{Z}/n$. Estos aspectos tienen que ver con el Teorema de Hasse y el problema del logaritmo discreto para curvas elípticas, respectivamente.

Teorema VII.3.6 — Teorema de Hasse-Weil. Sea $\Bbbk = \mathbb{F}_q$ un cuerpo finito con q elementos y sea E una curva elíptica sobre $\mathbb{F}_q$.

Entonces, $E(\mathbb{F}_q)$ es finito y el número de puntos racionales de E, $\#E(\mathbb{F}_q)$, verifica la siguiente desigualdad

$$|\#E(\mathbb{F}_q) - (q+1)| \leq 2\sqrt{q}\,.$$

Teorema VII.3.7 Sea E una curva elíptica sobre un cuerpo $\Bbbk$. Entonces, $E(\Bbbk) \simeq \mathbb{Z}/m \times \mathbb{Z}/n$ con $m|n$ (se admite el caso $m=1$, i. e. $E(\Bbbk)$ cíclico).

Concluimos que $E(\Bbbk)$ es un grupo con un número de elementos del orden de $\#\Bbbk$ y con una aritmética al menos tan compleja como los grupos cíclicos (estamos pensado en $\Bbbk^*$). Por todo ello, es razonable estudiar la criptografía basada en este tipo de grupos.

En particular, dada E y un punto P, tendremos un grupo finito (y cíclico) dado por

$$\langle P \rangle := \{O, P, 2P, 3P, \ldots\}$$

El procedimiento geométrico anterior permite obtener fórmulas para las coordenadas del punto suma $P+Q$ en función de las coordenadas de los puntos P y Q.

Fórmulas de adición en curvas elípticas

Las expresiones explícitas para $E = y^2 = x^3 + ax + b$ (en característica distinta de 2 y 3), y para $P = (x_P, y_P) \neq Q = (x_Q, y_Q)$, son las siguientes:

- denotamos $R = P+Q$ cuyas coordenadas (x_R, y_R) queremos determinar,
- si $Q = O$, entonces $P+Q = P+O = P = R$, es decir $(x_R, y_R) = (x_P, y_P)$;
- si $Q = -P$, entonces $P+Q = P+(-P) = O$;
- en los demás casos, $R = P+Q$ donde

$$x_R = \lambda^2 - x_P - x_Q \quad , \quad y_R = \lambda(x_P - x_R) - y_P$$

donde $\lambda = \frac{y_Q - y_P}{x_Q - x_P}$ si $x_P \neq x_Q$ y $\lambda = \frac{3x_P^2 + a}{2y_P}$ si $x_P = x_Q$.

Observación VII.3.8 Para hacer la demostración de las fórmulas anteriores, pruébese en primer lugar que la recta que une $P = (x_P, y_P)$, $Q = (x_Q, y_Q)$ tiene ecuación $y = \lambda x + \nu$ con $\nu = y_P - \lambda x_P$. A partir de esto, siguiendo la construcción geométrica, es fácil deducir las fórmulas para $R = (x_R, y_R)$.

Ejercicio VII.3.9

- Estudiar la expresión analítica de la suma de una curva elíptica en forma de Weierstrass para los casos de característica 2 y 3.
- Estudiar el caso $P = Q$, es decir, las coordenadas de $2P$. En este caso, la recta $\overline{PQ}$ es la tangente a E en P y, entonces, $R = 2P$ y las correspondientes fórmulas se denominan fórmulas de duplicación.

Ejercicio VII.3.10 Hacer la suma de $P = (-1,4)$ y $Q = (2,5)$ en la curva $E = y^2 = x^3 + 17$ sobre $\mathbb{Q}$.

Solución: Siguiendo las fórmulas, tenemos $\lambda = \frac{y_Q - y_P}{x_Q - x_P} = \frac{5-4}{2-(-1)} = \frac{1}{3}$. Entonces R tiene coordenadas $x_R = (\frac{1}{3})^2 - (-1) - 2 = -\frac{8}{9}$, $y_R = \frac{1}{3}((-1) + \frac{8}{9}) - 4 = -\frac{109}{27}$.

Problema VII.3.11 Consideremos la cúbica $y^2 = x^3 - x + 1$ y E la curva proyectiva que define. Se pide

- el discriminante de la curva y determinar si es irreducible,
- los puntos racionales,
- la ley de grupo,
- el subgrupo generado por $(1,0)$,

para $\Bbbk = \mathbb{F}_2, \mathbb{F}_4, \mathbb{F}_3$.

Problema VII.3.12 Calcular todos los puntos de la curva $y^2 = x^3 + 2x + 4$ sobre $\mathbb{F}_7$. Calcular sus subgrupos cíclicos, es decir, los generados por cada uno de los puntos.

Solución: Si el cuerpo base es muy grande, el cálculo explícito de todos los puntos puede llegar a ser un problema de una gran complejidad. Precisamente por esto, los ataques de fuerza bruta no son viables. En este caso, el cuerpo es pequeño, cuando x va tomando los valores $0, 1, \ldots, 6 \in \mathbb{F}_7$, queda una ecuación cuadrática que podemos resolver. De este modo, se obtienen todos sus puntos afines $(0,2), (0,5), (1,0), (2,3), (2,4), (3,3), (3,4), (6,1), (6,6)$.

Siguiendo las fórmulas anteriores, tenemos

$$\begin{aligned}\langle (0,2)\rangle = \{&O, (0,2), 2\cdot(0,2) = (2,4), 3\cdot(0,2) = (6,6),\\ &4\cdot(0,2) = (3,3), 5\cdot(0,2) = (1,0), 6\cdot(0,2) = (3,4),\\ &7\cdot(0,2) = (6,1), 8\cdot(0,2) = (2,3), 9\cdot(0,2) = (0,5)\}\end{aligned}$$

y, de hecho, $(0,2)$ tiene orden 10, equivalentemente $10\cdot(0,2) = O$ por lo que es un generador y $E(\mathbb{F}_7) \simeq \mathbb{Z}/10$. Utilizando este hecho, es fácil determinar los subgrupos generados por los demás elementos. Véase `https://andrea.corbellini.name/ecc/interactive/modk-mul.html`.

Problema VII.3.13 Consideramos la curva $E = y^2 = x^3 + x + 1$ definida sobre $\mathbb{F}_5$. Calcular sus puntos racionales. Probar que $E(\mathbb{F}_5)$ es un grupo de orden 9.

VII.4 Aplicaciones a la Criptografía

Un problema previo a los algoritmos de cifrado y descifrado basados en curvas elípticas consiste en la asociación de entre mensajes m y puntos M de la curva. Es el análogo a cuando en los cifrados de tipo César establecíamos una correspondencia entre letras y números.

Si bien es cierto que no tenemos algoritmos eficientes para determinar todos los puntos de $E(\mathbb{F}_q)$, sí se puede abordar utilizando métodos probabilísticos: la asociación $m \leftrightarrow M$ funcionará con una probabilidad tan alta como se quiera y fijada a priori.

Correspondencia entre números y puntos de una curva

Supongamos que los símbolos del alfabeto se representan por números $m = 1, 2, \ldots, m_p$. Tomamos $k \gg 0$ (p. ej. $k > 50$) y q tales que $q = \#\mathbb{F}_q > m_p \cdot k$, que nos asegura que la curva $E(\mathbb{F}_q)$ tiene muchos puntos racionales. La asociación $m \leftrightarrow M$ es como sigue:

- Dado m, hacemos $j = 1$ y consideramos $x = m \cdot k + j \in \mathbb{F}_q$.
- Para este x, intentamos resolver $y^2 = x^3 + ax + b$. Si y es la solución, entonces (x, y) es el punto de E asociado a m. Si no tiene solución, incrementamos j en 1 y volvemos a intentarlo.
- Recíprocamente, dado un punto $M = (x, y) \in E$, el valor m asociado es $m := [\frac{\bar{x}-1}{k}]$ siendo $\bar{x}$ un entero cuya clase en $\mathbb{F}_q$ sea x.

La probabilidad de que esta asociación falle (que no sea biyectiva en un m) es 2^{-k} ya que aproximadamente la mitad de los elementos de $\mathbb{F}_q$ son cuadrados.

A continuación ya podemos usar la aritmética de las curvas elípticas para diseñar un sistema de criptografía asimétrica. El siguiente sistema está inspirado en ElGamal.

Preparación de ElGamal con Curvas Elípticas

Se fijan los siguientes datos: p un número primo pues trabajaremos en $\mathbb{F}_p$, E una curva elíptica sobre $\mathbb{F}_p$, $P \in E(\mathbb{F}_p)$ un punto racional, n el orden del P esto es el orden del subgrupo generado por P. Estos datos son conocidos por todos. Cada usuario tiene una clave privada que consiste en un entero d entre 1 y $n-1$, y una clave pública que es el punto de E dado por $Q := d \cdot P = P + \ldots + P$. El usuario da a conocer la clave pública y mantiene en secreto la privada.

ElGamal con Curvas Elípticas

Cifrado:

- Sea m el mensaje que se va a enviar, y $M \in E$ el punto de la curva que le asociamos a m.
- Tomamos k al azar con $1 \leq k \leq n-1$.
- Calculamos la pareja de puntos $C_1 = k \cdot P$, $C_2 = M + k \cdot Q$, siendo Q la clave pública del destinatario.
- Se envía (C_1, C_2).

Descifrado:

- Recibimos los datos (C_1, C_2) que representan dos puntos de E.
- Siendo d nuestra clave privada, recuperamos el punto M mediante el cálculo $M = C_2 - d \cdot C_1$, ya que

$$(M + k \cdot Q) - d \cdot k \cdot P = (M + k \cdot d \cdot P) - d \cdot k \cdot P = M$$

- recuperamos el mensaje m como el texto asociado al punto M.

Conviene hacer la comparación paso a paso con el sistema del ElGamal, mientras allí era un grupo multiplicativo, ahora estamos usando un grupo aditivo. Mientras que allí la seguridad dependía del logaritmo discreto, aquí va a depender del problema del logaritmo discreto para curvas elípticas (ECDLP).

Definición VII.4.1 Fijada una curva elíptica $E(\Bbbk)$ y un punto P, el problema del logaritmo discreto para curvas elípticas es el siguiente: dado un punto $Q \in \langle P \rangle$, encontrar un k tal que $Q = k \cdot P$. En ese caso, escribiremos $k = \log_P Q$.

Observación VII.4.2 La criptografía de curvas elípticas se usa ampliamente desde 2004-2005, especialmente con protocolos para cifrado tipo Diffie-Hellman y para firma digital. El National Institute of Standards and Technology (NIST) ha establecido algoritmos para estos protocolos que, en particular, incluyen recomendaciones sobre las curvas y los cuerpos que se han de usar. Esto es así porque, de modo similar a como en RSA hay que elegir cuidadosamente los primos p,q, ahora la fortaleza del sistema depende de una buena elección de las curvas y los cuerpos. Algunos autores han expresado dudas sobre la completa seguridad de los consejos del NIST. En relación con esto, y de modo similar a la generación «aleatoria» de números primos, ahora será necesario dar construcciones de curvas elípticas «aleatorias».

VII.5 Temas para ampliar

VII.5.A Geometría Proyectiva y Geometría Algebraica

Aunque hemos trabajado con las cúbicas en el plano afín y realizado cálculos con sus puntos, la interpretación más natural de estos objetos surge dentro la Geometría Proyectiva. Así, dada una cúbica con polinomio $p(x,y)$ de grado 3 es fácil probar que existe una única curva irreducible de grado 3 en el plano proyectivo $\mathbb{P}_2$, con variables homogéneas (x_0,x_1,x_2) y $x = \frac{x_1}{x_0}$, $y = \frac{x_2}{x_0}$, que coincide con la cúbica en el abierto afín $x_0 \neq 0$ y corta a $x_0 = 0$ en un único punto, llamado punto del infinito, y que ha sido denotado anteriormente por O. Esta curva proyectiva viene dada por un polinomio homogéneo de grado 3, $P(x_0,x_1,x_2)$, tal que

$$P(1, \frac{x_1}{x_0}, \frac{x_2}{x_0}) = p(x,y) \tag{VII.1}$$

y sus puntos con valores en una extensión $\Bbbk \hookrightarrow \Bbbk'$ será el conjunto

$$E(\Bbbk') := \{(a_0,a_1,a_2) \in \mathbb{P}_2 \,|\, P(a_0,a_1,a_2) = 0\}$$

En la construcción geométrica de la ley de grupo sobre el conjunto $E(\Bbbk')$ es clave el Teorema de Bezout que afirma que dos curvas proyectivas de grados m y n sin componentes comunes se cortan exactamente en $m \cdot n$ puntos (contados con multiplicidad). Por tanto, usando geometría proyectiva podemos probar que la construcción geométrica vista (trazar rectas que unen dos puntos y cortar estas rectas con una cúbica no singular) permite definir una la ley de grupo en cualquier cúbica no singular del plano proyectivo. En ese sentido, se puede elegir cualquier punto de la curva como elemento neutro.

Además, podemos evaluar una función racional $\frac{A(x_0,x_1,x_2)}{B(x_0,x_1,x_2)}$, i. e. un cociente de dos polinomios homogéneos del mismo grado, sobre los puntos de E. De este modo, una función racional tiene asociado una suma formal $\sum_{P \in E} n_P P$ siendo n_P el orden del cero

(cero de A) o del polo (cero de B) de la función racional sobre E. Una suma de este tipo se denomina divisor. Es fácil probar que x_1 es una función con un polo doble en el punto del infinito y que x_2 es una función con polo triple. Consecuentemente, si definimos el grado de x igual a 2, el de y igual a 3, el de a_i, b_i igual a i, entonces cada uno de los sumandos de $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ tiene grado 6. En el caso $y^2 = x^3 + ax + b$, el grado de a es 4 y el de b es 6. El grado del discriminante sería 12 y el del invariante j es 0. Esto entronca con el estudio de las formas modulares.

Podemos ir más allá en nuestra interpretación y usar el lenguaje de la Geometría Algebraica. En este caso, una curva elíptica es una esquema propio de dimensión 1, no singular e íntegro. Usando las notaciones anteriores, escribimos

$$E = (P(x_0, x_1, x_2))_0 \subset \mathbb{P}_2 = \operatorname{Proj} \Bbbk[x_0, x_1, x_2] \,.$$

Así tendremos que E es el esquema proyectivo $\operatorname{Proj} \Bbbk[x_0, x_1, x_2]/(P(x_0, x_1, x_2))$ y las funciones anteriores, x_0, x_1, x_2 se interpretan como una base del espacio $H^0(E, \mathscr{O}(3O))$ siendo O el punto del infinito.

Con el formalismo de geometría algebraica, la intersección con el abierto afín $x_2 \neq 0$ viene descrita por los ceros de $p(x,y) \in \Bbbk[x,y]$, esto es, es la curva cuyo anillo de funciones es $\Bbbk[x,y]/p(x,y)$

$$\big(p(x,y)\big)_0 = \operatorname{Spec}\big(\Bbbk[x,y]/p(x,y)\big) \hookrightarrow \operatorname{Spec} \Bbbk[x,y] = \mathbb{A}_2$$

Desde esta perspectiva, se tiene una identificación

$$\text{Parte afín de } E(\Bbbk') \simeq \operatorname{Hom}_{\Bbbk-\text{álg}}(\Bbbk[x,y]/p(x,y), \Bbbk')$$

De este modo, la imagen de un polinomio $q(x,y) \in \Bbbk[x,y]$ via $\Bbbk[x,y] \to \Bbbk[x,y]/p(x,y)$ da lugar a una función sobre E. Intuitivamente esta función asocia a un punto (α, β) de la curva el valor $q(\alpha, \beta)$.

- [9] contiene un estudio sobre los puntos racionales de las curvas elípticas con técnicas de geometría algebraica y teoría de números.
- `https://homepages.warwick.ac.uk/~maskal/MA426_EllipticCurves_2018.pdf`

VII.5.B Algoritmos y Curvas Elípticas

ECDLP

La noción abstracta de logaritmo podría llevarse a cabo en cualquier grupo cíclico G con un generador fijado g o, más general, en el subgrupo de un grupo finito generado por un elemento prefijado. Así, el caso más simple, $G = \mathbb{Z}/n$, el problema del logaritmo discreto (DLP) consiste en resolver $mg = b \bmod n$ (conocido b), que tiene complejidad polinómica en el número de bits de n (basta aplicar el algoritmo de Euclides), es decir, $O(\log n)$. El siguiente caso corresponde al grupo $G = \mathbb{F}_q^*$ y la complejidad del mejor algoritmo conocido es $O(\exp(c(\log q)^{1/3}))$, por lo que es inviable para $p > 2^{2048}$. En el caso de curvas elípticas se considera el subgrupo de $E(\mathbb{F}_q)$ generado por un punto P, i. e. el ECDLP. Con una buena elección de E y P, el mejor algoritmo tiene una complejidad $O(\sqrt{p})$, por lo que en la actualidad tomar $p > 2^{200}$ sería tan seguro como $p > 2^{2048}$ en el DLP. Los ordenadores cuánticos podrían resolver estos problemas en tiempos polinómicos, por lo que estos sistemas dejarían de ser seguros.

Aplicaciones a la Factorización

Existe una generalización de la factorización con base de factores que utiliza curvas elípticas. Se conoce como factorización de Lenstra.

Ver pág 35 de `https://homepages.warwick.ac.uk/~maskal/MA426_EllipticCurves_2018.pdf`

También existen pruebas de primalidad basadas en curvas elípticas. Factorización con curvas elípticas de Lenstra (bueno si el factor más pequeño tiene entre 13 y 47 cifras y el siguiente menor es mucho mayor).

Referencias para este Capítulo

- [2] Capítulo 3
- [4] Capítulo 8
- [5] Capítulo VI
- [6] Capítulo 6
- [9] Para profundizar

VIII. Funciones Resumen (Hash): Aplicaciones

VIII.1 Funciones Resumen

Si bien la criptografía en un sentido clásico se ocupa del cifrado, es cierto que en un sentido moderno puede incorporar técnicas que permiten aplicaciones y protocolos muy interesantes y que se usan actualmente de modo muy amplio. En este capítulo veremos como la combinación de las técnicas criptográficas de los capítulos anteriores con un nuevo ingrediente, *las funciones hash*[1] *o resumen*, nos permiten diseñar e implantar tanto protocolos como aplicaciones de gran utilidad hoy en día. A modo de ilustración, entre estos protocolos encontramos los siguientes:

- la autenticación: indentificarse ante un tercero, impidiendo la suplantación de la identidad;
- el no repudio: no poder negar la autoría de un documento;
- la firma digital: demostrar la autoría de un documento;
- la integridad: bloquear un documento, de modo que se detecte si se modifica posteriormente;
- las pruebas de conocimiento cero: probar que conocemos una información sin necesidad de desvelarla.

Entre las aplicaciones, fruto de la combinación de los elementos anteriores, cabe citar la compra segura y pasarelas de pago, las votaciones electrónicas, sedes electrónica de instituciones, gestión de acceso mediante contraseña, juegos de cartas por internet, subastas electrónicas, etc.

[1] To hash: trocear. Hash: picadillo.

Este capítulo contiene una introducción básica a las funciones resumen. Supondremos de aquí en adelante que las funciones resumen usadas son conocidas públicamente y que el cómputo explícito (con una entrada dada) se puede realizar de modo eficiente.

Definición VIII.1.1 Una función $H : X \to Y$ se dice que es una función resumen de sentido único o función hash cuando verifique:

- el argumento de entrada $x \in X$ puede ser de longitud arbitraria pero el resultado, $H(x)$, tiene siempre una longitud fija.
- dado $y \in Y$, es difícil calcular un x con $H(x) = y$ (resistente a antiimágenes),
- dados $x \in X$ y $H(x)$, es difícil calcular $x' \neq x$ con $H(x) = H(x')$ (resistente a segundas antiimagénes),

donde fácil y difícil se refiere al tiempo de cómputo (complejidad).

Definición VIII.1.2 Una función $H : X \to Y$ se dice que es una función resumen resistente a colisiones o *collision resistant hash function* cuando sea una función resumen de sentido único y, además, sea resistente a colisiones, esto es, es difícil encontrar x, x' con $H(x) = H(x')$.

Entre las funciones hash utilizadas comúnmente destacaremos dos grandes familias, MD y SHA.

Observación VIII.1.3 La comprobación rigurosa de que una función cumple las propiedades anteriores no suele ser fácil. Incluso hay veces que se usan funciones solamente porque la experiencia muestra que funcionan pero sin demostración matemática rigurosa. Es una situación parecida a la comprobación de si un procedimiento de generación de números es o no aleatorio, o sobre la complejidad del problema del logaritmo discreto para curvas elípticas.

Como función hash «de juguete» (suficiente para poner ejemplos sencillos y aprender cómo operar con ellas) podemos tomar la exponencial, ya que el cálculo de antiimágenes (basándose en el logaritmo discreto) es difícil. Esto es, si p es un primo, g un generador de $(\mathbb{Z}/p)^*$, entonces definimos

$$\begin{aligned} H\colon \{1,2,\ldots,N\} &\to (\mathbb{Z}/p)^* \\ a &\mapsto g^a \end{aligned}$$

Observar que si $N < p$ entonces H será inyectiva. Obsérvese que aunque $N \geq p$, la función H seguirá siendo una función hash cuya salida tiene longitud fija, i. e. el número de bits es el mismo independiente de la entrada. Esto se suele tomar siempre así y de hecho es una ventaja.

VIII.2 Message-digest Hash Function

Message-digest hash function, también conocido por sus iniciales MD hash, combina una función de compresión f con una función de salida g y permite construir funciones resumen H resistentes a colisiones (*collision-resistant hash functions*) bajo ciertas hipótesis.

ITERACIÓN DE FUNCIONES RESUMEN

- Se fragmenta el mensaje en bloques de la misma longitud: $M_1 M_2 \dots M_m$,
- se fija un estado inicial S_0 (conocido públicamente),
- se evalúa sucesivamente:

$$S_i := f(S_{i-1}, M_i) \qquad \text{para } i = 1, \dots, m$$

- entonces $H(X) := g(S_m)$ es el resultado.

Basándose en esta idea, Damgård-Merkle propusieron una construcción de funciones resumen resistentes a colisiones (*collision-resistant hash functions*) por medio de una combinación de una función de compresión f con una función resumen resistente a colisiones con entradas de longitud fija H.

MD5 (*message-digest algorithm*) fue diseñado en 1991 por R. Rivest para sustituir el anterior sistema MD4 que había quedado obsoleto. Consiste en la división de la cadena original en fragmentos sobre los que se aplica una función de compresión (igual que SHA).

Aunque inicialmente se diseñó para usarla como función hash, se han descubierto posibles vulnerabilidades (concretamente, en 2013 Xie Tao, Fanbao Liu, and Dengguo Feng dieron un método para encontrar colisiones que necesita menos de un segundo en un ordenador común). Desde 2012 se la considera criptográficamente insegura, pero lo cierto es que se sigue utilizando ampliamente. La recomendación es no usarla para fines criptográficos y restringir su uso a una función de comprobación de corrupción no intencionada (al igual que el bit de paridad, o el CRC).

No obstante lo anterior, es ilustrativo aprender su funcionamiento.

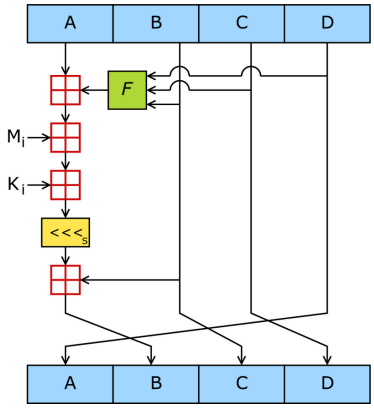

Figura VIII.1: Representación esquemática de la iteración i-ésima en el sistema MD5. Las cajas azules en la parte superior representan el estado inicial. Las cajas rojas representan la suma o XOR y la caja amarilla $<<<_s$ una rotación de s bits. Las cajas azules en la parte inferior representan el estado final. (Wikimedia Commons).

Algoritmo MD5

Dada una mensaje/cadena/fichero de longitud arbitraria, MD5 genera un resumen/huella de 128 bits del siguiente modo:

- Al mensaje de entrada se le añade un bit 1 al final, después se añaden bits 0 hasta llegar a una longitud que sea múltiplo de 512 menos 64. Finalmente se añaden los últimos 64 bits que contienen la longitud del fichero original (módulo 2^{64}). A continuación, el resultado obtenido, cuya longitud es ahora múltiplo de 512, se divide en bloques de 512 bits. Cada bloque de 512 bits se divide en palabras de 32 bits (16 palabras en total, $M_1 \dots M_{16}$).
- Se opera con un estado formado por 128 bits, divididos en cuatro palabras de 32 bits denominadas A, B, C y D. Estas palabras tienen unos valores iniciales fijos:

$$\begin{aligned} A &= 0X67452301, \\ B &= 0XEFCDAB89, \\ C &= 0X98BADCFE, \\ D &= 0X10325476 \end{aligned}$$

- El procesamiento de cada bloque modifica el estado y consta de cuatro «rondas». En cada una de estas cuatro rondas se utiliza una función no lineal. La función no lineal usada en cada una de las rondas es:

$$\begin{aligned} F(X,Y,Z) &= (X \text{ AND } Y) \text{ OR } ((\text{ NOT } X) \text{ AND } Z) \\ G(X,Y,Z) &= (X \text{ AND } Z) \text{ OR } (Y \text{ AND } (\text{ NOT } Z)) \\ H(X,Y,Z) &= X \text{ XOR } Y \text{ XOR } Z \\ I(X,Y,Z) &= Y \text{ XOR } (X \text{ OR } (\text{ NOT } Z)) \end{aligned}$$

  es decir, se usa F en la primera ronda, G en la segunda, H en la tercera e I en la cuarta.
- Cada ronda realiza 16 iteraciones (una por cada palabra de $M_1 \dots M_{16}$). Cada iteración consiste básicamente en la aplicación de la función correspondiente a la ronda, una suma modular y una rotación a la izquierda (junto con una clave conocida K_i y una rotación final de s_i posiciones). Está descrito con precisión en la imagen siguiente.
- El resultado son los nuevos $ABCD$ que se emplean en la segunda ronda, en la que vuelven a intervenir $M_1 \dots M_{16}$. Luego la tercera y la cuarta.

VIII.3 Familia de algortimos SHA

La familia de algoritmos SHA (*secure hash algorithm* comprende seis algoritmos: SHA-N siendo $N = 0, 1, 224, 256, 384, 512$. Los cuatro primeros operan sobre bloques de entrada de 512 bits mientras que los dos últimos lo hacen sobre bloques de 1024 bits. La longitud del bloque de salida es 160 para $N = 0, 1$ y N para los restantes.

El primero de estos algoritmos SHA-0, inicialmente conocido simplemente como SHA, data de 1993 y fue introducido por el NIST. Desde entonces, se han sucedido las distintas versiones que han sido homologadas por distintas instituciones. Aunque se conocen ataques, esto es, algoritmos para determinar colisiones, para SHA-0 y SHA-1, las últimas generaciones de esta familia están adaptadas para prevenir colisiones al nivel de seguridad requerido por los tres niveles de seguridad del AES(128, 192 y 256 bits respectivamente).

El algoritmo SHA-1 se emplea ampliamente en multitud de protocolos, como son TLS, SSL, PGP, SSH, S/MIME y IPsec. Actualmente ya está diseñada y estandarizada la siguiente versión, SHA-3. Tanto el MD5 como el SHA-1 son descendientes del MD4 y su estructura tiene claras similitudes. Para ilustrar este hecho, describimos sucintamente los pasos del SHA-1.

- SHA-1 actúa sobre bloques de entrada de 512 bits, por lo que el mensaje de entrada se divide en bloques de 512 bits, y el algoritmo se aplica sucesivamente sobre cada bloque.
- Se inicializan 5 variables de estado A, B, C, D, E, de 32 bits cada una con unos valores dados para el primer bloque y con el resultado de la iteración anterior para los bloques sucesivos.
- Partiendo del bloque de entrada con 512 bits, para $i = 0, \ldots, 15$, se define W_i igual a cada subbloque de 32 bits en los que se puede dividir. Y para $i = 16, \ldots, 79$, se define W_i por operaciones lógicas a partir de los W_j con $j < i$.
- Se consideran unas constantes, subclaves, K_i para $0 \leq i \leq 79$. Realmente toman cuatro valores, un valor en cada uno de los siguientes intervalos de la variable i que indica la iteración: $0 \leq i \leq 19$, $20 \leq i \leq 39$, $40 \leq i \leq 59$ y $60 \leq i \leq 79$.
- Se consideran unas funciones no lineales, f_i, construidas con operaciones lógicas. Al igual que antes, es una función para el intervalo $0 \leq i \leq 19$, otra para los intervalos $20 \leq i \leq 39$ y $60 \leq i \leq 79$, y una última para $40 \leq i \leq 59$.
- Se realizan 80 rondas siguiendo el esquema del diagrama adjunto.

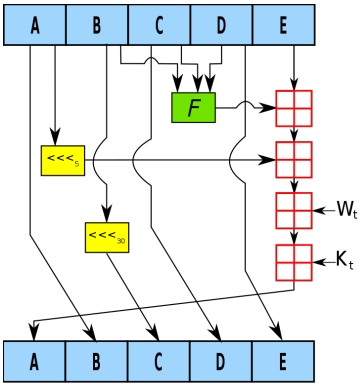

Figura VIII.2: Representación esquemática de la iteración i-ésima en el sistema SHA-1. Las cajas azules en la parte superior representan el estado inicial. Las cajas rojas representan la suma o XOR y la caja amarilla $<<<_s$ una rotación de s bits. La caja verde con la función F consiste en aplicar la función f_i. Las cajas azules en la parte inferior representan el estado final. (Wikimedia Commons).

VIII.4 Firma Digital

En mensajes cortos se puede emplear el método anterior de firma digital que consiste en firmar todo el mensaje.

En mensajes largos m resulta que es más rápido firmar $H(m)$ (que tiene longitud fija) que firmar todo el mensaje.

En el apartado §VI.1.B se explicó cómo se puede firmar con el sistema criptográfico de ElGamal. Pero lo cierto es que dicha firma se puede falsificar, esto es, es posible suplantar la identidad (*existential forguery*). Veamos cómo.

Recordemos que si C quiere suplantar a A, cuya firma es s, entonces tiene que ser capaz de mandar una terna del tipo (x,y,s) tal que:

$$(g^{a_A})^x \cdot x^y = g^s \qquad \mod p$$

donde p es el número primo, g es el generador, y g^{a_A} es la clave pública de A. Para conseguir su objetivo, C toma dos enteros u,v con $(v,p-1)=1$. Entonces considera:

$$\begin{aligned} x &= g^u (g^{a_A})^v && \mod p \\ y &= -xv^{-1} && \mod p-1 \\ s &= yu && \mod p-1 \end{aligned}$$

y envía la terna (x,y,s) ya que esta verifica la condición de validez. (Ojo, aunque se verifique la condición, s no tiene porque ser la firma de A).

A continuación veamos que podemos evitar suplantaciones de este tipo. Para ello se modifica el proceso de firma del siguiente modo que requiere la utilización de una función *hash*.

Firma Digital con ElGamal

- Si A quiere mandar su firma s a B, entonces elige k al azar con $(k,p-1)=1$, calcula $r = g^k \mod p$ que se toma como un entero y determina:

  $$t = k^{-1}(H(s) - a_A r) \qquad \mod p-1$$

  y envía la terna (r,t,s).
- B recibe (x,y,s) y debe comprobar si $1 \le r < p$ y que $(g^{a_A})^x \cdot x^y$ y $g^{H(s)}$ coinciden módulo p. En caso contrario, rechaza la firma.
- Si C quiere suplantar la identidad de A entonces tendría que ser capaz de encontrar s tal que $H(s) = y$ módulo $p-1$ pero esto es imposible por ser H una función *one-way*.

Aún con esto se podría conseguir la suplantación. Véase [1, pág. 226] para la explicación y una solución a este problema.

El DSA (*digital signature algorithm*) es una variante eficiente de ElGamal que reduce el número de exponenciaciones de 3 a 2 y, de modo más relevante, el número de dígitos en los exponentes es 160 en vez de tantos bits como el número primo p (al menos 768). Veamos las distintas etapas ([1, pág. 228]).

DIGITAL SIGNATURE ALGORITHM (DSA)

- Generación de claves. A toma un primo q con 160 bits, es decir, $2^{159} < q < 2^{160}$. También toma un primo p con $2^{511+64t} < p < 2^{512+64t}$ para algún t entre 0 y 8 y de modo que q divida a $p-1$. Entonces resulta que el número de bits de p está entre 512 y 1024 bits y es múltiplo de 64, es decir, p escrito en binario tiene entre 8 y 16 bloques de 64 bits. La condición $q|p-1$ implica que $(Z/p)^*$ contiene elementos de orden q. Toma ahora un generador x de $(Z/p)^*$ y computa:

  $$g = x^{(p-1)/q} \mod p$$

  que tiene orden q. Por último, elige a al azar con $1 \leq a \leq q-1$ y computa:

  $$g^a \mod p$$

  Entonces la clave pública de este usuario es (p,q,g,g^a) y la clave privada es a. Determinar la clave privada a partir de estos datos equivaldría a computar un logaritmo discreto en $(Z/q)^*$ que tiene unos 2^{160} elementos.
- Generación de firma. Se elige una función *hash* H. Entonces A toma un número al azar y calcula:

  $$r = (g^k \mod p) \mod q$$

  que se interpretará como un entero y

  $$s = k^{-1}(H(x) + ar) \mod q$$

  La firma del mensaje x es (r,s).
- Verificación. B recibe la firma (r,s) del mensaje x. B conoce la clave pública de A que es (p,q,g,g^a) y la función H. En primer lugar, comprueba si

  $$1 \leq r \leq q-1 \text{ y } 1 \leq s \leq q-1$$

  si no es así, rechaza el mensaje. En otro caso B continua y comprueba si se verifica que:

  $$r \equiv \left((g^{s^{-1}H(x) \bmod q} (g^a)^{rs^{-1} \bmod q}) \bmod p \right) \bmod q$$

  si no es así, rechaza el mensaje.

Si en este contexto, nos proponemos ahora usar el sistema RSA para firmar entonces hemos de fijar una función hash H y procederemos de la siguiente forma.

Firma con RSA

Si tengo claves públicas (n_A, e_A) y privada d_A y quiero enviar un mensaje firmado M a un destinatario con claves (n_B, e_B) y d_B, realizo lo siguiente:

- calculo $N = M^{e_B} \bmod n_B$.
- calculo $M' = H(M)$, y lo firmo $F = (M')^{d_A} \bmod n_A$.
- envío (N, F).
- B recibe (N, F) y descifra calculando $N^{d_B} \bmod n_B$ (recupera M).
- verifica la firma comprobando que $F^{e_A} = H(M)$.

Finalmente, vamos a ver el protocolo de firma digital usando RSA sobre curvas elípticas. Recordemos que en esta situación, tenemos fijados los siguientes datos y que son conocidos por todos los usuarios:

$$(q, \mathbb{F}_q, S, a, b, P, n, h, H)$$

donde:

- $\mathbb{F}_q$ es el cuerpo base, es un cuerpo finito con q elementos,
- S es una semilla,
- a, b son los valores a, b que determinan la curva elíptica en la forma de Weierstrass, $E = y^2 = x^3 + ax + b$,
- P es un punto de la curva elíptica,
- n es el orden del subgrupo generado por P,
- h es el orden del grupo cociente $E(\mathbb{F}_q)/\langle P \rangle$, igual a $\#E(\mathbb{F}_q)/n$,
- H es una función hash.

Firma con RSA sobre curvas elípticas

Supongamos que tenemos clave privada d y clave pública Q. Entonces para firmar un mensaje m, se procede así:

- se toma k al azar con $1 \leq k \leq n-1$,
- se calcula $kP = (x_1, y_1)$, elijo un entero $\bar{x}_1$ cuya clase coincida con x_1,
- se toma $r := \bar{x}_1 \bmod n$, si $r = 0$, se repite desde el principio,
- se calcula $e := H(m)$,
- se calcula $s = k^{-1}(e + dr) \bmod n$. Si $s = 0$, se repite desde el principio,
- se envía la pareja (r, s) como firma del mensaje m.

El receptor, para comprobar la validez, procede del siguiente modo:

- se comprueba si r, s están en el rango $\{1, \ldots, n-1\}$; si no están, se rechaza la firma,
- se calculan los siguientes
  - $e = H(m)$,
  - $w = s^{-1} \bmod n$,
  - $u_1 = ew$ y $u_2 = rw$ módulo n,
- se calcula $X = u_1 P + u_2 Q$,
- si X es el elemento neutro (el punto del infinito), se rechaza la firma,
- sean (x_1, y_1) las coordenadas de X, elegimos $\bar{x}_1$ un entero con clase x_1 mod n,
- se acepta la firma si y solamente si $\bar{x}_1 = r \bmod n$.

Veamos porqué este proceso es correcto si la pareja (r,s) es una firma válida:

$$k = s^{-1}(e+dr) = s^{-1}e + s^{-1}rd = we + wrd = u_1 + u_2 d$$

luego

$$X = u_1 P + u_2 Q = (u_1 + u_2 d)P = kP$$

y, por tanto, $\bar{x}_1 = r$.

VIII.5 Temas para ampliar

Esta última sección del capítulo está dedicada a ilustrar cómo la criptografía moderna es capaz de plantear y resolver nuevas aplicaciones que sobrepasan ampliamente la aplicación clásica al cifrado de la información pero que son de gran utilidad en la actualidad. Además de las que detallaremos en los apartados siguientes, hay que mencionar otros usos y protocolos como pueden ser: PGP, esquemas para compartir secretos, pruebas de conocimiento cero, sello de tiempo o timestamp, etc.

VIII.5.A Almacenamiento de Contraseñas

Un uso básico de las funciones hash debería ser el almacenamiento de contraseñas. Decimos «debería» porque en la práctica muchos portales web, etc no están correctamente diseñados y son vulnerables. Basta recordar el típico titular «se ha filtrado la contraseña de 10 millones de usuarios». Mencionaremos de modo sucinto algunas de estas situaciones así como posibles ataques. Evidentemente, si se guarda en el servidor un fichero con nombres de usuario y sus correspondientes contraseñas en texto plano, la filtración de dicho fichero compromete directamente la seguridad. Otro ataque sería la intercepción de la comunicación(*man in the middle*), que debería evitarse estableciendo un canal seguro con las técnicas estándar de criptografía.

La siguiente opción consistiría en almacenar los datos de usuario junto con el *hash* de su correspondiente contraseña. Los ataques diseñados para esta situación son, básicamente, los siguientes:

- fuerza bruta: si la función hash usada ha quedado obsoleta (es decir, el avance tecnológico permite encontrar preimágenes), entonces es fácil conseguir quizás no la contraseña pero sí otra cadena con el mismo hash, que nos daría acceso al sistema.
- diccionario: son colecciones de palabras con sus correspondientes hash calculados. Pueden incluir las palabras de cierto idioma, combinaciones frecuentes, contraseñas frecuentes, etc. Bastaría localizar el hash deseado en dicha tabla.
- tablas *rainbow*: muy parecido al anterior pero con una estructura interna que busca un equilibrio entre espacio usado en el disco y tiempo de acceso o búsqueda y que suelen estar disponibles en internet.

Como cabe suponer, se han diseñado ciertas defensas ante estos ataques:

- utilización de funciones hash avanzadas o iteración de varias funciones hash. De hecho, que el cálculo de $H(x)$ sea rápido es una desventaja, pues da opciones a los ataques de fuerza bruta. Realizar varias iteraciones de H puede hacer perder unos milisegundos por intento, lo que tras millones de intentos, puede convertir el ataque en inviable.

- utilización de sal y pimienta (*salt and pepper*). Consiste en, antes de realizar el hash, combinar la contraseña con un texto específico para ese usuario concreto (sal) o con un texto común para todos (pimienta). De este modo los diccionarios y tablas rainbow quedarían inutilizadas. Obviamente, estos textos han de protegerse de modo muy seguro.

VIII.5.B Computación segura multiparte

Sea f una función de n-variables. Supongamos que tenemos n personas $P_1,\ldots,P_n$, que la persona i-ésima proporciona una valor x_i a la variable i-ésima y que queremos evaluar $f(x_1,\ldots,x_n)$ de modo seguro, esto es, sin que ninguna persona llegue a conocer los valores x_i de las variables correspondientes a las demás personas.

Un procedimiento para realizar este cálculo se denomina protocolo de computación segura multiparte. Se puede entender como un caso de un tercero de confianza ya que el protocolo actúa como un agente que proporciona el valor correcto de f sin desvelar las x_i a ninguna de las partes.

Para ilustrar este procedimiento, tomaremos la versión más simple, conocida como el problema de los millonarios de Yao. En este problema dos millonarios, Alice y Bob, tienen respectivamente i y j millones, y quieren saber cuál es más rico de los dos, pero no quieren revelar exactamente cuánto dinero tienen.

Además, requerimos que se satisfagan las siguientes propiedades:

- seguridad: debe ser inmune a ataques de posibles saboteadores que intenten conocer la riqueza de Alice y Bob,
- privacidad: debe permitir que los dos conozcan la respuesta sin adquirir ningún otro conocimiento sobre la riqueza del otro,
- complejidad: la cantidad de información transmitida es proporcional al rango al que pertenecen i y j.

Hay que indicar que nada impide que uno de los dos o los dos participantes mienta en el valor de su variable. Es una dificultad inherente al problema y no al protocolo en sí, y por tanto no tiene solución. El protocolo debe asegurar que esa es la única trampa que pueden realizar.

Vamos a partir de que Alice y Bob acuerdan emplear el RSA como sistema criptográfico, que el número de millones de cada uno está en el rango de 1 a 10. La clave pública de Alice es $(79,3337)$, su clave privada es 1019 y tiene $i=5$ millones. Bob tiene $j=6$ millones. El procedimiento es el siguiente.

- Bob elige un número x de N bits, por ejemplo, $x=1234$, $N=14$. Bob calcula $C:=RSA(x)=1234^{79} \bmod 3337 = 901$.
- Bob envía a Alice $D:=C-j+1 = 901-6+1 = 896$.
- Alice calcula $Y_1,\ldots Y_{10}$ donde Y_k es el resultado de $(D-k+1)^{1019} \bmod 3337$.

	1	2	3	4	5	6	7	8	9	10
Y_k	1059	1156	2502	2918	385	**1234**	296	1596	2804	1311

- Alice genera un número primo p de $N/2$ bits, por ejemplo, $p=107$. Alice calcula $Z_k:=Y_k \bmod p$.

Z_k	96	86	41	29	64	**57**	82	98	22	27

donde hemos marcado en negrita la entrada $j=6$ para seguir mejor el funcionamiento.

- Alice envía p a Bob. Seguidamente le envía los números: $Z_1, Z_2, \ldots, Z_i, Z_{i+1}+1, Z_{i+2}+1, \ldots, Z_{10}+1$

$$\begin{array}{cccccccccc} Z_k & 96 & 86 & 41 & 29 & 64 & \mathbf{58} & 83 & 99 & 23 & 28 \end{array}$$

- Bob calcula $G := x \bmod p = 1234 \bmod 107 = 57$.
- Bob toma el número en la posición j y lo comprueba con G. Si son iguales, concluye que $i \geq j$, y si son distintos concluye que $i < j$.
- El número en la posición $j = 6$ es 58, como $58 \neq 57$, entonces $i < j$.

Este tipo de protocolo es una parte esencial en las votaciones electrónicas ya que interesa calcular $f(x_1, \ldots, x_n) = \sum_i x_i$, siendo x_i el sentido del voto de la persona P_i. Podría aplicarse también para diseñar un procedimiento de subastas electrónicas, por ejemplo, tomando la función $f(x_1, \ldots, x_n) = \{i | x_i = \max\{x_1, \ldots, x_n\}\}$.

Las votaciones electrónicas tienen unos requisitos adicionales, como son:

- *Democracia*: solo las personas registradas en el censo pueden emitir su voto y solo lo pueden hacer una vez.
- *Transparencia*: ningún voto puede ser alterado o modificado.
- *Privacidad*: no puede establecerse relación entre un voto y un votante.
- *No coercibilidad*: para evitar coacciones el votante no puede demostrar cual ha sido el sentido de su voto.
- *Verificabilidad*: cada votante y eventualmente un auditor pueden comprobar que el voto ha sido correctamente contabilizado.
- El proceso de comunicaciones debe ser práctico y realizable.

En una votación presencial estas condiciones quedan garantizadas por la urna transparente, la cabina de votación y el escrutinio público. Pero su versión electrónico requiere de la combinación de multitud de protoclos, tales como: autentificación, firma digital y/o firma ciega, pruebas de conocimiento cero, reparto de secretos, computación segura multiparte, criptografía homomórfica, etc.

- Schoenmakers, B., «*Multiparty Computation*», en Encyclopedia of Cryptography and Security, van Tilborg, H. C. A. y Jajodia, S. (eds), Springer, Boston, MA.
- Sako, K., «*Electronic Voting Schemes*», en Encyclopedia of Cryptography and Security, van Tilborg, H. C. A. y Jajodia, S. (eds), Springer, Boston, MA.
- [8] §6.4, pág. 192.
- Fuster, A., «*Técnicas criptográficas de protección de datos*», Editorial Ra-ma.

VIII.5.C Compromisos

Se trata de establecer un esquema para que dos partes puedan establecer un compromiso vinculante pero cuyo contenido no se revelará hasta transcurrido un tiempo. De modo más intuitivo, podemos pensar en que A escribe su compromiso en un documento, lo guarda en un cofre cerrado que entrega a B mientras A se queda con la llave. Pasado el tiempo, A entrega la llave a B que puede leer el documento con la seguridad de que no ha sido modificado. Este tipo de procedimientos son parte fundamental en los protocolos de conocimiento cero así como en el reparto de secretos.

De nuevo, recurrimos a continuación a una ejemplo simplificado que esencializa el modo de proceder de este tipo de esquemas. Alice y Bob han de realizar un sorteo o tomar una decisión de manera aleatoria y deciden tirar una moneda al aire. Esta situación cotidiana es fácil si ambos están presentes, pero la pregunta es cómo se puede adaptar para realizar este proceso por internet (o por teléfono, en la versión original que

propuso Blum) para que tanto Alice como Bob tengan confianza en la transparencia del resultado.

Lanzamiento de moneda (Blum)

- Alice elige dos primos p,q tales que $p = 3 \bmod 4$ y $q = 3 \bmod 4$. Sea $n = p \cdot q$. Alice envía n a Bob mientras mantiene p,q en secreto.
- Bob elige al azar un número entero x entre 1 y $n-1$. Calcula su cuadrado $a = x^2 \bmod n$ y le envía el resultado a Alice. Bob mantiene x en secreto.
- Alice, que conoce la descomposición de n, puede computar las raíces cuadras de a módulo n (usando el símbolo de Jacobi y el Teorema Chino de los restos, ver [5, pág. 48]). Estas cuatro raíces tendrán la forma $x, n-x, x', n-x'$ pero no sabe qué número ha elegido Bob. Alice elige una de estas cuatro y se la envía a Bob.
- Bob recibe un número de Alice, si el número es x o $n-x$, entonces Bob comunica que Alice ha ganado. Para comprobar la limpieza, Alice envía a Bob los factores p,q y Bob comprueba que $n = p \cdot q$.
- Si el número que recibe Bob es x' o $n-x'$, entonces conociendo x y x' Bob es capaz de descomponer n. Bob se declara ganador y proporciona a Alice la descomposición de n para demostrarlo.

Si preferimos evitar el uso de funciones resumen, podemos diseñar una alternativa a la propuesta original de Blum.

- Alice y Bob eligen una función hash $H : X \to Y$ donde X es un conjunto de números con el mismo número de pares que de impares (por ejemplo, $(\mathbb{Z}/p)^*$ con p primo impar). La función H será una función resumen resistente a colisiones (ver Definición VIII.1.2).
- Alice elige al azar $x \in X$, calcula $y = H(x)$ y le envía y a Bob.
- Bob guarda y (le servirá como comprobación de que Alice no le engaña).
- Bob elige «par» o «impar» (cara o cruz). Se lo dice a Alice.
- Alice comprueba si la elección de Bob coincide con la paridad de x. Si Bob ha acertado con la paridad de x, entonces Bob gana. En otro caso, Bob pierde y Alice gana.
- Alice comunica a Bob el ganador y también le envía x.
- Bob se asegura de que Alice no le ha engañado (por ejemplo, cambiando x a mitad del proceso) comprobando que $H(x)$ coincide con el valor y guardado. Aquí es fundamental que H sea una función resumen resistente a colisiones, para que Alice no pueda tener dos elementos x, x' con mismo hash $H(x) = H(x')$.

Una adaptación de estas ideas para volver a la idea original del documento guardado en un cofre, sería la siguiente.

- Se acuerda de nuevo usar una función hash $H : X \to Y$. Sea m el mensaje.
- Para el envío se procede así:
  1. A elige $x \in X$, y calcula $y = H(x \,\|\, m)$ (donde $\|$ denota la concatenación de cadenas),
  2. A envía a B los datos x y y. Observemos que B no puede adivinar m a partir de estos datos.
- Llegado el momento para la revelación:
  1. A envía m a B,

2. B se asegura que A no ha cambiado el mensaje comprobando que $y = H(x \parallel m)$.

La aplicación sería válida para proteger la identidad real de unos escritores participantes en un concurso literario bajo pseudónimos, o para la presentación de unas plicas o pujas en una licitación.

Pueden consultarse también las siguientes referencias:

- Blum, M., «*Coin flipping by telephone*», Advances in Cryptography, Santa Barbara, California, USA.
- Fuster, A., «*Técnicas criptográficas de protección de datos*», Editorial Ra-ma.
- [8] §6.2, págs. 184-185.

VIII.5.D SET: Secure Electronic Transaction

Es un protocolo para compras con tarjeta de crédito por internet que fue desarrollado en 1996 por Mastercard, Visa, Microsoft, Netscape, IBM entre otros. Su diseño responde a las siguientes características:

- privacidad: la información ha de ser confidencial, por tanto, no queremos dar nuestro número de tarjeta a la tienda T, y tampoco queremos que el banco B sepa que artículo hemos adquirido,
- integridad: se ha de garantizar la integridad de los datos, evitando modificaciones de estos para que una vez enviado un mensaje, este no se puede repudiar o negar su autoría, etc.
- autenticación: se ha de garantizar las identidades de comprador y vendedor por procesos de autenticación, evitando la suplantación,
- confidencialidad: las comunicaciones han de ser cifradas, de modo que si alguien intercepta alguna o todas las comunicaciones no sea capaz de deducir ningún dato.

Imaginemos que deseamos comprar un artículo en la tienda T y pagar con la tarjeta de crédito del banco B. Se procede así:

1. Preparamos dos documentos: la orden de compra C (contiene que artículo queremos) y la orden de pago P (contiene el número de tarjeta y el importe de la compra). Computamos las imágenes $c = H(C)$ y $p = H(P)$. Además producimos una firma digital para la imagen de la concatenación

   $$S = D_A(H(c \parallel p))$$

   donde D_A es la función que usamos para firmar basada en una clave privada. Encriptamos la concatenación de c, P, S con la clave pública del banco B

   $$M_B = E_B(c \parallel P \parallel S)$$

   y la concatenación de C, p, S con la clave pública de la tienda T

   $$M_T = E_T(C \parallel p \parallel S)$$

   y mandamos M_B y M_T a la tienda T.
2. La tienda T usa su clave privada para descifrar M_T y recupera C, p, S. La tienda verifica la autenticidad de la orden de compra comprobando que

   $$E_A(S) = H(H(C) \parallel p)$$

(siendo E_A la función de encriptación con nuestra clave pública). En caso afirmativo, reenvía M_B al banco B.

3. El banco B usa su clave privada para descifrar M_B y recupera c, P, S. El banco verifica la autenticidad de la orden de pago comprobando que

$$E_A(S) = H(c \,\|\, H(P))$$

de donde P es un pago a favor de la tienda T. Entonces el banco crea un mensaje de autorización M que consiste en un número de transacción, nuestro nombre, y el importe. El banco firma este mensaje (M, F), lo encripta con la clave pública de la tienda T, $E_T(M \,\|\, F)$ y lo envía a la tienda.
4. La tienda recibe el mensaje y lo descifra usando su clave privada y comprueba la autenticidad de la firma verificando que $E_B(F) = M$. La tienda comprueba que nuestro nombre está en M y que el importe es correcto. La tienda nos envía el artículo comprado. La tienda solicita el pago enviando al banco el número de transacción encriptado con la clave pública del banco.
5. El banco recibe el mensaje, lo descifra, y procede al pago correspondiente a dicho número de transacción cargando el importe en nuestra tarjeta.

Referencias:

- «*Secure Electronic Transactions: Credit Card Payment on the Web in Theory and Practice*», IBM International Technical Support Organization, Raleigh Center, Junio 1991, `http://ps-2.kev009.com/rs6000/redbook-cd/SG244978.PDF`
- `https://www.geeksforgeeks.org/secure-electronic-transaction-set-protocol/`

REFERENCIAS PARA ESTE CAPÍTULO

- [1] Capítulos 10 y 11
- [2] Capítulo 4
- [3] Capítulo 3
- [4] Capítulos 6 y 9
- [7] Capítulos 9 y 11

Aritmética Modular

¿Qué es la aritmética modular y por qué es tan relevante en Criptografía?

Es obvio que utilizamos símbolos para comunicarnos. Al conjunto de estos símbolos lo denominamos alfabeto. La idea básica detrás de todos los protocolos criptográficos es muy simple, se ha de realizar una serie de cambios y manipulaciones sobre los símbolos del texto claro del mensaje de modo que el resultado, esto es, el texto cifrado, sea ininteligible salvo para el destinatario legítimo.

Pues bien, si establecemos una correspondencia entre el alfabeto y un conjunto de números, entonces podríamos emplear el conocimiento matemático fruto de siglos de estudio para realizar dichas manipulaciones. Por tanto, ¡qué más sencillo que considerar unos números enteros y la suma y producto! Sin embargo, el alfabeto es un conjunto finito mientras que el conjunto de los números enteros, $\mathbb{Z}$, es infinito. Es aquí donde aparece la aritmética modular y, en consecuencia, trabajaremos con clases de restos al dividir por un número fijado, que sí son un conjunto finito. No obstante, en las últimas décadas se ha desarrollado la criptografía sobre conjuntos que no son números, sino puntos de curvas (ver el Capítulo VII).

Aunque se presupone que el lector tiene unos conocimientos básicos de álgebra, este capítulo resume las nociones básicas de la aritmética modular, ayuda a introducir y fijar las notaciones utilizadas en todo el texto, y recuerda ciertos resultados clave como son el Algoritmo de Euclides y el Teorema de Bezout.

Algunas propiedades de los números enteros

Denotaremos por $\mathbb{Z}$ el anillo de los números enteros, esto es, el conjunto de los números $\{\ldots,-2,-1,0,1,2,\ldots\}$ junto con las operaciones suma $+$ y producto $\cdot$. Las propiedades aritméticas más importantes de $\mathbb{Z}$ son consecuencia de ser un anillo euclídeo.

La primera consecuencia es el Teorema Fundamental de la Aritmética que prueba que todo número entero positivo se expresa de modo único (salvo orden) como un producto de potencias de primos distintos.

Otra consecuencia es que si $a\mathbb{Z}$ denota el ideal consistente en los múltiplos de un entero a

$$a\mathbb{Z} := \{\ldots,-2a,-a,0,a,2a,\ldots\}$$

o equivalentemente

$$\langle a\rangle := \{n\cdot a \,|\, n\in\mathbb{Z}\}$$

entonces dados $a,b\in\mathbb{Z}$ se cumple que $a\mathbb{Z}+b\mathbb{Z}$ está generado por un elemento que es, por definición, el máximo común divisor de a y b y que se denota por (a,b). Es decir:

$$a\mathbb{Z}+b\mathbb{Z} = (a,b)\mathbb{Z}.$$

donde, por simplicidad (a,b) denota el máximo común divisor de a y b. Observar que el mínimo común múltiplo verifica $a\mathbb{Z}\cap b\mathbb{Z} = \text{m.c.m.}\{a,b\}\cdot\mathbb{Z}$.

Desde un punto de vista práctico, el máximo común divisor se puede calcular como el producto de los primos comunes a las descomposiciones de a y b elevadas a la menor potencia. Este cálculo, basado en factorizar a y b, no es eficiente en general. Afortunadamente disponemos de una forma eficiente de hacer el cálculo, concretamente el algoritmo de Euclides.

Algoritmo de Euclides

Sean a,b dos números enteros tales que $a>b>0$.

1. Hacemos $r_0=a$, $r_1=b$, $i=0$.
2. Calculamos la división $r_i = r_{i+1}c_i + r_{i+2}$, siendo c_i el cociente y r_{i+2} el resto, que verifica $r_{i+1} > r_{i+2} \geq 0$.
3. Si $r_{i+2}\neq 0$, entonces incrementamos i y volvemos al punto 2.
4. Si $r_{i+2}=0$, entonces se cumple que $r_{i+1}=(a,b)$.

Teorema — Teorema de Bezout. Sean a,b dos números enteros. Entonces existen números $x,y\in\mathbb{Z}$ tales que:

$$x\cdot a + y\cdot b = (a,b).$$

De nuevo nos preguntamos por formas efectivas, explícitas para calcular estos números x,y, y de nuevo, la respuesta está basada en el algoritmo de Euclídes.

TEOREMA DE BEZOUT

Sean a, b dos números enteros tales que $a > b > 0$.

- Hacemos $r_0 = a$, $r_1 = b$, $i = 0$.
- Realizamos el algoritmo de Euclídes, y consideramos las divisiones resultantes como si fueran un sistema de ecuaciones lineales en $r_0, r_1, \ldots, r_{i+1} = (a,b)$.
- Despejamos $(a,b) = r_{i+1} = r_{i-1} - r_i c_i$.
- Sustituimos en la última ecuación la expresión de r_i obtenida en la división anterior, $r_i = r_{i-2} - r_{i-1}c_{i-1}$:

$$(a,b) = r_{i-1} - r_i(r_{i-2} - r_{i-1}c_{i-1})c_i$$

- Sucesivamente, vamos sustituyendo las expresiones de $r_{i-1}, r_{i-2}, \ldots, r_1$.
- La última relación es una expresión en $r_0 = a$ y $r_1 = b$ por lo que, llamando x e y a sus coeficientes respectivos, concluimos, ya que:

$$(a,b) = \ldots = x \cdot a + y \cdot b\,.$$

Definición — Clases de restos. Dado un entero $a \in \mathbb{Z}$, consideramos el conjunto de las clases de restos al dividir por a

$$\mathbb{Z}/a\mathbb{Z} := \{\bar{0}, \bar{1}, \ldots, \overline{a-1}\}$$

con las operaciones suma y producto siguientes:

- $\bar{x} + \bar{y} := \overline{x+y}$
- $\bar{x} \cdot \bar{y} := \overline{x \cdot y}$

La relevancia de $\mathbb{Z}/a\mathbb{Z}$ consiste en su relación con $\mathbb{Z}$. De la propia definición se observa que para operar con dos clases en $\mathbb{Z}/a\mathbb{Z}$, se toman sendos representantes en $\mathbb{Z}$, se opera con ellos en los enteros, y se calcula el resto del resultado al dividir por a. Se verifica que no depende de los representantes elegidos. Estos argumentos prueban la siguiente proposición.

Proposición Dado $a \in \mathbb{Z}$, la aplicación que asigna a cada entero x su clase módulo a (el resto al dividir x entre a)

$$\begin{aligned} \mathbb{Z} &\longmapsto \mathbb{Z}/a\mathbb{Z} \\ x &\longmapsto \bar{x} \end{aligned}$$

es un morfismo de anillos. Esto es, $\mathbb{Z}/a\mathbb{Z}$ es el anillo cociente de $\mathbb{Z}$ por el ideal $a\mathbb{Z}$.

Teorema — Teorema Chino de los restos. Sean m, n dos números primos entre sí. Entonces la aplicación $\mathbb{Z}/mn \to \mathbb{Z}/m \times \mathbb{Z}/n$ que asigna a cada clase su resto módulo m y módulo n es un morfismo de anillos. Las correspondientes expresiones explícitas (por ejemplo, para calcular la aplicación inversa) se obtienen a partir del algoritmo de Euclides y del Teorema de Bezout.

De nuevo, usando el Teorema de Bezout, se prueban fácilmente los siguientes resultados.

Proposición Sea $a \in \mathbb{Z}$ y $\bar{x} \in \mathbb{Z}/a\mathbb{Z}$. Entonces $\bar{x}$ es invertible (es decir, existe $\bar{y} \in \mathbb{Z}/a\mathbb{Z}$ tal que $\bar{x} \cdot \bar{y} = \bar{1}$) si y solo si a y x son primos entre sí.

Proposición — Cuerpo finito $\mathbb{F}_p$. Sea $p \in \mathbb{Z}$. Entonces $\mathbb{Z}/p$ es cuerpo si y solo si p es un número primo. En este caso, denotaremos este cuerpo por $\mathbb{F}_p$.

Aunque no sea estrictamente necesario para la teoría desarrollada en este manual, sí puede ser conveniente recordar algún resultado básico sobre la estructura de los cuerpos finitos. En primer lugar, se tiene que el grupo de invertibles de todo cuerpo finito es cíclico, por tanto, los cuerpos finitos son siempre conmutativos y dos cuerpos finitos son isomorfos si y solo si tienen el mismo número de elementos. Además, existe un cuerpo finito con n elementos si y solamente si n es una potencia de un número primo. Téngase en cuenta que $\mathbb{F}_{p^r}$ no es isomorfo a $\mathbb{Z}/p^r$ para $r > 1$ ya que el primero es íntegro pero el segundo no. Para construir $\mathbb{F}_{p^r}$ se toma un polinomio $q(x) \in \mathbb{F}_p[x]$ irreducible y de grado r, y entonces $\mathbb{F}_p[x]/q(x)$ es un cuerpo con p^r elementos.

Definición — Característica de un anillo. Dado un anillo con unidad cualquiera (por ejemplo, $\mathbb{Z}$, $\mathbb{Z}/n$, $\mathbb{Q}$, $\mathbb{F}_{p^r}$,...), llamamos característica al menor número entero no negativo n tal que $n \cot 1 = 1 + \overset{n}{\ldots} + 1 = 0$.

Así, la característica de $\mathbb{Z}$ y $\mathbb{Q}$ es 0, la de $\mathbb{Z}/n$ es n y la de $\mathbb{F}_{p^r}$ es p (siendo p primo).

Definición — Función Indicador de Euler. Dado un número natural n se define su indicador de Euler como el número de elementos invertibles de $\mathbb{Z}/n$

$$\phi(n) := \#\{a \in \mathbb{Z}/n \text{ t.q. } (a,n) = 1\}$$

Proposición Sea $n = p_1^{n_1} \cdot \ldots \cdot p_r^{n_r}$ la descomposición de n como producto de potencias de números primos distintos entre si, entonces se tiene

$$\phi(n) = \prod_j (p_j - 1) p_j^{n_j - 1}.$$

Definición — Expresión en base b. Decimos que un número n se escribe o expresa en base b como:

$$a_r a_{r-1} \cdots a_1 a_0$$

donde $a_i \in \{0, 1, 2, \ldots, b\}$ cuando:

$$n = a_r b^r + a_{r-1} b^{r-1} + \cdots + a_1 b + a_0$$

La expresión binaria de un número corresponde a la base $b = 2$; la octal a la base $b = 8$; la decimal a $b = 10$; la hexadecimal a $b = 16$ (en este caso se utilizan los símbolos $0, 1, \ldots, 9, A, B, C, D, E, F$ donde se entiende que A representa el número 10, B el 11, etc. y F el 15.

Ejercicios Propuestos

Problema Calcular las sumas:

- $3+5$, $17+38$ en $\mathbb{Z}/2$;
- $3+5$, $17+38$ en $\mathbb{Z}/5$;
- $876+8976+654$ en $\mathbb{Z}/26$.

Problema Calcular los productos:
- $3 \cdot 5$, $17 \cdot 38$ en $\mathbb{Z}/2$;
- $5 \cdot 67$, $27 \cdot 78 \cdot 9654$ en $\mathbb{Z}/18$;
- $79 \cdot 89 \cdot (67 + 54 + 34 \cdot 68)$ en $\mathbb{Z}/18$

Problema Determinar todos los divisores de: 7, 12, 129 y 1550.

Problema Utilizando el algoritmo de Euclides, dar la expresión general de una antimagen en $\mathbb{Z}/mn \to \mathbb{Z}/m \times \mathbb{Z}/n$ de un par (a,b). ¿Hay más de una?

Problema Aplicar el problema anterior para calcular 7^{-1} mod 300.

Problema Calcular los inversos:
- 5, 17 en $\mathbb{Z}/7$;
- 67, 9654 en $\mathbb{Z}/26$;
- 765, 33 en $\mathbb{Z}/23$.
- 6547 en $\mathbb{Z}/17$.

Problema Calcular el máximo común divisor de: 7 y 12, 125 y 1550, 14 y 26.

Problema Aplicar el algoritmo de Euclides a a,b para calcular x,y tales que $ax+by =$ m.c.d.(a,b); para las parejas 7 y 12, 125 y 1550, 14 y 26.

Problema Calcular el indicador de Euler de 72, 80, 143, 1911, 2499, 2772, 3234, 4032 y 14400.

Problema Calcular las potencias:
- 765^{1565} en $\mathbb{Z}/23$
- 456^{654} en $\mathbb{Z}/9$
- 127^{1565} en $\mathbb{Z}/24$
- $127^{987} + 53^{76} \cdot 67$ en $\mathbb{Z}/24$

Problema Resolver los sistemas:
- $\begin{cases} x = 2 & \text{en } \mathbb{Z}/3 \\ x = 3 & \text{en } \mathbb{Z}/5 \\ x = 4 & \text{en } \mathbb{Z}/11 \end{cases}$
- $\begin{cases} 19x = 103 & \text{en } \mathbb{Z}/900 \\ 10x = 511 & \text{en } \mathbb{Z}/841 \end{cases}$

Problema Invertir las siguientes matrices:
- $\begin{pmatrix} 1 & 3 \\ 4 & 3 \end{pmatrix}$ en $\mathbb{Z}/5$;
- $\begin{pmatrix} 1 & 3 \\ 4 & 3 \end{pmatrix}$ en $\mathbb{Z}/29$;
- $\begin{pmatrix} 15 & 17 \\ 4 & 9 \end{pmatrix}$ en $\mathbb{Z}/26$;

Problema Resolver los siguientes sistemas de ecuaciones:
- $\begin{cases} 2x+3y = 1 \\ 7x+8y = 2 \end{cases}$ en $\mathbb{Z}/26$;
- $\begin{cases} x+4y = 1 \\ 5x+7y = 1 \end{cases}$ en $\mathbb{Z}/9$.

Problema Resolver las ecuaciones:

- $3x = 4$ en $\mathbb{Z}/4$;
- $27x = 25$ en $\mathbb{Z}/256$;

Problema Determinar cuales de las siguientes transformaciones afines tiene una transformación inversa y, en caso afirmativo, calcularla.

- $x \mapsto 3x+2$ en $\mathbb{Z}/4$;
- $x \mapsto 11x-5$ en $\mathbb{Z}/26$;
- $x \mapsto 12x+7$ en $\mathbb{Z}/16$.

Solución: Sabemos por teoría que una transformación afín del tipo $T(x) = Ax+b$ tiene inversa si y solo si A es invertible y en ese caso $T^{-1}(x) = A^{-1}x - A^{-1}b$. En nuestro caso (dimensión 1), A es un escalar y la homotecia que define es invertible en $\mathbb{Z}/n$ si y solo si $(A,n) = 1$. Por tanto, para el primer caso tenemos que $(3,4) = 1$ por lo que es invertible y el algoritmo de Euclides prueba que $(-1)*3+1*4 = 1$, por lo que $3^{-1} = -1 = 3$ módulo 4. Así, la inversa será $x \mapsto 3x+3*2 = 3x+2$. En el tercer caso planteado, $(12,16) = 4 \neq 1$, por lo que no tiene inversa.

Problema Encontrar las transformaciones afines inversas de las siguientes (si existen):

- $\begin{pmatrix} x \\ y \end{pmatrix} \longmapsto \begin{pmatrix} 1 & 3 \\ 4 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 1 \\ 3 \end{pmatrix}$ en $\mathbb{Z}/29$
- $\begin{pmatrix} x \\ y \end{pmatrix} \longmapsto \begin{pmatrix} 15 & 17 \\ 4 & 9 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 1 \\ 7 \end{pmatrix}$ en $\mathbb{Z}/26$

Problema Rellenar la siguiente tabla:

base	2	10	16	26
	10110			
		2005		
			$1AF$	
				QW

donde cada fila corresponde a las distintas expresiones del *mismo* número.

Problema Observar que un número n es múltiplo de b si la expresión de n en base b termina en 0 (es decir, con la notación anterior $a_0 = 0$).

Probar que el resto de dividir un número decimal $a_r a_{r-1} \cdots a_1 a_0$ entre 9 (su clase en $\mathbb{Z}/9$) coincide con el resto de dividir $a_r + a_{r-1} + \cdots + a_1 + a_0$ entre 9.

Concluir que un número decimal $a_r a_{r-1} \cdots a_1 a_0$ es múltiplo de 9 si y solo si $a_r + a_{r-1} + \cdots + a_1 + a_0$ es múltiplo de 9.

Bibliografía

[1] Johannes A. Buchmann. *Introduction to cryptography*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, 2001. ISBN: 0-387-95034-6.

[2] Pino Caballero. *Introducción a la criptografia*. Editorial Ra-MA, 2002. ISBN: 8478975209.

[3] Raúl Durán Dıaz, Luis Hernández Encinas y Jaime Muñoz Masqué. *El criptosistema RSA*. Editorial Ra-MA, 2005. ISBN: 978-84-7897-651-5.

[4] Amparo Fuster Sabater y col. *Criptografía, protección de datos y aplicaciones. Una guıa para estudiantes y profesionales*. Editorial Ra-MA, 2012. ISBN: 978-84-9964-136-2.

[5] Neal Koblitz. *A course in number theory and cryptography*. Second. Volumen 114. GTM. Springer-Verlag, New York, 1994. ISBN: 0-387-94293-9.

[6] Neal Koblitz. *Algebraic aspects of cryptography*. Volumen 3. Algorithms and Computation in Mathematics. Springer-Verlag, Berlin, 1998. ISBN: 3-540-63446-0.

[7] Alfred J. Menezes y Scott A. van Oorschot Paul C. y Vanstone. *Handbook of applied cryptography*. CRC Press Series on Discrete Mathematics and its Applications. CRC Press, Boca Raton, FL, 1997. ISBN: 0-8493-8523-7.

[8] Arto Salomaa. *Public-key cryptography*. Second. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 1996. ISBN: 3-540-61356-0.

[9] John T. Silverman Joseph H. y Tate. *Rational points on elliptic curves*. Second. Undergraduate Texts in Mathematics. Springer, Cham, 2015. ISBN: 978-3-319-18587-3.

Índice Temático

Semblanza

Francisco José Plaza Martín (Torredelcampo, Jaén, 1970). Catedrático de Geometría y Topología en la Universidad de Salamanca. En su labor docente ha impartido clases en los estudios de matemáticas, ingeniería informática y físicas entre otros estudios universitarios abarcando materias de álgebra y geometría, pero también de teoría de códigos, criptografía o de lenguajes de cálculo simbólico. Sus intereses investigadores se enmarcan dentro de la geometría algebraica, área en la que ha dirigido tesis y proyectos de investigación y ha sido invitado a congresos internacionales y a centros de reconocido prestigio.

Documentos Didácticos, 169

Detrás de la comunicación cifrada de WhatsApp, del DNI electrónico y de los protocolos de navegación seguros https:// encontramos la criptografía. Este libro, concebido como un manual, pretende ser una introducción a los fundamentos matemáticos sobre los que se apoya la criptografía moderna y está orientado a estudiantes universitarios tanto de matemáticas y física como de informática e ingeniería. El texto presenta aspectos teóricos de forma accesible y los ilustra y complementa con ejercicios y algoritmos que pueden ser programados fácilmente. Entender la base matemática subyacente a la criptografía proporcionará al lector la capacidad de analizar las fortalezas y debilidades de un criptosistema, así como de comprender el diseño y los ataques de un protocolo criptográfico. Debido al carácter intrínsecamente interdisciplinar de esta materia y con vistas a una capacitación profesional plena, un ulterior estudio debería incorporar aspectos de probabilidad, de algoritmia, de diseño de *hardware*, etc.

ISBN: 978-84-1311-463-7