Facultad
de
Ciencias

ATAQUES AL CRIPTOSISTEMA RSA

(ATTACKS ON RSA)

Trabajo de fin de Grado
para acceder al

GRADO EN MATEMÁTICAS

Autor: David Balbás Gutiérrez

Director: Daniel Sadornil Renedo

Julio - 2019

Índice general

Agradecimientos **III**

Resumen **V**

1. Introducción **1**

2. El criptosistema RSA **4**
2.1. Protocolo y características 4
2.2. Vulnerabilidades en la implementación 9
2.3. Algunos primeros ataques 10

3. Ataques por Factorización **13**
3.1. Algoritmos de Factorización 13
3.2. Generación de primos 20
3.3. Elección de primos para RSA 21

4. Ataques basados en Fracciones Continuas **22**
4.1. Fracciones Continuas 22
4.2. Ataque de Wiener 26
4.3. Variantes del ataque de Wiener 29

5. Ataques basados en Retículos **32**
5.1. Retículos 32
5.2. Reducción de bases de retículos. Bases LLL-reducidas 34
5.3. Algoritmo de Coppersmith 36
5.4. Ataques en el caso univariado 40
5.5. Algoritmo de Coppersmith para polinomios en dos variables 44

6. Conclusiones **49**

Bibliografía **51**

A. Algunos conceptos y resultados **55**
A.1. Teoría de números 55
A.2. Complejidad computacional 57
A.3. Operaciones con polinomios 59

B. Ejemplo del Algoritmo de Coppersmith **61**

Agradecimientos

Han sido muchas las personas que, directa o indirectamente, han hecho posible este trabajo, y me es imposible nombrarlas a todas. En primer lugar, me gustaría agradecerles a los profesores de la Universidad de Cantabria su dedicación y la pasión que me han transmitido por las matemáticas, especialmente durante estos dos últimos años. Entre ellos he de destacar al director del trabajo, Dani. Te estoy totalmente agradecido por tu implicación total con el proyecto, por tu entusiasmo, y sobre todo por el apoyo que me has brindado dentro y fuera del ámbito académico.

También quiero agradecerle su apoyo a todos aquellos que forman en mi vida el conjunto complementario de las matemáticas. Tengo la suerte de haber estado en todo momento arropado por personas fantásticas, tanto en Santander como en Bergen y en Providence. Especialmente, quiero mencionar a mi grupo de amigos, que habéis estado a mi lado en todos los buenos y malos momentos, siempre capaces de sacarme una sonrisa cuando más falta me hace.

Por otro lado quiero agradecerles a mis compañeros, con quienes he compartido ideas, conversaciones y, en muchos casos, verdaderas amistades que trascienden lo académico. De manera muy especial, a mis once compañeros de Doble Grado. Embarcamos juntos a ciegas en esta aventura, y no quiero imaginarme cómo hubieran sido estos años sin vosotros, sin vuestro talento, amistad, y sobre todo, sin vuestro compañerismo.

Finalmente, quiero agradecerle a mi familia su apoyo incondicional en absolutamente todos los ámbitos de mi vida. A mi madre Ana, que siempre has estado a mi lado para guiarme de la mejor forma posible. A mi padre Juan, que desde pequeño has cultivado mi curiosidad y pasión por la ciencia. Y a mi hermana Victoria, por tu cariño y por todo lo que hemos compartido. Este trabajo está dedicado a ti; las matemáticas no son más que una mezcla de talento y creatividad, habilidades que en ti desbordan y sobre las que me has enseñado mucho más de lo que imaginas.

Resumen / Abstract

El criptosistema RSA, propuesto en 1977 por Rivest, Shamir y Adleman y pionero en el paradigma de la criptografía de clave pública, prevalece como uno de los sistemas más utilizados en la actualidad cuatro décadas después de su nacimiento. En este trabajo se presentan, analizan y conectan varios de los ataques más notables que se han empleado para el criptoanálisis de RSA. Los ataques se estudian desde un punto de vista matemático, complementándolos con algunos ejemplos ilustrativos.

En un principio, se muestran las propiedades y vulnerabilidades más elementales que presenta RSA. Posteriormente, se analizan distintas formas de atacar al criptosistema mediante tres enfoques distintos: factorización, fracciones continuas y retículos. El primero busca factorizar el módulo característico n, problema sobre el que se sustenta la seguridad del criptosistema. El segundo utiliza las fracciones continuas partiendo de las ideas de Wiener, dando lugar a importantes ataques si la clave de descifrado es pequeña. El tercero propone distintas aplicaciones para el ataque a RSA del algoritmo de Coppersmith, que determina raíces modulares pequeñas de polinomios con coeficientes enteros, basado en la reducción de bases de retículos.

RSA, the first public-key cryptosystem proposed by Rivest, Shamir and Adleman in 1977, prevails among the most common cyphers four decades after its publication. In this thesis, several of the most remarkable results obtained by cryptanalysing RSA are presented, analysed and connected. The attacks are studied from a mathematical perspective, including some illustrative examples.

To begin, some elementary properties and vulnerabilities of RSA are shown. Next, three different cryptanalytical approaches are studied: integer factorization, continued fractions and lattices. The first aims to factor the distinctive RSA modulus n, the problem which provides the cryptosystem with security. The second employs continued fractions starting from Wiener's ideas, leading to relevant attacks when the decryption key is small. The third suggests several applications to RSA of Coppersmith's algorithm to find small modular roots of polynomials with integer coefficients, which relies on lattice basis reduction.

Palabras Clave / Key Words

Criptología, RSA, Factorización, Fracciones Continuas, Retículos.
Cryptology, RSA, Factorization, Continued Fractions, Lattices.

Capítulo 1

Introducción

"The Magic Words are
Squeamish Ossifrage"

Rivest, Shamir, Adleman. 1977

La necesidad de cifrar mensajes para comunicarse de forma segura es casi tan antigua como la escritura en sí misma, y distintos métodos criptográficos se han sucedido a lo largo de la historia. Uno de los primeros y más sencillos fue el cifrado de César, consistente en reemplazar cada letra del alfabeto por la correspondiente al avanzar un número k de posiciones. Por ejemplo, utilizando el alfabeto español de 27 letras y $k = 3$, la palabra *RSA* se escribiría *UVD*. Este método es simple pero altamente inseguro, ya que en un alfabeto de 27 letras hay únicamente 27 posibles formas de cifrar cada mensaje (aunque una de ellas sea el cifrado trivial en el que no se altera el mismo), y un sencillo ataque por fuerza bruta puede ser llevado a cabo sin gran dificultad. Una pequeña modificación de este sistema, conocido como cifrado de sustitución monoalfabética, es reemplazar cada letra del mensaje original por otra distinta, sin que sea necesariamente la que se encuentra al avanzar k posiciones. El sistema es más seguro frente a ataques de fuerza bruta, con $27! \approx 10^{27}$ cifrados posibles. No obstante, la tarea de cifrar y descifrar es más compleja, siendo la clave una 27-tupla con las posiciones de cada una de las letras en orden, equivalente a una permutación del alfabeto.

Aparentemente, el método anterior parece indescifrable a mano. Las 27! posibilidades están fuera del alcance de cualquiera. Sin embargo, un análisis del criptosistema puede dar lugar a que un observador externo ajeno a la clave pueda descifrar el mensaje. En este caso, una forma adecuada de actuar es realizar un análisis de frecuencias. En cada idioma, hay una serie de letras que se repiten mucho más a menudo que otras. Detectando las más repetidas en el mensaje y probando con ellas, buena parte del mensaje será descifrado y se podrán deducir el resto de letras con facilidad. En adelante, un procedimiento que permita descifrar un mensaje bajo unas ciertas condiciones, sin el conocimiento de las claves necesarias para ello, se denominará un ataque. Si para un sistema criptográfico existe un ataque que permite descifrar *cualquier* mensaje en un tiempo mucho menor al de un ataque por fuerza bruta (como ocurre en el cifrado anterior), el criptosistema queda roto.

En los dos párrafos anteriores se han introducido las dos partes fundamentales de la criptología, que es la disciplina que se dedica a la información y escritura secreta. Estas dos partes son la criptografía, que estudia los métodos, algoritmos y protocolos utilizados para garantizar la seguridad de la información; y el criptoanálisis, que trata de romper los sistemas criptográficos proponiendo diversos ataques. Otras ramas, como la esteganografía y el estegoanálisis, se dedican a la ocultación de la información.

En general, la criptografía y el criptoanálisis son complementarios. No es posible tener la certeza de que un método de cifrado sea seguro sin que se haya realizado un criptoanálisis ex-

haustivo del mismo. La criptografía se encuentra hoy presente en todo tipo de comunicaciones, transacciones bancarias, comercio electrónico, y muchos otros ámbitos. En todos ellos, los métodos que se utilizan han sido y siguen siendo sometidos a un fuerte criptoanálisis en un intento de buscar sus vulnerabilidades. Formalmente, se le llama sistema criptográfico o criptosistema a lo siguiente [45]:

Definición 1.1. *Un criptosistema es una 6-tupla* $(\mathcal{P}, \mathcal{C}, \mathcal{K}_e, \mathcal{K}_d, \mathcal{E}, \mathcal{D})$ *donde:*

1. $\mathcal{P}$ *es el conjunto de posibles textos no cifrados.*
2. $\mathcal{C}$ *es el conjunto de posibles textos cifrados.*
3. $\mathcal{K}_e$ *es el conjunto de posibles claves de cifrado y* $\mathcal{K}_d$ *de descifrado.*
4. $\mathcal{E} : \mathcal{P} \times \mathcal{K}_e \longrightarrow \mathcal{C}$ *y* $\mathcal{D} : \mathcal{C} \times \mathcal{K}_d \longrightarrow \mathcal{P}$ *son aplicaciones que verifican que para cada* $e \in \mathcal{K}_e$*, existe una clave* $d \in \mathcal{K}_d$ *tal que* $\mathcal{D}(\mathcal{E}(M, e), d) = M$ *para todo* $M \in \mathcal{P}$*.*

Parece lógico pensar que un buen sistema criptográfico debe cumplir ciertos requisitos básicos: que sea sencillo cifrar un mensaje, que sea sencillo descifrarlo conociendo la clave, y que sea resistente al criptoanálisis. Estos principios fueron formulados por primera vez en 1883 por Kerckhoffs [22]. Una formulación actual de los mismos es que un buen criptosistema debe cumplir:

1. El sistema debe ser prácticamente, si no matemáticamente, indescrifrable sin el conocimiento de la clave de descifrado correspondiente.
2. El sistema debe ser público, y no debe suponer un problema que se conozca su funcionamiento.
3. Debe ser posible almacenar las claves de forma segura, y los usuarios deben poder modificar las claves a su gusto.
4. La información cifrada debe ser fácilmente transmisible.
5. El descifrado y el cifrado deben ser sencillos y realizables en unos pocos pasos.

Históricamente, ha sido frecuente la aparición y el uso de distintos sistemas de cifrados que cumplían, en mayor o menor medida, estas condiciones. Simon Singh, en su libro "The Code Book" [43], presenta una amplia colección de los mismos en su contexto histórico, siendo algunos de ellos muy conocidos e incluso presentes en el cine y la literatura popular, como la máquina Enigma utilizada durante la II Guerra Mundial. Sin embargo, todos ellos presentaban un problema; era necesario que las dos partes interesadas en comunicarse se pusieran de acuerdo de antemano en una clave.

La criptografía de clave pública surge como solución a la cada vez más frecuente incapacidad de acordar una clave por un medio seguro y al número de claves necesarias para conectar varios usuarios entre sí. Por ejemplo, dos usuarios compartiendo información por primera vez a través de la red. Whitfield Diffie y Martin Hellman describieron en 1976 las condiciones necesarias para que un criptosistema pudiera considerarse de clave pública [12], que se citan a continuación.

Definición 1.2. *Un criptosistema de clave pública satisface que para cada usuario* U *del sistema, se puede determinar una clave pública* $e_U \in \mathcal{K}_e$ *y una clave privada* $d_U \in \mathcal{K}_d$ *tales que:*

1. *Para todo mensaje* $M \in \mathcal{P}$*, se verifica que* $\mathcal{D}(\mathcal{E}(M, e_U), d_U) = M$*.*
2. *Las operaciones de cifrado* $C' = \mathcal{E}(M, e_U)$ *y descifrado* $M' = \mathcal{D}(C, d_U)$ *son fácilmente calculables cualesquiera* $M \in \mathcal{P}$ *y* $C \in \mathcal{C}$*.*

3. *Para casi todo mensaje $M \in \mathcal{P}$, hallar un equivalente a d_U o descifrar un criptograma $C \in \mathcal{C}$ a partir únicamente de C y de e_U es computacionalmente intratable.*

4. *Es computacionalmente factible determinar el par de claves e_U y d_U.*

El esquema más general de la criptografía de clave pública consiste en que si un emisor Alicia (en adelante A) desea enviarle un mensaje M a un receptor Bonifacio (en adelante B), podrá cifrarlo como $C = \mathcal{E}(M, e_B)$, y B podrá recuperarlo aplicando $M = \mathcal{D}(C, d_B)$, de acuerdo con la definición anterior. Para satisfacer la tercera condición de Diffie-Hellman, se buscan funciones de una vía [39]:

Definición 1.3. *Una función $f : X \longrightarrow Y$ es de una vía cuando dados $x \in X, y \in Y$ es sencillo calcular $f(x)$ pero es computacionalmente intratable calcular $f^{-1}(y)$.*

La intratabilidad de un problema a lo largo de este trabajo (ver Apéndice A para más detalles) se refiere a que su resolución conlleva un coste, en memoria o en capacidad de procesamiento, finito pero imposible de asumir [12]. Se asocia habitualmente con una complejidad computacional subexponencial o peor. No se conocen funciones de una vía con certeza, pero se conocen muchas tales que calcular $f^{-1}(y)$ es *probablemente* intratable.

Diffie y Hellman implementaron por primera vez su idea fundamental de la clave pública en 1976, desarrollando un protocolo de intercambio de claves cuya función de una vía es la exponenciación modular. Su protocolo no es un criptosistema ya que no permite cifrar información, pero facilita que dos usuarios que nunca se han comunicado entre sí puedan acordar una clave de forma segura. Aproximadamente un año después, Rivest, Shamir y Adleman publican RSA [37], que sí permite cifrar mensajes, y cuya función de una vía es el producto de dos números primos de tamaño similar. Posteriormente a su aparición, diversos criptoanálisis dieron lugar a ataques que utilizan técnicas distintas de la factorización para intentar romper el sistema con mayor o menor éxito. A día de hoy, existen en la literatura numerosos ataques que son eficaces frente a implementaciones de RSA bajo ciertas condiciones. Por ejemplo, si se conocen determinadas características del mensaje o de las claves. El objetivo de este trabajo es presentar, analizar y conectar entre sí varios de ellos.

En el capítulo 2 se realiza una descripción del criptosistema RSA y de sus características, junto con un detalle de algunas vulnerabilidades y ataques elementales. El capítulo 3 versa sobre los ataques por factorización, donde no únicamente se estudian algoritmos de factorización de enteros sino también la generación y la elección de primos adecuada para protegerse frente a estos ataques. En el capítulo 4 se estudian ataques basados en aproximaciones mediante fracciones continuas, como el ataque de Wiener, que suponen una amenaza importante cuando la clave de descifrado de RSA es pequeña. En el capítulo 5 se presenta el algoritmo de Coppersmith para encontrar raíces modulares de polinomios, que se basa en la reducción de bases de retículos. El algoritmo da lugar a una colección de ataques a RSA que se introducen a continuación del mismo. Las conclusiones del trabajo se muestran en el capítulo 6, y en las últimas páginas se encuentran dos pequeños apéndices. El primero incluye algunos conceptos sobre álgebra, teoría de números y complejidad computacional que se utilizan en el texto, y el segundo un ejemplo detallado de criptoanálisis a RSA utilizando el algoritmo de Coppersmith.

Capítulo 2

El criptosistema RSA

El criptosistema RSA fue propuesto en 1977 por Ron Rivest, Adi Shamir y Leonard Adleman, investigadores del Massachusetts Institute of Technology (MIT) [37], y es el más extendido entre los criptosistemas de clave pública actuales. Años antes de la publicación del RSA, los trabajadores del "Communications Electronics Security Group" James Ellis y Clifford Cocks, pertenecientes al servicio secreto británico, anticiparon las ideas de Diffie y Hellmann sobre la criptografía de clave pública y propusieron un sistema muy similar a RSA. Sin embargo, estos artículos se mantuvieron confidenciales hasta 1997, (como puede verse en [26] o en [45]), hecho por el cual los investigadores del MIT son generalmente considerados como los creadores de RSA.

A pesar de que RSA puede ser utilizado para cifrar mensajes completos, suele ser utilizado para la transmisión de claves destinadas a asegurar una comunicación cifrada segura mediante un sistema de clave simétrica debido a su lentitud [40]. En este capítulo se expondrá el protocolo RSA; la generación de las claves, el cifrado, y el descifrado de los mensajes, a la par que se probará el buen funcionamiento del mismo. Posteriormente, se mencionarán sus características fundamentales, y se verá una colección de vulnerabilidades y de ataques que dejan al descubierto algunas de sus debilidades más evidentes.

2.1. Protocolo y características

La seguridad de RSA se basa en la dificultad de factorizar un número entero en primos, un problema que hasta la fecha se considera computacionalmente inabordable. El procedimiento que se sigue para generar las claves, cifrar y descifrar mensajes [16] se detalla a continuación, siguiendo el esquema Alicia-Bonifacio (A-B) visto en la introducción. Posteriormente, se comprueba que el criptosistema satisface las condiciones de Diffie-Hellman.

Generación de claves

Cada usuario U sigue los siguientes pasos:

1. Elige dos primos p y q, calcula un entero $n_U = pq$ y su función de Euler $\varphi(n_U) = (p-1)(q-1)$.
2. Elige un entero positivo $e_U > 2$ tal que $mcd(e_U, \varphi(n_U)) = 1$. Generalmente, $e_U < n_U$.
3. Calcula el inverso d_U de e_U en $\left(\mathbb{Z}/\varphi(n_U)\mathbb{Z}\right)^*$, es decir, $d_U e_U \equiv 1$ mód $\varphi(n_U)$.
4. La clave pública del usuario U es el par (n_U, e_U) y su clave privada es d_U.

Cifrado de mensajes

Si A quiere enviarle un mensaje M a B, deberá obtener su clave pública (n_B, e_B) y representar M como un elemento de $\mathbb{Z}/n_B\mathbb{Z}$. Posteriormente, calculará $C \equiv M^{e_B} \bmod n_B$ y enviará C a B.

Descifrado de mensajes

Tras recibir el criptograma C, el receptor B simplemente calculará $C^{d_B} \bmod n_B$ para descifrar el mensaje.

2.1.1. Condiciones de Diffie-Hellman

La primera condición de Diffie-Hellman (Definición 1.2) es el correcto funcionamiento del descifrado, lo que se prueba a continuación [37].

Proposición 2.1. *Dado un usuario con clave pública (n, e) donde $n = pq$, y clave privada d, para cualquier criptograma $C = M^e \bmod n$ donde $M \in \mathbb{Z}/n\mathbb{Z}$, se verifica:*

$$C^d \equiv M \bmod n.$$

Demostración. Para comenzar, es claro que $C^d \equiv M^{ed} \bmod n$. Como e, d son inversos en $\left(\mathbb{Z}/\varphi(n)\mathbb{Z}\right)^*$, $ed \equiv 1 \bmod \varphi(n)$, que implica que $ed = 1 + k\varphi(n)$ con $k \in \mathbb{Z}$. Por tanto, $M^{ed} = M^{k\varphi(n)+1}$.

Por el Pequeño Teorema de Fermat, si $mcd(M, p) = 1$, entonces $M^{p-1} \equiv 1 \bmod p$. Como $p - 1$ divide a $\varphi(n)$:

$$M^{ed} = M^{k\varphi(n)+1} \equiv M \bmod p.$$

Además, si p divide a M, se tiene $M \equiv 0 \bmod p$ y la identidad anterior también se verifica, con lo que es válida para cualquier M. Razonando análogamente para q, se obtiene

$$M^{ed} = M^{k\varphi(n)+1} \equiv M \bmod q.$$

Las dos ecuaciones implican que tanto p como q dividen a $M^{k\varphi(n)+1} - M$, con lo que $n = pq$ también lo divide, y

$$M^{k\varphi(n)+1} \equiv M \bmod n. \qquad \square$$

La segunda condición de Diffie-Hellman es que el cifrado y el descifrado son fácilmente calculables. Ambas fases del protocolo se realizan mediante exponenciación modular, la cual se puede calcular mediante el algoritmo de los cuadrados repetidos, que es polinomial en el tamaño de n (ver Apéndice A). A pesar de ello, cifrar y descifrar puede ser un proceso lento, especialmente en dispositivos con una capacidad de computación reducida. Para acelerarlo, se recurre frecuentemente a dos opciones:

- Utilizar una clave de cifrado e pequeña, como $e = 3$, $e = 17$ o $e = 65537$.
- Descifrar utilizando el Teorema Chino de los Restos [2]. Sea $d_p = d \bmod p - 1$ y $d_q = d \bmod q - 1$. Si ambos son pequeños, es posible descifrar un mensaje cifrado $C \equiv M^e \bmod n$ calculando $M_p = C^{d_p} \bmod p$ y $M_q = C^{d_q} \bmod q$. Entonces, $M_p \equiv C^d \equiv M^{ed} \bmod p$ y análogamente para q. Se puede calcular M como la única solución del sistema de congruencias módulo n tal que $M \equiv M_p \bmod p$, $M \equiv M_q \bmod q$, que satisface $M \equiv M^{ed} \equiv C^d \bmod n$. La ventaja de este método es que aunque d_p y d_q sean pequeños, d podría ser del tamaño de n.

Utilizar una clave de descifrado d pequeña da lugar a una ruptura total del criptosistema, como se verá en el capítulo 4.

La tercera condición hace referencia a la seguridad del criptosistema. Un atacante cualquiera puede conocer n, e y C pero necesitaría calcular una clave de descifrado para recuperar el mensaje. Esto no es posible *a priori* sin conocer $\varphi(n)$, lo que dota de seguridad al sistema. Es evidente que descomponer n en sus factores primos p y q permite automáticamente al atacante determinar $\varphi(n) = (p-1)(q-1)$ y con ello calcular d. Sin embargo, factorizar un número en primos es una tarea generalmente intratable, que actualmente requiere un tiempo subexponencial, como se ve en el capítulo 3. No obstante, no está probado que no sea posible factorizar en tiempo polinomial [39].

Por otro lado, a partir de $\varphi(n)$ es posible obtener la factorización de n [21]. A pesar de que esto no es cierto en general, como se puede ver en [40], lo es para un esquema RSA donde $n = pq$. Basta con observar que la ecuación cuadrática

$$x^2 - (n - \varphi(n) + 1)x + n = 0$$

tiene como soluciones $x = p$ y $x = q$. A la vista de lo anterior, se puede concluir que factorizar n y determinar $\varphi(n)$ son problemas computacionalmente equivalentes.

Por último, **la cuarta condición** requiere que la generación de las claves sea un problema computacionalmente tratable. Por un lado, encontrar dos primos p y q del tamaño buscado no supone un problema. Como se verá en la sección 3.2 con detalle, es sencillo encontrarlos aplicando sucesivos tests de primalidad, de coste polinomial en el tamaño de n. Por otro lado, el cálculo de la clave de descifrado a partir de la de cifrado se puede realizar mediante el algoritmo de Euclides extendido, también polinomial en el tamaño de n (ver Apéndice A).

Lo primero que un criptoanalista debe preguntarse al estudiar RSA es si realmente es tan seguro como parece. Parece claro que la tarea de factorizar n es ardua, pero, ¿existe otro camino para descifrar un mensaje cifrado mediante RSA? A día de hoy, nadie ha sido capaz de encontrarlo, pero tampoco se ha probado que factorizar n sea equivalente a descifrar un mensaje. La seguridad de RSA depende de la veracidad de la siguiente afirmación, no probada [51]:

Romper RSA equivale computacionalmente a factorizar el módulo n.

2.1.2. Características fundamentales

En el artículo original de Rivest, Shamir y Adleman [37] no se impone la condición de que $mcd(M, n) = 1$, que por el contrario sí aparece frecuentemente en la literatura (ver por ejemplo [2], [16] o [45]). Esto se debe a que si M no es primo con n, el criptosistema queda roto.

Proposición 2.2. *Dado un par de clave pública RSA (n, e) y un mensaje M tal que $mcd(M, n) \neq 1$, es inmediato recuperar la factorización de n a partir del mensaje cifrado $C = M^e$ mód n.*

Demostración. Como M, n no son primos entre sí, y $n = pq$, necesariamente $M = kp$ con $k \in \mathbb{Z}$ (análogamente si $M = kq$). Entonces, $M^e = k^e p^e = rp$, donde $r \in \mathbb{Z}$, y se tiene que $C = rp$ mód n. Por tanto, $p = mcd(C, n)$, y $q = n/p$. □

Sin embargo, el caso en el que M no es primo con n es, a efectos prácticos, apenas relevante. El número de mensajes primos con n en $\mathbb{Z}/n\mathbb{Z}$ es $\varphi(n) = (p-1)(q-1)$, por lo que la probabilidad de que un M al azar no lo sea (supuesto $p > q$) es:

$$P[(mcd(M, n) \neq 1] = 1 - \frac{\varphi(n)}{n} = \frac{p + q - 1}{pq} < \frac{2}{q}.$$

Si el menor de los primos es grande, esta probabilidad es despreciable.

En párrafos anteriores se ha visto que, conocidos p y q, y elegida una clave pública de cifrado e, es sencillo calcular un exponente de descifrado d. Es razonable plantearse la pregunta inversa: conocidos e, d y n, ¿se pueden obtener p y q? La respuesta es afirmativa y se prueba a continuación.

Teorema 2.3. *Sea (n, e) un par de clave pública RSA. Conocida la clave privada d, es posible recuperar la factorización $n = pq$ de forma eficiente.*

Demostración. La prueba a continuación requiere de un algoritmo probabilístico. Existe una alternativa, por Coron y May [9], que lo hace de forma determinista cuando $ed < n^2$ utilizando una variante del algoritmo de Coppersmith (este último se verá en el capítulo 5).

Dados e, d, se puede calcular $k = ed - 1$, donde k es un múltiplo de $\varphi(n)$ por definición de e y d. Como $\varphi(n)$ es par, $k = 2^t r$ donde $t \geq 1$ y r es impar. Además, para todo $m \in \left(\mathbb{Z}/n\mathbb{Z}\right)^*$, por el Teorema de Euler, $m^k \equiv 1 \bmod n$ y entonces $m^{k/2}$ es una raíz cuadrada de la unidad módulo n.

Aplicando ahora el Teorema Chino de los Restos, 1 tiene cuatro raíces cuadradas módulo $n = pq$. Dos de ellas son ± 1, y las otras dos son $\pm x$, donde x es un entero tal que $x \equiv 1 \bmod p$ y $x = -1 \bmod q$. Si se obtiene x, como p divide a $x - 1$, bastará con calcular $mcd(x - 1, n)$ para factorizar n (y análogamente con $-x$).

Para calcular x, se va a probar que si se escoge aleatoriamente $g \in \left(\mathbb{Z}/n\mathbb{Z}\right)^*$, uno de los elementos en la secuencia: $g^{k/2}, g^{k/4}, \ldots, g^{k/2^t} \bmod n$ será una raíz no trivial de la unidad (es decir, $\pm x$) con probabilidad mayor a $1/2$. Para comenzar, dado un g aleatorio, $g^{k/2}$ es una raíz cuadrada de la unidad dado que $g^k \equiv 1 \bmod n$. Se tienen cuatro posibilidades: $g^{k/2} \equiv \pm 1 \bmod n$ ó $g^{k/2} \equiv \pm x \bmod n$. Si se da el último caso, es sencillo determinar x. Por otra parte, si $g^{k/2} \equiv 1 \bmod n$, entonces $g^{k/4}$ es una raíz cuadrada de la unidad y se puede repetir el argumento anterior, mientras que si $g^{k/2} \equiv -1 \bmod n$ esto no es posible. De esta manera, se considera un algoritmo que evalúe la secuencia $g^{k/2}, g^{k/4}, \ldots, g^{k/2^t} \bmod n$ iterativamente empezando por el primer elemento y pare con éxito si encuentra una raíz no trivial de la unidad. El algoritmo fracasa si $g^{k/2^t} \equiv 1 \bmod n$ o si para algún $i \leq t$ se tiene que $g^{k/2^i} \equiv -1 \bmod n$.

Sea ahora $X = \left\{ g \in \left(\mathbb{Z}/n\mathbb{Z}\right)^* : \text{ el algoritmo fracasa} \right\}$. Bastará contar el número de elementos de X para obtener la proporción entre los g's válidos y los no válidos. Se define F como el producto de las potencias de primos impares de la descomposición en factores primos de $p - 1$ y $q - 1$. Es decir, $k = 2^t r$ es múltiplo de $(p - 1)(q - 1)$, y

$$\begin{aligned} p - 1 &= 2^{t_1} p_1^{\alpha_1} \cdots p_{k_1}^{\alpha_{k_1}} l_1^{\gamma_1} \cdots l_s^{\gamma_s} \\ q - 1 &= 2^{t_2} q_1^{\beta_1} \cdots q_{k_2}^{\beta_{k_2}} l_1^{\delta_1} \cdots l_s^{\delta_s} \\ F &= p_1^{\alpha_1} \cdots p_{k_1}^{\alpha_{k_1}} q_1^{\beta_1} \cdots q_{k_2}^{\beta_{k_2}} l_1^{\gamma_1 + \delta_1} \cdots l_s^{\gamma_s + \delta_s}. \end{aligned}$$

Se tiene que $t \geq t_1 + t_2$, y además, $t_1, t_2 \geq 1$. Es fácil ver que F divide a r. Por otra parte, $\left|\left(\mathbb{Z}/n\mathbb{Z}\right)^*\right| = \varphi(n) = 2^{t_1 + t_2} F$. Antes de continuar, es conveniente enunciar el lema siguiente, cuya prueba es inmediata a partir del Corolario A.9 (Apéndice A).

Lema 2.4. *Sea $n = pq$. La ecuación $x^k \equiv a \bmod n$ tiene solución si y solo si $a^{(p-1)/d_p} \equiv 1 \bmod p$ y $a^{(q-1)/d_q} \equiv 1 \bmod q$, donde $d_p = mcd(k, p - 1)$ y $d_q = mcd(k, q - 1)$. Además, si se da esta condición, la ecuación tiene exactamente $d_p d_q$ soluciones distintas módulo n.*

El conjunto X está formado por dos tipos de elementos: aquellos tales que $g^r \equiv 1$, y aquellos tales que $g^{2^i r} \equiv -1 \bmod n$ para algún i. El número de elementos del primer tipo es el número

de soluciones de la ecuación $g^r = 1$ mód n, que por el Lema anterior, siempre tiene exactamente $mcd(r, p-1) \cdot mcd(r, q-1) = F$ soluciones.

Para contar los elementos del segundo tipo, cabe notar que la ecuación $g^{2^i r} \equiv -1$ mód p tiene solución únicamente si $(-1)^{(p-1)/d} \equiv 1$ mód p, siendo $d = mcd(2^i r, p-1) = p-1$ si $i \geq t_1$ y $d = (p-1)/2^{t_1 - i}$ en otro caso. Por tanto, existe solución si $i < t_1$, y el número de soluciones de la ecuación en este caso es $(p-1)/2^{t_1-i}$. El escenario es análogo para q y t_2.

Se considera ahora la ecuación $g^{2^i r} \equiv -1$ mód n, que por el Lema 2.4 tiene solución cuando las respectivas ecuaciones para p y q la tienen. Es decir, asumiendo sin pérdida de generalidad que $t_2 \geq t_1$, solo existe solución cuando $i < t_1$, y hay exactamente $(p-1)(q-1)/2^{t_1+t_2-2i} = 2^{2i}F$ soluciones. Sumando sobre $i = 0, \ldots, t_1 - 1$, el número de elementos del segundo tipo es $\sum_{i=0}^{t_1-1} 2^{2i}F = \frac{4^{t_1}-1}{3}F$.

Finalmente, se divide la suma del número de elementos de ambos tipos entre el cardinal del grupo para obtener la proporción de g's no válidos:

$$\frac{|X|}{\varphi(n)} = \frac{F + \frac{4^{t_1}-1}{3}F}{2^{t_1+t_2}F} \leq \frac{4^{t_1}+2}{3 \cdot 2^{2t_1}} = \frac{1}{3} + \frac{1}{3 \cdot 2^{2t_1-1}} \leq \frac{1}{2}.$$

Con lo que la probabilidad de escoger un g válido al azar es mayor que $1/2$. Por último, en cada iteración del algoritmo se calculan a lo sumo $t < \log_2 ed$ exponenciaciones modulares. Como cada una de ellas tiene un coste polinomial en los tamaños de e, d, n, cada iteración tendrá también un coste polinomial en los tamaños de e, d y n. □

La demostración anterior recuerda al test de primalidad de Rabin-Miller, y de hecho, se inspira en un argumento de este último [29]. La diferencia principal entre ambos procedimientos es el exponente k de la prueba anterior; en el test de Rabin-Miller $k = n-1$, mientras que aquí k es un múltiplo de $\varphi(n)$.

A continuación, se describen tres propiedades destacadas de RSA, que son: su homomorfía, la existencia de varias claves de descifrado, y la presencia de mensajes inalterables.

Proposición 2.5. *El cifrado RSA es homomórfico. Dados dos mensajes M y M', una misma clave pública (n, e), y sus respectivos criptogramas C y C', el cifrado del mensaje $M \cdot M'$ es $C \cdot C'$.*

Demostración. Basta ver que $C \equiv M^e$ mód n, $C' \equiv M'^e$ mód n, y por tanto $C \cdot C' \equiv (M \cdot M')^e$ mód n. □

Proposición 2.6. *Dado un par de clave pública (n, e), la clave de descifrado d definida como $ed \equiv 1$ mód $\varphi(n)$ no es única.*

Demostración. Un entero r sirve como clave de descifrado si para todo mensaje M cumple que $M^{er-1} \equiv 1$ mód n, lo que es equivalente a $M^{er-1} \equiv 1$ mód p y $M^{er-1} \equiv 1$ mód q. Estas dos ecuaciones se cumplen para cualquier $er - 1$ que sea múltiplo de $p-1$ y de $q-1$, y si $L = mcm(p-1, q-1)$, la condición es equivalente a

$$er \equiv 1 \text{ mód } L.$$

De hecho, si d es la clave original, cualquier $\tilde{d} = d + kL$ mód n será una clave de descifrado válida, habiendo en total $\varphi(n)/L = mcd(p-1, q-1)$ claves distintas. Como $p-1$ y $q-1$ son ambos pares, $mcd(p-1, q-1) \geq 2$ y se da la igualdad si $p = 2s+1$, $q = 2t+1$ con t, s primos entre sí. □

Esta característica implica que la condición de que e, d sean inversos módulo $\varphi(n)$ es condición suficiente, pero no necesaria, para el buen funcionamiento de RSA. La condición necesaria es que $ed \equiv 1$ mód $\lambda(n)$ y se deriva de la definición de la función de Carmichael (ver Apéndice

A) que es el menor entero positivo t tal que $a^t \equiv 1$ mód n para todo a coprimo con n. En concordancia con la Proposición anterior, $\lambda(n) = mcm(p-1, q-1) = L$.

Aunque a lo largo del trabajo se sigue la definición original de RSA con $ed \equiv 1$ mód $\varphi(n)$, la generación de claves módulo $\lambda(n)$ se utiliza en algunos estándares como PKCS#1 (ver [21] para más detalles).

Proposición 2.7. *Dado un par de clave pública (n, e), existen al menos 9 mensajes inalterables, esto es, $M \in {}^{\mathbb{Z}}\!/_{n\mathbb{Z}}$ tales que $M^e \equiv M$ mód n.*

Demostración. Aplicando el Teorema Chino de los Restos, que un mensaje sea inalterable equivale a $M^e \equiv M$ mód p y $M^e \equiv M$ mód q. El número de soluciones de la ecuación de congruencia $M^{e-1} \equiv 1$ mód p es $mcd(e-1, p-1)$, con lo que, añadiendo la solución trivial, la ecuación $M^e \equiv M$ mód p tiene $1 + mcd(e-1, p-1)$ soluciones. El razonamiento es similar para q, y por tanto, el número de mensajes inalterables será el de todas las combinaciones posibles de cada caso:

$$(1 + mcd(e-1, p-1))(1 + mcd(e-1, q-1)).$$

Como $e-1$, $p-1$ y $q-1$ son todos pares, este número será mayor o igual que 9. □

2.2. Vulnerabilidades en la implementación

En esta sección se verán algunas vulnerabilidades que dan lugar a ataques utilizando técnicas relativamente sencillas. Estas se aprovechan de algunas de las características mencionadas en la sección anterior o de una implementación deficiente o débil del criptosistema. Es por ello que se proponen varias precauciones básicas a tener en cuenta en la práctica.

Padding

Para comenzar, a partir de lo expuesto en párrafos anteriores se pueden deducir dos vulnerabilidades:

- RSA es maleable. Si un atacante E intercepta un criptograma $C = M^{e_A}$ dirigido hacia A, puede alterar el mensaje de forma consciente. Por ejemplo, si E le envía a A el criptograma alterado $C' = C \cdot 2^{e_A}$ mód n, el mensaje descifrado por A será $2M$ mód n, lo que se deduce de la propiedad de homomorfía. Este hecho puede incluso dar lugar a que el adversario recupere el mensaje original[1].

- RSA es determinista. Cada vez que B le envíe un mensaje M a A, este será cifrado de forma idéntica. Por tanto, si E sospecha que B le está enviando un mensaje M al usuario A, tan sólo necesita calcular M^{e_A} y compararlo con el mensaje interceptado para cerciorarse de estar en lo correcto.

Ambos problemas se solucionan añadiendo una cadena aleatoria de caracteres a cada mensaje, técnica conocida como "padding". Esto se puede realizar dado que en la práctica, cada mensaje enviado se divide en bloques con una longitud determinada por el tamaño de n. Si se añaden caracteres aleatorios en cierta parte fija del mensaje, es fácil desecharlos tras descifrar para recuperar el mensaje original, en una tarea análoga a la que se realiza aplicando códigos correctores. De esta manera, ni E puede introducir cambios previsibles en los mensajes, ni podrá comparar criptogramas, ya que B cifrará M de una forma distinta cada vez. El padding no solamente es habitual en RSA, sino en muchos sistemas criptográficos en general.

Sumado a lo anterior, Boneh, Joux y Nguyen presentan en [4] un algoritmo que permite recuperar cierto tipo de mensajes cifrados mediante RSA plano (es decir, sin aplicar ninguna

[1] En el apartado 4.5.3 de Durán [16] se puede encontrar más información al respecto.

técnica de padding) en tiempo polinomial en la longitud del mensaje con un 18 % de probabilidad de éxito.

Autentificación

Por otra parte, se encuentra el problema de la autentificación de los mensajes. Siguiendo el protocolo RSA, A no tiene forma de saber si B es quien realmente ha cifrado el mensaje, ya que podría hacerlo cualquiera con el conocimiento de la clave pública de A. Por este motivo, RSA es susceptible a un ataque por suplantación de identidad, en el que un atacante E podría enviarle un mensaje a A haciéndose pasar por B. Para evitar esto, RSA permite autentificar los mensajes, asegurando que el mensaje no ha sido suplantado y que es B quien realmente lo ha emitido [37].

Para autentificar un mensaje M, B puede cifrarlo inicialmente con su clave privada d_B, y luego con la clave pública de A, e_A. A recibirá entonces el mensaje $\tilde{C} = M^{e_A d_B}$, que podrá descifrar utilizando su propia clave privada d_A y la clave pública de B e_B. El hecho de que lo pueda descifrar utilizando la clave pública de B prueba que el mensaje es auténtico, ya que solo él es conocedor de su clave privada. Además, el mensaje no puede ser alterado por E ya que necesitaría conocer d_B para simular que el mensaje proviene de B.

Elección de módulo n común

Una de las mayores tentaciones para una autoridad o entidad central que proponga una red de comunicación segura entre sus usuarios mediante RSA puede ser utilizar el mismo módulo n para todos los usuarios de la red [2]. En principio, los usuarios no requieren del conocimiento de p y q, sino tan solo de sus claves de cifrado y descifrado respectivas, que pueden ser únicas para cada usuario.

Por el contrario, de acuerdo con el Teorema 2.3, cada usuario puede recuperar la factorización de n a partir de sus claves, y así calcular las claves privadas de todos los usuarios de la red, rompiendo por completo la seguridad de la misma. Por ello, nunca se debe utilizar el mismo módulo para varios usuarios de RSA, a pesar de que ello implique un mayor número de cálculos.

Cifrado de mensajes cortos

Si se desea cifrar un mensaje M muy corto, tal que $M^e < n$, entonces $C = M^e$ y es trivial recuperar el mensaje, calculando $M = \sqrt[e]{C}$. Este ataque, que puede parecer demasiado evidente, no lo es tanto cuando se usan exponentes de cifrado pequeños o al descomponer mensajes largos en varios bloques.

Para evitar este tipo de ataques, se utiliza una técnica denominada *salado,* que se asemeja al padding, basada en añadir caracteres aleatorios con el objetivo de que el tamaño del mensaje sea similar al de n o al del tamaño de bloque fijado por defecto [51].

2.3. Algunos primeros ataques

2.3.1. Ataque de Håstad

No es infrecuente que varios usuarios utilicen una clave de cifrado e pequeña. Como se ha visto, $e = 3$, $e = 17$ son elecciones comunes, que agilizan mucho el cifrado de los mensajes. El ataque siguiente, debido a Håstad [19], muestra que es posible recuperar un mismo mensaje M cifrado para un número $r \geq e$ de usuarios distintos si todos ellos utilizan la misma clave de cifrado.

Proposición 2.8. *Sean $(n_1, e), \ldots, (n_r, e)$ pares de clave pública RSA tales que los n_i son coprimos dos a dos y $r \geq e$. Sean $n_0 = \min\{n_1, \ldots, n_r\}$ y $n = n_1 \cdots n_r$. Entonces, para cualquier mensaje $M < n_0$, dados $C_i \equiv M^e \bmod n_i$ y las claves (n_i, e), es sencillo recuperar M.*

Demostración. En primer lugar, como $e \leq r$ y $M < n_0$, es claro que $M^e < n_1 \cdots n_r = n$. Aplicando el Teorema Chino de los Restos sobre las congruencias $C_i \equiv M^e \bmod n_i$, se tiene que la solución única C módulo n verifica que $C \equiv M^e \bmod n$, y como $0 < M^e < n$, entonces $C = M^e$. Obtener M es directo calculando $M = \sqrt[e]{C}$. □

Existe una versión más fuerte de este ataque que se verá en el capítulo 5.

2.3.2. Ataques Cíclicos

Los ataques cíclicos fueron propuestos inicialmente por Simmons y Norris [42]. Obedecen a una propiedad de RSA que en cierta forma es similar a la de los mensajes inalterables, y se basan en que dados un par de clave pública (n, e), un mensaje M y su criptograma C coprimo con n, se verifica que $C^{e^l} \equiv C \bmod n$ para un cierto entero l. Si un atacante consigue encontrar l, inmediatamente puede descifrar M aplicando que, como $C \equiv M^e \bmod n$, entonces $C^{e^{l-1}} \equiv M \bmod n$.

El procedimiento a seguir es aumentar sucesivamente l, cifrando repetidamente el criptograma, hasta que $C^{e^l} \equiv C \bmod n$ (es decir, hasta encontrar un ciclo). Por tanto, presenta una amenaza importante cuando existe un exponente l pequeño. El siguiente resultado ofrece una cota superior para l.

Proposición 2.9. *Sea (n, e) un par de clave pública y sea C un criptograma primo con n. Entonces, existe un entero positivo l que divide a $\lambda(\lambda(n))$ tal que $C^{e^l} \equiv C \bmod n$, donde λ es la función de Carmichael.*

Demostración. Para comenzar, cabe recordar que el orden de todo elemento del grupo multiplicativo $\left(\mathbb{Z}/\varphi(n)\mathbb{Z}\right)^*$ divide a $\lambda(n)$ (ver Teorema A.6), y que e es primo con $\varphi(n)$ y por tanto con $\lambda(n)$. Como $C^{e^l} \equiv C \bmod n$, el orden de C divide a $e^l - 1$, con lo que $\lambda(n) \mid e^l - 1$. Se tiene entonces que $e^l \equiv 1 \bmod \lambda(n)$. Finalmente, si se elige l como el menor entero tal que esta ecuación se satisface, $l \mid \lambda(\lambda(n))$ por definición de la función de Carmichael. La existencia de l está garantizada ya que con $l = \lambda(\lambda(n))$ se da la condición del enunciado. □

De la Proposición anterior se deduce que si $\lambda(\lambda(n))$ es pequeño, las posibilidades de encontrar un exponente l en un tiempo razonable son mucho mayores. Es posible defenderse de este ataque eligiendo p y q de forma que $l = \lambda(\lambda(n)) = \lambda(mcm(p-1, q-1))$ sea grande, lo que se verifica cuando $p-1$ y $q-1$ tienen factores primos grandes.

En 1979, Williams y Schmid [49] propusieron una alternativa al procedimiento anterior, que consiste en buscar ciclos módulo p o q en lugar de módulo n. Para ello, proponen evaluar repetidamente el valor

$$g = mcd(C^{e^l}, n)$$

hasta encontrar que $g \neq 1$. Si $g \neq 1$ y $g \neq n$, la factorización de n es revelada, mientras que si $g = n$ se puede hallar $M \equiv C^{e^{l-1}} \bmod n$ como en el ataque original. El número de iteraciones en este caso está acotado por $\lambda(\lambda(p)) = \lambda(p-1)$ (o por $\lambda(q-1)$), luego el ataque puede ser efectivo nuevamente si $p-1$ y $q-1$ tienen factores primos pequeños. Sin embargo, cada iteración de este ataque es más lenta debido al cálculo del máximo común divisor.

Finalmente, cabe mencionar que Rivest y Silverman [36] argumentan que no es necesaria una protección adicional frente a ellos, ya que eligiendo números primos suficientemente grandes de forma aleatoria, la probabilidad de que un ataque cíclico tenga más éxito que un ataque por

factorización es mínima. En la sección 3.3 se discute con más detalle la elección de primos para RSA.

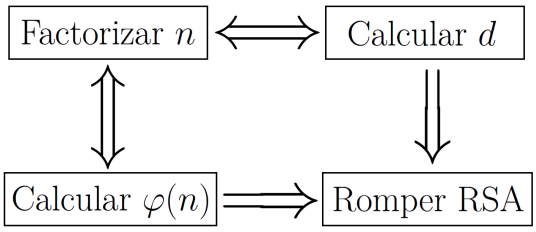
A continuación se muestra un ejemplo de ataque cíclico, en su forma original. Alicia desea enviarle a Bonifacio el mensaje '*RSA*', para lo cual lo representa como $M = 18 \cdot 26^2 + 19 \cdot 26 + 1 = 12663$. Alicia cifra el mensaje con la clave pública de Bonifacio, $n = 241001$, $e = 13$, obteniendo $C = M^e$ mód $n = 187047$, y se lo envía.

Un atacante intercepta el mensaje cifrado C y decide aplicar un ataque cíclico, para lo que calcula $C_{i+1} = C_i^e$ mód n repetidas veces comenzando con $C_1 = C$, hasta obtener $C_k = C = 187047$. Los sucesivos C_i son:

187047	107270	101496	89285	157006	46012	193238
30164	39498	163200	52203	180442	230817	7747
101341	169448	108635	157773	33724	***12663***	**187047**

El criptograma C se repite tras 20 iteraciones, luego el mensaje M es el resultado previo a la última iteración, $M = 12663 = RSA$. El ataque ha sido muy eficaz en este caso ya que $n = 241001 = 601 \cdot 401$, luego $\lambda(\lambda(n)) = \lambda((mcm(600, 400)) = \lambda(1200) = 20$. Por tanto, 20 es el número máximo de iteraciones que necesitará un ataque cíclico para cualquier mensaje cifrado con $n = 241001$.

A lo largo de este capítulo, se ha visto el funcionamiento de RSA, algunas de sus características más importantes, y varias vulnerabilidades y ataques elementales que ilustran claros ejemplos de un uso inapropiado del criptosistema. La seguridad de RSA depende de una serie de factores clave, que tras lo visto hasta ahora, se resumen en el siguiente diagrama de reducciones, donde $A \implies B$ significa que B es reducible polinomialmente a A (es decir, que a partir de A es sencillo resolver B, ver Apéndice).



Capítulo 3

Ataques por Factorización

En el capítulo anterior se ha visto que la seguridad de RSA está fuertemente ligada al problema de la factorización de enteros, y en particular, de aquellos que tienen únicamente dos factores primos. La importancia de la factorización es fundamental, hasta el punto en el que hoy en día la longitud de los primos utilizados en RSA se determina en base a la capacidad actual de los ordenadores de descomponer números en sus factores. De hecho, para realizar un seguimiento de los avances en factorización, los Laboratorios RSA ofrecían premios económicos a quien lograra factorizar ciertos módulos, en los llamados RSA Challenges[1]. En este capítulo se estudiará esta relación entre RSA y la factorización. Para comenzar, se verán algoritmos de factorización de diversa índole. Posteriormente, se tratará brevemente la generación de primos; y por último se estudiarán aquellos tipos de primos que dotan a RSA de una mayor seguridad.

3.1. Algoritmos de Factorización

A pesar de que a lo largo de la historia siempre ha habido un interés por factorizar números enteros, no ha sido hasta las últimas décadas del siglo XX, con la aparición de RSA y la mejora en la capacidad de computación, cuando han surgido algoritmos de factorización eficaces. Una gran parte de estos algoritmos, considerados de propósito especial, son indicados para descomponer números con ciertas características, pero su complejidad (asintótica) no es buena en el caso general. Por ejemplo, utilizar la criba de Eratóstenes es ideal cuando el número es producto de primos pequeños, pero resulta extremadamente lento en otro caso. El tiempo de ejecución de otros, de propósito general, depende fundamentalmente del tamaño del número a factorizar. En esta sección se estudian varios algoritmos de ambos tipos. Las principales referencias generales utilizadas son Crandall y Pomerance [10], Riesel [35], y Bressoud [5].

3.1.1. Algoritmos de propósito especial

Algoritmo de Fermat

Es razonable que los factores p y q del módulo n de un usuario sean de la misma longitud, dado que si uno de ellos es mucho más pequeño que otro, se obtendrá más fácilmente mediante ciertos algoritmos de factorización (algunos de ellos se verán a lo largo de esta sección). Sin embargo, también se tiene que tomar precaución en que p y q no sean demasiado cercanos. El método que se describe a continuación, ideado por Fermat, permite romper RSA en este escenario y es válido para factorizar enteros en general. Si $n = pq$, se tiene:

$$(p+q)^2 - 4n = (p-q)^2 .$$

[1] Más información en: `https://en.wikipedia.org/wiki/RSA_Factoring_Challenge`

Si $p \approx q$, el término de la derecha es pequeño y $(p+q) \gtrsim 2\sqrt{n}$. El método consiste en encontrar un par x, y distintos de $n \pm 1$ tales que $4n = x^2 - y^2$. Si se encuentran, entonces $p = \frac{1}{2}(x-y)$, $q = \frac{1}{2}(x+y)$, revelando la factorización de n. La búsqueda se realiza mediante tanteo, probando $x = \lceil 2\sqrt{n} \rceil, \lceil 2\sqrt{n} \rceil + 1, \ldots$ hasta que $x^2 - 4n$ sea un cuadrado perfecto. Los resultados siguientes [11] prueban la eficacia del método.

Lema 3.1. *Sean $n = pq$ y $\delta = |p-q|$. Entonces, $0 < p + q - 2n^{1/2} < \frac{\delta^2}{4n^{1/2}}$.*

Demostración. La desigualdad de la izquierda es consecuencia de la desigualdad entre las medias aritmética y geométrica. Para probar la de la derecha,

$$\delta^2 = p^2 + q^2 - 2n = (p+q)^2 - 4n = (p + q + 2n^{1/2})(p + q - 2n^{1/2}).$$

Como $p + q > 2n^{1/2}$, el resultado se obtiene dividiendo ambos lados por $p + q + 2n^{1/2}$. □

Proposición 3.2. *La complejidad computacional del algoritmo de Fermat es $O(\delta^2/n^{1/2})$, donde $\delta = |p-q|$.*

Demostración. Despreciando el coste de la aritmética elemental, se cuenta el número de iteraciones del método. Como $x = \lceil 2\sqrt{n} \rceil, \lceil 2\sqrt{n} \rceil + 1, \ldots$, cuando el algoritmo encuentre la solución, el número de iteraciones habrá sido $x + 1 - \lceil 2\sqrt{n} \rceil$, que por el Lema 3.1 está acotado por

$$x + 1 - 2n^{1/2} = 1 + p + q - 2n^{1/2} < 1 + \frac{\delta^2}{4n^{1/2}}.$$

□

Se observa que si $\delta < cn^{1/4}$, el número de iteraciones está acotado por $1 + c^2/4$, con lo que el algoritmo es muy eficiente en este caso. Este resultado es bastante sorprendente, ya que tomando por ejemplo $p = 1000000087$, $q = 1000010029$, se tiene $c < 1$, luego el algoritmo debe ser capaz de hallar p, q a partir de $n = 1000010116000872523$ en un solo paso. En efecto, tomando $x = \lceil 2\sqrt{n} \rceil = 2000010116$, se obtiene el resultado:

$$x^2 - 4n = 98843364 = 9942^2 = (p-q)^2.$$

ρ de Pollard

Este algoritmo probabilístico, creado por Pollard en 1975 [32] es especialmente eficaz cuando el número a factorizar n tiene factores primos pequeños. La idea fundamental del algoritmo es crear una sucesión de números mediante una función recursiva aleatoria f módulo n hasta que se encuentra un ciclo.

Sean $p > 2$ un factor primo de n, x_0 un valor inicial, y $x_i \equiv f(x_{i-1}) \bmod n$ la sucesión que define la función f. Como existe un número finito de clases de equivalencia módulo p, si se calculan sucesivamente los pares x_k, x_{2k}, se tendrá que $x_k \equiv x_{2k} \bmod p$ para algún k. Entonces, como p divide a n, se tiene que $d = mcd(n, |x_{2k} - x_k|) > 1$ es un factor de n. Pollard propuso originalmente comprobar los pares x_i, x_j para distintos índices, pero calcular x_k, x_{2k} en su lugar reduce el tiempo de ejecución del algoritmo.

Una elección frecuente para la función es $f(x) = x^2 + 1$. Si dicha función fracasa, lo que puede ocurrir en el improbable caso de que $d = n$, es común elegir $f(x) = x^2 + c$ con $c \neq 0, 2$. Si c toma uno de estos dos valores, el algoritmo no funciona adecuadamente (ver [10]).

Proposición 3.3. *Si f se comporta como una función aleatoria, la complejidad del algoritmo ρ de Pollard es $O(\sqrt{p})$, donde p es el menor factor primo de n. En el peor caso, $O(\sqrt{p}) = O(n^{1/4})$.*

Una justificación (para un análisis detallado ver [17]) del método se basa en la conocida paradoja del cumpleaños, que en este contexto puede verse de la siguiente forma: dados $x_1, \ldots, x_r$ números aleatorios, la probabilidad de que dos de ellos sean congruentes módulo p es cercana

a $1/2$ cuando r es del orden de $\sqrt{p}$. Por lo tanto, la longitud esperada del ciclo del algoritmo ρ es $O(\sqrt{p})$.

Se muestra un ejemplo del algoritmo. Sea $n = 23711$ y $f(x) = x^2 + 1$. Comenzando con $x_0 = 2$, se tienen los valores:

n	1	2	3	4	5	6	7
x_n	5	26	677	7821	17373	3811	12590
x_{2n}	26	7821	3811	66	14650	1129	8660
$mcd(n, \lvert x_n, x_{2n}\rvert)$	1	1	1	1	1	1	131

Con lo que se encuentra el factor primo 131 tras 7 iteraciones, luego $n = 131 \cdot 181$.

$p - 1$ de Pollard

El método que se detalla a continuación, desarrollado también por Pollard [31], es muy rápido factorizando números compuestos n con un factor primo p tal que $p - 1$ es producto de factores pequeños. La idea fundamental es que por el Pequeño Teorema de Fermat, si para un cierto número M se cumple que $p - 1 \mid M$, entonces cualquier a verifica $a^M \equiv 1 \bmod p$, con lo cual p divide a $mcd(a^M - 1, n)$. En este algoritmo, se prueba a calcular este máximo común divisor con un cierto M producto de muchos factores primos pequeños y de sus potencias. Si $p - 1$ es producto de potencias de factores de M, el algoritmo devuelve p.

Se necesita, para comenzar, elegir una cota superior B que se da como parámetro de entrada al algoritmo. A continuación se siguen los siguientes pasos:

1. Se establece una base de primos $q_1 < q_2 < \cdots < q_m \leq B$, y para cada primo, el máximo exponente a_i cumpliendo que $q_i^{a_i} \leq B$.

2. Tomando $c = 2$, se calcula $c = c^{\prod q_i^{a_i}} \bmod n$. Esto puede hacerse calculando a_i veces la operación $c = c^{q_i} \bmod n$ para cada primo q_i.

3. Se calcula $l = mcd(c - 1, n)$. Si $l \neq 1$ y $l \neq n$, se ha encontrado un divisor no trivial de n.

Si el algoritmo devuelve $l = 1$, la cota B es insuficiente para el número a factorizar. Si devuelve $l = n$, lo cual ocurrirá muy raramente, se puede repetir el proceso variando el orden en el que se multiplican los primos q_i y calculando el máximo común divisor antes de terminar todas las multiplicaciones [10]. Si B es insuficiente, existe una fase 2 del algoritmo que permite aumentar la cota sin tener que repetir el proceso al completo. Consiste en elegir una nueva cota B', rescatar el último valor calculado c en la fase anterior, y efectuar:

4. Se establece una nueva base de primos $B < r_1, \cdots, r_k \leq B'$.

5. Se calcula recursivamente $c^{r_{i+1}} = c^{r_i} \cdot c^{r_{i+1} - r_i} \bmod n$.

6. Se devuelve $l = mcd(c - 1, n)$ análogamente a la fase 1.

Como las diferencias $r_{i+1} - r_i$ van a ser pares y en general pequeñas, conviene precalcular los valores c^2, c^4, etc. módulo n para acelerar el paso 5. En cada recursión de este paso se emplea una única multiplicación modular, con lo que se realizan un total de k productos. Esta fase 2 es más rápida que la primera y permite encontrar primos tales que $p - 1 = rh$, donde $h \mid M$ siendo M el exponente alcanzado en la fase 1, y r es un número primo tal que $B < r \leq B'$ (o un producto de primos en tal intervalo). La fase 2 descrita se corresponde con la original, aunque existen varias versiones en la literatura que ofrecen mejoras o formulaciones diversas. La mejora en agilidad respecto a la fase 1 conlleva sin embargo ignorar aquellos factores que son potencias de primos q_i con un exponente mayor que el a_i alcanzado en la fase 1.

Proposición 3.4. *La complejidad de la fase 1 del algoritmo $p-1$ es $O(B \log B (\log n)^2)$.*

Un logro bastante ilustrativo de este algoritmo [51] fue descomponer el número de Mersenne $2^{257}-1 = p_{15} \cdot p_{25} \cdot p_{39}$, donde $p_{25}-1 = 2^3 \cdot 3^2 \cdot 19^2 \cdot 47 \cdot 67 \cdot 257 \cdot 439 \cdot 119173 \cdot 1050151$. Al estar compuesto por potencias de primos pequeñas, el algoritmo es muy eficaz para encontrar este factor.

Se muestra un ejemplo con $n = 1526407$. Se elige $B = 10$, luego la base de primos es $2, 3, 5, 7$ con exponentes $3, 2, 1, 1$ respectivamente. Por tanto, $c^{\prod q_i^{a_i}} \bmod n = c^{2520} \bmod n = 1041125$. Como $l = mcd(c-1, n) = 1$, la cota $B = 10$ es insuficiente y se pasa a la fase 2.

Se elige la nueva cota $B' = 20$, con lo que la nueva base de primos es $11, 13, 17, 19$. Se calcula recursivamente $c = c^{11 \cdot 13 \cdot 17 \cdot 19} \bmod n = 104109$. Ahora, $l = mcd(c-1, n) = 1531$, que es un factor de $n = 1531 \cdot 997$. El factor 1531 se ha encontrado en la fase 2 dado que $1531 - 1 = 2 \cdot 5 \cdot 9 \cdot 17$.

$p+1$ de Williams

En 1982, Williams propuso un método [50] muy similar al $p-1$ de Pollard, que permite encontrar un factor p de un número tal que $p+1$ se descompone en factores primos pequeños. El método se basa en aplicar, en lugar de potencias de primos, las sucesiones de Lucas, también utilizadas en algunos test de primalidad. Se muestra su definición y un resultado fundamental para el funcionamiento del algoritmo.

Definición 3.5. *Dados dos enteros r, s, sean α y β las raíces del polinomio $x^2 - rx + s$. Se define la sucesión de Lucas $U_n(r,s)$ como $U_n(r,s) = \frac{\alpha^n - \beta^n}{\alpha - \beta}$. En su formulación recursiva, $U_n(r,s) = rU_{n-1}(r,s) - sU_{n-2}(r,s)$.*

Proposición 3.6. *Dado un primo p que no es divisor de s y dado el discriminante $\triangle = r^2 - 4s$, sea $\varepsilon = \left(\frac{\triangle}{p}\right)$ el símbolo de Legendre correspondiente. Entonces, $U_{(p-\varepsilon)m}(r,s) \equiv 0 \bmod p$ para cualquier entero $m \geq 0$.*

El algoritmo requiere una cota B al igual que el de Pollard, y busca encontrar divisores p de n de la forma $p+1 = \prod_{i=1}^{k} q_i^{b_i}$, donde cada $q_i^{b_i} \leq B$. Para ello, se calcula una base idéntica de primos q_i y exponentes a_i idéntica a la del algoritmo $p-1$, determinando $c = \prod_{i=1}^{k} q_i^{a_i}$. Es claro que $(p+1) \mid c$. Si además se dan las condiciones del Teorema 3.6 con $\varepsilon = -1$, entonces $p \mid U_c(r,s)$, luego también $p \mid mcd(U_c(r,s), n)$. Encontrar un no-residuo cuadrático junto con la sucesión de Lucas apropiada conlleva un procedimiento relativamente complejo que se detalla en el artículo original.

En la práctica, al algoritmo $p+1$ también se le puede incorporar una segunda fase muy similar a la del $p-1$, y es capaz de encontrar factores que el algoritmo $p-1$ no puede detectar. Sin embargo, es aproximadamente 9 veces más lento que el anterior en promedio, según sus autores. El interés para RSA de los dos algoritmos combinados reside en que, al elegir $n = pq$, puede ser sencillo encontrar los factores p, q si $p \pm 1$ o $q \pm 1$ se descomponen en potencias de primos pequeñas.

Curvas Elípticas de Lenstra

El algoritmo de Curvas Elípticas es uno de los más rápidos existentes hasta la fecha para enteros de cualquier tipo, pero se considera de propósito especial porque su tiempo de ejecución, subexponencial, depende del factor p más pequeño del número n sobre el que se aplica y no de n en sí. Fue propuesto por Henrik Lenstra en 1987 [24], y como se indica en el propio artículo, es una mejora del algoritmo $p-1$ de Pollard en el que se opera con los puntos de una curva elíptica aleatoria definida sobre el anillo $\mathbb{Z}/n\mathbb{Z}$.

Definición 3.7. *Sean a, b elementos en el anillo $\mathbb{Z}/n\mathbb{Z}$, con $mcd(n, 6) = 1$ y $mcd(4a^3+27b^2, n) = 1$; y sea O el punto en el infinito. El conjunto de puntos que define la curva elíptica $E_{a,b}$ sobre el anillo es*

$$E_{a,b}(\mathbb{Z}/n\mathbb{Z}) = \left\{(x, y) \in \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z} : y^2 = x^3 + ax + b\right\} \cup \{O\}$$

Si en lugar de definir la curva sobre un anillo cualquiera se define sobre un cuerpo, los puntos de la curva pueden ser dotados de una estructura de grupo aditivo. Como n no es primo, al definir la curva sobre $\mathbb{Z}/n\mathbb{Z}$ existirán pares de puntos P, Q tales que $P+Q$ no está bien definido. La idea fundamental de Lenstra se basa precisamente en encontrar puntos en la curva donde esta aritmética falla, que pueden revelar un factor no trivial de n.

Proposición 3.8. *La complejidad computacional del algoritmo de Curvas Elípticas es $O\left(\exp\left[(1 + o(1))\sqrt{2 \ln p \ln \ln p}\right]\right)$, donde p es el menor factor primo de n.*

Esta complejidad es subexponencial en el tamaño de p, luego el método es especialmente adecuado para descomponer enteros que poseen factores primos relativamente pequeños. A día de hoy, continúa siendo utilizado ampliamente.

3.1.2. Algoritmos de propósito general

Los algoritmos de propósito general más eficaces hoy en día se basan en la idea fundamental del algoritmo de Fermat, que no es otra que encontrar congruencias cuadráticas módulo n, es decir, un par de enteros x e y verificando

$$x^2 \equiv y^2 \text{ mód } n.$$

Los algoritmos de esta sección son sucesivas variantes y mejoras que aplican esta idea. Para justificar este propósito, se considera n un número impar con al menos dos factores primos diferentes. Si se encuentra una solución de la ecuación anterior, donde $x \not\equiv \pm y$, entonces $x - y$ divide a n. Bastará calcular $mcd(|x - y|, n)$ para determinar un factor no trivial de n.

Algoritmo de Dixon

Propuesto en 1981 por Dixon [13], tiene gran interés fundamentalmente por dos motivos. Primero, utiliza los conceptos clave que llevaron al desarrollo de la Criba Cuadrática o la Criba General, entre otros, pero de manera más sencilla. Y segundo, el autor proporciona una prueba rigurosa del tiempo de ejecución del mismo (subexponencial en el tamaño de n) que no requiere de argumentos heurísticos.

Este método se divide principalmente en dos etapas. La primera es buscar cuadrados z^2 módulo n en una lista de N enteros cuyos factores primos se encuentran en un conjunto P, compuesto por todos los primos menores que una cota B. Posteriormente, se utilizan relaciones entre los exponentes y álgebra lineal para construir los mencionados x e y que cumplen la ecuación de congruencias. El punto crucial para que el algoritmo sea rápido es la elección adecuada de las cotas B y N. Los pasos detallados aparecen a continuación.

Inicial Se crea una lista L con N enteros en el intervalo $[1, n]$, y un conjunto ordenado $P = \{p_1, \ldots, p_h\}$ con todos los primos menores que B. Se toman $V, Z = \emptyset$ y $j = 0$.

1. Mientras $L \neq \emptyset$, se realiza $j = j + 1$ y se extrae el primer elemento z_j de L.
2. Se calcula $w_j = z_j^2$ mód n, y se factoriza $w_j = w'_j \prod_i p_i^{a_i}$, donde w'_j no tiene factores en P. Si $w'_j \neq 1$, se vuelve al paso 1.

3. Sea $a_j = (a_{j1}, \ldots, a_{jh})$ un vector con los exponentes de los factores primos de w_j, en el orden en el que se encuentran en P. Se añade a_j en V, y z_j en Z. Si $|V| \leq h$, se vuelve al paso 1.

4. Cada elemento $a_j \in V$ se representa como un vector $f_j \in \mathbb{F}_2^h$. Se recorren los f_j y se encuentra el primer f_k linealmente dependiente de los anteriores (existe porque $|V| > h$). Se elimina a_k de V y z_k de Z, y se calculan $c_i \in \mathbb{F}_2$ tales que $f_k \equiv \sum_{i<k} f_i c_i$ mód 2.
   Se calcula $d = (d_1, \ldots, d_h) = (a_k + \sum f_i c_i)/2$.

5. Se toman $x = z_k \prod_{i<k} z_i^{c_i}$, e $y = \prod_{i=1}^{h} p_i^{d_i}$. De esta forma, se cumple que $x^2 \equiv \prod_{i=1}^{h} p_i^{2d_i} \equiv y^2$ mód n. Si $x \equiv \pm y$ mód n, se vuelve al paso 1. En caso contrario, $mcd(n, |x - y|)$ es un factor no trivial de n.

Así pues, las claves de la velocidad del algoritmo son fundamentalmente dos: reducir el número de cuadrados que se calculan combinando los exponentes de las distintas factorizaciones, y utilizar herramientas de álgebra lineal para agilizar los cálculos.

Proposición 3.9. *[13] Sea n un entero divisible por al menos dos primos distintos. Tomando $B = \exp\left[\sqrt{2 \ln n \ln \ln n}\right]$, y $N = B^2 + 1$, el número promedio de operaciones que el algoritmo de Dixon realiza sobre una lista L de tamaño N es $O(B^3)$, y la proporción de listas para las que el algoritmo fracasa es $O(n^{-1/2})$.*

Criba Cuadrática de Pomerance

La criba cuadrática fue introducida en 1982 por Pomerance [33], y aunque su autor no lo menciona directamente, puede estudiarse como una optimización del algoritmo de Dixon. Hasta la fecha, continúa siendo el algoritmo más rápido para factorizar números de en torno a 100 cifras decimales, como el propio Pomerance afirma en [34].

El algoritmo de criba cuadrática realiza exactamente los mismos pasos que el de Dixon, pero introduciendo dos mejoras. La primera consiste en elegir un conjunto L favorable, en lugar de hacerlo aleatoriamente. El objetivo es aumentar la probabilidad de que los elementos de L tengan factores primos menores que B. Para ello, conjetura que la probabilidad P_b de que si $x < n$ entonces x^2 mód n posea esta propiedad, es aproximadamente $P_b = u^{-u}$, donde $u = \ln n / \ln B$. Por otro lado, todos los enteros $x \in (\sqrt{n}, \sqrt{2n})$ verifican x^2 mód $n = x^2 - n$, luego si se escogen $x \in (\sqrt{n}, \sqrt{n} + N)$ y $N \ll n$ el orden de magnitud de $x^2 - n$ es $n^{1/2+\varepsilon}$, donde $0 < \varepsilon \ll 1$. Por lo tanto, en ese rango P_b aumenta, ya que en la expresión de u, $\ln n$ pasa a ser aproximadamente $\frac{1}{2} \ln n$.

La segunda mejora se basa en que al factorizar los distintos $x^2 - n$ mediante criba con todos los primos menores que B, solamente aquellos primos p para los que la ecuación $x^2 - n \equiv 0$ mód p puede tener solución son necesarios. Equivalentemente, son aquellos que verifican $\left(\frac{n}{p}\right) = 1$. Esto hace que, de nuevo heurísticamente, aproximadamente la mitad de los primos sean descartables.

En resumen, el esquema del algoritmo de Dixon prevalece con modificaciones en dos pasos. En el inicial, $L = \{\lceil\sqrt{n}\rceil, \lceil\sqrt{n}\rceil + 1, \ldots, \lceil\sqrt{n}\rceil + N\}$, y en el 2, se utilizan únicamente primos cuyo símbolo de Legendre es 1.

Proposición 3.10. *Asumiendo las hipótesis anteriores, la complejidad computacional de la criba cuadrática es $O\left(\exp\left[(1 + o(1))\sqrt{\ln n \ln \ln n}\right]\right)$.*

A continuación se muestra un ejemplo sencillo del algoritmo, siguiendo los pasos del algoritmo de Dixon y aplicando las mejoras indicadas. Sea $n = 1739$, y se eligen $B = 20, N = 100$. Como $\lceil\sqrt{n}\rceil = 42$, entonces $L = \{42, 43, \ldots, 141\}$. El conjunto P formado por los primos $p_i < B$ tales que $\left(\frac{n}{p_i}\right) = 1$ es $P = \{2, 5, 11, 13\}$. Seguidamente, se realiza:

- Se calcula $w_1 = z_1^2 - n = 42^2 - n = 25$ y se factoriza, $w_1 = 5 \cdot 5$. Como es un cuadrado perfecto, se elimina y se repite con z_2.
- Se calcula $w_2 = z_2^2 - n = 43^2 - n = 110 = 2 \cdot 5 \cdot 11$. Como todos sus factores están en P, se añade el vector (de exponentes de los p_i) $a = (1, 1, 1, 0)$ a V y $z_a = 43$ a Z.
- Se repite el proceso con todos los números de L hasta que V tiene 5 elementos (ya que $|P| = 4$). Se calculan los f_j a partir de los vectores a_j módulo 2, obteniendo:

j	z_j	a_j	$f_j \in \mathbb{F}_2^5$
2	43	$(1, 1, 1, 0)$	$(1, 1, 1, 0)$
4	45	$(1, 0, 1, 1)$	$(1, 0, 1, 1)$
17	58	$(0, 3, 0, 1)$	$(0, 1, 0, 1)$
45	86	$(3, 1, 1, 0)$	$(1, 1, 1, 0)$
49	90	$(3, 0, 1, 1)$	$(1, 0, 1, 1)$

- El primer vector f en $\mathbb{F}_2^5$ linealmente dependiente es el tercero (luego en este caso particular hubiera bastado con iterar hasta $j = 17$), correspondiente a $z_{17} = 58$. Se elimina la tercera fila de la tabla, es decir, $z_{17} = 58$ se elimina de Z y $a_{17} = (0, 3, 0, 1)$ de V, y $c_2 = c_4 = 1$ ya que el tercer vector de la tabla es suma de los dos primeros. Por tanto, $d = [(0, 3, 0, 1) + (1, 1, 1, 0) + (1, 0, 1, 1)]/2 = (1, 2, 1, 1)$.
- Se toman $x = 43 \cdot 45 \cdot 58 \equiv 934$ mód n e $y = 2 \cdot 5^2 \cdot 11 \cdot 13 \equiv 194$ mód n, que verifican que $x^2 \equiv y^2$ mód n. Finalmente, $mcd(x - y, n) = 37$ que es un factor no trivial de n, luego $n = 47 \cdot 37$.

Como se puede ver, el algoritmo requiere una inicialización compleja, y por ello no es óptimo para números con pocos dígitos como el del ejemplo. De hecho, cuando $\sqrt{2n} - \sqrt{n} < N$ como en el ejemplo, las mejoras de Pomerance respecto del algoritmo de Dixon desaparecen. Existen múltiples optimizaciones de la criba cuadrática y algunas variantes, entre las que cabe destacar dos: la criba cuadrática de polinomios múltiples, y la criba general del cuerpo de números. La primera se basa en la idea de sustituir la búsqueda directa de congruencias $x^2 \equiv y^2$ mód n por encontrar ternas (u, v, w) tales que $u^2 \equiv v^2 w$, donde w es sencillo de factorizar [51]. La segunda se introduce brevemente a continuación.

Criba General del Cuerpo de Números

A finales de la década de 1980, Lenstra y Pomerance habían conseguido revolucionar el paradigma de la factorización de enteros con sus respectivos algoritmos. Sin embargo, una carta de Pollard dirigida a varios matemáticos sugería un nuevo acercamiento al problema utilizando cuerpos de números [34]. A pesar de que Pollard lo propuso pensando en números particulares, como los de Mersenne, análisis de complejidad realizados por Lenstra, Pomerance y otros autores revelaron que el método podía factorizar enteros en el caso general en menor tiempo que cualquier otro algoritmo conocido. Su implementación práctica, sin embargo, es compleja. Debido a estas dificultades técnicas, el algoritmo no es el idóneo para enteros por debajo de las 130 cifras decimales.

La criba general se basa en buscar congruencias cuadráticas como las anteriores, pero a través de pares θ, $\phi(\theta)$, donde θ pertenece a un cierto anillo de enteros algebraicos, y ϕ es un homomorfismo del mismo en $\mathbb{Z}/n\mathbb{Z}$. Una explicación detallada puede encontrarse en [10].

Proposición 3.11. *La complejidad computacional de la criba general del cuerpo de números, asumiendo ciertas hipótesis heurísticas, es* $O\left(\exp\left[(1{,}923 + o(1))(\ln n)^{1/3}(\ln \ln n)^{2/3}\right]\right)$.

Algoritmo Cuántico de Shor

El algoritmo cuántico de Shor fue publicado en 1994 [41], y como su nombre indica, es un algoritmo diseñado para ser ejecutado por ordenadores cuánticos. Este algoritmo utiliza la transformada cuántica de Fourier (QFT) como herramienta principal para realizar exponenciaciones modulares, consiguiendo un tiempo de ejecución polinómico en el tamaño de n. Sin embargo, los computadores cuánticos actuales distan mucho de ser capaces de factorizar enteros de un tamaño considerable mediante este algoritmo.

El interés que tiene el algoritmo de Shor para este trabajo es que hasta la fecha es la única herramienta con la que se puede romper RSA a partir de la factorización de n en tiempo polinomial, independientemente de la implementación utilizada. Hoy en día existe un interés creciente por los sistemas criptográficos de clave pública resistentes a computadores cuánticos, en el paradigma de la conocida como criptografía postcuántica. Por ejemplo, el National Institute of Standards and Technology (NIST), ha propuesto una competición para elegir un estándar con estas características que culminará a mediados de la década de 2020[2].

3.2. Generación de primos

La necesidad de generar primos es común a todas las implementaciones de RSA. La forma más práctica de hacerlo es generar enteros impares del tamaño deseado al azar, aplicando sucesivamente un test de primalidad hasta que alguno sea efectivamente primo.

Para que este procedimiento sea computacionalmente eficiente se deben verificar dos condiciones: que los números primos sean suficientemente abundantes entre los enteros, y que los test de primalidad no sean costosos. El Teorema del Número Primo [38] muestra lo primero.

Teorema 3.12. *Sea* $\pi : \mathbb{N} \longrightarrow \mathbb{N}$, $\pi(x) = \sharp(\{p : p \text{ es primo}, \ p \leq x\})$ *la función contador del número de primos menores o iguales que* x*, y sea*

$$Li(x) = \int_2^x \frac{dt}{\ln(t)}.$$

Entonces, $\pi(x) \approx Li(x)$*. Si además* $x \geq 55$*, se verifica*

$$\frac{x}{\ln(x)+2} \leq \pi(x) \leq \frac{x}{\ln(x)-4}.$$

Si se desea encontrar un primo p de tamaño similar a un número dado a, y si a es suficientemente grande, $\pi(a) \approx a/\ln a$. Entonces, la probabilidad de que un cierto número aleatorio de este tamaño sea primo es aproximadamente $(\ln a)^{-1}$, y el número de test de primalidad que será necesario realizar en promedio será $O(\ln a)$.

La segunda condición requiere que los test de primalidad no sean costosos. Dado que los números primos abundan, es suficiente que un test de primalidad tenga una complejidad polinomial en el tamaño del número sobre el que se aplica para poder encontrar números primos en tiempo polinomial. Históricamente, distintos tests se han propuesto en la clase RP, como el de Solovay-Strassen en 1977 o el de Rabin-Miller en 1980. Estos algoritmos permiten determinar si un número es primo con una probabilidad de falso positivo que se puede acotar tanto como sea requerido, a base de realizar sucesivas iteraciones de los mismos. En 2004, fue publicado el test determinístico AKS, en la clase P. A pesar de ello, el test de Rabin-Miller es el más utilizado ya que permite alcanzar una probabilidad de error muy baja manteniendo un coste computacional notablemente menor que el del AKS.

[2]Más información sobre este proceso puede encontrarse en `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/`

Antes de realizar un test de primalidad sobre los enteros en un cierto intervalo, es conveniente realizar una criba para descartar múltiplos de primos pequeños y así evitar ejecutar el test en tantas ocasiones. En [35] se muestra que si se quiere encontrar un primo cercano a 10^{60}, y se toman los 5000 números impares siguientes a este número, cribar con todos los números primos menores de 1000 reducirá el número de candidatos a 81. De estos 81, de acuerdo con el Teorema 3.12, aproximadamente 72 serán primos. Este cribado no requiere una gran cantidad de cómputo (se puede llevar a cabo realizando el máximo común divisor entre el producto de los primos y el número), y reduce en gran medida el número de test necesarios.

3.3. Elección de primos para RSA

En la sección 3.1, se ha visto que existen factores primos que hacen que un módulo RSA $n = pq$ sea más vulnerable frente a algunos ataques por factorización. Para comenzar, es claro que p y q deben ser de un tamaño similar. Si uno de los dos es mucho más pequeño que el otro, puede ser encontrado de forma rápida mediante algoritmos como el ρ de Pollard, y especialmente mediante el de Curvas Elípticas de Lenstra. Por otra parte, no pueden ser demasiado cercanos porque entonces serían vulnerables al algoritmo de Fermat.

Existen una serie de números primos, conocidos como primos robustos, que garantizan que la implementación de RSA ofrezca una mayor resistencia frente a ciertos ataques [40].

Definición 3.13. *Se dice que p y q son primos robustos si verifican:*

- *El máximo común divisor de $p-1$ y $q-1$ es pequeño.*
- *$p-1$ y $q-1$ tienen factores primos grandes p' y q'.*
- *$p'-1$ y $q'-1$ tienen a su vez factores primos grandes.*
- *$p+1$ y $q+1$ tienen factores primos grandes.*
- *Idealmente, $(p-1)/2$ y $(q-1)/2$ son primos.*

Además de su resistencia frente a los algoritmos $p-1$ y $p+1$, estas propiedades reducen el número de mensajes inalterables y de múltiples claves de descifrado, como puede deducirse de las Proposiciones 2.6 y 2.7. Los primos robustos pueden ser generados mediante un método propuesto por Gordon (ver [16]), que requiere un 19 % más de tiempo en promedio que generar primos utilizando Rabin-Miller como se ha mencionado. También presentan una protección añadida frente a los ataques cíclicos que se han visto en el capítulo anterior. Una pareja de primos seguros cumple que $\lambda(n) = mcm(p-1, q-1)$ es grande y factoriza en primos grandes, luego la cota para el exponente de los ataques cíclicos (ver Proposición 2.9) es alta.

A pesar de todo, debido a la capacidad de cómputo actual, que obliga a elegir primos de gran tamaño, algunos autores (por ejemplo Schneier [40], Yan [51], y en especial Rivest y Silverman [36]) manifiestan que ya no es necesario tomar estas precauciones a la hora de elegir p y q para RSA. En principio, un atacante que desee romper el criptosistema mediante factorización nunca va a elegir algoritmos como el $p-1$ o el $p+1$, ya que para números grandes son mucho más lentos que otros como la Criba General o el de Curvas Elípticas. Aunque el atacante se decidiese a intentarlo, las probabilidades de éxito en promedio son muy bajas.

Por otra parte, tampoco es necesario tener una especial precaución en el caso de los mensajes inalterables; estos pueden ser fácilmente detectados al cifrar y corregidos introduciendo un nuevo padding. En cuanto a los ataques cíclicos, los métodos de factorización actuales tienen mayor probabilidad de éxito que éstos, lo que también descarta su utilización.

Capítulo 4

Ataques basados en Fracciones Continuas

En los capítulos anteriores se ha visto que una elección apropiada de los primos p, q es un factor fundamental a la hora de garantizar la seguridad de RSA. Sin embargo, no solo juegan un papel importante estos dos números sino también las claves de cifrado y descifrado e, d. Según lo visto hasta ahora, un usuario podría inocentemente elegir una clave privada d pequeña para acelerar el proceso de descifrado, ya que requiere de $O(\log d \log^2 n)$ operaciones. En 1990, Michael J. Wiener [48] propuso un ataque capaz de romper RSA cuando d es muy pequeño en comparación con n. El ataque de Wiener se basa en determinar d mediante fracciones continuas, una forma de realizar aproximaciones fraccionales de números reales con una serie de propiedades particulares.

En este capítulo, se estudiarán el ataque de Wiener, sus características, y distintas variantes del mismo basadas en la aproximación por fracciones continuas, que se introducen a continuación.

4.1. Fracciones Continuas

Definición 4.1. *Una fracción continua es una expresión de la forma*

$$a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cfrac{b_3}{a_3 + \ddots}}}$$

Donde los a_i, b_i son enteros positivos no nulos para $i \geq 1$. Una fracción continua se denomina finita si el conjunto formado por los a_i (y por los b_i) tiene cardinal finito.

En lo que sigue, el trabajo se centra en el caso en el que cada $b_i = 1$, de tal forma que una fracción continua queda determinada unívocamente por los a_i. Así, para representar una fracción continua finita f, se utiliza la notación $f = \langle a_0, a_1, \ldots, a_m \rangle$. Es fácil ver que un número es racional si y solo si su representación en fracción continua es finita.

Dado $x \in \mathbb{Q}$, su fracción continua asociada $x = \langle a_0, a_1, \ldots, a_m \rangle$ se calcula mediante el algoritmo de Euclides. En particular, si $x = c/d$, con $c, d \in \mathbb{Z}$, se obtiene:

$$\begin{aligned} c &= a_0 d + r_0 \\ d &= a_1 r_0 + r_1 \\ &\cdots \\ r_{m-2} &= a_m r_{m-1} + 0. \end{aligned}$$

En consecuencia, el número de cocientes a_i, así como el número de pasos necesarios para calcularlos, es lineal en el tamaño de d (de acuerdo con el Teorema de Lamé, ver Apéndice A). A raíz de lo anterior, se pueden observar dos propiedades básicas. La primera, que $a_m \geq 2$, ya que si $a_m = 1$ entonces $r_{m-2} = 1$ lo cual es imposible (nótese que $r_{m-1} > 0$). La segunda es la siguiente.

Proposición 4.2. *Para cualquier $k > 0$, se verifica:*

$$\begin{cases} \langle a_0, a_1, \ldots, a_r \rangle < \langle a_0, a_1, \ldots, a_r + k \rangle & \textit{si } r \textit{ es par} \\ \langle a_0, a_1, \ldots, a_r \rangle > \langle a_0, a_1, \ldots, a_r + k \rangle & \textit{si } r \textit{ es impar} \end{cases}$$

Por otra parte, dado $s > 0$,

$$\begin{cases} \langle a_0, a_1, \ldots, a_r \rangle < \langle a_0, a_1, \ldots, a_r, \ldots, a_{r+s} \rangle & \textit{si } r \textit{ es par} \\ \langle a_0, a_1, \ldots, a_r \rangle > \langle a_0, a_1, \ldots, a_r, \ldots, a_{r+s} \rangle & \textit{si } r \textit{ es impar} \end{cases}$$

Demostración. Para la primera afirmación, basta con reconstruir hacia atrás la fracción continua hasta llegar a una fracción simple. Si a_r aumenta, el denominador definido por $a_{r-1} + 1/a_r$ disminuye, lo que hará crecer el valor del siguiente denominador, y así sucesivamente. Como la fracción tiene r niveles, si r es par, se alterna un número par de veces, luego el valor total de la fracción aumenta. Si r es impar, sucede lo contrario.

Para la segunda afirmación es suficiente con observar que $\langle a_0, a_1, \ldots, a_{r+s} \rangle = \langle a_0, a_1, \ldots, a_r + k \rangle$, donde $k = 1/\langle a_{r+1}, \ldots, a_{r+s} \rangle$. Como $k > 0$, se tiene el resultado a partir de la primera afirmación. □

Para $0 \leq i \leq m$, se define la convergente i-ésima de la fracción continua $\langle a_0, a_1, \ldots, a_m \rangle$ como $f_i = \langle a_0, a_1, \ldots, a_i \rangle$. Cada f_i se corresponde con una fracción irreducible c_i/d_i, que es una aproximación de c/d que mejora sucesivamente a medida que i crece, como muestran los resultados siguientes [18].

Proposición 4.3. *Sea $f_i = c_i/d_i$ la convergente i-ésima de una fracción continua f. Se verifica:*

$$\begin{cases} c_0 = a_0 \\ c_1 = a_1 a_0 + 1 \\ c_i = a_i c_{i-1} + c_{i-2} \end{cases} \qquad \begin{cases} d_0 = 1 \\ d_1 = a_1 \\ d_i = a_i d_{i-1} + d_{i-2} \end{cases}$$

para $2 \leq i \leq m$. Además,

$$c_i d_{i-1} - c_{i-1} d_i = (-1)^{i-1}, \quad 1 \leq i \leq m.$$

Demostración. Para la primera recurrencia, los casos $i = 0, 1$ son triviales operando con las fracciones. El caso general se prueba por inducción. Es fácil ver que se cumple para $i = 2$, luego suponiéndolo cierto para $i \leq k$, se tiene

$$\frac{c_k}{d_k} = \langle a_0, a_1, \ldots, a_k \rangle = \frac{a_k c_{k-1} + c_{k-2}}{a_k d_{k-1} + d_{k-2}}$$

donde los coeficientes c_i, d_i, $i \leq k$ dependen únicamente de $a_0, \ldots, a_k$. La convergente k+1-ésima será

$$\begin{aligned} \frac{c_{k+1}}{d_{k+1}} = \langle a_0, a_1, \ldots, a_k, a_{k+1} \rangle = \langle a_0, a_1, \ldots, a_k + \frac{1}{a_{k+1}} \rangle &= \frac{\left(a_k + \frac{1}{a_{k+1}}\right) c_{k-1} + c_{k-2}}{\left(a_k + \frac{1}{a_{k+1}}\right) d_{k-1} + d_{k-2}} \\ &= \frac{a_{k+1}(a_k c_{k-1} + c_{k-2}) + c_{k-1}}{a_{k+1}(a_k d_{k-1} + d_{k-2}) + d_{k-1}} = \frac{a_{k+1} c_k + c_{k-1}}{a_{k+1} d_k + d_{k-1}}. \end{aligned}$$

La segunda propiedad también se puede probar por inducción utilizando la recurrencia anterior. Para $i = 1$, se tiene $c_1 d_0 - c_0 d_1 = 1$. Suponiéndolo cierto para $i = k$, se tiene, para $i = k + 1$:

$$c_{k+1}d_k - c_k d_{k+1} = (a_{k+1}c_k + c_{k-1})d_k - c_k(a_{k+1}d_k + d_{k-1}) = -[c_k d_{k-1} - c_{k-1}d_k] = (-1)^{(k+1)-1}$$

□

Corolario 4.4. *Sean $f_0, \ldots, f_m$ las convergentes de una fracción continua, $f_i = c_i/d_i$. Si $1 \leq i \leq m$,*

$$f_i - f_{i-1} = \frac{c_i}{d_i} - \frac{c_{i-1}}{d_{i-1}} = \frac{(-1)^{i-1}}{d_i d_{i-1}}.$$

Proposición 4.5. *Dada una fracción continua f, sus convergentes $f_0, \ldots, f_m$, y dados $r, s < \lfloor m/2 \rfloor$, las convergentes f_i están ordenadas de la forma $f_0 < f_2 < \cdots < f_{2r} < f < f_{2s-1} < f_{2s-3} < \cdots < f_1$.*

Demostración. Cabe notar en primer lugar que por la Proposición 4.2, la sucesión de convergentes de índice par, $f_0, f_2, \ldots, f_{2r}$ es estrictamente creciente, mientras que la de índice impar, $f_1, \ldots, f_{2s-1}$, es estrictamente decreciente.

Se prueba a continuación que toda convergente de índice impar es mayor que cualquier convergente de índice par por reducción al absurdo. Sean $t, \mu < m$ que verifican $f_{2t+1} \leq f_{2\mu}$. Es claro que $f_{2t+1} > f_{2t}$ por el Corolario 4.4. Si $\mu \leq t$, entonces por la monotonía de las fracciones convergentes de índice par, $f_{2t+1} \leq f_{2\mu} \leq f_{2t}$, lo que contradice lo anterior. Si $\mu > t$, análogamente $f_{2\mu+1} < f_{2t+1} \leq f_{2\mu}$, lo que también es una contradicción.

Por último, dado que $f = f_m$ es la mayor de las convergentes pares o la menor de las convergentes impares, $f_{2r} < f < f_{2s-1}$ para cualesquiera r, s, con lo que termina la prueba. □

Corolario 4.6. *Sean $f_0, \ldots, f_m$ las convergentes de la fracción continua asociada a α, $f_i = c_i/d_i$. Si $1 \leq i \leq m$,*

$$\left| \alpha - \frac{c_i}{d_i} \right| < \frac{1}{d_i d_{i+1}}.$$

Demostración. La prueba es inmediata a partir de la Proposición anterior y del Corolario 4.4, notando que α siempre se encuentra en el intervalo entre dos convergentes consecutivas. □

Hasta este punto, se han introducido una serie de generalidades de las fracciones continuas. Cabe notar que a pesar de que se han obtenido estos resultados en el caso de fracciones continuas con un número finito de términos, son ampliables de manera inmediata al caso infinito. Los dos teoremas siguientes [23] muestran el potencial de las fracciones continuas como herramienta de aproximación, fundamental en los ataques que se tratan en este capítulo. Previamente, es conveniente definir el concepto de aproximación óptima de segunda clase.

Definición 4.7. *Sean $\alpha \in \mathbb{R}$ y $a, b \in \mathbb{Z}$ con $b > 0$. Se dice que a/b es una aproximación óptima de segunda clase de α si para cualesquiera $c, d \in \mathbb{Z}$ tales que $c/d \neq a/b$ y $0 < d \leq b$ se cumple que $|d\alpha - c| > |b\alpha - a|$.*

Teorema 4.8. *Toda aproximación óptima de segunda clase de un cierto $\alpha \in \mathbb{Q}$ es una convergente de la fracción continua asociada a α.*

Demostración. Sea a/b una aproximación óptima de segunda clase de $\alpha = \langle a_0, a_1, \ldots, a_m \rangle$. Se desea probar que $a/b = c_j/d_j$ para algún j, donde $c_j/d_j = f_j$ es la j-ésima convergente de α. En primer lugar, si $a/b < a_0$, se tiene que $|1 \cdot \alpha - a_0| < |\alpha - a/b| \leq |b\alpha - a|$, luego a/b no podría ser una aproximación óptima de segunda clase. Por lo tanto, $a/b \geq a_0$.

Se procede ahora por reducción al absurdo. Si a/b no es una convergente de α, es decir, $a/b \neq f_i \ \forall i \leq m$, se tienen dos posibles casos debido a la ordenación de las fracciones continuas

dada por la Proposición 4.5: $f_{k\pm1} < a/b < f_{k\mp1}$ para algún índice k, o $a/b > f_1$. En el primer caso, en el que se incluye el caso en el que $f = f_k = f_{k+1}$, se tiene

$$\left|\frac{a}{b} - \frac{c_{k-1}}{d_{k-1}}\right| = \left|\frac{ad_{k-1} - bc_{k-1}}{bd_{k-1}}\right| \geq \frac{1}{bd_{k-1}}$$

La última desigualdad se debe a que $a/b \neq c_{k-1}/d_{k-1}$. Paralelamente,

$$\left|\frac{a}{b} - \frac{c_{k-1}}{d_{k-1}}\right| < \left|\frac{c_{k+1}}{d_{k+1}} - \frac{c_{k-1}}{d_{k-1}}\right| \leq \left|\frac{c_k}{d_k} - \frac{c_{k-1}}{d_{k-1}}\right| = \frac{1}{d_k d_{k-1}}$$

donde la última igualdad está dada por el Corolario 4.4. Juntando ambos resultados,

$$\frac{1}{bd_{k-1}} \leq \left|\frac{a}{b} - \frac{c_{k-1}}{d_{k-1}}\right| < \frac{1}{d_k d_{k-1}}$$

luego $b > d_k$. Por otra parte, como f_{k+1} es mejor aproximación de α que a/b, se tiene

$$\left|\alpha - \frac{a}{b}\right| \geq \left|\frac{c_{k+1}}{d_{k+1}} - \frac{a}{b}\right| \geq \frac{1}{bd_{k+1}}.$$

Multiplicando todo por b, se llega a que $|b\alpha - a| \geq 1/d_{k+1}$. Además, por el Corolario 4.6, $|d_k\alpha - c_k| \leq 1/d_{k+1}$. En consecuencia, $|b\alpha - a| \geq |d_k\alpha - c_k|$ con $b > d_k$, luego a/b no es una aproximación óptima de segunda clase de α, lo que es una contradicción.

En el segundo caso, $a/b > c_1/d_1$. Entonces,

$$\left|\alpha - \frac{a}{b}\right| > \left|\frac{c_1}{d_1} - \frac{a}{b}\right| \geq \frac{1}{bd_1}, \quad \text{luego } |b\alpha - a| > \frac{1}{d_1}.$$

Por otro lado, $|1 \cdot \alpha - a_0| \leq 1/d_1$. Esto implica que $|b\alpha - a| > |1 \cdot \alpha - a_0|$, que es nuevamente una contradicción. □

Teorema 4.9. *Sean $\alpha \in \mathbb{Q}$ y $c, d \in \mathbb{Z}$ tales que*

$$\left|\alpha - \frac{c}{d}\right| < \frac{1}{2d^2}.$$

Entonces, c/d es una de las convergentes de la fracción continua asociada a α.

Demostración. Es claro que la propiedad del enunciado es equivalente a $|d\alpha - c| < 1/2d$. Por el Teorema anterior, bastará con probar que c/d es una aproximación óptima de segunda clase de α. Si existen $a, b \in \mathbb{Z}$ con $a/b \neq c/d$ tales que $|b\alpha - a| < |d\alpha - c| < 1/2d$, es decir, que a/b es una mejor aproximación que c/d,

$$\left|\frac{a}{b} - \frac{c}{d}\right| \leq \left|\alpha - \frac{c}{d}\right| + \left|\alpha - \frac{a}{b}\right| < \frac{1}{2d^2} + \frac{1}{2bd} = \frac{b+d}{2d^2 b},$$

donde se ha utilizado la desigualdad triangular. Por otra parte,

$$\left|\frac{a}{b} - \frac{c}{d}\right| = \left|\frac{ad - cb}{bd}\right| \geq \frac{1}{bd}.$$

Juntando ambas desigualdades,

$$\frac{1}{bd} \leq \left|\frac{a}{b} - \frac{c}{d}\right| < \frac{b+d}{2d^2 b}$$

con lo que $b > d$, y por definición, c/d es una aproximación óptima de segunda clase de α. □

4.2. Ataque de Wiener

Wiener fue el primero en revelar una vulnerabilidad importante en RSA. Hasta la publicación de su ataque [48] en 1990, elegir una clave de descifrado pequeña para acelerar el proceso de descifrado para el usuario no entrañaba ningún riesgo aparente, siempre y cuando no se cometiesen errores como los que se han visto a lo largo de los capítulos 2 y 3. El ataque se fundamenta en la aproximación mediante fracciones continuas, y se encuentra en la literatura en múltiples versiones. La más común de ellas, que se puede encontrar por ejemplo en Boneh [2] o Yan [51], es un caso particular de la original de Wiener. En este trabajo se probará primero la versión original, más general, siguiendo la prueba de Hinek [21], y se incluirá la primera como un corolario.

Cabe recordar, previamente al teorema, que la clave de descifrado d que se define como el inverso de e módulo $\varphi(n)$ no es única. De acuerdo con la Proposición 2.6, basta con que una clave de descifrado $\tilde{d}$ verifique la ecuación $e\tilde{d} \equiv 1 \text{ mód } mcm(p-1, q-1)$.

Teorema 4.10. *Sea (n, e) un par RSA de clave pública y $\tilde{d}$ una clave de descifrado válida. Sea K un entero tal que $e\tilde{d} = 1 + K\, mcm(p-1, q-1)$, $G = mcd(p-1, q-1)$, y sean g, k enteros positivos primos entre sí tales que $kG = Kg$. Si la clave de descifrado satisface*

$$\tilde{d} < \frac{n}{2sgk}$$

donde $s = p + q - 1$, es posible determinar $\tilde{d}$ en tiempo polinomial en el tamaño de n y en g/k.

Demostración. En primer lugar, teniendo en cuenta que $K\, mcm(p-1, q-1) = K(p-1)(q-1)/G$, y sustituyendo en la ecuación del enunciado,

$$e\tilde{d} = 1 + \frac{K}{G}(p-1)(q-1) = 1 + \frac{k}{g}(n-s).$$

Dividiendo por $n\tilde{d}$ en ambos lados,

$$\frac{e}{n} = \frac{k}{n\tilde{d}g}(n-s) + \frac{1}{n\tilde{d}} = \frac{k}{\tilde{d}g} + \frac{g - ks}{n\tilde{d}g}.$$

Reorganizando, como $\tilde{d}$ verifica la condición del enunciado y además $s > g$,

$$\left|\frac{e}{n} - \frac{k}{\tilde{d}g}\right| = \left|\frac{ks - g}{n\tilde{d}g}\right| < \frac{ks}{n\tilde{d}g} = \frac{ksg}{n(\tilde{d}g)^2}\tilde{d} < \frac{1}{2(\tilde{d}g)^2},$$

con lo que las condiciones del Teorema 4.9 se cumplen, y se puede afirmar que $k/\tilde{d}g$ es una de las convergentes de la expansión en fracciones continuas de e/n. En particular, existirá un j tal que $k/\tilde{d}g$ sea igual a la j-ésima convergente f_j, y será posible deducir de la ecuación $e\tilde{d} = 1 + (k/g)\varphi(n)$ que

$$\varphi(n) = e\frac{\tilde{d}g}{k} - \frac{g}{k} = \left\lfloor \frac{e}{f_j} \right\rfloor - \left\lfloor \frac{g}{k} \right\rfloor.$$

Esta relación entre $\varphi(n)$ y f_j es la que permite determinar cuál es la convergente correcta (es decir, el valor de j). Dada una convergente f_i cualquiera y un entero $m \geq 0$, es posible proponer una suposición de $\varphi(n)$ como $\gamma_{i,m} = \lfloor e/f_i \rfloor - m$. Se puede comprobar si $\gamma_{i,m} = \varphi(n)$ ya que si lo es, $p + q = n + 1 - \gamma_{i,m} = n + 1 - \varphi(n)$ y se da la igualdad siguiente,

$$\left(\frac{p+q}{2}\right)^2 - n = \left(\frac{p-q}{2}\right)^2$$

con lo que si el término de la izquierda es un cuadrado perfecto, la suposición es correcta, y obtener d es inmediato a partir de $f_j = k/dg$, sabiendo que $g/k = e/f_j - \varphi(n)$.

Para determinar la convergente correcta f_j, se pueden calcular todas las convergentes f_i de e/n, y comprobar si $\gamma_{i,0} = \varphi(n)$. En caso contrario, se repite el proceso con $m = 1, 2, \ldots$ hasta dar con el resultado.

En total, el número de convergentes es lineal en el tamaño de n y $m < \lfloor g/k \rfloor + 1$. Por último, todas las operaciones que se realizan tienen a lo sumo un coste polinomial en el tamaño de n, luego es posible determinar d en tiempo polinomial en el tamaño de n y en g/k. □

Cabe notar que en el escenario más usual, en el que $ed > n$, se da que $k > g$ y únicamente se requiere probar el caso $m = 0$. Una sutileza de este resultado es que la vulnerabilidad que explota el ataque depende realmente de la elección de e y no de d. En efecto, las claves de descifrado están asociadas a cada par (n, e), y basta con que una sola de ellas satisfaga la desigualdad inicial. El siguiente corolario es la versión más común, pero menos general, del teorema anterior.

Corolario 4.11. *Sea $n = pq$ con $q < p < 2q$, y sea $d < \frac{1}{3}n^{1/4}$. Dados (n, e) tales que $ed \equiv 1$ mód $\varphi(n)$ y $e < \varphi(n)$, es posible recuperar d en tiempo polinomial en el tamaño de n.*

Demostración. En primer lugar, $\varphi(n) = n - p - q + 1$ y $p + q - 1 < 3\sqrt{n}$, luego $|n - \varphi(n)| < 3\sqrt{n}$. Sea ahora $t \in \mathbb{Z}$ tal que $ed - t\varphi(n) = 1$. Entonces,

$$\left|\frac{e}{n} - \frac{t}{d}\right| = \left|\frac{ed - t\varphi(n) - tn + t\varphi(n)}{nd}\right| = \left|\frac{1 - t(n - \varphi(n))}{nd}\right| \leq \frac{3t}{d\sqrt{n}}.$$

Como $t\varphi(n) = ed - 1$ y $e < \varphi(n)$, se tiene que $t < d < \frac{1}{3}n^{1/4}$. Se concluye que

$$\left|\frac{e}{n} - \frac{t}{d}\right| \leq \frac{1}{dn^{1/4}} \leq \frac{1}{3d^2} < \frac{1}{2d^2}.$$

Mediante una aproximación por fracciones continuas similar a la del resultado anterior, es posible encontrar t/d como una de las $O(\log n)$ convergentes de e/n. □

El corolario anterior es aplicable en las implementaciones más típicas de RSA, en las que p y q son del mismo tamaño, y las claves de cifrado y descifrado son menores que $\varphi(n)$. Por ello, el resultado da lugar a lo que se conoce como cota de Wiener: RSA es vulnerable cuando el tamaño de la clave de descifrado es, aproximadamente, una cuarta parte de la de n. Es posible llegar a una conclusión similar mediante aproximaciones a partir del Teorema 4.10, como se ve a continuación. Es claro que

$$k \approx \frac{e\tilde{d}}{mcm(p-1, q-1)} \approx \frac{e\tilde{d}g}{n}$$

Asumiendo $e \approx n$, $g \ll n$ y $s \approx \sqrt{n}$, se tiene:

$$\tilde{d} < \frac{n}{2sgk} \approx \frac{n^2}{2se\tilde{d}g^2}, \quad \text{luego } \tilde{d}^2 < \frac{n}{2sg^2} \approx n^{1/2},$$

de donde se obtiene la aproximación $\tilde{d} < n^{1/4}$. Las principales diferencias entre las dos versiones son dos: la generalización de la cota, por un lado; y la posibilidad de encontrar una de las múltiples claves de descifrado en lugar del (único) inverso de e módulo $\varphi(n)$.

La forma más directa de defenderse frente al ataque de Wiener preservando que la clave de descifrado sea pequeña es aumentar el denominador $2sgk$ de la cota superior de $\tilde{d}$. Esto se puede lograr de dos formas. La primera es aumentar g, que se puede conseguir si $mcd(p-1, q-1)$ es grande. Esta solución no es muy recomendable dado que debilita el criptosistema frente a

otros ataques, como por ejemplo por factorización, y dado que aumenta el número de claves de descifrado. La segunda es aumentar k, lo que se puede lograr escogiendo una clave de cifrado e grande.

Proposición 4.12. *Si la clave pública e satisface $e > n^{3/2}$, el ataque de Wiener fracasa para cualquier clave de descifrado $\tilde{d}$.*

Demostración. Se probará que si $e > n^{3/2}$, entonces la cota de Wiener $\tilde{d} < \frac{n}{2sgk}$ implica que $\tilde{d} < 1$. De la ecuación

$$\frac{e}{n} = \frac{k}{\tilde{d}g} - \frac{ks - g}{n\tilde{d}g} = \frac{k}{\tilde{d}g}(1 - \delta), \quad \delta \ll 1,$$

se tiene que si $e > n^{3/2}$, $k/\tilde{d}g > n^{1/2}$. Sustituyendo $k > n^{1/2}\tilde{d}g$ en la cota de Wiener, se obtiene

$$\tilde{d}^2 < \frac{n}{2sg^2\sqrt{n}} < 1,$$

donde la última desigualdad viene dada por $s > \sqrt{n}$. Se concluye que $\tilde{d} < 1$. □

Una alternativa para mantener la rapidez en el descifrado es utilizar los exponentes reducidos d_p y d_q que surgen de aplicar el Teorema Chino de los Restos, como se explica en el apartado 2.1.1. Así, d puede ser del tamaño de n mientras que d_p y d_q son pequeños.

4.2.1. Ejemplo

Por último, es interesante incluir un ejemplo sencillo del ataque de Wiener. Siguiendo la notación del Teorema 4.10, se eligen $p = 1861$, $q = 2089$, luego $n = pq = 3887629$. Se tiene que $\varphi(n) = (p-1)(q-1) = 3883680$. Las claves de cifrado y descifrado serán $e = 2969873$ y $d = 17$, que son inversos módulo $\varphi(n)$.

El atacante únicamente conoce (n, e). Sin embargo, para hacer el ejemplo más ilustrativo, se va a comprobar que estos números cumplen la cota. Se da que $G = 12$, $K = 156$, luego $g = 1$ y $k = 13$. Por tanto, la cota es $d < 38$, lo que garantiza que el ataque de Wiener funciona. El siguiente paso es calcular las convergentes de e/n. Se muestran las seis primeras (suficientes en este ejemplo) y las dos últimas:

f_0	f_1	f_2	f_3	f_4	f_5	$\cdots$	f_{15}	$f = f_{16}$
$\frac{0}{1}$	$\frac{1}{1}$	$\frac{3}{4}$	$\frac{13}{17}$	$\frac{55}{72}$	$\frac{233}{305}$	$\cdots$	$\frac{1149339}{1504510}$	$\frac{2969873}{3887629}$

La convergente correcta se corresponderá con k/dg. Para $i = 0, i = 1$ es claro que la suposición no es válida (d no puede ser 1). Para $i = 2$, $\gamma_{2,0} = \lfloor 4e/3 \rfloor = 3959830$, luego:

$$\left(\frac{p+q}{2}\right)^2 - n = \left(\frac{n+1-\gamma_{2,0}}{2}\right)^2 - n = 1303210000$$

el cual no es un cuadrado perfecto. Para $i = 3$, $\gamma_{3,0} = \lfloor 17e/13 \rfloor = 3883680$, y se tiene que

$$\left(\frac{p+q}{2}\right)^2 - n = \left(\frac{n+1-\gamma_{3,0}}{2}\right)^2 - n = 12996 = 114^2$$

Finalmente, $g/k = e/f_3 - \varphi(n) = 1/13$, de lo cual se concluye que $d = 17$.

4.3. Variantes del ataque de Wiener

Desde la publicación del artículo de Wiener en 1990, diversos autores han profundizado en el criptoanálisis del caso en el que la clave de descifrado es pequeña, mejorando la cota de Wiener mediante ataques diversos. Probablemente, el mejor de todos ellos hasta la fecha es una generalización por Boneh y Durfee [3], que eleva la cota hasta $d < n^{0,292}$, y que se verá en el capítulo 5 ya que utiliza retículos. Esta sección se centra en aquellas variantes del ataque de Wiener que se fundamentan en la aproximación por fracciones continuas. Para comenzar, el propio Wiener propone una serie de mejoras a su algoritmo, que son las siguientes:

- Permitir que el algoritmo pueda iterar hasta algo más allá de la cota. Aunque la probabilidad de obtener una solución de esta forma es pequeña, podría darse el caso de que funcione.
- Buscar una mejor aproximación de $k/\tilde{dg}$ en lugar de e/n. Una estimación más cercana a $e/\varphi(n)$ es tomar $e/(\sqrt{n}-1)^2$ (utilizada en la variante de De Weger, que se muestra en el apartado 4.3.2).
- Intentar encontrar g o factores de g para acelerar el algoritmo, e incluso, para mejorar la cota. Sin embargo, no es sencillo encontrar factores de g (de hecho, Wiener propone factorizar $n-1$ para ello).

Los enfoques propuestos son bastante sencillos y no aportan una novedad real por sí solos. De hecho, un resultado de Steinfeld [44] prueba que si toda clave de descifrado supera la cota, la primera de estas estrategias tiene una probabilidad de éxito despreciable a efectos prácticos.

4.3.1. Variante de Verheul-Van Tilborg y Dujella

La primera mejora propuesta por Wiener falla porque no hay una única convergente de e/n que contenga la información suficiente como para revelar d por sí misma. Sin embargo, aprovechando la información parcial que revelan varias convergentes, se puede obtener una aproximación suficientemente buena como para realizar *a posteriori* una búsqueda entre posibles candidatos para $k/\tilde{dg}$. Verheul y Van Tilborg [46] probaron el siguiente resultado.

Proposición 4.13. *Sea (n,e) un par de clave pública RSA tal que $e < n$. Si el ataque de Wiener fracasa, es posible determinar una clave de descifrado $\tilde{d}$ a partir de la expansión en fracciones continuas de e/n realizando una búsqueda exhaustiva sobre aproximadamente 2^{2r+8} valores, siendo $r = \log_2(\tilde{d}/n^{1/4})$.*

La idea fundamental de la demostración es que dadas las convergentes c_i/d_i de $e/n = \langle a_0, \ldots, a_m\rangle$, existen enteros positivos u, v tales que

$$\frac{k}{\tilde{dg}} = \frac{uc_{j'+1} + vc_{j'}}{ud_{j'+1} + vd_{j'}},$$

donde j' es el mayor entero que satisface $\frac{c_{j'}}{d_{j'}} - \frac{e}{n} > \frac{2{,}122e}{n^{3/2}}$. El ataque consiste en probar múltiples pares u, v, comprobando en cada paso si la fracción es la buscada de la misma forma que en el ataque de Wiener. Posteriormente, Dujella introdujo (separadamente) dos mejoras en el método. La primera, [14] es un resultado que permite acelerar el algoritmo anterior considerando las convergentes para $j \in \{j', j'+1, j'+2\}$ y reduciendo el número de candidatos a $0 < u, v < 4D$, donde $\tilde{d} = Dn^{1/4}$. De esta forma se consigue una complejidad $O(D^2)$ (el número de candidatos a comprobar), y el algoritmo tiene éxito con una probabilidad del 98 %. La segunda [15] es una optimización computacional del algoritmo, incluyendo la siguiente forma alternativa de comprobar si un par u, v candidato es correcto.

Se quiere comprobar si el denominador de la fracción candidata es realmente $\tilde{dg}$. Se asume que g es conocido; en otro caso se puede repetir el método con distintas suposiciones para g. De hecho, en las condiciones del Corolario 4.11 con la clave privada d superando ligeramente la cota, la clave que se encuentre será d y se puede tomar $g = 1$. Es sencillo ver que si $\tilde{dg} = ud_{j+1} + vd_j$, entonces para cualquier mensaje M,

$$M^{e(ud_{j+1}+vd_j)} \equiv M^{e\tilde{dg}} \equiv M^g \text{ mód } n.$$

En particular, la identidad se cumple para $M = 2$. Definiendo $a = 2^{ed_{j+1}} \text{ mód } n$, $b = \left(2^{ed_j}\right)^{-1} \text{ mód } n$, la condición anterior se convierte en

$$a^u \equiv 2^g \cdot b^v \text{ mód } n.$$

Como a y b son fijos para cada j, se puede evaluar si se satisface la congruencia anterior calculando todos los a^u, colocándolos en una lista ordenada o en una tabla hash, y comprobando para cada v si $2^g b^v$ está en la estructura anterior. La optimización reside en que calcular y almacenar previamente los a^u en una estructura de datos apropiada es más eficiente que comprobar los denominadores individualmente. El proceso tiene una complejidad $O(D \log D)$, mucho mejor que la anterior a pesar de que sigue siendo exponencial en el tamaño de D.

Dujella muestra que, por restricciones de memoria, es factible realizar búsquedas hasta aproximadamente $D = 2^{30}$. Esto lleva a ampliar la cota de Wiener hasta $\tilde{d} < 2^{30} n^{1/4}$, que a pesar de que asintóticamente es igual que la original, a efectos prácticos supone un aumento notable. Por ejemplo, si se considera una implementación de RSA donde n tiene un tamaño de 1024 bits, un cálculo rápido lleva a que $\tilde{d} < n^{0,28}$.

Los ataques de Verheul-Van Tilborg y Dujella, aunque eficaces, no dejan de ser una extensión del ataque de Wiener basada en una búsqueda con complejidad exponencial. Es interesante preguntarse si existe un algoritmo que a partir de la expansión en fracciones continuas de e/d pueda mejorar la cota de Wiener en tiempo al menos subexponencial. La respuesta es negativa, nuevamente gracias a un resultado de Steinfeld [44].

4.3.2. Variante de De Weger

Cuando los primos p y q son muy cercanos, se produce una vulnerabilidad que puede ser explotada mediante el algoritmo de factorización de Fermat, como se ha visto en la Proposición 3.2. Concretamente, si $\Delta = |p - q| = n^\beta$, el algoritmo es eficaz cuando $\beta < 1/4$. De Weger [11] propone una modificación al ataque de Wiener, que consiste principalmente en mejorar la aproximación de $\varphi(n)$ y eleva la cota cuando $\beta < 1/2$.

Proposición 4.14. *Sea (n, e) un par de clave pública RSA donde $d = n^\delta$ es la clave privada y $\Delta = |p - q| = n^\beta$. Si $e < \varphi(n)$ y $\delta < \frac{3}{4} - \beta$, es posible recuperar d en tiempo polinomial en el tamaño de n.*

Demostración. La demostración sigue un esquema muy similar a la del Corolario 4.11. En primer lugar, es sabido que $\frac{e}{\varphi(n)} - \frac{k}{d} = \frac{1}{\varphi(n)d}$. Como $\varphi(n) = n + 1 - p - q$ se considera la aproximación $\varphi(n) \approx n + 1 - 2n^{1/2}$ y se tiene, aplicando el Lema 3.1 relativo al Algoritmo de Fermat:

$$(n + 1 - 2n^{1/2}) - \varphi(n) = p + q - 2n^{1/2} < \frac{\Delta^2}{4n^{1/2}}.$$

Sustituyendo $\varphi(n)$ por su aproximación y aplicando la desigualdad anterior, además de la desigualdad triangular y $e < \varphi(n)$, se tiene:

$$\left|\frac{e}{n + 1 - 2n^{1/2}} - \frac{k}{d}\right| < e\left|\frac{1}{n + 1 - 2n^{1/2}} - \frac{1}{\varphi(n)}\right| + \left|\frac{e}{\varphi(n)} - \frac{k}{d}\right| <$$

$$< e \left| \frac{(n+1-2n^{1/2}) - \varphi(n)}{n+1-2n^{1/2}} \right| - \frac{1}{\varphi(n)d} < \frac{1}{\varphi(n)} \left(\frac{\Delta^2}{4n^{1/2}} + \frac{1}{d} \right).$$

Se puede asumir con seguridad que $\varphi(n) > \frac{3}{4}n$ y que $n > 8d$. Sustituyendo $\Delta = n^\beta$, $d = n^\delta$ en la ecuación anterior,

$$\left| \frac{e}{n+1-2n^{1/2}} - \frac{k}{d} \right| < \frac{4}{3n} \left(\frac{n^{2\beta}}{4n^{1/2}} + \frac{1}{n^\delta} \right) < \frac{1}{3} n^{2\beta-3/2} + \frac{1}{6} n^{-2\delta}.$$

Finalmente, como $\delta < \frac{3}{4} - \beta$, o equivalentemente, $-2\delta > 2\beta - 3/2$,

$$\left| \frac{e}{n+1-2n^{1/2}} - \frac{k}{d} \right| < \frac{1}{2} n^{-2\delta} = \frac{1}{2d^2},$$

lo que garantiza que k/d es una de las convergentes de la expansión en fracciones continuas de $e/(n+1-2n^{1/2})$. Por tanto, es posible obtener d de forma similar al procedimiento del ataque de Wiener. □

De Weger no solo incluye en su artículo una mejora al ataque de Wiener para $\Delta < n^{1/2}$, sino que además presenta una mejora del ataque de Boneh y Durfee ($d < n^{0,292}$, ver sección 5.5) también cuando la diferencia entre p y q es pequeña.

4.3.3. Variante de Bunder y Tonien

En 2017, Bunder y Tonien presentaron una variante del ataque de Wiener que mejora el original cuando la clave de cifrado e también es pequeña. Su resultado principal es el siguiente [6], cuya prueba no es compleja pero se omite por su similitud con la del resultado anterior.

Proposición 4.15. *Sea $n = pq$ con $q < p < 2q$ y $n > 20\,000\,000$, y sean e, d las claves de cifrado y descifrado con $e < \varphi(n)$. Entonces, si $d < n^{1/4}\sqrt{8n/e}$, es posible recuperar d en tiempo polinomial en el tamaño de n a partir de n, e.*

El algoritmo se basa en una aproximación mediante fracciones continuas siguiendo el mismo esquema que el ataque de De Weger. Se utiliza la siguiente aproximación de $\varphi(n)$:

$$\varphi(n) \approx n - \left(1 + \frac{3}{2\sqrt{2}}\right) n^{1/2} + 1$$

Una forma alternativa de expresar la cota de Bunder y Tonien es $d^2 e < 8n^{1/2}$. Para terminar, se muestra un esquema del alcance de los ataques mencionados en el capítulo.

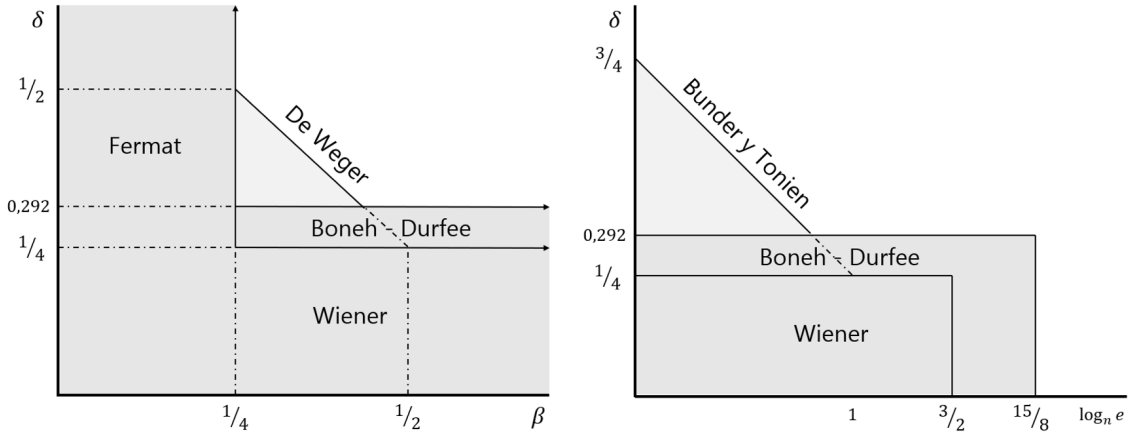


Figura 4.1: Representación aproximada del alcance de los ataques de Wiener, Fermat, Boneh-Durfee, Bunder y Tonien, De Weger. El área sombreada representa la zona donde RSA es vulnerable. La notación utilizada es $\delta = \log_n d$, $\beta = \log_n |p - q|$. No se tiene en cuenta la variación de Boneh-Durfee y Wiener con respecto a $\log_n e$.

Capítulo 5

Ataques basados en Retículos

Un retículo es un conjunto formado por combinaciones enteras de vectores linealmente independientes de $\mathbb{R}^n$. La geometría discreta que define en el espacio euclídeo está completamente determinada por los vectores que lo generan. Además de ser de gran importancia en teoría de números, los retículos se encuentran en la naturaleza con frecuencia. Por ejemplo, describen las estructuras cristalinas de muchos materiales.

A partir de 1982, cuando Lenstra, Lenstra y Lóvasz presentaron el algoritmo LLL, comenzaron a surgir aplicaciones de los retículos en criptografía, especialmente de clave pública. Posteriormente, Coppersmith publicó en 1996 un algoritmo para encontrar soluciones pequeñas de ecuaciones polinomiales modulares basado en LLL, que ha dado lugar a múltiples ataques al RSA. Hoy en día, continúa habiendo una intensa actividad investigadora sobre las ideas de Coppersmith, tanto aplicado al criptoanálisis de RSA como a otros ámbitos.

5.1. Retículos

Definición 5.1. *Sea $\mathcal{B} = \{b_1, \ldots, b_r\}$ una base de un subespacio vectorial de $\mathbb{R}^m$. El retículo L generado por $\mathcal{B}$ es el conjunto formado por las combinaciones lineales enteras de los vectores de $\mathcal{B}$. Es decir,*

$$L = \left\{ \sum_{i=1}^{r} a_i b_i : a_i \in \mathbb{Z}, b_i \in \mathcal{B} \right\}$$

El número de vectores r de la base se conoce como dimensión o tamaño de L. Todo retículo es isomorfo a $\mathbb{Z}^r$ y admite distintas bases generadoras. En adelante, se hará referencia únicamente a retículos de dimensión máxima, es decir, en los que $r = m$. Como los vectores de $\mathcal{B}$ son linealmente independientes, se puede aplicar sobre ellos el algoritmo de ortogonalización de Gram-Schmidt.

Proposición 5.2. *Sean $b_1, \ldots, b_m \in \mathbb{R}^m$ vectores linealmente independientes, y sea $\langle \cdot, \cdot \rangle$ el producto escalar usual de $\mathbb{R}^m$. Los vectores $\tilde{b_i}$ definidos inductivamente a partir de*

$$\tilde{b_i} = b_i - \sum_{j=1}^{i-1} \mu_{i,j} \tilde{b_j},$$

$$\mu_{i,j} = \langle b_i, \tilde{b_j} \rangle / \langle \tilde{b_j}, \tilde{b_j} \rangle,$$

forman una base ortogonal de vectores de $\mathbb{R}^m$, denominada base de Gram-Schmidt.

Con frecuencia en el capítulo se construirán retículos a partir de una base de polinomios. En este caso, los vectores que generan el retículo se corresponden con los vectores formados por los coeficientes de los polinomios de la base. A continuación se introduce la noción de determinante de un retículo.

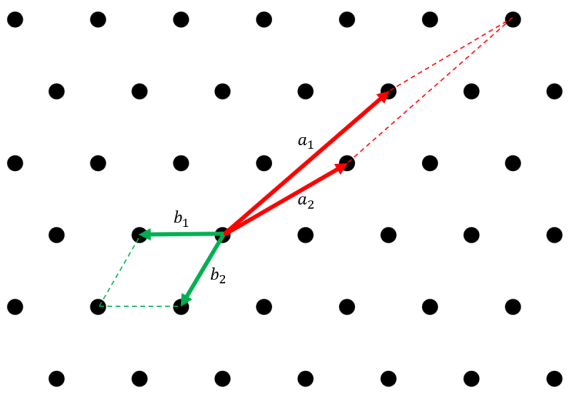


Figura 5.1: Ejemplo de un retículo de dimensión 2 con dos bases generadoras distintas.

Definición 5.3. *Sea L el retículo generado por la base $\mathcal{B}$, y sea B la matriz formada por los vectores de $\mathcal{B}$. El determinante de L, denotado por* $\det(L)$*, es el determinante de la matriz de los vectores de la base,* $\det(L) = |\det(B)|$.

El determinante es un invariante del retículo, el cual coincide con el volumen del paralelepípedo formado por los vectores de cualquier base generadora del mismo. Por tanto, no depende de la base elegida [17]. Debido a esto, a menudo es conveniente utilizar la caracterización siguiente del determinante.

Proposición 5.4. *Sea L el retículo generado por $\mathcal{B}$ y sean $\tilde{b_1}, \ldots, \tilde{b_m}$ los vectores resultantes de aplicar el algoritmo de Gram-Schmidt a $\mathcal{B}$. Entonces,* $\det(L) = \prod_{i=1}^{m} \|\tilde{b_i}\|$.

Demostración. Aplicar el algoritmo de Gram-Schmidt equivale a efectuar una factorización QR de la matriz B. De esta forma, se obtiene $B = QR$, donde Q es una matriz ortogonal y R es triangular superior. Es sencillo ver que la matriz R tiene la forma siguiente:

$$R = \begin{bmatrix} \|\tilde{b_1}\| & \star & \cdots & \star \\ 0 & \|\tilde{b_2}\| & \cdots & \star \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \|\tilde{b_m}\| \end{bmatrix}$$

Como Q es ortogonal, $|\det(Q)| = 1$, luego $\det(L) = |\det(B)| = |\det(Q)||\det(R)| = \det(R) = \prod_{i=1}^{m} \|\tilde{b_i}\|$. □

Dado un retículo, un problema habitual es determinar su vector más corto, entendiendo la longitud como la norma ℓ_2 usual. Este problema es conocido como SVP por sus siglas en inglés, *Shortest Vector Problem*. En la actualidad, el SVP es considerado intratable en el caso general, y de hecho se sospecha que es un problema *NP*-duro [17]. El resultado siguiente, conocido como cota de Minkowski, ofrece una cota superior para la longitud del vector más corto.

Teorema 5.5. *Para todo retículo L de dimensión máxima m, existe un vector $x \in L \setminus \{\mathbf{0}\}$ tal que* $\|x\| \leq \sqrt{m}\det(L)^{1/m}$.

La prueba del Teorema no es constructiva. Afortunadamente, para algunas aplicaciones de los retículos basta con encontrar uno o varios vectores con una longitud próxima a la del vector más corto. Una manera de hacerlo es transformar la base original del retículo en una equivalente cuyos primeros vectores sean suficientemente cortos, para lo cual se utilizan algoritmos de reducción de bases.

5.2. Reducción de bases de retículos. Bases LLL-reducidas

A partir de una base $\mathcal{B}$ de un retículo L, un algoritmo de reducción de bases devuelve una base $\mathcal{B}'$ tal que la longitud de sus vectores está acotada. Idealmente, uno de estos vectores debería ser el más corto del retículo, lo cual no es fácil de conseguir. Para el caso de retículos de dimensión 2, Gauss propuso un sencillo algoritmo [17] que permite reducir una base de forma óptima, es decir, encontrar los dos vectores más cortos de L que forman una base. Su funcionamiento es el siguiente.

1. Se parte de dos vectores independientes $b_1, b_2 \in \mathbb{R}^2$, con $\|b_1\| \leq \|b_2\|$. Si $\|b_1\| > \|b_2\|$ se intercambian b_1 y b_2.
2. Se calcula $u = \langle b_1, b_2\rangle / \langle b_1, b_1\rangle$ y se redondea a los enteros (si $u = \pm\frac{1}{2}$, se redondea a 0).
3. Si $u \neq 0$, se reemplaza $b_2 = b_2 - ub_1$, se intercambian b_1 y b_2 y se vuelve al paso 1. Si $u = 0$, se termina el algoritmo.

Por ejemplo, si se parte de los vectores $b_1 = (5, 4)$ y $b_2 = (7, 5)$ de $\mathbb{R}^2$, se procede:

- $u = (5, 4)(7, 5)/(5, 4)(5, 4) = 55/41$, luego redondeando, $u = 1$. $b_2 = (7, 5) - u \cdot (5, 4) = (2, 1)$. Se intercambian los vectores: $b_1 = (2, 1)$, $b_2 = (5, 4)$.
- $u = (2, 1)(5, 4)/(2, 1)(2, 1) = 14/5$, luego $u = 3$. $b_2 = (5, 4) - 3 \cdot (2, 1) = (-1, 1)$. Se intercambian los vectores: $b_1 = (-1, 1)$, $b_2 = (2, 1)$.
- $u = (-1, 1)(2, 1)/(-1, 1)(-1, 1) = 1/2$, luego $u = 0$. b_1 y b_2 forman la base reducida buscada.

El algoritmo anterior es muy intuitivo, pero no permite una generalización eficiente para el caso m-dimensional. En 1982, Lenstra, Lenstra y Lóvasz introdujeron el algoritmo LLL de reducción de bases de retículos [25]. Desde entonces, LLL ha demostrado tener muchas otras aplicaciones donde se requieren este tipo de bases. Por ejemplo, se utiliza en teoría de números, programación entera o en algunos sistemas criptográficos [30].

Por otro lado, el resultado principal del artículo en el que se presenta el algoritmo es probar que factorizar un polinomio en $\mathbb{Q}[x]$ se puede realizar en tiempo polinomial en el grado del polinomio y en el tamaño de sus coeficientes, aunque en la práctica existen algoritmos más rápidos en promedio como el algoritmo de Berlekamp-Zassenhaus o el de Van Hoeij (ver por ejemplo [30]). Este resultado es de gran importancia y de hecho será necesario más adelante en el trabajo.

Definición 5.6. *Sea L un retículo, $\mathcal{B} = \{b_1, \ldots, b_m\}$ una base de L y $\{\tilde{b_1}, \ldots, \tilde{b_m}\}$ su base ortogonal de Gram-Schmidt asociada. $\mathcal{B}$ se denomina LLL-reducida si satisface:*

$$\begin{cases} |\mu_{i,j}| \leq 1/2 & \forall\ 1 \leq j < i \leq m \\ \|\tilde{b_i} + \mu_{i,i-1}\tilde{b}_{i-1}\|^2 \geq \frac{3}{4}\|\tilde{b}_{i-1}\|^2 & \forall\ 1 < i \leq m \end{cases}$$

El algoritmo LLL recibe por entrada una base de un retículo L y devuelve una base LLL-reducida del mismo. Para obtener la base, realiza únicamente operaciones de intercambio, suma y resta de múltiplos enteros sobre los vectores iniciales, similares a las del algoritmo de dimensión 2, con lo que el retículo subyacente no varía en ningún momento. Un tratamiento completo del algoritmo puede verse en [25]. Se incluyen los resultados más relevantes de las bases LLL-reducidas para su aplicación al criptoanálisis de RSA.

Teorema 5.7. *Sea $\mathcal{B} = \{b_1, \ldots, b_m\}$ una base LLL-reducida de L y sea $\{\tilde{b_1}, \ldots, \tilde{b_m}\}$ su base ortogonal de Gram-Schmidt asociada, con $\mu_{i,j} = \langle b_i, \tilde{b_j}\rangle / \langle \tilde{b_j}, \tilde{b_j}\rangle$. Se cumple que:*

$$\begin{aligned} &i) \quad \|b_j\|^2 \leq 2^{i-1}\|\tilde{b_i}\|^2, \quad \forall\ 1 \leq j \leq i \leq m, \\ &ii) \quad \det(L) \leq \textstyle\prod_{i=1}^{m}\|b_i\| \leq 2^{m(m-1)/4}\det(L), \\ &iii) \quad \|b_1\| \leq 2^{(m-1)/4}\det(L)^{1/m}. \end{aligned}$$

Demostración. De la definición de base LLL-reducida, se tiene que

$$\|\tilde{b_i}\|^2 + |\mu_{i,i-1}|^2\|\tilde{b}_{i-1}\|^2 \geq \frac{3}{4}\|\tilde{b_i}\|^2$$

por ser $\tilde{b_i}, \tilde{b}_{i-1}$ ortogonales. Reagrupando términos y aplicando que $|\mu_{i,j}| \leq 1/2$, se obtiene que $\|\tilde{b}_{i-1}\|^2 \leq 2\|\tilde{b_i}\|^2$. Inductivamente, la desigualdad puede escribirse como

$$\|\tilde{b_j}\|^2 \leq 2^{i-j}\|\tilde{b_i}\|^2, \quad \forall\ 1 \leq j \leq i \leq n. \tag{5.1}$$

Por otro lado, de la definición de base de Gram-Schmidt (Proposición 5.2), utilizando nuevamente que $\tilde{b_i}, \tilde{b_j}$ son ortogonales,

$$\|b_j\|^2 = \|\tilde{b_j}\|^2 + \sum_{k=1}^{j-1}|\mu_{j,k}|^2\|\tilde{b}_k\|^2. \tag{5.2}$$

Aplicando la desigualdad (5.1) y que $|\mu_{i,j}| \leq \frac{1}{2}$,

$$\|b_j\|^2 \leq \|\tilde{b_j}\|^2 + \sum_{k=1}^{j-1}\frac{1}{4}2^{j-k}\|\tilde{b_j}\|^2 = \left(1 + \frac{1}{4}(2^j - 2)\right)\|\tilde{b_j}\| \leq 2^{j-1}\|\tilde{b_j}\|. \tag{5.3}$$

Finalmente, combinando (5.1) y (5.3), se prueba i):

$$\|b_j\|^2 \leq 2^{i-j}2^{j-1}\|\tilde{b_i}\|^2 \leq 2^{i-1}\|\tilde{b_i}\|^2 \quad \forall\ 1 \leq j \leq i \leq m.$$

En segundo lugar, de (5.2) se deduce también que $\|\tilde{b_i}\|^2 \leq \|b_i\|^2$ para todo $1 \leq i \leq m$. Como $\det(L) = \prod_{i=1}^{m}\|\tilde{b_i}\|$, se obtiene la primera desigualdad de ii). Aplicando ahora i) con $i = j$,

$$\prod_{i=1}^{m}\|b_i\| \leq \prod_{i=1}^{m}2^{(i-1)/2}\|\tilde{b_i}\| = 2^{m(m-1)/4}\det(L),$$

con lo que queda probado ii). Para demostrar iii), se efectúa el producto en $i = 1, \ldots, m$ en la ecuación i), con $j = 1$:

$$\|b_1\|^m \leq \prod_{i=1}^{m}2^{(i-1)/2}\|\tilde{b_i}\|^2 = 2^{m(m-1)/4}\det(L).$$

Elevando la expresión a $1/m$ se obtiene iii). □

También pueden obtenerse cotas para el resto de vectores. Boneh [3] prueba el resultado siguiente relativo al segundo vector de la base.

Proposición 5.8. *Sea $b_1, \ldots, b_m$ una base LLL-reducida. Entonces, $\|b_2\| \leq 2^{m/2}\det(L)^{1/(m-1)}$.*

Por último, se muestra la complejidad computacional de calcular una base LLL-reducida [28], correspondiente al tiempo de ejecución del algoritmo L^2, una variante de LLL.

Proposición 5.9. *Dada una base $\mathcal{B} = \{b_1, \ldots, b_m\}$ de un retículo L de dimensión m, existe un algoritmo que devuelve una base LLL-reducida de L en un tiempo $O(m^5(m + \log s)\log s)$, donde $s = \max_i\|b_i\|$.*

5.3. Algoritmo de Coppersmith

Aunque la factorización de polinomios en $\mathbb{Z}[x]$ es un problema computacionalmente abordable, existen dos problemas relacionados que pueden ser difíciles: encontrar raíces módulo n de polinomios univariados cuando la factorización de n no es conocida, y encontrar raíces de polinomios en varias variables. En 1996, Coppersmith [7] propuso un método basado en reducción de bases de retículos que es capaz de resolver ambos problemas cuando las raíces buscadas están suficientemente acotadas.

El objetivo de esta sección es mostrar el funcionamiento del algoritmo en el caso univariado dando una prueba de la corrección del mismo, mientras que el caso multivariado se introducirá al final del capítulo con menor detalle. Se sigue principalmente el desarrollo de May [28].

El algoritmo de Coppersmith ofrece una solución al problema siguiente: *dado un entero n con un divisor $b \geq n^\beta$ (no necesariamente conocido), una cierta cota X, y un polinomio $f(x) \in \mathbb{Z}[x]$, determinar todas las soluciones $|x_0| < X$ que satisfacen $f(x_0) \equiv 0 \bmod b$.*

La idea fundamental del método es buscar un polinomio $g(x) \in \mathbb{Z}[x]$ que tenga las mismas raíces que $f(x)$ sobre los enteros, es decir,

$$f(x) \equiv 0 \bmod b \implies g(x_0) = 0, \quad \forall\, |x_0| \leq X.$$

Encontrado g, se determinan sus raíces mediante un algoritmo de factorización en $\mathbb{Z}[x]$. Para construirlo, se siguen los siguientes pasos:

1. Fijados dos enteros m, γ, se construye una base de polinomios $f_1(x), \ldots, f_\gamma(x)$ que comparten la raíz x_0 módulo b^m. Esto se puede hacer a partir de potencias y múltiplos de $f(x)$.

2. Se construye un retículo de tamaño γ a partir de los coeficientes de los polinomios de la base.

3. Se obtiene una base LLL-reducida del retículo. Su primer vector será una combinación lineal $g(x) = \sum_{i=1}^{\gamma} a_i f_i(x)$, con $a_i \in \mathbb{Z}$, que satisface la condición $|g(x_0)| < b^m$.

4. Como b^m, por construcción, divide a cada $f_i(x_0)$, divide también a $g(x_0)$. Además, como $|g(x_0)| < b^m$, necesariamente $g(x_0) = 0$ sobre los enteros, y basta con factorizar $g(x)$ para encontrar x_0.

Para garantizar que el polinomio $g(x)$ satisface la condición del paso 3 anterior, será de utilidad el siguiente resultado de Howgrave-Graham.

Lema 5.10. *Sea $g(x) = \sum_i a_i x^i \in \mathbb{Z}[x]$ un polinomio formado por a lo sumo d monomios, y sean b, m, X enteros positivos. Si $\|g(xX)\| < b^m/\sqrt{d}$ donde $\|g(x)\|^2 = \sum_i |a_i|^2$, dado un $x_0 \in \mathbb{Z}$ que cumpla*

$$g(x_0) \equiv 0 \bmod b^m, \quad |x_0| < X,$$

entonces x_0 es una raíz de $g(x)$ sobre los enteros.

Demostración. Partiendo de la la desigualdad triangular, se tiene,

$$|g(x_0)| = \left| \sum_i a_i x_0^i \right| \leq \sum_i \left| a_i x_0^i \right| \leq \sum_i |a_i| X^i.$$

Aplicando la desigualdad de Cauchy-Schwarz sobre los vectores $(|a_i|X^i)_i$, y $(1)_i$, se obtiene que $\left(\sum_i |a_i| X^i\right)^2 \leq \left(\sum_i (|a_i|X^i)^2\right)(1 + 1 + \overset{(d)}{\cdots} + 1)$. Tomando raíces cuadradas y dividiendo por d,

$$|g(x_0)| \leq \sum_i |a_i| X^i \leq \sqrt{d} \|g(xX)\| < b^m.$$

Como $g(x_0)$ es un múltiplo de b^m, entonces $g(x_0) = 0$ sobre los enteros. □

Teorema 5.11. *Sea n un entero que tiene un divisor $b \geq n^{\beta}$, $0 < \beta \leq 1$, y sea $0 < \epsilon < \frac{1}{7}\beta$. Además, sea $f(x) \in \mathbb{Z}[x]$ un polinomio mónico univariado de grado δ. Entonces, existe un algoritmo que encuentra todas las soluciones x_0 que satisfacen*

$$f(x_0) \equiv 0 \bmod b, \quad |x_0| \leq \frac{1}{2} n^{\frac{\beta^2}{\delta} - \epsilon}.$$

El tiempo empleado está dominado por el requerido para encontrar una base LLL-reducida de dimensión $O(\epsilon^{-1}\delta)$ a partir de una base cuyos elementos son de tamaño $O(\epsilon^{-1} \log n)$, que se puede conseguir en tiempo $O(\epsilon^{-7}\delta^5 \log^2 n)$.

Demostración. En primer lugar, se define $X = \frac{1}{2} n^{\frac{\beta^2}{\delta} - \epsilon}$, y se fija

$$m = \left\lceil \frac{\beta^2}{\delta \epsilon} \right\rceil. \tag{5.4}$$

Se genera el siguiente conjunto $\mathcal{P}$ de polinomios a partir de f en función de un parámetro t por determinar:

$$\mathcal{P} = \left\{ x^j n^{m-i} f^i : 0 \leq i < m, 0 \leq j < \delta \right\} \bigcup \left\{ x^i f^m : 0 \leq i < t \right\}.$$

Nótese que hay exactamente un polinomio de grado k en $\mathcal{P}$ para todo $0 \leq k \leq \delta m + t - 1$, luego $\mathcal{P}$ es una base. Este conjunto puede expresarse de la siguiente manera:

$$\begin{array}{ccccccc}
n^m & xn^m & \cdots & x^j n^m & \cdots & x^{\delta-1} n^m \\
\vdots & \vdots & & \vdots & & \vdots \\
n^{m-i} f^i & x n^{m-i} f^i & \cdots & x^j n^{m-i} f^i & \cdots & x^{\delta-1} n^{m-i} f^i \\
\vdots & \vdots & & \vdots & & \vdots \\
n f^{m-1} & x n f^{m-1} & \cdots & x^j n f^{m-1} & \cdots & x^{\delta-1} n f^{m-1} \\
\\
f^m & x f^m & \cdots & x^i f^m & \cdots & x^{t-1} f^m
\end{array}$$

El siguiente paso es definir un retículo formado a partir de la base $\mathcal{P}$. Dado que hay exactamente $\gamma = \delta m + t$ vectores, si $p_i(x)$ es el polinomio de grado $i - 1$ de $\mathcal{P}$, se define

$$\mathcal{B} = \left\{ v_i \in \mathbb{Z}^{\gamma} : p_i(xX) = \sum_{k=1}^{\gamma} v_{i,k} x^{k-1},\ i = 1, \ldots, \gamma \right\}.$$

Es decir, $\mathcal{B}$ es la base formada por los vectores de enteros correspondientes a los coeficientes de $p_i(xX)$. La matriz B asociada a esta base, que se puede ver en la página siguiente, es una matriz cuadrada triangular inferior de rango γ. Por tanto, $\mathcal{B}$ genera un retículo L de dimensión γ cuyo determinante se puede calcular fácilmente como el producto de los elementos de la diagonal de la matriz B,

$$\det(L) = \det(B) = n^{\frac{1}{2}\delta m(m+1)} X^{\frac{1}{2}\gamma(\gamma-1)}. \tag{5.5}$$

El siguiente paso es determinar el valor del parámetro t, que es equivalente a fijar el tamaño del retículo $\gamma = \delta m + t$. Esto se realiza con el objetivo de que el primer vector de la base LLL-reducida de L satisfaga la condición del Lema 5.10. Se puede realizar una justificación heurística de la elección de t en base a que los términos de la diagonal de B sean menores que $N^{\beta m}$ (ver por ejemplo [28]), pero quedará probado más adelante que la condición

$$\gamma \leq \frac{\delta m}{\beta} \tag{5.6}$$

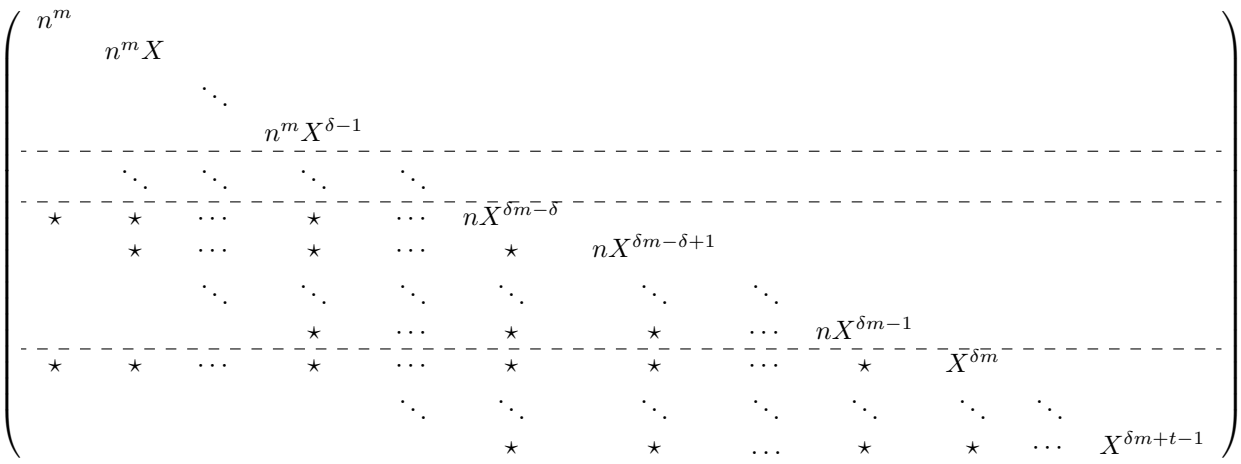


Figura 5.2: Matriz B asociada a la base $\mathcal{B}$.

es suficiente. Se escoge γ como el mayor entero que satisface esta desigualdad. Como, por (5.4), $m < \beta^2/\delta\epsilon + 1$, se obtiene la cota para el tamaño del retículo $\gamma < \beta/\epsilon + \delta/\beta$, que junto con la condición del enunciado $\epsilon \leq \frac{1}{7}\beta$, permite obtener $\gamma = O(\delta\epsilon^{-1})$. A partir de (5.4) y de la elección de γ en (5.6) también se puede obtener una cota inferior para la dimensión,

$$\gamma \geq \frac{\delta m}{\beta} - 1 \geq \frac{\delta}{\beta}\frac{\beta^2}{\delta\epsilon} - 1 \geq \frac{7\epsilon}{\epsilon} - 1 = 6.$$

Se prueba a continuación que el primer vector v de la base LLL-reducida de L es suficientemente corto. Es decir, que v (cuyas componentes serán los coeficientes de un polinomio de grado $\gamma - 1$) satisface la condición del Lema 5.10, $\|v\| \leq b^m/d$. Por el Teorema 5.7 se verifica que $\|v\| \leq 2^{(\gamma-1)/4}\det(L)^{1/\gamma}$. Por tanto, bastará con que

$$2^{\frac{\gamma-1}{4}}\det(L)^{\frac{1}{n}} \leq \frac{b^m}{\sqrt{\gamma}}. \tag{5.7}$$

Utilizando (5.5) y que $b \geq n^\beta$, se obtiene la nueva condición (que implica la anterior):

$$2^{\frac{\gamma-1}{4}} n^{\frac{\delta m(m+1)}{2\gamma}} X^{\frac{\gamma-1}{2}} \leq \gamma^{-\frac{1}{2}} n^{\beta m}.$$

Despejando, se obtiene una cota en el tamaño de X. Además, notando que $\gamma^{-\frac{1}{\gamma-1}} = (2^{\log\gamma})^{\frac{-1}{\gamma-1}} \geq 1/\sqrt{2}$ para $\gamma \geq 6$ se obtiene la nueva condición

$$X \leq \frac{1}{2} n^{\frac{2\beta m}{\gamma-1} - \frac{\delta m(m+1)}{\gamma(\gamma-1)}}.$$

Como $X = \frac{1}{2}n^{\frac{\beta^2}{\delta}-\epsilon}$ basta con probar la desigualdad en los exponentes, es decir

$$\frac{\beta^2}{\delta} - \epsilon \leq \frac{2\beta m}{\gamma - 1} - \frac{\delta m(m+1)}{\gamma(\gamma-1)}.$$

Multiplicando la parte de la derecha por $(\gamma-1)/\gamma < 1$, se obtiene una cota inferior para el término de la izquierda,

$$\frac{\beta^2}{\delta} - \epsilon \leq \frac{2\beta m}{\gamma} - \frac{\delta m(m+1)}{\gamma^2},$$

Aplicando (5.6) y sustituyendo en el lado de la derecha, se obtiene otra nueva cota inferior,

$$\frac{\beta^2}{\delta} - \epsilon \leq 2\frac{\beta^2}{\delta} - \frac{\beta^2}{\delta}(m+1).$$

Finalmente, cancelando términos la expresión se simplifica como

$$m \geq \frac{\beta^2}{\delta\epsilon},$$

lo cual es cierto siempre por la elección establecida en (5.4). Como todas las desigualdades anteriores implican (5.7), queda probado que las propiedades de la base LLL-reducida son suficientes para garantizar la condición del Lema 5.10.

Por último, falta probar que el tiempo de ejecución del algoritmo es el indicado, para lo cual se utilizará el tiempo de ejecución mostrado en la Proposición 5.9. En primer lugar, se ha de acotar superiormente el tamaño s de los coeficientes de B. Los coeficientes de cada polinomio f^i de $\mathcal{P}$ se pueden reducir módulo n^i ya que x_0 tiene que ser una raíz módulo b^i y b es un divisor de n. Por lo tanto, los productos $n^{m-i}f^i$ tienen tamaño a lo sumo $m \log n = O(\epsilon^{-1} \log n)$.

Por otra parte, las potencias de $X = \frac{1}{2}n^{\frac{\beta^2}{\delta}-\epsilon}$ no tienen un exponente mayor que $\gamma = O(\delta\epsilon^{-1})$ en ningún caso, luego su tamaño está acotado por $\gamma\frac{\beta^2}{\delta}\log n = O(\epsilon^{-1}\log n)$.

Se concluye que el tamaño s de los coeficientes de B es $s = O(\epsilon^{-1}\log n)$. De acuerdo con la Proposición 5.9, se puede obtener una base LLL-reducida en un tiempo

$$O\left((\delta\epsilon^{-1})^5(\delta\epsilon^{-1} + \epsilon^{-1}\log n)\epsilon^{-1}\log n\right).$$

Como $\delta \leq \log n$, ya que en caso contrario la cota X sería menor que 1, el tiempo requerido para que el algoritmo determine el vector v es $O(\epsilon^{-7}\delta^5 \log^2 n)$. Los cálculos requeridos para construir la base de polinomios requieren un tiempo polinomial en la dimensión $\gamma = O(\delta\epsilon^{-1})$ y en el tamaño de los coeficientes s, al igual que la factorización del polinomio $g(x)$ de grado $\gamma - 1$. Por ello, se dice que el tiempo está dominado por el necesario para obtener la base LLL-reducida. □

Posteriormente, May [28] presentó una generalización del resultado de Coppersmith.

Teorema 5.12. *Sea n un entero que tiene un divisor $b \geq n^\beta$, $0 < \beta \leq 1$. Sea $f(x) \in \mathbb{Z}[x]$ un polinomio mónico univariado de grado δ. Entonces, existe un algoritmo que encuentra todas las soluciones x_0 que satisfacen*

$$f(x_0) \equiv 0 \text{ mód } b, \quad |x_0| \leq cn^{\frac{\beta^2}{\delta}}$$

en un tiempo dominado por $O(c\delta^5 \log^9 n)$.

Demostración. Aplicando el Teorema 5.11 con $\epsilon = \log^{-1} n$, la cota pasa a ser

$$|x_0| \leq \frac{1}{2}n^{\frac{\beta^2}{\delta}}\left(n^{-\frac{1}{\log n}}\right) = \frac{1}{4}n^{\frac{\beta^2}{\delta}}$$

y el tiempo de ejecución es $O(\delta^5 \log_9 n)$. Para aumentar la cota en un factor $4c$, se puede dividir el intervalo $\left[-cN^{\beta^2/\delta}, cN^{\beta^2/\delta}\right]$ en $4c$ subintervalos, cada uno de longitud $\frac{1}{2}N^{\beta^2/\delta}$ centrados en el punto x_i correspondiente. El resultado se obtiene aplicando el Algoritmo de Coppersmith en las condiciones anteriores al polinomio $f(x - x_i)$ sobre cada intervalo. □

Originalmente, Coppersmith propuso un algoritmo para el caso particular del Teorema 5.12 en el que $\beta = 1$, $c = 1$. Para muchas aplicaciones a RSA, como algunos de los ataques que se muestran en la sección siguiente, basta con esta versión del enunciado.

Teorema 5.13. *Sea n un entero, y sea $f(x) \in \mathbb{Z}[x]$ un polinomio mónico de grado δ. Entonces, existe un algoritmo que encuentra en tiempo polinomial todas las soluciones x_0 que satisfacen*

$$f(x_0) \equiv 0 \text{ mód } n, \quad |x_0| \leq n^{\frac{1}{\delta}}.$$

Cabe tener en cuenta que el Algoritmo de Coppersmith, en todas sus versiones, tiene un tiempo de ejecución polinómico siempre y cuando el número de raíces modulares $|x_0| < X$ del polinomio $f(x)$ sea también polinomial en el tamaño de la entrada. Como el polinomio $g(x)$ generado por el retículo contiene todas estas raíces modulares, y su grado está acotado por la dimensión del mismo, esto no supone un problema ya que el Teorema Fundamental del Álgebra proporciona una cota superior para el número de raíces de g. Cuando los resultados se extienden a varias variables esto sí puede ser un inconveniente, como se verá al final del capítulo.

Por otro lado, si en lugar de LLL-reducir la base del retículo se consiguiera encontrar el vector más corto del mismo, la mejora de la cota sería mínima. Reemplazando la aproximación que proporciona LLL (y en concreto, el valor $2^{(n-1)/4}$) en la prueba del Teorema 5.11 por el valor dado por la cota de Minkowski, X aumenta en un factor cercano a 2.

En el Apéndice B se encuentra un ejemplo detallado de criptoanálisis a RSA mediante un ataque por mensajes estereotipados (apartado 5.4.1) que emplea el algoritmo.

5.4. Ataques en el caso univariado

Es frecuente que distintos problemas relativos a RSA se puedan reducir al problema de encontrar una raíz modular de un polinomio en $\mathbb{Z}[x]$. En este apartado, se muestran cuatro tipos de ataques al RSA que utilizan el algoritmo de Coppersmith para polinomios en una variable. Aunque las vulnerabilidades que se revelan son importantes, a tener en cuenta especialmente si la clave de cifrado e es pequeña, ninguna es tan crítica como los ataques sobre claves de descifrado d pequeñas del capítulo anterior.

5.4.1. Mensajes estereotipados

En el capítulo 2 se observó que si un mensaje M es muy corto, tal que $M < n^{1/e}$, se puede descifrar calculando la raíz e-ésima de M^e. Mediante el algoritmo de Coppersmith, es posible extender este ataque a criptogramas tales que el contenido de su mensaje original es conocido, salvo por una pequeña parte cuya longitud relativa en el mensaje sea menor que $1/e$. Por ejemplo, el ataque se puede aplicar a criptogramas de mensajes como el siguiente:

Hoy, día 6 de Junio de 1944, la clave es: xxxxxxxx. ¡Ánimo, valientes!

Proposición 5.14. *Dado un par de clave pública RSA (n, e) y el criptograma $C = M^e \bmod n$ correspondiente a un mensaje M, si se conocen todos los bits de M salvo una fracción consecutiva de los mismos menor que $1/e$, el mensaje M puede ser recuperado en tiempo polinomial en el tamaño de n y en e.*

Demostración. Como se desconoce únicamente una fracción consecutiva menor que $1/e$ de M, se puede representar este último como $M = 2^{a_2}M_2 + 2^{a_1}x + M_0$, donde todo es conocido salvo x. Nótese que, en función de la posición de la incógnita en el mensaje, M_2 o M_0 podrían ser iguales a 0. Además $|x| < n^{1/e}$. Por tanto, se puede modelar el problema como encontrar una raíz modular del siguiente polinomio,

$$f_n(x) = (2^{a_2}M_2 + 2^{a_1}x + M_0)^e - C \bmod n$$

tal que $|x| < n^{1/e}$. Finalmente, para poder aplicar el Teorema 5.13, el polinomio debe ser mónico, luego tomando

$$f_n(x) = 2^{-a_2 e}(2^{a_2}M_2 + 2^{a_1}x + M_0)^e - C \bmod n,$$

donde todos los coeficientes son enteros porque $2^{a_2 e}$ tiene inverso módulo n, se obtiene el resultado. □

Este ataque se puede evitar fácilmente, por ejemplo, aplicando un padding cuya longitud relativa sea mayor que $1/e$. Por otra parte, como la dimensión del retículo utilizado en el Algoritmo de Coppersmith depende de e, el ataque es muy lento si la clave pública es grande.

5.4.2. Mensajes relacionados

Sean M_1 y M_2 dos mensajes que están relacionados mediante una ecuación lineal, $M_2 = \alpha M_1 + \beta$. Franklin y Reiter observaron que si estos mensajes se cifran utilizando el mismo n y $e = 3$, se pueden descrifrar de forma muy sencilla a partir de sus respectivos criptogramas cuando α, β son conocidos. Si C_1, C_2 son los respectivos criptogramas de M_1, M_2, entonces M_1 se puede obtener simplemente a partir de

$$\frac{\beta(C_2 + 2\alpha^3 C_1 - \beta^3)}{\alpha(C_2 - \alpha^3 C_1 + 2\beta^3)} = \frac{3\alpha^3\beta M_1^3 + 3\alpha^2\beta^2 M_1^2 + 3\alpha\beta^3 M_1}{3\alpha^3\beta M_1^2 + 3\alpha^2\beta^2 M_1 + 3\alpha\beta^3} = M_1 \text{ mód } n.$$

Coppersmith y Patarin [8] generalizan este ataque a mensajes que están relacionados polinomialmente, es decir, $M_2 = f(M_1)$ con $f(x) \in \mathbb{Z}[x]$ de grado δ; y a e arbitrario (aunque pequeño). Si f y los criptogramas $C_1 = M_1^e$ mód n, $C_2 = M_2^e$ mód n son conocidos, es posible plantear el siguiente sistema de ecuaciones, del que M_1 es solución:

$$\begin{aligned} x^e - C_1 &\equiv 0 \text{ mód } n \\ f(x)^e - C_2 &\equiv 0 \text{ mód } n \end{aligned}$$

Ambos polinomios son divisibles por $x - M_1$ en $\mathbb{Z}/n\mathbb{Z}[x]$, luego el polinomio

$$g(x) = mcd(x^e - C_1, f(x)^e - C_2) \in \mathbb{Z}/n\mathbb{Z}[x]$$

también lo será. En la práctica, $g(x) = x - M_1$ salvo en algunos casos especiales (por lo cual no se puede enunciar el resultado como proposición), lo cual revela el mensaje. Este ataque se puede aplicar para cualquier e, aunque está sujeto al coste de calcular el máximo común divisor de dos polinomios en el anillo $\mathbb{Z}/n\mathbb{Z}[x]$. Para efectuar esta operación, se puede recurrir al algoritmo de Euclides para polinomios reduciendo todos los coeficientes intermedios módulo n. En el improbable caso de que algún coeficiente no tenga inverso y el algoritmo falle, se encuentra un factor no trivial de n. El coste del algoritmo de Euclides, como los coeficientes están acotados por n, es polinomial en δe y en el tamaño de n (ver Apéndice A para más detalles).

El argumento anterior se puede aplicar a una situación aún más general. Si se conoce que $M_1, \ldots, M_k$ satisfacen una relación polinomial del tipo $f(M_1, \ldots, M_k) \equiv 0$ mód N, y sus respectivos criptogramas $C_1, \ldots, C_k$ son conocidos, se obtiene el siguiente sistema de $k+1$ ecuaciones:

$$\begin{aligned} f(x_1, \ldots, x_k) &\equiv 0 \text{ mód } N \\ g_1(x_1) = x_1^e - C_1 &\equiv 0 \text{ mód } N \\ &\ldots \\ g_k(x_k) = x_k^e - C_k &\equiv 0 \text{ mód } N \end{aligned}$$

Los autores sugieren dos métodos para solucionar este sistema. El primero es calcular la base de Gröbner del ideal generado por $f, g_1, \ldots, g_k$, cuyo resultado esperado es $x_1 - M_1, \ldots, x_k - M_k$. El segundo es mediante el cálculo de resultantes para la sucesiva eliminación de variables en el sistema. Un mayor detalle se puede encontrar en el artículo de Coppersmith y Paitarin [8].

Padding corto

El ataque anterior parece difícil de llevar a la práctica. Al fin y al cabo, parece improbable que se conozca una relación polinomial entre varios mensajes de antemano. Sin embargo, se puede combinar con el algoritmo de Coppersmith para dar lugar a una vulnerabilidad bastante más probable.

Supóngase que B desea enviarle un mensaje M a A, que es interceptado por un atacante. Como A no responde, B vuelve a cifrar un mensaje M', esta vez aplicando un nuevo padding s tal que $M' = M + s$. El resultado siguiente prueba que si s es excesivamente corto, el atacante es capaz de descifrar M.

Proposición 5.15. *Sea (n, e) un par de clave pública RSA, y sean M un mensaje con criptograma C y $M' = M + s$ con criptograma C'. Si $|s| < n^{1/e^2}$, es posible recuperar s a partir de C, C' en tiempo polinomial en el tamaño de n y en e.*

Demostración. Sean $g(x, y) = x^e - C$, $h(x, y) = (x+y)^e - C'$. Cuando $y = s$, ambos polinomios tienen una raíz en común, $x = M$. Por lo tanto, la resultante de ambos polinomios

$$f(y) = \mathrm{res}_x(g, h) \in {}^{\mathbb{Z}}\!/_{n\mathbb{Z}}[y]$$

también se anula en $y = s$. El coste computacional de calcular la resultante (ver Apéndice A) es polinomial en e. Además, el grado de $f(y)$ es a lo sumo $\deg_y(g) \cdot \deg_y(h) = e^2$, luego como $|s| < n^{1/e^2}$, las condiciones del Teorema 5.13 se verifican y es posible obtener s en las condiciones del enunciado. □

Finalmente, para obtener el mensaje original M a partir de s, basta aplicar el ataque del apartado anterior, ya que $M' = f(M) = M + s$.

La implicación más importante de este resultado es que aplicar un padding corto es poco recomendable, especialmente para casos como $e = 3$, en los que un padding cuya longitud sea menor que $1/9$ veces la del mensaje original es inseguro.

5.4.3. Ataque fuerte de Håstad

El ataque original de Hastad se puede aplicar bajo las condiciones de la Proposición 2.8. Es decir, cuando el mismo mensaje M se envía a distintos usuarios utilizando la misma clave de cifrado e y distintos módulos n_i. El ataque es posteriormente generalizado por el propio Hastad [20] al caso en el que los mensajes están relacionados mediante ciertos polinomios conocidos $f_i \in \mathbb{Z}[x]$, y donde las claves de cifrado no son necesariamente iguales.

Proposición 5.16. *Sean $(n_1, e_1), \ldots, (n_r, e_r)$ pares de clave pública RSA tales que los n_i son primos dos a dos, $n_0 = \min\{n_1, \ldots, n_r\}$ y $N = n_1 \cdots n_r$. Sean $f_i(x) \in {}^{\mathbb{Z}}\!/_{n_i\mathbb{Z}}[x], i = 1, \ldots, r$ polinomios conocidos de grado δ_i. Si $\Delta = \max_i\{e_i\delta_i\}$ y $r \geq \Delta$, a partir de los criptogramas $C_i = f_i(M)^{e_i}$ de un mensaje $M < n_0$ es posible recuperar M en tiempo polinomial en el tamaño de n y en Δ.*

Demostración. Se puede asumir que los f_i son mónicos. De no serlo, se multiplica por el inverso de su coeficiente director en $\left({}^{\mathbb{Z}}\!/_{n_j\mathbb{Z}}\right)^*$; en el caso improbable de que no exista su inverso, se revela un factor primo no trivial de n_j con lo que el criptosistema se rompe por completo. Se construye, mediante el Teorema Chino de los Restos, el polinomio

$$g(x) = \prod_{i=1}^{k} b_i x^{r_i} (f_i(x)^{e_i} - C_i) \in {}^{\mathbb{Z}}\!/_{N\mathbb{Z}}[x],$$

donde $r_i = \max_j\{\delta_j\} - \delta_i$, y los b_i son los coeficientes del Teorema Chino tales que $b_i \bmod n_j = 1$ si $i = j$, y $b_i \bmod n_j = 0$ en otro caso. El polinomio $g(x)$ es mónico y su grado es Δ, además de que satisface

$$g(M) \equiv 0 \bmod N.$$

Como $M < n_0 < N^{1/r} < N^{1/\Delta}$, el resultado se obtiene de aplicar el Teorema 5.13. □

Este resultado difiere del ataque para mensajes relacionados dado que intervienen distintos módulos n_i, luego es necesario un mayor número de mensajes relacionados y no únicamente dos de ellos.

5.4.4. Conocimiento parcial de un factor de n

En el Capítulo 3 se estudiaron distintos algoritmos para factorizar n sin ningún conocimiento adicional sobre sus factores. A continuación, se muestra una aplicación del Algoritmo de Coppersmith en su versión más general, que consiste en factorizar n si se conoce al menos la mitad más significativa de los bits de uno de los primos p (es decir, los $\frac{1}{2}\log_2(p)$ bits de la izquierda del número); o la mitad menos significativa de los mismos. En general, existe una gran variedad de ataques a instancias de RSA donde se conocen algunos bits de d, p o q. Una muestra bastante completa se puede encontrar por ejemplo en el Capítulo 6 de Hinek [21], donde también aparece el resultado que se detalla a continuación.

Proposición 5.17. *Sea $n = pq$ tal que p, q son primos balanceados, es decir, $r^{-1}\sqrt{n} < p, q < r\sqrt{n}$ para algún número real $r > 1$ de tamaño despreciable respecto al de n. Si se conoce o la mitad más significativa o la menos significativa de los bits de p, es posible factorizar n en tiempo polinomial en su tamaño.*

Demostración. En primer lugar, se prueba el caso para la mitad más significativa de bits. Sea p_0 una aproximación de p con estas características. Es decir, omitiendo constantes (como r es pequeño esto no altera la cota en más de unos pocos bits), $|p - p_0| \leq n^{1/4}$. Además, $p - p_0$ es una raíz del polinomio

$$f(x) = p_0 + x \bmod p.$$

Aplicando el Teorema 5.12 con $\beta = \frac{1}{2}$ y $\delta = 1$, se pueden determinar todas las raíces de f menores que $n^{\beta^2/\delta} = n^{1/4}$ en tiempo polinomial en el tamaño de n, y por tanto hallar p.

En segundo lugar, conocer la mitad menos significativa de los bits de p es equivalente a conocer s, t tales que $p = p_0 s + t$ donde $0 \leq t < s$, $p^{1/2} < s < p$, y la incógnita p_0 cumple que

$$|p_0| = \left|\frac{p - t}{s}\right| < |p^{1/2}| < \sqrt{r} n^{1/4}.$$

Cabe notar que si s es una potencia de 2, entonces la representación binaria de t se corresponde exactamente con los bits menos significativos de p. Sea ahora $S = s^{-1} \bmod n$ (nótese que si s no tiene inverso, entonces $s \mid n$ y se puede encontrar un factor no trivial de n). El polinomio mónico $f(x) = x + St$ cumple que

$$sf(p_0) \equiv p_0 s + tSs \equiv p \equiv 0 \bmod p,$$

y como no existen divisores de cero módulo p y $s \not\equiv 0 \bmod p$, entonces $f(p_0) \equiv 0 \bmod p$. Por lo tanto, se puede aplicar el Teorema 5.12 en las mismas condiciones que en el caso anterior, determinar p_0 y finalmente hallar p. □

El resultado no es únicamente válido en el contexto de RSA. Al no estar implicadas en ningún momento las claves pública y privada, puede verse como un algoritmo de factorización de enteros en general.

5.5. Algoritmo de Coppersmith para polinomios en dos variables

El algoritmo de Coppersmith presentado en la sección 5.3 admite una extensión natural a polinomios de varias variables, proponiendo una solución al problema siguiente: *dado un polinomio* $f(x_1, \ldots, x_k) \in \mathbb{Z}[x_1, \ldots, x_k]$*, encontrar todas las raíces* $(x_{0,1}, \ldots, x_{0,k}) \in \mathbb{Z}^k$ *de* f *tales que* $|x_{0,i}| < X_i$ *para ciertas cotas* X_i*,* $i = 1, \ldots, k$*.* En esta sección se introducen las ideas generales del algoritmo para polinomios en dos variables por su interés para RSA, siguiendo principalmente el trabajo original de Coppersmith [7], May [27], y en menor medida Hinek [21].

El algoritmo es muy similar al del caso univariado. Se comienza generando un retículo a partir de una base de polinomios que comparten la raíz modular buscada (x_0, y_0) con el polinomio inicial $f(x, y) \in \mathbb{Z}[x, y]$. Posteriormente, se obtiene una base LLL-reducida cuyo primer vector satisfaga las condiciones del resultado siguiente (la demostración es análoga a la del Lema 5.10).

Lema 5.18. *Sea* $g(x, y) = \sum_{i,j} a_{i,j} x^i y^j \in \mathbb{Z}[x, y]$ *un polinomio formado por a lo sumo* d *monomios, y sean* b, m, X *enteros positivos. Si* $\|g(xX, yY)\| < b^m/\sqrt{d}$ *donde* $\|g(x, y)\|^2 = \sum_{i,j} |a_{i,j}|^2$*, dados* $x_0, y_0 \in \mathbb{Z}$ *que cumplan*

$$g(x_0, y_0) \equiv 0 \text{ mód } b^m, \quad |x_0| < X, |y_0| < Y,$$

entonces $g(x_0, y_0) = 0$ *sobre los enteros.*

A partir de los coeficientes del primer vector se obtiene un polinomio $g(x, y)$ de dos variables que se anula en (x_0, y_0). No es sencillo encontrar dicha raíz, y de hecho, podría haber un número infinito de ellas (considérese por ejemplo el polinomio $g(x, y) = x^2 - y^2$). Una posibilidad es determinar un polinomio $h(x, y)$ que también se anule en (x_0, y_0). Coppersmith propone construirlo a partir de los coeficientes del segundo vector más corto de la base LLL-reducida (ver Proposición 5.8), que también debe verificar las condiciones del Lema 5.10. Para encontrar la raíz común de g y h, se resuelve el sistema $g(x, y) = h(x, y) = 0$ por medio de resultantes, de la forma siguiente:

1. Se calcula la resultante respecto de una de las variables, $r(y) = res_x(g, h) \in \mathbb{Z}[y]$.
2. Se determinan las raíces $y_1, \ldots, y_k$ de $r(y)$ en $\mathbb{Z}$.
3. Se factoriza $mcd(g(x, y_i), h(x, y_i)) \in \mathbb{Z}[x]$, obteniendo las soluciones x_i asociadas a cada y_i (en caso de que existan).
4. Se determina si $f(x_i, y_i) \equiv 0$ mód n para comprobar si la raíz es el par (x_0, y_0) buscado.

Para garantizar que el algoritmo funcione, se requiere asumir dos suposiciones que suelen cumplirse en la práctica. La primera es que los polinomios con la solución pequeña buscada tienen a lo sumo un número polinomial de soluciones pequeñas. La segunda, que los polinomios obtenidos a partir de los dos vectores más cortos de la base LLL-reducida son algebraicamente independientes[1].

La primera suposición es necesaria para que el algoritmo termine en tiempo polinomial, ya que devuelve todas las soluciones pequeñas y no únicamente la buscada. Esta suposición a menudo puede comprobarse en cada aplicación del algoritmo en particular. La segunda es imprescindible para resolver el sistema $g(x, y) = h(x, y) = 0$, ya que si $\deg(mcd(g, h)) \geq 1$, entonces $res_x(g, h) = 0$ (ver Apéndice A) y no se pueden determinar las raíces comunes.

Las aplicaciones de estas ideas a RSA son muchas. Por ejemplo, en ataques con conocimiento parcial de la clave privada o a variantes del RSA (ver [21] o [51] para más detalles). Se presentan dos de las más notables: una reinterpretación del ataque de Wiener (que no requiere del algoritmo al completo), y el ataque de Boneh-Durfee.

[1] Dos polinomios $g, h \in \mathbb{Z}[x]$ son algebraicamente independientes si $mcd(g, h) \in \mathbb{Z}$.

5.5.1. Revisitando el Ataque de Wiener

El ataque de Wiener admite una formulación mediante retículos [27] que ofrece alguna ventaja respecto del original. Las dos más significativas son la obtención directa de d sin necesidad de iterar sobre varios candidatos, y la capacidad de revelar directamente la factorización de n. A continuación se enuncia nuevamente el resultado de Wiener en las condiciones más habituales, desarrollando una prueba alternativa.

COROLARIO 4.11. *Sea $n = pq$ con $q < p < 2q$, y sea $d < \frac{1}{3}n^{1/4}$. Dados (n, e) tales que $ed \equiv 1 \bmod \varphi(n)$ y $e < \varphi(n)$, es posible recuperar d en tiempo polinomial en el tamaño de n.*

Demostración. Se comienza con la ecuación que relaciona las claves pública y privada, $ed \equiv 1 \bmod \varphi(n)$, equivalente a escribir $ed = 1 + k\varphi(n)$ para algún entero $k < d$ (ya que $e < \varphi(n)$). Como $\varphi(n) = n - p - q + 1$, se obtiene la expresión

$$ed + k(p + q - 1) - 1 = kn. \tag{5.8}$$

Tomando el polinomio $f(x, y) = ex + y$, el problema de encontrar d se reduce a encontrar la raíz $(x_0, y_0) = (d, k(p + q - 1) - 1)$ módulo n de f.

En lo sucesivo, se construirá un polinomio $g(x, y)$ tal que $g(x_0, y_0) = 0$ sobre los enteros y se probará que (x_0, y_0) se puede determinar a partir únicamente de g. El procedimiento no requiere la construcción de un segundo polinomio, por lo que no es necesario asumir la suposición de que los polinomios sean algebraicamente independientes.

En primer lugar es necesario acotar x_0 e y_0, para lo cual se definen $X = \frac{1}{3}n^{1/4}$, $Y = \frac{3}{\sqrt{2}}X\sqrt{n}$. Es sencillo ver que $x_0 = d < X$ y que $y_0 = k(p + q - 1) - 1 < 2d\sqrt{pq} < Y$.

Como el polinomio f está formado por dos monomios y la raíz (x_0, y_0) está acotada por (X, Y), las condiciones del Lema 5.18 en este caso particular son las siguientes: $f(x_0, y_0) \equiv 0 \bmod n$, $\|f(xX, yY)\| < \frac{1}{\sqrt{2}}n$. La primera condición se satisface, pero como

$$\|f(xX, yY)\| = \sqrt{e^2X^2 + Y^2} > eX > ed > k\varphi(n),$$

entonces $f(x, y)$ no satisface la segunda condición, luego se debe construir un polinomio $g(x, y)$ que verifique ambas condiciones. El polinomio $f_0(x, y) = nx$ cumple la primera condición ya que es idénticamente cero módulo n, luego toda combinación lineal

$$g(x, y) = af(x, y) + bf_0(x, y), \quad a, b \in \mathbb{Z}$$

verifica que $g(x_0, y_0) \equiv 0 \bmod n$. Para que g verifique la segunda condición, los parámetros a, b se determinan a partir de un vector corto $v = (a, b) \cdot B$ del retículo L generado por los coeficientes de $\{f(xX, yY), f_0(xX, yY)\}$, cuya matriz es

$$B = \begin{pmatrix} nX & 0 \\ eX & Y \end{pmatrix}.$$

Como L es de dimensión 2, se puede aplicar el algoritmo de reducción de bases de Gauss, que garantiza que se obtiene el vector más corto y por tanto que cumple la cota de Minkowski (Teorema 5.5), $\|v\| < \sqrt{2\det(L)}$. Esta condición implica que

$$\|v\| < \sqrt{2\det(L)} = \sqrt{2}\sqrt{nXY} = \frac{2^{1/4}}{3^{1/2}}n < \frac{1}{\sqrt{2}}n,$$

luego el polinomio $g(xX, yY) = (an+be)Xx+bYy$ tiene norma menor que $\frac{1}{\sqrt{2}}n$. En consecuencia, $g(x, y) = (an + be)x + by$ verifica también la segunda condición del Lema y $g(x_0, y_0) = 0$ sobre los enteros.

A continuación se prueba que se puede obtener esta raíz a partir de los coeficientes de f. De hecho, se tiene que $(x_0, y_0) = (|b|, |an + be|)$. En efecto, como $g(x_0, y_0) = (an + be)x_0 + by_0 = 0$, se tiene la igualdad de fracciones

$$\frac{x_0}{y_0} = \frac{-b}{an + be}.$$

En primer lugar, $mcd(d, k) = 1$ por la ecuación $ed - k\varphi(n) = 1$. Además, como $d < n^{1/4} < q$, entonces $mcd(d, n) = 1$, lo cual implica que $mcd(d, kn) = 1$. Por otro lado, $mcd(d, k(p+q-1)-1)$ divide a $mcd(ed, k(p + q - 1) - 1)$ que a su vez divide a kn por (5.8), con lo que $mcd(d, k(p + q - 1) - 1) = 1$ y la fracción de la izquierda es irreducible.

En segundo lugar, como v es el vector más corto de L, necesariamente $mcd(b, a) = 1$ ya que en caso contrario se podría dividir v por este valor, obteniendo un vector más corto. Por tanto, $mcd(b, an + be) = mcd(b, an) = mcd(b, n)$. Por otra parte, la condición $\|v\| < \frac{1}{\sqrt{2}}n$ implica que $bY < \frac{1}{\sqrt{2}}n$. Sustituyendo, $b < n^{1/4} < q$ y $mcd(b, n) = 1$.

Por tanto, ambas fracciones son irreducibles y deben ser iguales salvo el signo. Finalmente, como x_0 es un entero positivo, $|b| = d = x_0$, y también $|an + be| = k(p + q - 1) - 1 = y_0$. □

Se puede recuperar la factorización de n a partir de (x_0, y_0) de forma muy sencilla, continuando con el procedimiento anterior. Por (5.8), $k = (ex_0 + y_0)/n$, y es inmediato calcular $\varphi(n)$ aplicando que $ed + k\varphi(n) = 1$. Como se indica en el apartado 2.1.1, p, q son las soluciones de la ecuación $x^2 - (n - \varphi(n) + 1)x + n = 0$.

La combinación del ataque de Wiener con retículos ha dado lugar a otros resultados interesantes. El más destacado es probablemente el ataque de Blomer y May [1] que, dado un par de clave privada (n, e), logra factorizar n si existen x, y tales que

$$ex + y \equiv 0 \text{ mód } \varphi(n), \qquad x < \frac{1}{3}n^{1/4}, \quad |y| \leq cn^{-3/4}ex.$$

Este resultado se conoce como ataque de Wiener generalizado, ya que la situación del ataque de Wiener en la que d es pequeño y $ed \equiv 1 \text{ mód } \varphi(n)$ es un caso particular del mismo. En el algoritmo, se combina el Teorema 5.17 (factorizar con una aproximación de p o q) junto con aproximaciones por fracciones continuas.

5.5.2. Ataque de Boneh-Durfee

El ataque de Boneh y Durfee [3] es una aplicación del Algoritmo de Coppersmith en dos variables. Este ataque es efectivo cuando $d < n^{0,292}$ en las condiciones más habituales de RSA, es decir, cuando e y n son del mismo tamaño. Si e es más pequeño, la cota anterior puede aumentar. Su justificación requiere asumir las dos suposiciones mencionadas anteriormente, por lo que no se enuncia como teorema. En este apartado se presenta una versión simplificada del ataque que logra la cota $d < n^{0,285}$.

Para comenzar, como 2 divide a $gcd(p - 1, q - 1)$, existe una clave de descifrado $\tilde{d}$ (ver Proposición 2.6) que satisface $e\tilde{d} \equiv 1 \text{ mód } \varphi(n)/2$. Como $\varphi(n) = n - p - q + 1$, se tiene que

$$e\tilde{d} + k\left(\frac{n+1}{2} - \frac{p+q}{2}\right) = 1. \tag{5.9}$$

Tomando $s = (p + q)/2$, $A = (n + 1)/2$, la igualdad implica que $k(A + s) \equiv 1 \text{ mód } e$, y es claro que determinar s y k es equivalente a encontrar $\tilde{d}$ (de hecho, determinar s es suficiente).

En adelante, $\tilde{d} < n^{\delta}$ y $e = n^{\alpha}$. Por la ecuación (5.9), se pueden acotar k y s de la forma siguiente:

$$|k| = \frac{2ed - 1}{\varphi(n)} < \frac{3de}{n} < 3n^{\delta}ee^{-1/\alpha} = 3e^{1+(\delta-1)/\alpha},$$

$$|s| = \frac{1}{2}(p+q) < 2\sqrt{n} = 2e^{1/2\alpha}.$$

Por lo tanto, si $X = 3e^{1+(\delta-1)/\alpha}$ e $Y = 2e^{1/2\alpha}$, el problema en términos del algoritmo de Coppersmith se reduce a, dado el polinomio $f(x,y) = x(A+y) - 1$, encontrar los pares de enteros x_0, y_0 tales que

$$f(x_0, y_0) = x_0(A + y_0) - 1 \equiv 0 \text{ mód } e, \qquad |x_0| < X, \quad |x_0| < Y,$$

entre los que se encuentran $x_0 = k$ e $y_0 = s$. El objetivo, por tanto, es construir un par de polinomios $g(x,y)$, $h(x,y)$ que cumplan las condiciones del Lema 5.18 para un cierto entero m (cuyo rango de valores se discutirá posteriormente), a fin de resolver el sistema mediante resultantes. Para ello, se definen las clases de polinomios

$$g_{i,k}(x,y) = x^i f^k(x,y) e^{m-k}, \qquad h_{j,k}(x,y) = y^j f^k(x,y) e^{m-k},$$

a los que se denominará, respectivamente, x-desplazamientos e y-desplazamientos. Estos polinomios comparten la raíz (x_0, y_0) módulo b^m para $k = 0, \ldots, m$.

Para un parámetro t por determinar, se construye el retículo L formado por los vectores de los coeficientes de los polinomios $g_{i,k}(xX, yY)$ para $i = 0, \ldots, m-k$, y $h_{j,k}(xX, yY)$ para $j = 1, \ldots, t$. Como el monomio de mayor grado de cada polinomio es distinto, los vectores forman una base de dimensión $\gamma = (m+1)(m+2)/2 + t(m+1)$, que genera una matriz triangular B cuyos elementos se muestran en la siguiente tabla (representada para el caso $m = 2$, $t = 1$).

	$g_{0,0}$	$g_{1,0}$	$g_{2,0}$	$g_{0,1}$	$g_{1,1}$	$g_{0,2}$	$h_{1,0}$	$h_{1,1}$	$h_{1,2}$
1	e^2			$\star$		$\star$			
x		Xe^2		$\star$	$\star$	$\star$			
x^2			X^2e^2		$\star$	$\star$			
xy				eXY		$\star$		$\star$	$\star$
x^2y					eX^2Y	$\star$			$\star$
x^2y^2						X^2Y^2			$\star$
y							e^2Y	$\star$	$\star$
xy^2								eXY^2	$\star$
x^2y^3									eX^2Y^3

El determinante de L se puede calcular como el producto de los elementos de la diagonal de B. Se hará en dos pasos, en primer lugar el correspondiente a los x-desplazamientos, $\det_x(L)$, y posteriormente a los y-desplazamientos, $\det_y(L)$. Se tiene:

$$\det_x(L) = e^{m(m+1)(m+2)/3} X^{m(m+1)(m+2)/3} Y^{m(m+1)(m+2)/6} = e^{(m^3/3\alpha)(2\alpha+\delta-\frac{3}{4})+o(m^3)},$$

$$\det_y(L) = e^{tm(m+1)/2} X^{tm(m+1)/2} Y^{t(m+1)(m+t+1)/2} = e^{(tm^2/2\alpha)(2\alpha+\delta-\frac{1}{2})+mt^2/4\alpha+o(tm^2)}.$$

La derivación completa con todos los términos se puede encontrar en el artículo original [3].

A continuación, para garantizar que el primer vector b_1 de la base LLL-reducida sea suficientemente pequeño, debe cumplir $\|b_1\| \leq e^m/\sqrt{\gamma}$, de acuerdo con el Lema 5.18. Combinando este resultado con el Teorema 5.7, el determinante de L se puede acotar por

$$\det(L) \leq \frac{e^{m\gamma}}{\gamma^{\gamma/2} 2^{\gamma(\gamma-1)/4}} < e^{m\gamma}.$$

Paralelamente, puede verse que

$$\gamma = (m+1)(m+2)/2 + t(m+1) = \frac{m^2}{2} + tm + o(m^2). \tag{5.10}$$

Por lo tanto, para satisfacer la desigualdad $\det(L) = \det_x(L)\det_y(L) < e^{m\gamma}$, se debe cumplir la relación entre los exponentes de la clave pública e siguiente

$$\frac{m^3}{3\alpha}\left(2\alpha+\delta-\frac{3}{4}\right)+\frac{tm^2}{2\alpha}\left(2\alpha+\delta-\frac{1}{2}\right)+\frac{mt^2}{4\alpha}+o(m^3+tm^2)<\frac{m^3}{2}+tm^2+o(m^3).$$

Despreciando los términos pequeños y operando, se concluye que

$$m^2(2\alpha+4\delta-3)+tm(6\delta-3)+3t^2<0.$$

Para que el término de la izquierda sea lo más pequeño posible, se elige t de forma óptima. Derivando e igualando a cero, se obtiene que t minimiza la expresión cuando $t = m(\frac{1}{2}-\delta)$, que al insertarlo en la ecuación anterior, da lugar a

$$m^2(2\alpha+4\delta-3)+m^2\left(\frac{1}{2}-\delta\right)(6\delta-3)+3m^2\left(\frac{1}{2}-\delta\right)^2=m^2\left(2\alpha+7\delta-3\delta^2-\frac{15}{4}\right)<0.$$

Resolviendo la ecuación de segundo grado en δ, se obtiene que la desigualdad es cierta cuando

$$\delta<\frac{7}{6}-\frac{1}{3}\sqrt{1+6\alpha}-\epsilon. \tag{5.11}$$

Lo cual, si $\alpha \approx 1$ (es decir, si e es del mismo tamaño que n), implica que $\delta < 0{,}285 - \epsilon$. El término ϵ se introduce para compensar los redondeos necesarios en la práctica y los términos que se han despreciado de menor orden, que tienden a cero cuando m es grande.

Lo anterior prueba que $g(x, y)$, construido a partir del vector más corto de la base LLL-reducida, cumple que $g(x_0, y_0) = 0$ sobre los enteros. Falta ver que el segundo vector b_2 del retículo da lugar a un polinomio $h(x, y)$ que también cumple las condiciones del Lema 5.18. En efecto, combinándolo con la Proposición 5.8, se tiene que

$$\det(L)<\frac{e^{\gamma-1}}{(\gamma 2^\gamma)^{(\gamma-1)/2}}<e^{\gamma-1}.$$

Aplicando ahora (5.10), $\gamma - 1 = \frac{m^2}{2} + tm + o(m^2) - 1$, luego se puede despreciar -1 como un término de orden inferior, garantizando que la desigualdad también se cumple para $h(x, y)$ si m es suficientemente grande.

El parámetro m debe ser elegido al comienzo del algoritmo y juega un papel importante en el mismo. Si m es pequeño, la dimensión del retículo disminuye y el algoritmo tiene un coste computacional menor, pero sus probabilidades de éxito también se reducen ya que ϵ puede hacerse más grande en la cota (5.11). En la práctica es importante optimizar el valor de m para que el algoritmo no fracase o no sea excesivamente costoso.

Experimentalmente, Boneh y Durfee observan que la suposición de que g, h son algebraicamente independientes es válida. Para elevar la cota (5.11) hasta $d < n^{0{,}292}$, los autores realizan cambios en la base de polinomios sobre la que se genera el retículo, sustituyendo algunos de los y-desplazamientos por otros con menor peso en el determinante.

Por otro lado, el ataque de Wiener pierde totalmente su efectividad cuando $e > n^{3/2}$. Puede obtenerse a partir de (5.11) que el ataque de Boneh-Durfee es inocuo si $e > n^{15/8}$, que es una mejora notable respecto de Wiener. En los últimos años han surgido numerosos ataques que complementan al de Boneh-Durfee, pero ninguno mejora la cota $d < n^{0{,}292}$ en el caso general.

Capítulo 6

Conclusiones

Después de cuarenta años sometido a un intenso criptoanálisis, RSA prevalece como uno de los estándares criptográficos más utilizados en la actualidad. Tras haberlo atacado desde múltiples ángulos a lo largo del trabajo, la conclusión más evidente es que RSA continúa siendo confiable si su implementación es adecuada. A pesar de ello, es muy sencillo caer en una vulnerabilidad si no se toman ciertas precauciones. Factores como la elección de las claves, la elección de los primos, el padding o el envío del mismo mensaje a varios usuarios deben ser tenidos muy en cuenta si se desea alcanzar un nivel de seguridad óptimo.

En caso de que los primos sean grandes, balanceados y no excesivamente cercanos, las vulnerabilidades más importantes se producen cuando las claves son pequeñas, especialmente la de descifrado. Frecuentemente se utilizan claves públicas estándares, como por ejemplo $e = 2^{16} + 1 = 65537$, para la que el proceso de cifrado es ágil. Valores como $e = 3$ o $e = 17$, utilizados en el pasado, han quedado en desuso a raíz de las aplicaciones del algoritmo de Coppersmith. Una ventaja añadida de utilizar una clave pública pequeña es la garantía que la clave privada será grande, ya que $e \cdot d$ en ningún caso puede ser menor que $\lambda(n)$. De este modo, se evitan por completo los ataques del capítulo 4 y de la sección 5.5.

En los últimos años, la mayor parte del criptoanálisis de RSA ha girado en torno al algoritmo de Coppersmith en una y varias variables. Este último da lugar a los ataques más importantes que se conocen, como por ejemplo el de Boneh-Durfee. El carácter heurístico del algoritmo invita a la reflexión sobre la importancia que tiene la experimentación en la criptología, así como la posibilidad de encontrar formas de atacar un criptosistema careciendo de resultados probados. Al fin y al cabo, un algoritmo efectivo cuya corrección no está demostrada matemáticamente le será igualmente útil a cualquier atacante.

La mayor parte de ataques presentados en cada capítulo comparten una idea común, como la búsqueda de congruencias modulares de cuadrados, la aproximación por fracciones continuas o la reducción de bases de retículos. De hecho, a menudo se presenta un algoritmo inicial, como el de Wiener, y sus sucesores son refinamientos del mismo. Esto prueba que una gran parte del criptoanálisis, al igual que muchas otras áreas, consiste en subirse a hombros de gigantes y buscar cómo mejorar lo existente. Por ejemplo, optimizando un parámetro o reduciendo la complejidad computacional de los métodos. En muchos casos, una variación aparentemente pequeña supone una diferencia muy grande en el tiempo de ejecución o un cambio significativo en la cota, como sucede con los algoritmos de Dixon y de Criba Cuadrática o con las variantes del ataque de Wiener.

El trabajo deja de lado algunos ataques matemáticos, como los de conocimiento parcial de la clave, pero también otros como la suplantación y los ataques a la implementación (temporización, fallos aleatorios), más experimentales. Estos tienen un carácter muy diverso, más pícaro en cierto sentido, pudiendo tener consecuencias devastadoras. Una extensión de este trabajo podría consistir en trabajar en ellos y en profundizar en el algoritmo de Coppersmith en varias variables.

A un nivel más amplio, se encuentran dos problemas abiertos de gran interés. El primero es probar la equivalencia (o la no equivalencia) entre romper RSA y la factorización de enteros. El segundo, quizás menos probable, es encontrar a través del criptoanálisis de RSA una vulnerabilidad que de lugar a un algoritmo de factorización de enteros en tiempo polinómico. En cuanto a los algoritmos de factorización de enteros en sí, los avances realizados en las últimas dos décadas son escasos salvo por la posible irrupción de la computación cuántica. Cualquier usuario debe ser consciente de que toda información cifrada mediante RSA será vulnerable tan pronto como se pueda implementar el algoritmo de Shor (o un equivalente) en un computador cuántico de potencia suficiente. RSA quedará en desuso en este momento, dando paso a una generación de criptosistemas *post-cuánticos* que buscan funciones de una vía no basadas en el logaritmo discreto o en la factorización.

En general, la temporalidad de la información cifrada es común a todos los criptosistemas, y no únicamente a aquellos amenazados por la computación cuántica. La elección de un criptosistema debe hacerse pensando en cuánto tiempo es deseable que la seguridad prevalezca y nunca dar por hecho que el cifrado será irrompible por mucho tiempo. La confidencialidad de la información, en la mayor parte de los casos, es relativa, y se ha de tener en cuenta no únicamente la mejora en la capacidad de cómputo, sino también la futura aparición de nuevos ataques. Un ejemplo ilustrativo de este fenómeno es la frase que abre la introducción, "*The Magic Words are Squeamish Ossifrage*", el primer texto cifrado por RSA y publicado por sus autores. En 1977, se estimaba en varios billones de años el tiempo necesario para descifrarla, pero la aparición de la Criba Cuadrática hizo que el problema fuese resuelto tan solo 17 años más tarde. El ataque de Wiener es otro indicativo claro de esto último, ya que hasta su aparición en 1990 el uso de claves de descifrado pequeñas se consideraba seguro. Por todo ello, cada vez que se decide estandarizar un criptosistema, ha de acompañarse con un criptoanálisis exhaustivo. En caso contrario, promover su uso sería muy arriesgado.

Mi última reflexión personal consiste en admirar el entramado de resultados, métodos y enfoques que han surgido a raíz de un cifrado tan sencillo como RSA. La cantidad y variedad de los mismos es sorprendente teniendo en cuenta que todo surge de la dificultad de factorizar enteros, o más concretamente de calcular un inverso módulo $\varphi(n)$ sin conocer la factorización de n. El criptoanálisis, además de tener un interés por sí mismo, también proporciona una motivación importante para empujar las fronteras del estado del arte en matemáticas, dando lugar al desarrollo de una gran diversidad de herramientas interesantes y con aplicaciones en muchos ámbitos.

Bibliografía

[1] Blömer, J. y May, A. A generalized Wiener attack on RSA. In *International Workshop on Public Key Cryptography* (2004), Springer, pp. 1–13.

[2] Boneh, D. Twenty years of attacks on the RSA cryptosystem. *Notices-American Mathematical Society 46* (1999), 203–213.

[3] Boneh, D. y Durfee, G. Cryptanalysis of RSA with private key d less than $n^{0,292}$. *IEEE transactions on Information Theory 46*, 4 (2000), 1339–1349.

[4] Boneh, D., Joux, A. y Nguyen, P. Q. Why textbook ElGamal and RSA encryption are insecure. In *International Conference on the Theory and Application of Cryptology and Information Security* (2000), Springer, pp. 30–43.

[5] Bressoud, D. M. *Factorization and primality testing*. Springer Science & Business Media, 2012.

[6] Bunder, M. W. y Tonien, J. A new attack on the RSA cryptosystem based on continued fractions. *Malaysian Journal of Computational Sciences* (2017).

[7] Coppersmith, D. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *Journal of Cryptology 10*, 4 (1997), 233–260.

[8] Coppersmith, D., Franklin, M. K., Patarin, J. y Reiter, M. K. Low-Exponent RSA with Related Messages. In *EUROCRYPT* (1996).

[9] Coron, J. S. y May, A. Deterministic polynomial-time equivalence of computing the rsa secret key and factoring. *Journal of Cryptology 20*, 1 (2007), 39–50.

[10] Crandall, R. y Pomerance, C. B. *Prime numbers: a computational perspective*, vol. 182. Springer Science & Business Media, 2006.

[11] De Weger, B. Cryptanalysis of RSA with small prime difference. *Applicable Algebra in Engineering, Communication and Computing 13*, 1 (2002), 17–28.

[12] Diffie, W. y Hellman, M. New directions in cryptography. *IEEE Trans. Inf. Theor. 22*, 6 (1976).

[13] Dixon, J. D. Asymptotically fast factorization of integers. *Mathematics of Computation 36* (1981), 255–260.

[14] Dujella, A. Continued fractions and RSA with small secret exponent. *Tatra mountains mathematical publications 29* (2004), 101–112.

[15] Dujella, A. A variant of Wiener's attack on RSA. *Computing 85*, 1-2 (2009), 77–83.

[16] Durán, R., et al. *El Criptosistema RSA*, 1ª ed. Ra-Ma Editorial, Madrid, 2003.

[17] Galbraith, S. D. *Mathematics of Public Key Cryptography*. Cambridge University Press, 2012.

[18] Hardy, G. H. y Wright, E. M. *An introduction to the theory of numbers*. Oxford University Press, 1979.

[19] Hastad, J. On using RSA with low exponent in a public key network. In *Conference on the Theory and Application of Cryptographic Techniques* (1985), Springer, pp. 403–408.

[20] Hastad, J. Solving simultaneous modular equations of low degree. *SIAM Journal on Computing 17*, 2 (1988), 336–341.

[21] Hinek, M. J. *Cryptanalysis of RSA and its variants*. Chapman and Hall/CRC, 2009.

[22] Kerckhoffs, A. La cryptographic militaire. *Journal des sciences militaires* (1883), 5–38.

[23] Khinchin, A. I. *Continued fractions*, vol. 525. P. Noordhoff, 1963.

[24] Lenstra, H. W. Factoring integers with elliptic curves. *Annals of Mathematics 126*, 3 (1987), 649–673.

[25] Lenstra, A. K., Lenstra, H. W. y Lovász. L. Factoring polynomials with rational coefficients. *Mathematische Annalen 261*, 4 (1982), 515–534.

[26] Levy, S. The Open Secret. *Wired*. [En línea; consultado el 7-06-2019]. Disponible en `https://www.wired.com/1999/04/crypto/`.

[27] May, A. *New RSA vulnerabilities using lattice reduction methods*. Tesis doctoral. University of Paderborn, 2003.

[28] May, A. Using LLL-reduction for solving RSA and factorization problems. In *The LLL algorithm*. Springer, 2009, pp. 315–348.

[29] Miller, G. L. Riemann's hypothesis and tests for primality. *Journal of computer and system sciences 13*, 3 (1976), 300–317.

[30] Nguyen, P. Q. y Vallée, B. *The LLL algorithm*. Springer, 2010.

[31] Pollard, J. M. Theorems on factorization and primality testing. In *Mathematical Proceedings of the Cambridge Philosophical Society* (1974), vol. 76, Cambridge University Press, pp. 521–528.

[32] Pollard, J. M. A Monte Carlo method for factorization. *BIT Numerical Mathematics 15*, 3 (1975), 331–334.

[33] Pomerance, C. Analysis and comparison of some integer factoring algorithms. *Computational methods in number theory* (1982), 89–139.

[34] Pomerance, C. A tale of two sieves. *Biscuits of Number Theory 85* (2008), 175.

[35] Riesel, H. *Prime numbers and computer methods for factorization*, vol. 126. Springer Science & Business Media, 2012.

[36] Rivest, R. L. y Silverman, R. D. Are Strong Primes Needed for RSA? In *The 1997 RSA Laboratories Seminar Series, Seminars Proceedings* (1997).

[37] Rivest, R. L., Shamir, A. y Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM 21*, 2 (1978), 120–126.

[38] Rosser, B. Explicit bounds for some functions of prime numbers. *American Journal of Mathematics 63*, 1 (1941), 211–232.

[39] Salomaa, A. *Public-key Cryptography*, 2nd ed. Springer-Verlag, 1996.

[40] Schneier, B. *Applied Cryptography*, 2nd ed. Wiley & Sons, 1996.

[41] Shor, P. W. Algorithms for quantum computation: Discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on* (1994), IEEE, pp. 124–134.

[42] Simmons, G. J. y Norris, M. J. Preliminary comments on the MIT public-key cryptosystem. *Cryptologia 1*, 4 (1977), 406–414.

[43] Singh, S. *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots, to Quantum Cryptography*, 1st ed. Doubleday, New York, NY, USA, 1999.

[44] Steinfeld, R., Contini, S., Wang, H. y Pieprzyk, J. Converse results to the Wiener attack on RSA. In *International Workshop on Public Key Cryptography* (2005), Springer, pp. 184–198.

[45] Stinson, D. *Cryptography: Theory and Practice*, 3rd ed. CRC/Chapman and Hall, 2006.

[46] Verheul, E. R. y van Tilborg, H. C. Cryptanalysis of 'less short' RSA secret exponents. *Applicable Algebra in Engineering, Communication and Computing 8*, 5 (1997), 425–435.

[47] Von Zur Gathen, J., and Gerhard, J. *Modern computer algebra*. Cambridge University Press, 2013.

[48] Wiener, M. J. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information theory 36*, 3 (1990), 553–558.

[49] Williams, H. C y Schmid, B. Some remarks concerning the MIT public-key cryptosystem. *BIT Numerical Mathematics 19*, 4 (1979), 525–538.

[50] Williams, H. C. A $p + 1$ method of factoring. *Mathematics of Computation 39*, 159 (1982), 225–234.

[51] Yan, S. Y. *Cryptanalytic attacks on RSA*. Springer Science & Business Media, 2007.

Apéndice A

Algunos conceptos y resultados

Con el fin de facilitar el seguimiento del trabajo, se presentan en este apéndice varias definiciones, aclaraciones y resultados que pueden ser útiles en la lectura del texto. Las pruebas de los teoremas que se omiten pueden encontrarse en cualquier libro general de álgebra, de teoría de números, o de complejidad computacional. Como referencia, se siguen principalmente Hardy y Wright [18] y Von Zur Gathen y Gerhard [47].

A.1. Teoría de números

Para comenzar, se introduce la función φ de Euler, indispensable para el funcionamiento de RSA, junto con la función λ de Carmichael.

Definición A.1. *Dado $n \in \mathbb{N}$, la función $\varphi(n)$, conocida como función de Euler, es el cardinal del conjunto $\{m \in \mathbb{N}, m < n : mcd(m, n) = 1\}$.*

Teorema A.2. *La función φ de Euler satisface las siguientes propiedades.*

1. *$\varphi(p) = p - 1$ si y solo si p es primo.*
2. *Si n, m son coprimos, $\varphi(nm) = \varphi(n)\varphi(m)$.*
3. *Si p, q son primos distintos, $\varphi(pq) = (p-1)(q-1)$.*
4. *El orden del grupo multiplicativo $\left(\mathbb{Z}/n\mathbb{Z}\right)^*$ es $\varphi(n)$.*
5. *(Teorema de Euler-Fermat) Para todo m coprimo con n, $m^{\varphi(n)} \equiv 1 \text{ mód } n$.*

Corolario A.3 (Pequeño Teorema de Fermat)**.** *Sea p un primo. Para todo $a > 0$ coprimo con p, se tiene que $a^{p-1} \equiv 1 \text{ mód } p$.*

Definición A.4. *Dado $n \in \mathbb{N}$, la función $\lambda(n)$, conocida como función de Carmichael, se define como el menor entero t tal que $a^t \equiv 1 \text{ mód } n$ para todo a coprimo con n. Alternativamente, es el orden máximo de todos los subgrupos cíclicos propios de $\left(\mathbb{Z}/n\mathbb{Z}\right)^*$.*

Proposición A.5. *La función λ de Carmichael toma los siguientes valores:*

- *Si p es un primo impar, $\lambda(p^k) = p^{k-1}(p-1)$.*
- *Si $k \leq 2$, $\lambda(2^k) = 2^{k-1}$. Si $k \geq 3$, $\lambda(2^k) = 2^{k-2}$.*
- *$\lambda(p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_r^{\alpha_r}) = mcm(\lambda(p_1^{\alpha_1}), \lambda(p_2^{\alpha_2}), \ldots, \lambda(p_r^{\alpha_r}))$.*

En particular, $\lambda(pq) = mcm(p-1, q-1)$.

Teorema A.6. *El orden de todo elemento* $a \in \left(\mathbb{Z}/n\mathbb{Z}\right)^*$ *divide a* $\lambda(n)$.

Se introduce a continuación el conocido Teorema Chino de los Restos. Bastará para este trabajo presentarlo en su versión en $\mathbb{Z}$.

Teorema A.7. *Dados* $n_1, \cdots, n_r \in \mathbb{N}$ *coprimos dos a dos, es decir,* $mcd(n_i, n_j) = 1\ \forall i \neq j$, *y* $N = \prod_{i=1}^{r} n_i$, *la aplicación siguiente es un isomorfismo de anillos:*

$$\begin{array}{rccc} \Phi: & \mathbb{Z}/N\mathbb{Z} & \longrightarrow & \mathbb{Z}/n_1\mathbb{Z} \times \cdots \times \mathbb{Z}/n_r\mathbb{Z} \\ & x \bmod N & \longmapsto & (x \bmod n_1, \cdots, x \bmod n_r) \end{array}$$

Además, dado un sistema de r *congruencias* $x \equiv a_i \bmod n_i$ *con* $i = 1, \ldots, r$, *su única solución* x *módulo* N *es* $x = \sum_{i=1}^{r} a_i N_i t_i$, *donde* $N_i = n_1 n_2 \cdots n_r / n_i$, *y* t_i *es el coeficiente que que satisface la identidad de Bézout* $t_i N_i + s_i n_i = 1$.

Los dos resultados siguientes estudian el número de soluciones de ecuaciones de congruencias.

Teorema A.8. *Sea* p *un primo impar,* $a, k \in \mathbb{N}$ *y* $1 \leq a < p$. *La ecuación* $x^k \equiv a \bmod p$ *tiene solución si y solo si* $a^{(p-1)/d} \equiv 1 \bmod p$, *donde* $d = mcd(k, p-1)$. *Además, esta ecuación tiene exactamente* d *soluciones en* $\left(\mathbb{Z}/p\mathbb{Z}\right)^*$.

Demostración. Por un lado, sea $1 \leq x < p$ una solución de $x^k \equiv a \bmod p$. Como $\left(\mathbb{Z}/p\mathbb{Z}\right)^*$ es un grupo cíclico, existe al menos un elemento g que lo genera. Por tanto, $x \equiv g^\alpha \bmod p$ para algún $\alpha \geq 1$. De igual modo, existirá un $\beta \geq 1$ tal que $a \equiv g^\beta \bmod p$. Se tiene entonces que $x^k \equiv g^{\alpha k} \equiv g^\beta \equiv a \bmod p$. Como el orden de g en el grupo es $p-1$, multiplicando la ecuación por $g^{-\beta}$ puede verse que $\alpha k - \beta$ es múltiplo de $p-1$.

Sea $r \in \mathbb{Z}$ tal que $\alpha k - \beta = r(p-1)$. Entonces, $\alpha k - r(p-1) = \beta$, luego por la identidad de Bézout, $d = mcd(k, p-1)$ divide a β. Esto implica finalmente que $a^{(p-1)/d} \equiv g^{\beta(p-1)/d} \equiv 1 \bmod p$.

Por otro lado, supóngase que el orden de a divide a $(p-1)/d$. Nuevamente, si g es un generador de $\left(\mathbb{Z}/p\mathbb{Z}\right)^*$, $a \equiv g^\beta \bmod p$ luego $a^{(p-1)/d} \equiv g^{\beta(p-1)/d} \equiv 1 \bmod p$, con lo cual d divide a β. Por otra parte, un elemento $x = g^\alpha$ verifica que $x^k \equiv g^{k\alpha} \equiv a \bmod p$ si y solo si $k\alpha \equiv \beta \bmod p-1$. Esta ecuación diofántica tiene solución (en α) dado que $d = mcd(k, p-1)$ divide a β, y de hecho, tiene exactamente d soluciones distintas módulo $p-1$. Finalmente, como $x = g^\alpha$ y g es un elemento generador del grupo, la ecuación $x^k \equiv a \pmod p$ tiene exactamente d soluciones. □

Corolario A.9. *Dado* $n = p_1, \ldots, p_r$ *donde los* p_i *son primos impares, la ecuación* $x^k \equiv a \bmod n$ *tiene solución si y solo si* $x^k \equiv a \bmod p_i$ *tiene solución para todo* $1 \leq i \leq r$. *Además, tiene* $d_1 \cdots d_r$ *soluciones, donde* $d_i = mcd(k, p_i - 1)$.

Demostración. El resultado se deduce del Teorema A.8 considerando el isomorfismo que define el Teorema Chino de los Restos. Si $x^k \equiv a \bmod p_i$ no tiene solución para algún i, tampoco la tendrá módulo n. En cambio, si la ecuación tiene solución para cada i, habrá $d_1 \cdots d_r$ posibles combinaciones de soluciones para cada p_i, que a través el isomorfismo darán lugar a ese mismo número de soluciones distintas módulo n. □

Se introduce a continuación la noción de residuo cuadrático y de símbolo de Legendre.

Definición A.10. *Sean* $p, r \in \mathbb{Z}$ *primos entre sí. Se dice que* r *es un residuo cuadrático módulo* p *si la ecuación de congruencia* $x^2 \equiv r \bmod p$ *tiene solución.*

Definición A.11. *Dados $a \in \mathbb{Z}$ y p un primo impar, el símbolo de Legendre de a y p es*

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{si } a \text{ es un residuo cuadrático módulo } p \\ -1 & \text{si } a \text{ no es un residuo cuadrático módulo } p \\ 0 & \text{si } p \text{ es divisor de } a \end{cases}$$

Se enuncian a continuación la fórmula de recurrencia de la sucesión de Fibonacci y el Teorema de Lamé, dos resultados encadenados de utilidad de cara a estudiar la complejidad computacional del Algoritmo de Euclides.

Teorema A.12. *Sean $F_1 = 1$, $F_2 = 1$, $F_i = F_{i-1} + F_{i-2}$ con $i > 2$ los elementos de la sucesión de Fibonacci. Entonces,*

$$F_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

Teorema A.13. *Sea $n \geq 1$, y sean $u > v > 0$ enteros tales que el Algoritmo de Euclides aplicado a u, v requiere exactamente de n divisiones, y tal que u es el entero más pequeño que satisface estas condiciones. Entonces, $u = F_{n+2}$ y $v = F_{n+1}$.*

A.2. Complejidad computacional

En primer lugar, se define el tamaño de un número entero, la notación O grande, la complejidad computacional de un algoritmo, y la notación o pequeña.

Definición A.14. *Dado un entero $m \in \mathbb{Z}$, el tamaño n de m es el número de bits que tiene la representación binaria de n. Se calcula como $n = \lfloor \log_2(|m|) \rfloor + 1$.*

Definición A.15. *Sean $t, f : \mathbb{N} \longrightarrow \mathbb{R}$ funciones. Se dice que $t \in O(f(n))$ si existen $c \in \mathbb{R}$, $n_0 \in \mathbb{N}$ tales que $t(n) \leq c \cdot f(n), \forall n \geq n_0$.*

Definición A.16. *Sea A un algoritmo cuyo tiempo de ejecución (o número de operaciones básicas) en el caso peor está dado por $t(n) \in O(f(n))$. Entonces, se dice que la complejidad computacional de A es $O(f(n))$.*

Definición A.17. *Sean $t, f : \mathbb{N} \longrightarrow \mathbb{R}$ funciones. Se denota $t \in o(f(n))$ si $\lim_{n\to\infty} \frac{t(n)}{f(n)} = 0$.*

Cuando se utilizan las notaciones estándar O y o, se asume que la función $f(n)$ es la función elemental más pequeña posible tal que $t \in O(f(n))$ y $t \in o(f(n))$, respectivamente, para que la complejidad tenga sentido práctico. Aunque es cierto que la función $t(n) = n^2 + 3n - \log n$ satisface que $t(n) \in O(n^4)$, es mucho más significativo dar la cota $t(n) \in O(n^2)$. En ese trabajo se comete a menudo un abuso de notación, indicando $t = O(f(n))$.

La complejidad de muchos algoritmos se puede encuadrar en una de las categorías siguientes. La complejidad computacional de un algoritmo cuyo tiempo de ejecución en el caso peor viene dado por $t(n)$ es:

- Logarítmica si $t \in O(\log(n))$.
- Lineal si $t \in O(n)$.
- Cuadrática si $t \in O(n^2)$.
- Polinómica si $t \in O(p(n))$, donde $p(n) \in \mathbb{R}[x]$.
- Subexponencial si $t \in O(k^{o(n)})$, donde $k > 1$.

- Exponencial si $t \in O(k^{p(n)})$, donde $k > 1$.

Proposición A.18. *Sean n, m enteros. La complejidad de algunas operaciones básicas sobre n, m es la siguiente:*

- *Suma, resta, suma modular y resta modular:* $O(\log n)$.
- *Producto:* $O(\log^a n)$, *con* $1 < a < 2$ *que depende del algoritmo utilizado. Por simplicidad, se tomará habitualmente* $a = 1$, *despreciando términos del orden de* $\log(\log n)$ *o inferior.*
- *División entera:* $O(\log^2 n)$.
- *Producto modular: Dado* $k \in \mathbb{N}$, $n \cdot m \bmod k$ *es* $O(\log^2 k)$.
- *Exponenciación modular: Dado* $k \in \mathbb{N}$, $n^m \bmod k$ *es* $O(\log m \log^2 k)$
- *Algoritmo de Euclides:* $O(\log^2 n)$.
- *Algoritmo de Euclides extendido (recuperando los coeficientes correspondientes a la identidad de Bézout):* $O(\log^2 n)$.

La complejidad tan baja que requiere el algoritmo de Euclides se debe al Teorema de Lamé y a la fórmula previa presentadas en la sección anterior.

Proposición A.19. *Sean $A, B \in \mathbb{Z}^{n \times n}$ dos matrices tales que el valor absoluto de sus coeficientes está acotado por B. El coste de algunas operaciones sobre las matrices es:*

- *Suma, resta:* $O(n^2 \log B)$.
- *Producto:* $O(n^\omega \log^a B)$, *con* $2 < \omega \leq 2{,}37$, *si se utilizan algoritmos de multiplicación rápida. Habitualmente, se toma* $O(n^3 \log B)$.
- *Cálculo del determinante,* $\det(A)$*:* $O(n^4 \log B + n^3 \log B^2)$.

Generalmente, aquellos problemas que se pueden resolver con un algoritmo con complejidad polinómica o mejor en el tamaño de la entrada se consideran tratables, mientras que aquellos para los que solamente se conocen algoritmos en tiempo subexponencial o peor, se denominan intratables. Para comparar la relación que existe entre dos problemas intratables, se pueden utilizar los conceptos siguientes.

Definición A.20. *Dados dos problemas A, B, se dice que A es reducible en tiempo polinómico (o reducible polinomialmente) a B si A se puede resolver en tiempo polinomial mediante una subrutina de un algoritmo que solucione B en tiempo polinomial.*

Definición A.21. *Dos problemas intratables A, B son computacionalmente equivalentes cuando son reducibles polinomialmente entre sí.*

Por último, se incluye una breve definición, poco formal, de algunas clases de complejidad.

Definición A.22. *Una clase de complejidad* $\boldsymbol{X}$ *es un conjunto que contiene a todos los problemas que pueden resolverse con algoritmos con complejidad computacional similar.*

Definición A.23. *Un lenguaje o problema de decisión (es decir, un problema para el que los posibles resultados son únicamente sí o no) L está en la clase* $\boldsymbol{P}$ *si existe una máquina de Turing determinista que lo decide en tiempo polinomial en el tamaño de la entrada. Informalmente, la clase* $\boldsymbol{P}$ *engloba los problemas tratables.*

Definición A.24. *Un lenguaje o problema de decisión L está en la clase* ***NP*** *si existe una máquina de Turing no determinista que lo decide en tiempo polinomial en el tamaño de la entrada. Informalmente, la clase* ***NP****, que contiene a* ***P****, engloba los problemas tratables y los intratables "menos difíciles".*

Definición A.25. *Un lenguaje o problema de decisión L está en la clase* ***RP*** *si existe una máquina de Turing determinista que lo decide probabilísticamente en tiempo polinomial en el tamaño de la entrada, con una probabilidad de falso positivo menor que 1/2. Informalmente, la clase* ***RP****, que contiene a* ***P*** *y está contenida en* ***NP****, engloba los problemas para los que existen algoritmos probabilísticos en tiempo polinomial.*

A.3. Operaciones con polinomios

Se detalla el coste computacional de algunas operaciones de polinomios en $\mathbb{F}[x]$, donde $\mathbb{F}$ es un cuerpo finito, junto con su complejidad computacional.

Proposición A.26. *Sean $f, g \in \mathbb{F}[x]$ dos polinomios de grados n, m respectivamente, con $n > m$, y sea $B = |\mathbb{F}|$. La complejidad de algunas operaciones básicas sobre $f(x)$ y $g(x)$ es la siguiente:*

- *Suma, resta: $O(n \log B)$.*
- *Producto: $O(n^2 \log B)$.*
- *División de polinomios: $O(n^2 \log B)$.*
- *Algoritmo de Euclides: a lo sumo $m+1$ cocientes y $2nm + O(n)$ sumas y productos en $\mathbb{F}$. Por tanto, $O(n^2 \log^2 B)$.*
- *Algoritmo de Euclides extendido: a lo sumo $m+1$ cocientes y $6nm + O(n)$ sumas y productos en $\mathbb{F}$. Por tanto, $O(n^2 \log^2 B)$.*

El algoritmo de Euclides (y, en general, todas las operaciones anteriores) se pueden realizar en caso de que los polinomios f, g pertenezcan a un anillo finito R (por ejemplo $R = {}^{\mathbb{Z}}\!/_{n\mathbb{Z}}$ con n no primo) siempre y cuando alguno de los coeficientes no sea un divisor de cero en R.

Cuando los polinomios son elementos de $\mathbb{Q}[x]$ o de $\mathbb{Z}[x]$, el coste computacional de realizar sumas, restas, productos y divisiones se mantiene (donde B pasa a ser una cota para el valor absoluto de los coeficientes de f, g). Sin embargo, los coeficientes del Algoritmo de Euclides, y por tanto el coste, pueden dispararse. Afortunadamente, existen algoritmos para calcular el máximo común divisor que trabajan de forma modular y que consiguen acotar su complejidad.

Proposición A.27. *Sean $f, g \in \mathbb{F}[x]$ dos polinomios de grados n, m respectivamente, con $n > m$, y sea B una cota superior del valor absoluto de los coeficientes de f, g. El coste computacional de calcular el máximo común divisor de f, g es $O(n^3 \log^2(n) m \log^2(nB))$.*

Es decir, polinomial en el grado y el tamaño de los coeficientes del polinomio. Se introduce la noción de resultante, que se utiliza en este trabajo para resolver sistemas de ecuaciones polinomiales en dos variables.

Definición A.28. *Sea R un anillo conmutativo con unidad y sean $f = \sum_{i=0}^{n} a_i x^i$, $g = \sum_{i=0}^{m} b_i x^i$ dos polinomios de grado n y m respectivamente, $f, g \in R[x]$. La matriz de Sylvester de f y de g, denotada por $Syl(f, g)$, es una matriz cuadrada de tamaño $n + m$ dada por*

$$Syl(f,g) = \left(\begin{array}{ccccccc} a_n & a_{n-1} & \cdots & a_0 & 0 & \cdots & 0 \\ 0 & a_n & \cdots & a_1 & a_0 & \cdots & 0 \\ & & \ddots & \ddots & \ddots & \ddots & \\ 0 & 0 & \cdots & a_n & \cdots & a_1 & a_0 \\ \hline b_m & b_{m-1} & \cdots & b_0 & 0 & \cdots & 0 \\ 0 & b_m & \cdots & b_1 & b_0 & \cdots & 0 \\ & & \ddots & \ddots & \ddots & \ddots & \\ 0 & 0 & \cdots & b_m & \cdots & b_1 & b_0 \end{array}\right)$$

donde el bloque superior tiene m filas y el inferior n filas. El determinante de la matriz de Sylvester se conoce como resultante de f y g, y se denota por $res(f,g) = \det(Syl(f,g))$.

Proposición A.29. *Sea $\mathbb{K}$ un cuerpo y sean $f, g \in \mathbb{K}[x]$. Entonces,* $\deg(mcd(f,g)) = (n+m) - rank(Syl(f,g))$.

Proposición A.30. *Sea R un dominio de factorización única y sean $f, g \in R[x]$ no ambos nulos. Entonces,* $\deg(mcd(f,g)) \geq 1$ *si y solo si* $res(f,g) = 0$ *en R.*

Si R es dominio de factorización única, por el Lema de Gauss $R[x_1, \ldots, x_k]$ también lo es. Por tanto, si $f, g \in R[x_1, \ldots, x_n]$, se puede suponer, para cada variable x_i, que los polinomios se encuentran en el anillo $R[x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n][x_i]$. La resultante de f, g respecto de la variable x_i se denota $res_{x_i}(f,g) \in R[x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]$.

A lo largo del trabajo aparecen polinomios en varias variables con coeficientes enteros. El cálculo de la resultante respecto de una variable conlleva un tiempo polinomial en el grado de los polinomios y el tamaño de sus coeficientes. Esto se deduce de la complejidad computacional de calcular un determinante junto con el coste de las operaciones de suma y producto de polinomios.

Por otro lado, las resultantes se utilizan para resolver sistemas de ecuaciones de polinomios en varias variables. En el trabajo aparecen sistemas en dos variables. Su resolución se lleva a cabo eliminando una de las variables mediante el cálculo de la resultante respecto de ella.

Proposición A.31. *Sean $f, g \in \mathbb{Z}[x,y]$ algebraicamente independentes, es decir, $mcd(f,g) \in \mathbb{Z}$, y sean $x_0, y_0 \in \mathbb{Z}$. Si $f(x_0, y_0) = g(x_0, y_0) = 0$, entonces $res_x(f,g)(y_0) = 0$. Además,* $\deg(res_x(f,g)) \leq \deg(f)\deg(g)$.

Para resolver el sistema $f(x,y) = g(x,y) = 0$, se determinan las raíces $y_1, \ldots, y_r$ de $h(y) = res_x(f,g) \in \mathbb{Z}[y]$. Posteriormente, se hallan las raíces comunes de $f(x, y_i)$ y de $g(x, y_i)$ calculando el máximo común divisor de ambos polinomios en $\mathbb{Z}[x]$.

Apéndice B

Ejemplo del Algoritmo de Coppersmith

A continuación, se presenta un ejemplo de ejecución de un ataque a RSA que utiliza el algoritmo de Coppersmith. El ejemplo es completo; se muestra la generación de claves, el cifrado del mensaje y su posterior criptoanálisis mediante un ataque de mensaje estereotipado (apartado 5.4.1). Los números involucrados son pequeños, ya que el fin del ejemplo es ilustrar el funcionamiento del algoritmo.

Algoritmo de Coppersmith

Se introduce de forma algorítmica el método de Coppersmith.

- **Entrada:** Un polinomio univariado $f(x)$ de grado δ, un entero n múltiplo de b, y cotas β, ϵ con $b \geq n^{\beta}, \epsilon \leq \frac{1}{7}\beta$.

1. Se calculan $m = \lceil \beta^2/\delta\epsilon \rceil$ y $t = \lfloor \delta m(-1 + 1/\beta) \rfloor$. Se determina la base $\mathcal{P}$ de polinomios,
$$\mathcal{P} = \left\{ x^j n^{m-i} f^i : 0 \leq i < m, 0 \leq j < \delta \right\} \bigcup \left\{ x^i f^m : 0 \leq i < t \right\}.$$
2. Se calcula la cota $X = \frac{1}{2} n^{\frac{\beta^2}{\delta} - \epsilon}$.
3. Se define la matriz triangular B formada por los coeficientes de los polinomios $p_i(xX)$, donde $p_i \in \mathcal{P}$. La matriz B genera un retículo de dimensión $\gamma = \delta m + t$.
4. Se calcula una base LLL-reducida del retículo generado por B y se elige su primer vector, v, que se corresponde con los coeficientes de un polinomio $g(xX)$. Se obtiene $g(x)$ a partir de v.
5. Se factoriza $g(x)$ sobre los enteros, obteniendo un conjunto de raíces R. Para cada raíz $x_0 \in R$, se determina si $mcd(f(x_0), n)) \geq n^{\beta}$, y en caso contrario, se elimina x_0 de R.

- **Salida:** El conjunto R de raíces x_0 tales que $f(x_0) = 0$ mód b con $|x_0| \leq X$.

Se muestra a continuación el desarrollo del ataque.

Protocolo RSA

Alicia, que se encuentra de viaje, desea enviarle a su hijo Bonifacio el PIN de su tarjeta de crédito para que Bonifacio pueda comprarle un regalo a su hermana por su cumpleaños. Alicia, sin embargo, no confía en la conexión a internet de su hotel y decide cifrar el mensaje por medio de RSA. Para ello, Bonifacio genera un par de clave pública y privada, de la forma siguiente:

- Elige dos primos, $p = 641$, $q = 509$. Calcula $n = pq = 326269$, $\varphi(n) = 325120$.
- Elige una clave de cifrado, $e = 3$.
- Calcula su clave de descifrado d tal que $ed \equiv 1$ mód $\varphi(n)$. Obtiene $d = 216747$.
- Le envía a Alicia su clave pública, el par (e, n).

Alicia cifra su PIN $M = 1997$, que de hecho es el año de nacimiento de su hija.

$$C = M^e \text{ mód } n = 153952.$$

Se puede comprobar que Bonifacio puede obtener M como $M = C^d$ mód $n = 1997$.

Ataque de mensaje estereotipado

Mariano, un huésped del hotel con malas intenciones, ha estado atento a la conversación entre Alicia y Bonifacio, interceptando todo su contenido. Mariano conoce e, n y C, y además, sospecha que el PIN de la tarjeta es el año de nacimiento de la hija de Alicia. Por tanto, decide efectuar un ataque de mensajes estereotipados con la estimación $\tilde{M} = 1990$. De acuerdo con la Proposición 5.14, $x = M - \tilde{M}$ será una raíz pequeña del polinomio

$$f(x) = (\tilde{M} + x)^3 - C = x^3 + 5970x^2 + 134616x + 69891 \text{ mód } n$$

si Mariano está en lo correcto. Mariano se dispone a buscar la raíz x aplicando el Teorema 5.11, ejecutando el Algoritmo de Coppersmith con los parámetros $\beta = 1$ (ya que $b = n$), $\delta = 3$, $\epsilon = 1/12$.

Paso 1:

Se calculan $m = \lceil \beta^2/\delta\epsilon \rceil = 4$ y $t = \lfloor \delta m(-1+1/\beta) \rfloor = 0$, con lo cual el tamaño del retículo será $\gamma = \delta m + t = 12$. Se determina la base de polinomios $\mathcal{P} = \left\{ x^j n^{4-i} f^i, j = 0, 1, 2,\ i = 0, 1, 2, 3 \right\}$:

$$\begin{array}{lll} n^4 & xn^4 & x^2n^4 \\ n^3f(x) & xn^3f(x) & x^2n^3f(x) \\ n^2f(x)^2 & xn^2f(x)^2 & x^2n^2f(x)^2 \\ nf(x)^3 & xnf(x)^3 & x^2nf(x)^3 \end{array}$$

Paso 2:

$$X = \frac{1}{2} n^{\frac{\beta^2}{\delta} - \epsilon} = \frac{1}{2} 326269^{\frac{1}{3} - \frac{1}{12}} = 11{,}8$$

Luego, truncando a los enteros, se toma $X = 11$ como cota. Es decir, el algoritmo funcionará si $1979 \leq M \leq 2001$.

Paso 3:

Se determina la matriz B triangular inferior de tamaño 12×12 formada por los coeficientes de los polinomios $p_i(xX)$, con $p_i \in \mathcal{P}$. Cada fila de la matriz es un vector de la base del retículo y la i-ésima columna corresponde al coeficiente de grado $(i-1)$-ésimo del polinomio correspondiente. Se muestra parcialmente a continuación.

$$B = \begin{pmatrix} 11331913412989554250321 & & & & \\ \vdots & \ddots & & & \\ 519988971233591689041 & \cdots & 188585255568593521 & & \\ \vdots & \ddots & \vdots & \ddots & \\ 0 & \cdots & 91876342468297038262378812 & \cdots & 93088353458580359 \end{pmatrix}$$

Paso 4:

Aplicando el algoritmo LLL sobre el retículo generado por B, se obtiene la matriz B' siguiente:

$$B' = \begin{pmatrix} 2522622606700616658 & 1268146000239181437 & \cdots & 1675590362254446462 \\ 5214713401293781385 & 10914323117616452886 & \cdots & 6888538155934946566 \\ \vdots & \vdots & \ddots & \vdots \\ 6745697945504613613274 & 429379859298243933827 0 & \cdots & 0 \end{pmatrix}$$

El vector más corto v de la base LLL-reducida se corresponde con la primera fila de B',

$$v^T = \begin{pmatrix} 2522622606700616658 \\ 1268146000239181437 \\ 3431473649249253850 \\ 4098174776141752779 \\ 5277868155684978447 \\ 4804372723377864241 \\ 2206476217022708873 \\ 8811673218247884094 \\ 091483069148757366 \\ 9558866047296371575 \\ 7193190949072118650 \\ 1675590362254446462 \end{pmatrix}$$

Dividiendo la componente i-ésima del vector por X^{i-1}, se obtienen los coeficientes del polinomio $g(x)$ buscado,

$$\begin{aligned} g(x) = {} & 2522622606700616658 + 1152860000217437 67x - 28359286357431850x^2 \\ & - 3079019365996809x^3 - 360485496597567x^4 - 29831374678691x^5 + 1245498301793x^6 \\ & + 452178164714x^7 + 37747365686x^8 - 4053892325x^9 - 277328650x^{10} - 5872842x^{11} \end{aligned}$$

Paso 5:

La factorización de g es la siguiente:

$$\begin{aligned} g(x) = {} & -326269(x-7)(18x^{10} + 976x^9 + 19257x^8 + 19105x^7 - 1252171x^6 - 12582594x^5 \\ & + 3353681x^4 + 1128347810x^3 + 17335492331x^2 + 208268399967x + 1104532327926). \end{aligned}$$

Como 7 es la única raíz entera de $g(x)$, es la raíz modular buscada de $f(x)$. En efecto, $M = \tilde{M} + x = 1990 + 7 = 1997$ y Mariano puede recuperar el PIN de la tarjeta de Alicia.