Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

Grado en Matemáticas

Firma electrónica con criptografía multivariante

Autor: Samuel Vicente Sánchez
Tutor: José Ignacio Farrán Martín
2024-2025

Resumen

El objetivo principal de este trabajo es el estudio de las firmas electrónicas resistentes a la computación post-cuántica, con especial énfasis en la criptografía multivariante como alternativa segura frente a estos avances en computación. Se analizan los fundamentos teóricos y los desafíos actuales de este tipo de criptografía en general, entrando en detalle en el caso del esquema de firma digital GeMSS, que se utilizará como caso particular sobre el que se aplica la teoría explicada. Además, se estudian distintos tipos de ataques conocidos sobre este esquema de firma, estudiando el trasfondo matemático de los mismos, con lo que se pretende evaluar la viabilidad de este sistema en un escenario post-cuántico.

Palabras clave: Criptografía post-cuántica, criptografía multivariante, *MinRank*, GeMSS

Abstract

The main objective of this work is the study of electronic signature schemes that are resistant to post-quantum computing, with a particular focus on multivariate cryptography as a secure alternative in light of advances in quantum computation. The theoretical foundations and current challenges of this type of cryptography are analyzed, with a detailed examination of the GeMSS digital signature scheme, which serves as a case study for applying the discussed theory. Furthermore, various known attacks on this signature scheme are explored, along with the mathematical background behind them, in order to assess the viability of this system in a post-quantum scenario.

Keywords: Post-Quantum Cryptography, Multivariate Cryptography, *MinRank*, GeMSS

Índice general

Capítulo 1

Introducción

La inminente rápida evolución de los ordenadores cuánticos hace que la criptografía, uno de los pilares fundamentales para la protección de la información en la era digital, se encuentre ante una amenaza histórica. Esto ha provocado que la comunidad internacional se vuelque intentando buscar una solución, ya que el principal problema no es la aparición de ordenadores cuánticos, presentes en nuestra sociedad desde 1998, sino su rápido desarrollo, que hace que muchos de los sistemas criptográficos usados actualmente, como RSA, DSA o ECDSA, se enfrenten a un desafío sin precedentes.

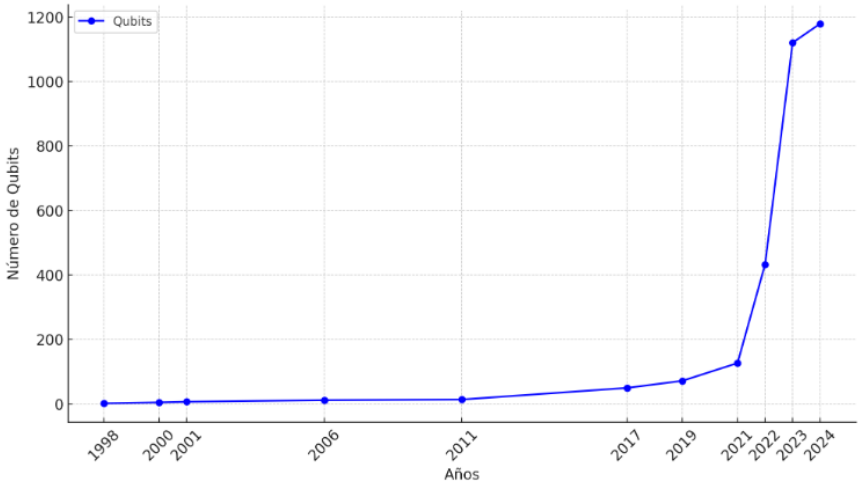

Figura 1.1: Evolución del número de *qubits* en ordenadores cuánticos.

Para tener una visión general de este rápido avance, podemos observar la Figura 1.1, en la que vemos cómo en los últimos años la cantidad de *qubits* en el ordenador cuántico más potente ha crecido de forma exponencial. Los ordena-

dores cuánticos se estima que serán capaces de resolver problemas matemáticos que hasta ahora se consideraban intratables, pero para ello necesitan alcanzar un mínimo de *qubits* para poder afrontarlos. Con esta rápida evolución, las posibilidades de que alcancen estos valores son cada vez más altas, poniendo en peligro la seguridad de las tecnologías basadas en estos problemas.

Recuperando el tema de interés para este trabajo, cabe destacar que la llegada de los ordenadores cuánticos no marca el fin de la criptografía, sino el inicio de una nueva etapa, que se conoce como criptografía post-cuántica. Además, es importante tener en cuenta que la criptografía que está en peligro con los ordenadores cuánticos es la criptografía asimétrica (de clave pública); la criptografía simétrica (de clave privada, el AES, y Vernam con secuencias cifrantes) hasta la fecha no corre peligro. Este segundo tipo de criptografía, cuyas características explicaremos en este trabajo, solo puede ser vulnerada por ataques de fuerza bruta. Los ataques de este tipo se pueden acelerar gracias al algoritmo de Grover, pero no lo suficiente como para asegurar que la criptografía simétrica esté en peligro.

Gracias a las nuevas investigaciones, se están encontrando sistemas criptográficos resistentes al ataque de ordenadores cuánticos. Estos nuevos sistemas mejoran los utilizados actualmente, ya que estos sistemas antiguos presentan el problema de que su complejidad exponencial es reducida a una complejidad polinomial utilizando un ordenador cuántico con una cantidad suficiente de *qubits*. Por tanto, no tenemos una razón para justificar el salto de "los ordenadores cuánticos destruyen RSA, DSA y ECDSA" a "los ordenadores cuánticos destruyen la criptografía", ya que hay muchos sistemas criptográficos importantes más allá de los clásicos y que, además, son resistentes a ordenadores cuánticos. Entre ellos, vamos a destacar los siguientes:

- **Criptografía basada en funciones *hash*:** este tipo de criptografía se aplica principalmente en la creación de firmas digitales. Su seguridad depende de las propiedades de las funciones *hash*, que transforman datos de entrada en cadenas de longitud fija. Sin embargo, estas firmas tienen una limitación: requieren generar una nueva clave para cada operación, lo que puede restringir su uso en ciertos escenarios.

- **Criptografía basada en códigos correctores de errores:** introducida por Robert McEliece en 1978, esta técnica utiliza códigos correctores de errores como base para el cifrado. El método convierte cada mensaje en palabras de un código lineal que tiene capacidad de corrección de errores. Aunque requiere tamaños de clave pública más grandes que otros sistemas, su resistencia a ataques cuánticos la hace muy interesante para aplicaciones prácticas.

- **Criptografía basada en retículos:** este enfoque se centra en problemas teóricos relacionados con la estructura de los retículos, conocidos como problemas difíciles. La dificultad radica en la resolución de estos problemas en el peor de los casos, lo que los hace especialmente adecuados para aplicaciones criptográficas. Por ello, los retículos se han convertido en una base prometedora para diseñar sistemas criptográficos seguros frente a ordenadores cuánticos.

- **Criptografía multivariante:** este campo utiliza sistemas de ecuaciones polinomiales en varias variables sobre cuerpos finitos como base para la construcción de esquemas criptográficos. Los sistemas multivariantes se consideran resistentes tanto a ataques clásicos como a ataques cuánticos, ya que resolver ecuaciones polinomiales de alto grado es un problema computacionalmente difícil. Este enfoque se utiliza principalmente para desarrollar esquemas de firma digital rápidos y eficientes.

A pesar de los avances, la criptografía post-cuántica enfrenta desafíos significativos en términos de eficiencia, confianza y usabilidad. Los algoritmos propuestos suelen requerir claves más grandes y recursos computacionales superiores, lo que plantea barreras para su adopción generalizada. Clave para solucionar estos problemas están siendo instituciones como el NIST (*National Institute of Standards and Technology*), que además de encargarse del proceso de estandarización y evaluación de estos algoritmos, ha alentado el desarrollo de la criptografía post-cuántica.

El NIST planteó en 2016 un concurso llamado *Post-Quantum Cryptography Standardization Process* con el que se buscaba una estandarización de algoritmos resistentes a ordenadores cuánticos tanto para el cifrado de clave pública, como para la firma digital y el intercambio de claves. Una vez llegada la fecha límite para presentar algoritmos, hacia finales de 2017, se permitió que los participantes pudieran ser sometidos a cualquier tipo de ataque, para poder elegir como ganadores a aquellos que fueran más resistentes. El concurso fue pasando por diferentes fases con el paso de los años hasta que se eligieron dos ganadores. El elegido para el establecimiento de claves fue CRYSTALS-KYBER, mientras que para la firma digital el ganador fue CRYSTALS-Dilithium, ambos basados en retículos. Con ello, podríamos llegar a pensar que la línea de desarrollo correcta son los algoritmos basados en retículos, pero nada más lejos de la realidad, pues en el año 2024 se publicó un artículo que parecía comprometer definitivamente toda la seguridad de la criptografía basada en retículos. Finalmente, se detectó un error que parece ser irremediable en el artículo, pero con ello se quiere hacer ver que con la criptografía en cualquier momento todo puede dar un giro inesperado y cambiar completamente la visión actual.

Por ello, en este trabajo nos centraremos en otra de las principales vías de desarrollo de la criptografía post-cuántica que ya hemos mencionado antes, la criptografía multivariante, basada en la resolución de sistemas de ecuaciones polinomiales en varias variables. Para ello, después de realizar una primera toma de contacto con la criptografía, la computación cuántica y la relación entre ambas, nos centraremos en los fundamentos, ventajas y desafíos de este tipo de criptografía, apoyándonos para ello en un esquema de firma de este tipo, GeMSS. Finalmente, analizaremos los posibles ataques a este criptosistema y cómo sería posible contrarrestarlos.

Capítulo 2

Computación cuántica

Antes de la aparición de la computación cuántica, había una clara línea divisoria entre los problemas que se podían resolver y aquellos que eran computacionalmente irresolubles con la tecnología disponible. La computación cuántica ha conseguido dejar esta división, de momento, en el aire, por la posibilidad de que aparezcan algoritmos que sean lo suficientemente eficientes para resolver problemas hasta ahora considerados imposibles, en los que se basan algunos criptosistemas que se creían seguros con la computación clásica. Por ejemplo, a finales del siglo XX [26], Shor encontró algoritmos cuánticos para la factorización y logaritmo discreto , que pueden usarse para romper criptosistemas como el RSA o el intercambio de claves de Diffie-Hellman, ampliamente utilizados actualmente.

Pero la verdadera duda sobre la posible vulnerabilidad de los sistemas criptográficos surgió antes, concretamente en 1982, cuando Richard P. Feynman publicó un artículo [15] donde se mostraba una evidente mejora de la eficiencia de la computación cuántica respecto a la computación de los ordenadores clásicos. Esta mejora se basa en la posibilidad de los *qubits* de encontrarse en un estado intermedio entre los estados 0 y 1, mientras que los bits usados en los ordenadores clásicos solo pueden estar en uno de los dos estados de forma excluyente.

En este capítulo encontraremos las nociones básicas de la computación cuántica, necesarias para comprender el funcionamiento del Algoritmo de Shor, que desarrollaremos también en este mismo capítulo. Para ello, explicaremos primero qué son los *qubits* y cómo se relacionan con los bits que ya conocemos, además de cómo se construyen algunas de las puertas cuánticas básicas, que se utilizarán posteriormente en la explicación del Algoritmo de Shor. Para terminar el capítulo, veremos de forma resumida otro importante algoritmo en el ámbito cuántico, el algoritmo de Grover.

2.1. Bits y qubits

Como sabemos, los *qubits* son la unidad mínima utilizada en los sistemas cuánticos, pero estos surgen a partir de los bits, que precisamente son la unidad mínima de información que se utiliza en la computación actual. De hecho, el *qubit* es la analogía del bit en la computación cuántica. Podríamos decir que un bit es una unidad mínima de información que representa una de dos situaciones disjuntas, representadas habitualmente con 0 o 1. Por ejemplo, un bit podría ser una forma de decir verdadero (1) o falso (0), un interruptor encendido o apagado, o la electricidad fluyendo por un circuito o no haciéndolo. Todos estos ejemplos dicen lo mismo: un bit es una forma de describir un sistema cuyo conjunto de estados tiene tamaño dos.

Diremos que tanto bits como *qubits* tienen un estado, que representaremos utilizando la notación estándar que se utiliza en mecánica cuántica, la de Dirac, en la que un estado se representa por $|x\rangle$. Como ya hemos mencionado, un bit puede estar en el estado $|0\rangle$ o en el estado $|1\rangle$, que es suficiente para el mundo de la computación clásica. Pero para el mundo cuántico necesitamos algo más, ya que tenemos situaciones en las que estamos en un estado y en otro simultáneamente. Haciendo analogía con uno de los ejemplos anteriores, en el mundo cuántico diríamos que hay sistemas donde un interruptor está en una superposición de estados encendido y apagado; de hecho, el poder de los ordenadores cuánticos radica en el hecho de que un sistema puede estar en muchos estados al mismo tiempo.

En el mundo cuántico tenemos muchos más estados, que se construyen a partir de los estados cero, $|0\rangle$, y uno, $|1\rangle$, que son los estados base. Para otros estados, podemos utilizar combinaciones lineales de los estados base, a lo que se le denomina superposición. Si tenemos un estado cualquiera $|\psi\rangle$, lo podemos representar como:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.1}$$

donde α y β son números complejos que cumplen que $|\alpha|^2 + |\beta|^2 = 1$, entendiéndose por $|\alpha|^2$ la probabilidad de que el *qubit* se encuentre en el estado cero y $|\beta|^2$ la probabilidad de que se encuentre en estado uno.

Por lo tanto, definimos un *qubit* (o bit cuántico) como una forma de describir un sistema cuántico de dimensión dos. La forma más habitual para representar un *qubit* es mediante un vector representado en el espacio vectorial $\mathbb{C}^2$ (espacio vectorial complejo de dimensión 2). Cabe destacar que estamos haciendo referencia en este ejemplo al caso en el que el sistema tiene un único *qubit*, pero se puede generalizar para n *qubits*, teniendo en esa situación un espacio vectorial de dimensión 2^n.

Continuando con el ejemplo para un único *qubit*, denominamos base compu-

tacional a la base ortonormal formada por los estados $|0\rangle$ y $|1\rangle$,

$$\mathcal{B} = \left\{ |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\} \tag{2.2}$$

y en este caso podríamos representar (2.1) como

$$|\psi\rangle = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \alpha, \beta \in \mathbb{C}, \tag{2.3}$$

donde $|\psi\rangle$ es un estado cualquiera, que se puede expresar como combinación lineal de los estados de la base.

Ejemplo 2.1.1. Veamos cómo cualquier elemento de $\mathbb{C}^2$ se puede convertir en un *qubit*, expresándolo en la base ortonormal que hemos definido en (2.2), y cómo podemos calcular las probabilidades de que esté en cada uno de los dos estados posibles. Tomemos, por ejemplo, el vector

$$\mathbf{v} = \begin{bmatrix} 5+3i \\ 6i \end{bmatrix}$$

que tiene norma

$$|\mathbf{v}| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle} = \sqrt{\begin{bmatrix} 5-3i, -6i \end{bmatrix} \begin{bmatrix} 5+3i \\ 6i \end{bmatrix}} = \sqrt{34+36} = \sqrt{70}.$$

Por lo tanto, $\mathbf{v}$ describe el mismo estado físico que el *qubit*

$$\frac{\mathbf{v}}{\sqrt{70}} = \begin{bmatrix} \frac{5+3i}{\sqrt{70}} \\ \frac{6i}{\sqrt{70}} \end{bmatrix}$$

y podemos expresarlo como

$$\frac{\mathbf{v}}{\sqrt{70}} = \frac{5+3i}{\sqrt{70}} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{6i}{\sqrt{70}} \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Observando entonces (2.3), tendríamos que

$$\alpha = \frac{5+3i}{\sqrt{70}}, \beta = \frac{6i}{\sqrt{70}}$$

y por tanto las probabilidades de estar en los estados $|0\rangle$ y $|1\rangle$ serán, respectivamente

$$|\alpha|^2 = \frac{34}{70}, |\beta|^2 = \frac{36}{70}$$

Ejemplo 2.1.2. Los estados con los que estamos constantemente trabajando no son más que vectores, como hemos explicado en el apartado previo, por tanto todas las propiedades de las operaciones con vectores se conservan al operar con estados. Por ejemplo, tomemos el *qubit*

$$\mathbf{w} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

cuya norma es 1. Podemos expresarlo en la base de la siguiente forma:

$$\mathbf{w} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

Simplemente por la propiedad conmutativa podemos asegurar que:

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{|1\rangle + |0\rangle}{\sqrt{2}}$$

y tendremos entonces dos formas de escribir el *qubit* $\mathbf{w}$.

De manera similar, si ahora consideramos

$$\mathbf{u} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}$$

que puede expresarse en la base computacional

$$\mathbf{u} = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

y también tendríamos que

$$-\mathbf{u} = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = -\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = \frac{|1\rangle - |0\rangle}{\sqrt{2}}.$$

por tanto, como $\mathbf{u} \neq 0, \mathbf{u} \in \mathbb{C}^2$, entonces $\mathbf{u} \neq -\mathbf{u}$ y podemos asegurar que

$$\frac{|0\rangle - |1\rangle}{\sqrt{2}} \neq \frac{|1\rangle - |0\rangle}{\sqrt{2}}.$$

Sin embargo, como hemos visto ambos están relacionados:

$$\frac{|0\rangle - |1\rangle}{\sqrt{2}} = (-1)\frac{|1\rangle - |0\rangle}{\sqrt{2}}.$$

2.2. Conceptos preliminares

Como veremos en el último apartado de este capítulo, el Algoritmo de Shor está basado en otros algoritmos tradicionales de factorización a los que se les añade una parte cuántica. Desarrollaremos una base teórica del cálculo del periodo de una función, necesaria para las etapas cuánticas, explicando también el funcionamiento de la puerta de Hadamard y la transformada cuántica de Fourier. En este apartado, además de profundizar en estos conceptos, entraremos en detalle acerca de la estimación cuántica de la fase, herramienta basada en estas puertas y que se utiliza en varios algoritmos cuánticos, entre ellos, en el algoritmo de Shor.

2.2.1. Definiciones y resultados previos

Se muestran a continuación una serie de definiciones y proposiciones necesarias para las distintas etapas del algoritmo de Shor. La parte cuántica del algoritmo, como veremos en el apartado 2.3, se basa en el cálculo del periodo de una función, sobre el que explicaremos la base teórica con los siguientes resultados.

Definición 2.2.1. Sean $a, b \in \mathbb{Z}$ enteros con al menos uno de ellos diferente a cero, supongamos que $a \neq 0$. El *máximo común divisor* de a y b, denotado por $\text{mcd}(a, b)$, es el entero positivo $d \in \mathbb{Z}_{>0}$, que satisface:

1) $d|a$ y $d|b$

2) Si $c|a$ y $c|b$, entonces $c|d$.

Además, si $\text{mcd}(a, b) = 1$, se dice que a y b son coprimos.

Definición 2.2.2. Sea $n \geq 2$. Si $\text{mcd}(a, n) = 1$, el *orden de a de módulo n* es el entero positivo más pequeño $r \in \mathbb{Z}_{>0}$ tal que

$$a^r \equiv 1 \text{ mód } n \tag{2.4}$$

Proposición 2.2.1. *Sea $f(x) = a^x \text{ mód } n$, $x \in \mathbb{Z}$, donde $a, n \in \mathbb{Z}_{>0}$ son enteros positivos. Entonces, $f(x)$ es periódica, y el periodo es exactamente el orden de a módulo n.*

Demostración:
Sea r el orden de a de módulo n. Por (2.4), sabemos que:

$$a^r \equiv 1 \text{ mód } n.$$

Esto implica que para cualquier entero k:

$$a^{k+r} = a^k \cdot a^r \equiv a^k \cdot 1 \equiv a^k \text{ mód } n,$$

es decir,

$$a^{k+r} \text{ mód } n = a^k \text{ mód } n, \quad \forall k \in \mathbb{Z},$$

en concreto, para $x \in \mathbb{Z}$,

$$f(x) = a^x \text{ mód } n = a^{x+r} \text{ mód } n = f(x+r).$$

y queda demostrado que f es periódica y además su periodo es r, que es el orden de a de módulo n.

□

La cuestión fundamental del Algoritmo de Shor es factorizar un entero positivo N como producto de primos, y para ello es necesario elegir otro entero positivo a entre 1 y N que cumpla ciertas condiciones que pide el algoritmo. Como hemos visto con el anterior resultado, será posible encontrar el orden r simplemente buscando el periodo de la función f que hemos definido, por tanto la cuestión que nos falta por resolver es la elección del entero a. Como veremos más adelante, esta elección es aleatoria, pero daremos los siguientes resultados para aportar información sobre una elección ventajosa de a. Cabe destacar que las siguientes demostraciones se han obtenido basándose en la idea proporcionada por el propio Peter Shor en [26].

Definición 2.2.3. Para cada $n \geq 1$, denotamos con $\varphi(n)$ a la función *"phi" de Euler*, que se define como el número de enteros positivos y menores que n que son coprimos con el propio n.

Lema 2.2.1. *Sean $p \in \mathbb{N}$ un número primo impar y $\alpha \in \mathbb{Z}_{>0}$ entero positivo. Sea 2^d la mayor potencia de 2 tal que divide a $\varphi(p^\alpha)$. Entonces, para algún elemento aleatorio h del grupo $\mathbb{Z}_{p^\alpha}^\times$, 2^d divide al orden de h módulo p^α con probabilidad $\frac{1}{2}$.*

Demostración:

Por las propiedades de la función $\varphi(n)$ definida anteriormente, al ser p impar, tenemos que $\varphi(p^\alpha) = p^{\alpha-1}(p-1)$ es par, luego tenemos que $d \geq 1$. De nuevo, por ser p impar y $\alpha \in \mathbb{Z}_{>0}$ entero positivo, entonces podemos asegurar que el grupo $\mathbb{Z}_{p^\alpha}^\times$ es cíclico y, en consecuencia, existe g elemento generador del grupo.

Sea r el orden de g^k módulo p^α, con $k = 1, \ldots, \varphi(p^\alpha)$, es decir,

$$(g^k)^r \equiv 1 \text{ mód } p^\alpha, \ k = 1, \ldots, \varphi(p^\alpha). \tag{2.5}$$

Tenemos los dos siguientes casos en función del valor de k:

- Si k es impar, entonces por (2.5), se tiene que $\varphi(p^\alpha)$ divide a $k \cdot r$, $\varphi(p^\alpha) \mid k \cdot r$, por ser g un generador de un grupo cíclico. Entonces, como $\varphi(p^\alpha)$ es par, por la definición de d tendrá el factor 2^d en su descomposición, y al ser k impar, podemos concluir que en $k \cdot r$ solo aporta factores pares r, por lo que $2^d \mid r$.

- En cambio, si k es par podemos escribir la siguiente cadena de igualdades

$$\begin{aligned} & g^{\frac{k\varphi(p^\alpha)}{2}} \text{ mód } p^\alpha \\ = & \left(g^{\varphi(p^\alpha)}\right)^{k/2} \text{ mód } p^\alpha \\ = & \, 1^{k/2} \text{ mód } p^\alpha \\ = & \, 1 \text{ mód } p^\alpha. \end{aligned}$$

  De esta cadena de igualdades podemos deducir que $r \mid \frac{\varphi(p^\alpha)}{2}$, y de ahí obtenemos que 2^d no divide a r, $2^d \nmid r$.

Para concluir, podemos ver que lo que hemos hecho es particionar el grupo $\mathbb{Z}_{p^\alpha}^\times$ en dos subconjuntos de igual cardinal. La igualdad de cardinal se deduce de haber dividido los valores de k entre pares e impares, con los valores de k comenzando por 1 (impar) y terminando por $\varphi(p^\alpha)$ (par). Por tanto, solo en los casos donde k es impar se da que 2^d divide al orden r de un elemento aleatorio del grupo $\mathbb{Z}_{p^\alpha}^\times$, y al ser estos la mitad de los casos totales, tenemos que la probabilidad de que esto ocurra es $\frac{1}{2}$, como queríamos probar.

□

Teorema 2.2.1. *Sea $n \in \mathbb{N}$ un número natural impar que se pueda descomponer en factores primos, es decir, podríamos expresarlo como: $n = p_1^{\alpha_1} \dots p_m^{\alpha_m}$. Sea $a \in \mathbb{Z}_{>0}$, $1 \leq a \leq n-1$ y a coprimo con n, y sea r el orden de a de módulo n. Entonces, si medimos la probabilidad de que r sea par y que $a^{r/2} \neq -1$ mód n, tenemos que:*

$$p(r \text{ es par y } \ a^{r/2} \neq -1 \text{ mód } n) \geq 1 - \frac{1}{2^m} \tag{2.6}$$

Demostración:

Podemos reescribir la desigualdad que queremos probar como

$$p(r \text{ es impar o } a^{r/2} = -1 \text{ mód } n) \leq \frac{1}{2^m} \tag{2.7}$$

donde hemos utilizado algunas propiedades básicas de la probabilidad.

Probaremos (2.7) y con ello, por lo mencionado antes, quedará probado el teorema. En la situación que se nos presenta en este teorema, se cumplen todas

las condiciones del Teorema Chino de los Restos, cuyo enunciado y demostración se pueden consultar, por ejemplo, en [25]. Por este resultado, elegir $a \in \mathbb{Z}_n^\times$ es equivalente a elegir $a_j \in \mathbb{Z}^\times_{p_j^{\alpha_j}}$ independientes y uniformemente distribuidos tales que

$$a = a_j \text{ mód } p_j^{\alpha_j}, \quad j = 1, \ldots, m$$

donde recordemos que m es el número de factores primos en los que se descompone el número n.

Antes de continuar, definamos la notación que se utilizará. Sean, para cada $j = 1, \ldots, m$, r_j el orden de cada elemento a_j módulo $p_j^{\alpha_j}$, y 2^{d_j} la mayor potencia de 2 tal que divide al orden r_j. Definimos de igual forma 2^d como la mayor potencia de 2 que divide al orden r.

Entonces, para que se cumpla una de las dos condiciones de (2.7), es decir, o bien que r es impar o, en su defecto, que $x^{r/2} = -1 \mod n$, es necesario que todos los elementos d_j posean el mismo valor. Por el lema previo aplicado a cada a_j, tenemos que la probabilidad de que d_j tome un valor correcto es de $\frac{1}{2}$, y, al tener m elementos, la probabilidad conjunta de que todos los d_j coincidan es de $\frac{1}{2^m}$.

Para completar la demostración, simplemente nos faltaría comprobar que los d_j solo pueden tomar los valores d o 0. Para probarlo, distinguiremos dos casos:

- Si r es impar, entonces, para cada $j = 1, \ldots, m$ se tiene que r_j divide a r, $r_j \mid r$. Por tanto, podemos saber que r_j es impar para todo j, y $d_j = 0$, $\forall j$, por definición.

- Si r es par, entonces $a^{r/2} = -1 \text{ mód } n$, pues recordemos que estamos estudiando la probabilidad (2.7). En ese caso, $a^{r/2} = -1 \text{ mód } p_j^{\alpha_j}$ para cada $j = 1, \ldots, m$. Deduciríamos en este caso que $r_j \nmid \frac{r}{2}$, y como $r_j \mid r$ se tiene $d_j = d$, $\forall j$, y queda probado.

□

Con este resultado, podemos garantizar que si n factoriza en al menos 2 factores primos, cada vez tendremos un 75 % de probabilidades de que se cumplan las condiciones que, como veremos luego, son necesarias para que el algoritmo de Shor funcione. Con ello, podemos concluir que basta con repetir el algoritmo 3 veces para que la probabilidad de obtener el resultado correcto al menos una vez esté por encima del 98 %.

2.2.2. Puerta de Hadamard

Las puertas cuánticas son una extensión natural de las puertas clásicas, al igual que los circuitos cuánticos, que son una extensión de los circuitos físicos

clásicos. En este apartado nos centraremos en las puertas cuánticas, especialmente en la puerta de Hadamard, la más importante y necesaria para desarrollar teóricamente el Algoritmo de Shor.

En realidad, la definición de puertas cuánticas es muy simple, ya que son simplemente operaciones que pueden ser descritas por matrices ortogonales y que operan sobre uno o más *qubits* para alterar su estado. La clave de estas puertas es que operan según las leyes de la mecánica cuántica, por lo que fenómenos como la superposición o el entrelazamiento pueden ocurrir al utilizar estas puertas.

En concreto, la puerta de Hadamard opera sobre un solo *qubit*, y es una de las puertas cuánticas más útiles e importantes que existen, ya que al aplicarla a cualquiera de los estados de la base computacional, definida en (2.2), se genera una superposición de estados. La puerta de Hadamard está definida por el operador H de la siguiente forma

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{2.8}$$

Como hemos mencionado previamente, la importancia de esta puerta radica en la superposición que se consigue al aplicarla sobre los estados $|0\rangle$ y $|1\rangle$,

$$H(|0\rangle) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \tag{2.9}$$

$$H(|1\rangle) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{2.10}$$

Nótese que al realizar el cuadrado a la matriz H, obtenemos la identidad, $H^2 = I$, por lo que la matriz H es su propia inversa. Esto quiere decir que si aplicamos el cuadrado de esta puerta a cualquier estado, este no se modifica.

2.2.3. Transformada cuántica de Fourier

La transformada cuántica de Fourier (denotada como QFT, por sus siglas en inglés provenientes de *Quantum Fourier Transformation*) usa el paralelismo cuántico para adaptar la transformada discreta de Fourier, de forma que esta pueda aplicar sobre estados en lugar de sobre números complejos. Recordemos que el caso discreto de la transformada de Fourier normalmente se escribe como la transformación de un conjunto $x_0, \ldots, x_{N-1}$ de N números complejos en otro conjunto de números complejos $y_0, \ldots, y_{N-1}$ definido por

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i j k/N} x_j. \tag{2.11}$$

Esta transformada tiene una gran cantidad de aplicaciones en diferentes ramas de la ciencia, de hecho, la resolución de ciertos problemas suele ser más sencilla planteando la versión transformada de Fourier, causa principal por la que la utilizaremos en el Algoritmo de Shor.

Las puertas de Hadamard, previamente explicadas en el apartado 2.2.2, son un caso de generalización de las transformadas de Fourier, y además de ello, son la base sobre la que se construye la transformada cuántica. De hecho, para un único *qubit*, la transformada cuántica de Fourier es matemáticamente equivalente a la puerta H. Además, la construcción de la matriz correspondiente a la transformada cuántica de Fourier, sigue una fórmula recursiva similar a la utilizada para construir $H^{\otimes n}$, donde $H^{\otimes n}$ denota el producto tensorial de n matrices de Hadamard.

A partir de este momento, trabajaremos en este apartado en un sistema formado por n *qubits*.

Definición 2.2.4. Se define el operador de la *transformada cuántica de Fourier* (QFT_n) como

$$|j\rangle \longmapsto \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i j k/2^n} |k\rangle, \tag{2.12}$$

donde $0 \leq j \leq 2^n - 1$.

Se puede demostrar que esta transformación es unitaria. Además, si escribimos su acción sobre una superposición de estados,

$$\sum_{j=0}^{2^n-1} x_j |j\rangle \longrightarrow \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} \left[\sum_{j=0}^{2^n-1} e^{2\pi i j k/2^n} x_j \right] |k\rangle = \sum_{k=0}^{2^n-1} y_k |k\rangle,$$

vemos que corresponde a una notación vectorial para la transformada de Fourier discreta (2.11) en el caso $N = 2^n$.

2.2.4. Estimación cuántica de fase

La transformada de Fourier, explicada en el apartado anterior 2.2.3, es la clave de un proceso general conocido como estimación de fase, que es la llave gracias a la cual funcionan muchos algoritmos cuánticos, entre ellos, el Algoritmo de Shor, que es el que nos concierne.

Se supone un operador unitario U que tiene un vector propio $|u\rangle$, cuyo valor propio correspondiente es $e^{2\pi i \varphi}$, donde el valor de φ es desconocido, de hecho, estimar este valor es el objetivo de esta subrutina cuántica. La estimación cuántica de fase utiliza dos registros. El primero de ellos tiene t *qubits* que están inicialmente en el estado $|0\rangle$, mientras que el segundo registro comienza en el

estado $|u\rangle$, y contiene tantos *qubits* como sean necesarios para almacenar este estado. La elección de t depende de dos factores: el número de dígitos que queremos en la estimación de φ y la probabilidad que queremos que tenga la estimación de fase para funcionar correctamente, lo que nos dice que la elección de este valor t será clave.

La subrutina de la estimación de fase se puede dividir en tres etapas. La primera de ellas consiste en aplicar la puerta de Hadamard al primer registro, seguida de la aplicación de operadores basados en U al segundo registro. Estos operadores estarán formados por el operador U elevado a las sucesivas potencias de 2, es decir, $U^{2^j},\ j = 0, 1, \ldots, t-1$. Utilizando las propiedades de los vectores propios, por ser $|u\rangle$ vector propio de U, es sencillo ver que, al terminar la primera etapa, el primer registro será:

$$\frac{1}{\sqrt{2^t}}\left(|0\rangle + e^{2\pi i 2^{t-1}\varphi}|1\rangle\right)\left(|0\rangle + e^{2\pi i 2^{t-2}\varphi}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 2^{0}\varphi}|1\rangle\right) =$$

$$= \frac{1}{\sqrt{2^t}}\sum_{k=0}^{2^t-1} e^{2\pi i\varphi k}|k\rangle. \tag{2.13}$$

mientras que el segundo registro no se verá modificado. Todo ello se puede observar en la Figura 2.1, obtenida de [23], donde se han omitido los factores $1/\sqrt{2}$ por simplicidad.

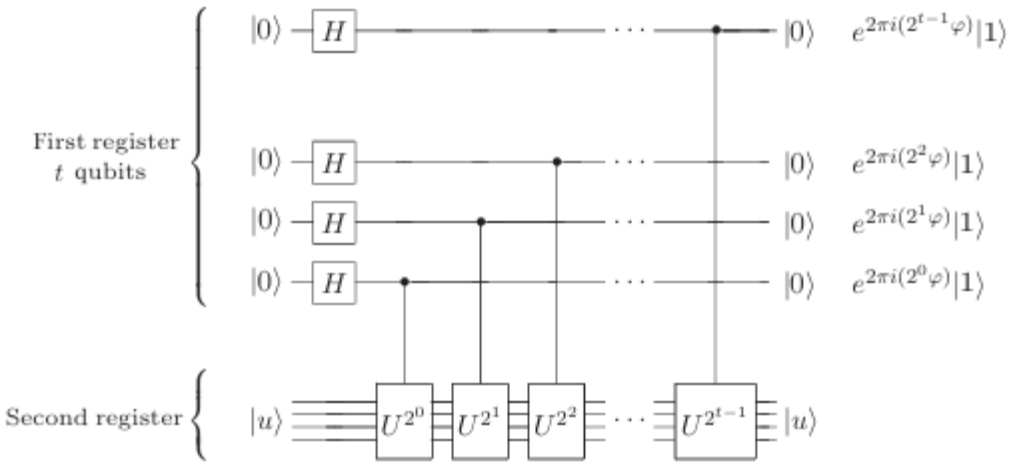

Figura 2.1: Primera etapa de la estimación cuántica de fase [23].

La segunda etapa de esta subrutina consiste en aplicar la inversa de la transformada cuántica de Fourier, que se obtiene invirtiendo la transformada definida

en (2.12), en el apartado anterior. Por último, la tercera etapa consistirá en leer el estado del primer registro para obtener la estimación cuántica buscada para φ, para lo que haremos una medición en la base computacional.

En resumen, la estimación de fase permite, como su propio nombre indica, estimar la fase φ de un valor propio de un operador unitario U, dado el correspondiente vector propio $|u\rangle$. Una de las partes clave en este procedimiento es la capacidad de la transformada de Fourier inversa para realizar la siguiente transformación

$$\frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} e^{2\pi i \varphi j} |j\rangle |u\rangle \rightarrow |\hat{\varphi}\rangle |u\rangle, \tag{2.14}$$

donde $|\hat{\varphi}\rangle$ denota un estado que es un buen estimador de φ cuando se mide.

2.3. Algoritmo de Shor

El algoritmo de Shor tiene como base una combinación de conceptos matemáticos y principios de la computación cuántica. Al igual que el resto de algoritmos de computación cuántica, es probabilístico; por tanto, no nos proporcionará la respuesta correcta en todos los casos. Aún así, varios estudios han indicado que, repitiendo el algoritmo 10 veces, se obtendría una solución con el 99 % de probabilidad. En esta sección, se proporciona una descripción más detallada de los pasos que sigue el algoritmo, combinando los conceptos matemáticos de los algoritmos tradicionales de factorización con la parte cuántica esencial para poder garantizar el funcionamiento del algoritmo.

El objetivo del Algoritmo de Shor es el mismo que el de los algoritmos tradicionales de factorización, de hecho, los pasos seguidos son muy similares. A continuación, haremos un recorrido paso por paso por los algoritmos tradicionales de factorización, pero deteniéndonos en un punto clave, la obtención del periodo r, cuando será necesario aplicar la parte cuántica del Algoritmo de Shor, con el fin de reducir la complejidad computacional de los algoritmos tradicionales de forma significativa.

Primera parte, tradicional

Para ilustrar cómo funciona el algoritmo de Shor, supongamos que tenemos un entero N y queremos factorizarlo como el producto de dos enteros p y q. Antes de comenzar, realicemos algunas suposiciones que simplifiquen el algoritmo:

- N **es impar:** si N fuera par, entonces 2 sería trivialmente un factor.
- N **no es potencia de primo:** existen algoritmos de factorización clásicos que resuelven este problema de forma muy eficiente.

Teniendo entonces que N no es par ni potencia de primo, comenzaremos el algoritmo eligiendo un número a tal que $1 < a < N$. Descartaremos otro caso trivial realizando el $\text{mcd}(a, N)$ mediante el algoritmo de Euclides. Si en este caso obtenemos un factor no trivial, es decir, o bien $\text{mcd}(a, N) \neq 1$ o $\text{mcd}(a, N) \neq N$, entonces podemos tomar como factores

$$p = \text{mcd}(a, N), \quad q = \frac{N}{\text{mcd}(a, N)}$$

En caso de no obtener un factor no trivial, tenemos que a y N son coprimos, y por tanto tendremos que continuar con el algoritmo.

Segunda parte, cuántica

En este punto comienza la parte cuántica del algoritmo de Shor, ya que el siguiente paso es encontrar el orden de a de módulo N, es decir, el entero r tal que

$$a^r \equiv 1 \text{ mód } N, \tag{2.15}$$

pero el proceso de encontrar ese valor r no es sencillo, de hecho, no es factible para la computación tradicional por su alto coste computacional. En cambio, con la computación cuántica parece factible resolver este problema, gracias a los pasos siguientes, que son la clave del algoritmo estudiado.

Definimos la función

$$f(x) = a^x \text{ mód } N, \tag{2.16}$$

con $x \in \mathbb{Z}_{>0}$ y a el valor elegido con anterioridad. Es sencillo ver, gracias a los resultados presentados en la sección anterior, que esta función es periódica, y además su periodo es precisamente el valor r que buscamos, que es el orden de a de módulo N. Esta función periódica es clave, y explotaremos sus propiedades para obtener el periodo r. Para ello, se seguirán los pasos previamente marcados en el apartado 2.2.4, donde se explicó la subrutina de la estimación cuántica de fase.

Primero, se considera un sistema compuesto por dos registros cuánticos: el primero con t *qubits* y el segundo con n *qubits*. El estado inicial del sistema se escribe como:

$$\frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle|0\rangle,$$

donde el primer registro se prepara en una superposición uniforme mediante la aplicación de puertas de Hadamard a cada uno de los t *qubits*.

A continuación, se aplica al sistema el operador unitario U, que actúa sobre los registros según la definición:

$$U|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle.$$

y aplicándolo al sistema, obtenemos:

$$\frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle|f(x)\rangle = \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{x=0}^{2^t-1} |x\rangle|\hat{f}(s)\rangle \quad .$$

El siguiente paso consiste en aplicar la transformada cuántica de Fourier inversa al primer registro. Esto permite extraer información del periodo r a partir de las propiedades de interferencia cuántica. El estado resultante del primer registro, tras la transformada de Fourier, se describe como:

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\widehat{s/r}\rangle|\hat{f}(s)\rangle,$$

donde $\widehat{s/r}$ es una estimación de la fase, eligiendo s aleatoriamente. La incorporación de la transformada cuántica de Fourier no solo amplía las posibilidades en la manipulación de estados cuánticos, sino que también proporciona un método robusto para identificar patrones periódicos complejos que son intratables mediante métodos clásicos.

Para obtener definitivamente r, realizaremos la última etapa que explicamos en la estimación cuántica de fase, en la que se mide el primer registro para obtener $\widehat{s/r}$, y haciendo uso de la expansión en fracciones continuas, en la que tampoco entraremos en detalle, se obtiene el periodo r que buscábamos. En este punto, aunque lo detallaremos más adelante, podemos descubrir si nuestro algoritmo nos va a dar un resultado válido o no. Para ello, basta con fijarse en el valor r obtenido, si este valor es par, entonces obtendremos una factorización válida para N; en cambio, si r es impar, no podremos continuar con el siguiente paso y tendremos que reelegir nuestro valor a inicial y comenzar de nuevo.

Tercera parte, tradicional

Una vez terminada la parte cuántica del algoritmo de Shor, volvemos al apartado tradicional para completar el algoritmo y obtener la factorización del número N. Recordemos que en este punto ya conocemos el valor r tal que se cumple (2.15) que recordemos que decía que

$$a^r \equiv 1 \text{ mód } N,$$

de donde deducimos que $N \mid a^r - 1$. Reescribiendo esta condición, obtenemos:

$$N \mid (a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) \quad . \tag{2.17}$$

Es importante destacar que la condición (2.17) no implica que N divida a uno de los dos factores, de hecho, pueden darse tres casos que trataremos a continuación. Pero antes de ello, cabe destacar que en este punto encontramos el

caso en el que el algoritmo no da resultado, que es cuando $r/2$ no es un número entero, es decir, si r no es par. Si se da esta situación deberíamos volver al punto donde elegimos el valor a de forma aleatoria y elegir otro número diferente para volver a probar. Como ya mencionamos al inicio, este caso no se debería dar en repetidas ocasiones, al ser la probabilidad de que el valor sea par muy alta, como vimos en el Teorema 2.2.1. Consideramos ahora los tres casos posibles:

1. $N \mid a^{r/2} - 1$.

   Si se diera este caso, entonces tendríamos que $a^{r/2} \equiv 1$ mód N, lo que es absurdo, pues por la definición de r sabemos que es el menor entero que satisface esa propiedad, por ser el orden de a módulo N. Por tanto, este caso no se dará nunca.

2. $N \mid a^{r/2} + 1$.

   En estos dos últimos casos el estudio va a estar relacionado con el cálculo de $d = \text{mcd}(N, a^{r/2} - 1)$. En el caso 2, se cumple que $d = 1$, lo que implica que $a^{r/2} \equiv -1$ (mód N). Este caso, a diferencia del primero si se puede dar, pero no nos ofrece ninguna solución válida, por tanto estaríamos en otro de los remotos casos en los que el algoritmo no nos brinda ninguna solución, ya que no podríamos encontrar un factor no trivial de N. En este caso deberíamos volver a elegir otro valor diferente de a y repetir el algoritmo.

3. N divide el producto $a^{r/2} - 1$ y $a^{r/2} + 1$, pero no divide a ninguno de los dos individualmente.

   Finalmente, tenemos el caso en el que el algoritmo nos da una solución, ofreciéndonos factores no triviales de N. En este caso tenemos que $d = \text{mcd}(N, a^{r/2} - 1) \neq 1$, y d será un factor no trivial de N, siendo el otro N/d, de forma que habremos obtenido con éxito una factorización como estábamos buscando.

Con este último caso concluimos la explicación teórica del algoritmo de Shor, del cual se puede consultar un resumen gráfico en la figura 2.2, que se ha obtenido de [11]. A modo de conclusión, recordemos que la aparición de este algoritmo junto con los ordenadores cuánticos son la clave del posible fin de la criptografía clásica, ya que son capaces de resolver de forma muchísimo más rápida el problema de encontrar el periodo de la función f definida. La forma en la que pueden afectar a algunos algoritmos clásicos se detallará en un apartado posterior, en concreto 3.2.4, donde se comentará el caso concreto de RSA.

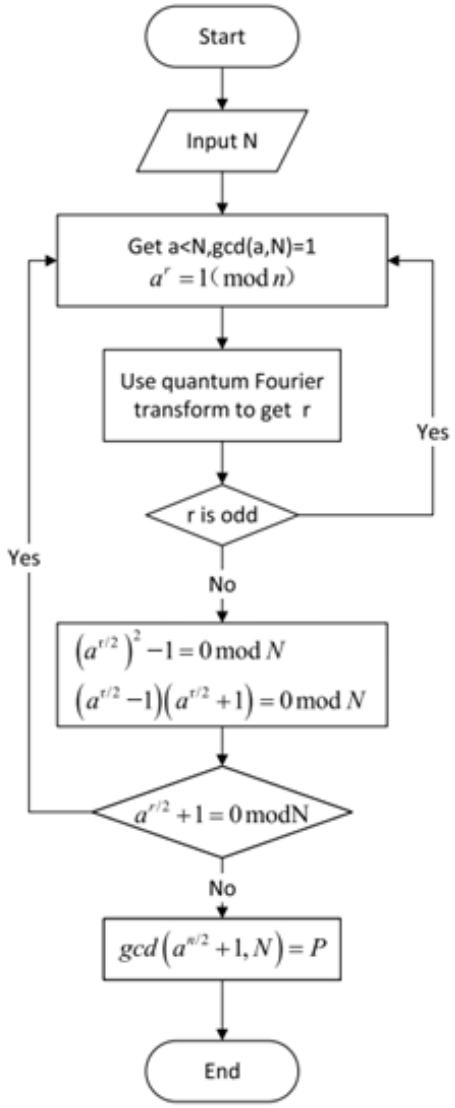

Figura 2.2: Esquema del algoritmo de Shor [11].

2.4. Algoritmo de Grover

El algoritmo de Grover [17] es un algoritmo de búsqueda exhaustiva en conjuntos no ordenados. Si un ordenador necesita, utilizando un algoritmo clásico, N operaciones para encontrar un elemento objetivo, con el algoritmo de Grover, un ordenador cuántico, lograría una aceleración cuadrática, requiriendo tan solo $\sqrt{N}$ operaciones. Aunque el crecimiento sigue siendo exponencial, el número de

operaciones se ve reducido considerablemente utilizando este algoritmo.

Veamos, de forma resumida, el funcionamiento de este algoritmo. El algoritmo parte de una secuencia desordenada con N elementos. Las posiciones de la secuencia se identifican con los autoestados $|0\rangle, |1\rangle, \ldots, |N-1\rangle$, y cada uno de ellos tiene un autovalor asociado $\lambda_0, \lambda_1, \ldots, \lambda_{N-1}$.

Se define un operador cuántico U_ω que actúa como un oráculo: transforma el estado buscado $|\omega\rangle$ en su opuesto, $-|\omega\rangle$, y deja los demás sin cambio. Este operador se representa como

$$U_\omega = I - 2|\omega\rangle\langle\omega|,$$

donde I es el operador identidad (la matriz identidad de dimensión N). Desde el punto de vista espectral, este operador tiene autovalor -1 para el estado $|\omega\rangle$ y autovalor $+1$ para todos los demás estados $|x\rangle$ con $x \neq \omega$. En otras palabras, U_ω refleja respecto al hiperplano ortogonal a $|\omega\rangle$.

Su objetivo es marcar el estado que cumple el criterio de búsqueda, para amplificar su probabilidad en pasos posteriores.

Una vez completada la inicialización del algoritmo, los pasos que hay que realizar son los siguientes.

1. Se inicializa el sistema en una superposición uniforme de todos los estados:

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle.$$

2. Se repite r veces (r será definido posteriormente) la siguiente secuencia de operaciones:

   - Aplicar el operador U_ω, que invierte la fase del estado buscado.
   - Aplicar el operador

$$U_s = 2|s\rangle\langle s| - I,$$

     llamado difusor de Grover, donde I es el operador identidad. Este operador actúa como una reflexión respecto al estado $|s\rangle$. Es decir, tiene autovalor $+1$ para el estado $|s\rangle$, y -1 para todos los vectores ortogonales a él. Su efecto es invertir las amplitudes respecto al promedio, lo que amplifica la probabilidad del estado marcado en combinación con el oráculo.

   Cada iteración incrementa la amplitud del estado correcto $|\omega\rangle$, haciendo más probable que aparezca como resultado de la medición final.

3. Finalmente, se realiza la media. Con alta probabilidad, el resultado será el autovalor λ_ω, correspondiente al estado $|\omega\rangle$ buscado. Si tomamos N grande, podremos obtener ω a partir de λ_ω.

El valor de r para obtener el resultado buscado debe de ser de, aproximadamente,

$$r \approx \left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor$$

iteraciones. Con ese valor de r, la probabilidad de medir el estado $|x_0\rangle$ se aproximará a 1.

Este algoritmo no convierte el problema de búsqueda en un problema polinomial, ya que la complejidad sigue siendo exponencial en muchos casos prácticos. Sin embargo, representa una mejora significativa frente a la búsqueda clásica, al reducir el número de operaciones de $O(N)$ a $O(\sqrt{N})$, lo que lo convierte en el algoritmo óptimo para la búsqueda no estructurada sobre conjuntos no ordenados en el paradigma cuántico.

Aunque Grover no permite romper criptografía asimétrica como lo hace el algoritmo de Shor, sí representa una amenaza para algoritmos simétricos y funciones hash. En particular, reduce la seguridad de una búsqueda exhaustiva sobre una clave de k bits de $O(2^k)$ a $O(2^{k/2})$. Por esta razón, en escenarios post-cuánticos se recomienda duplicar el tamaño de las claves simétricas para mantener niveles equivalentes de seguridad.

Capítulo 3

Introducción a la criptografía

En los últimos tiempos, la criptografía ha tomado cada vez más fuerza por su importancia para la seguridad de la información, con el objetivo de proteger la confidencialidad, integridad y autenticidad de los datos. Esta disciplina es una de las ramas de la criptología, ciencia que estudia los sistemas criptográficos y que se completa con el criptoanálisis. Mientras que la criptografía se encarga de implementar los criptosistemas, el criptoanálisis, aunque suene extraño, tiene como función romperlos. Estos sistemas tienen la necesidad de ser lo más robustos posible, y la mejor manera de probarlo es intentar realizar ataques contra ellos para descubrir sus vulnerabilidades.

El auge de la computación cuántica ha causado que los sistemas criptográficos antiguos, tradicionalmente basados en la dificultad para resolver ciertos problemas matemáticos, como la factorización de números primos o el logaritmo discreto, enfrenten un desafío significativo al abrirse la posibilidad de que la complejidad de esos problemas, tradicionalmente exponencial, se vea reducida con estos nuevos ordenadores. Con la aparición de algoritmos cuánticos como el Algoritmo de Shor, explicado previamente en el apartado 2.3 y que compromete la seguridad de estos sistemas, se ha impulsado la investigación de nuevas técnicas de criptografía resistentes a este tipo de algoritmos, la criptografía post-cuántica. En este contexto, resulta crucial estudiar tanto los fundamentos de la criptografía clásica como las innovaciones en este campo, aspectos que trataremos en este capítulo.

3.1. Conceptos básicos

Podemos resumir el funcionamiento de un sistema criptográfico en su caso más simplificado con la siguiente definición. Esta será la base sobre la que desarrollemos el contexto en el que vamos a trabajar en los siguientes apartados.

Definición 3.1.1. Un *criptosistema* o *sistema criptográfico* puede ser definido por una tupla (M, C, K, E, D), donde:

1. M es un conjunto finito formado por los mensajes en texto plano, denominado "espacio de texto plano".

2. C es un conjunto finito formado por los mensajes cifrados, conocido como "espacio de texto cifrado".

3. K es el "espacio de claves", un conjunto finito de claves.

4. $E = \{e_k : k \in K\}$ es el conjunto de las funciones de cifrado. Para cada $k \in K$, se define e_k de la forma

$$e_k : M \longrightarrow C,$$

que envía un elemento cualquiera de M a un elemento de C. e_k se conoce como la función de encriptado o cifrado para la clave k.

5. $D = \{d_k : k \in K\}$ es el conjunto de las funciones de descifrado. Para cada $k \in K$, se define d_k de la forma

$$d_k : C \longrightarrow M,$$

que envía un elemento cifrado de C a un elemento de M. d_k se conoce como la función de descifrado para la clave k.

Además, debe cumplirse que para cada $k_1 \in K$, existe $k_2 \in K$ tal que

$$D_{k_1}(E_{k_2}(m)) = m, \ \forall m \in M$$

Cabe destacar que esta definición suele ser modificada para distinguir los dos tipos de criptosistemas que existen, de clave pública y privada. En el siguiente apartado, nos centraremos en los sistemas criptográficos de clave pública, que son los que nos conciernen para este trabajo, pero antes vamos a dar unas pequeñas nociones de los criptosistemas de clave privada, que son los primeros que surgieron. La principal diferencia entre estos dos tipos de criptosistema es el uso de las claves pública y privada. Los sistemas de clave privada, también conocidos como criptosistemas simétricos, utilizan la misma clave tanto para el cifrado como para el descifrado, es decir, en la definición 3.1.1 tendríamos que necesariamente para cada $m \in M$, $k_1 = k_2$. En cambio, los sistemas de clave pública, conocidos también como asimétricos, tienen la condición de que las claves pública y privada son diferentes, es decir $k_1 \neq k_2$ en nuestra definición.

Los sistemas criptográficos de clave privada son los más antiguos, siendo uno de los más famosos en la historia el sistema de encriptación César, utilizado por el propio Julio César durante el imperio Romano. Estos sistemas están pensados para la comunicación únicamente entre un emisor y un receptor, que comparten la misma clave para cifrar y descifrar. Esto hace que la clave deba ser compartida en secreto, de forma que nadie más pueda acceder a los mensajes además de ellos. Estos criptosistemas tienen una ventaja muy apreciada en el ámbito de la criptografía, que es la velocidad de cifrado y descifrado, inmejorable por cualquier sistema de clave pública, pero presentan muchos otros problemas:

- **La distribución de claves:** Los dos usuarios que participan en la comunicación tienen que encontrar un canal secreto y seguro para compartir la clave, ya que cualquiera que la intercepte podrá descifrar todos los mensajes de la conversación e incluso suplantar la identidad de alguno de los usuarios enviando mensajes cifrados con esa clave.

- **Gestión de claves:** En caso que la comunicación sea en una red de n usuarios, cada par de usuarios tiene que tener su clave compartida particular, lo que implica un total de $\frac{n(n-1)}{2}$ claves, que sumado al primer problema puede ser potencialmente peligroso para una comunicación segura.

- **No hay firma digital:** La firma digital, en la que profundizaremos más adelante, es el equivalente a las firmas manuales en el caso digital. Generalmente, con la criptografía de clave privada no existe una forma de firmar los mensajes, por el hecho de que la clave la conocen al menos dos usuarios.

El afán por superar estos tres inconvenientes fue la principal razón por la que surgió la criptografía de clave pública, propuesta por W. Diffie y M.E. Hellman en *New Directions in Criptography* [10]. La idea de esta nueva criptografía era permitir un intercambio seguro de mensajes entre emisor y receptor sin necesidad de intercambiar previamente una clave común, de forma que se solucionaban con un simple cambio todos los problemas anteriormente presentados.

3.2. Criptografía de clave pública

Los criptosistemas de clave pública, a diferencia de los de clave privada, están pensados para una red de usuarios, más que solo para que dos interlocutores intercambien mensajes. En este tipo de criptosistemas, como hemos mencionado antes, se hace una diferencia entre la clave utilizada para cifrar y la que se

utiliza para descifrar. Por ello, cada usuario u tendrá asociado un par de claves $< P_u, S_u >$, generado por un algoritmo de generación de claves, donde:

- P_u es la clave pública del usuario u, que se publica en un directorio público y es conocida por cualquier persona.
- S_u es la clave privada del usuario u. En este caso, solo debe conocerla el propio u.

Supongamos ahora que queremos enviarle el mensaje m al usuario u. Entonces el procedimiento sería el siguiente:

1. Buscar la clave pública del usuario receptor, es decir, buscamos P_u en el directorio de claves públicas.
2. Calculamos el mensaje cifrado, para ello, haciendo uso de un algoritmo público de cifrado que denotaremos por e_{P_u} y que aplicado a m

$$e_{P_u}(m) = c$$

3. Enviamos el mensaje c al usuario u.

Una vez recibido el mensaje, ningún usuario será capaz de extraer la información que contiene el mensaje c recibido, ni siquiera u, ya que estará cifrado. Para poder comprender el mensaje recibido será necesario descifrar el mensaje, lo que en este caso será exclusivo para el usuario u, por ser el único que conoce su clave privada S_u. Esta clave es necesaria para deshacer la acción realizada por P_u, y, para ello, se aplicará un algoritmo de descifrado d_{S_u} tal que:

$$d_{S_u}(c) = d_{S_u}(e_{P_u}(m)) = m. \tag{3.1}$$

Entonces el usuario u podrá leer sin problema el mensaje que otro usuario le ha enviado. Es importante destacar que, para que esto funcione, la función de cifrado e_{P_u} y la de descifrado d_{S_u} tienen que cumplir (3.1). Con esto se quiere enfatizar en la importancia del algoritmo de generación de claves escogido, ya que debe garantizar que existen funciones de cifrado y descifrado que permitan que el sistema funcione correctamente.

3.2.1. Funciones unidireccionales y trampa

Las funciones de cifrado y descifrado tienen que ser unas funciones especiales, ya que la acción de la primera no debe poder deshacerse fácilmente. Aún así, la función de descifrado es necesaria, por lo que debe existir al menos una forma de conseguir que se revierta la acción tomada, para poder recuperar el mensaje enviado.

Definición 3.2.1. Las *funciones unidireccionales*, también conocidas como funciones de una vía, funciones sin retorno o *one-way functions* en inglés, son funciones $f : A \longrightarrow B$ tales que:

1. Admiten inversa, es decir, existe $f^{-1} : B \longrightarrow A$ tal que:

$$f \circ f^{-1} = id_B,\ f^{-1} \circ f = id_A$$

   donde id es la función identidad.

2. La imagen directa se calcula de forma eficiente, es decir, para cada $a \in A$ es sencillo encontrar $f(a) = b$, con $b \in B$.

3. La imagen inversa es computacionalmente muy costosa de calcular, es decir, dado $b \in B$ no es sencillo encontrar un $a \in A$ tal que $f^{-1}(b) = a$.

Normalmente, estas funciones suelen basarse en problemas matemáticos complejos (problemas NP-difíciles), como puede ser la multiplicación de números enteros (en contraposición a la complejidad de la factorización), o la exponenciación modular (contrarrestada con el logaritmo discreto).

Un caso particular de las funciones unidireccionales son las funciones trampa, que son las utilizadas para la criptografía de clave pública. La diferencia con el resto de funciones unidireccionales es que en este caso, si se tiene cierta información adicional, el cálculo de la inversa pasa de ser un problema difícil a solventarse de forma muy eficiente.

Definición 3.2.2. Sea $f : A \longrightarrow B$ una función unidireccional, es decir, que cumple las condiciones de la definición 3.2.1. Se dice que la función f es además una *función trampa*, trampilla o *trapdoor* (en inglés), si se conoce un certificado (clave privada) que permita calcular la inversa de la función de forma eficiente.

Ejemplo 3.2.1. Uno de los problemas difíciles que se suele utilizar para generar estas funciones trampa, y que está directamente relacionado con lo visto en apartados anteriores, es la factorización de números enteros. En el apartado 2.3 vimos un algoritmo para realizar esta factorización y vimos que era un problema costoso, imposible de resolver con un ordenador tradicional. Supongamos que tenemos un número entero n que es producto de primos, $n = p \cdot q$. En este caso, para factorizar n bastaría con hallar los primos p y q, y tendríamos que:

- **Función unidireccional:** multiplicar p y q para hallar n. Poco costoso para calcular.
- **Función inversa:** factorizar n, como hemos mencionado este es un problema difícil, cuya solución es imposible de encontrar en tiempo polinómico.

- **Función trampa:** si da como clave privada el valor p, entonces es sencillo factorizar n, ya que $q = n/p$. Este sería un problema muy eficiente para resolver si se conoce p.

El algoritmo RSA, que explicaremos más adelante, se basa en este problema.

3.2.2. Funciones hash

Además de las funciones unidireccionales y trampa mencionadas en el apartado anterior, tenemos otro tipo de funciones que son importantes en el ámbito de la criptografía: las funciones hash.

Definición 3.2.3. Una *función hash* es una función que, dado un mensaje m de longitud variable expresado como una cadena de bits, devuelve otra cadena de bits reducida de tamaño fijo. Normalmente, denominamos a esta cadena de bits de tamaño fijo como *valor hash del mensaje* y se representa por $H(m)$.

Las funciones hash también se conocen como función resumen, al ser como una huella del mensaje m. Uno de los usos más comunes es para verificar la autenticidad de un mensaje enviado; si el valor hash calculado no coincide con el mismo valor enviado por el emisor, significa que el mensaje ha sido modificado y no es auténtico.

Otra característica importante de las funciones hash es su longitud fija, es decir, hay un número finito de diferentes valores hash que es mucho menor que la variedad de mensajes que se pueden enviar. Esto causa un fenómeno que se denomina colisión, producido cuando dos mensajes diferentes tienen el mismo valor hash. Aunque parezca un hecho que podría complicar el uso de este tipo de funciones, no es así, ya que para funciones hash con suficiente cantidad de bits, la probabilidad de que esto ocurra es extremadamente baja, y no es un problema que preocupe en los casos reales. De hecho, las funciones hash suelen devolver valores hash de 128 bits, por lo que existen 2^{128} resúmenes posibles, y el número de intentos para encontrar dos mensajes con el mismo hash sería 2^{64}, que llevaría unos 600 milenios de computación con ordenador.

Pero las funciones hash verdaderamente interesantes son aquellas que no se pueden deshacer para obtener el mensaje, es decir, son unidireccionales, por el hecho de mantener la confidencialidad en la comunicación.

Definición 3.2.4. Una *función hash unidireccional* es una función hash (cumple la definición 3.2.3) que además, dado el valor hash de un mensaje m, es prácticamente imposible encontrar el otro mensaje que colisione con él, es decir, que tenga el mismo valor hash. En resumen, es una función hash que además es unidireccional (cumple, además, la definición 3.2.1 del apartado anterior).

3.2.3. Firmas digitales

La firma digital es uno de los descubrimientos más útiles de la criptografía moderna y supuso un gran avance, ya que los usuarios podían firmar mensajes de forma que cualquier persona pueda verificar que han sido enviados por ellos más tarde. La firma digital surgió de la mano de la criptografía de clave pública, ya que el par de claves (pública y privada) que tiene a su disposición cada usuario son las necesarias para firmar y comprobar que el mensaje ha sido enviado por el emisor en cuestión.

El funcionamiento consiste en que el usuario utilice su clave privada para firmar el mensaje, y, por la definición de las claves pública y privada, cualquiera podrá descifrar el mensaje, pero solo se podrá hacer utilizando la clave pública del usuario. De esta forma, cualquier receptor se queda convencido de que el mensaje enviado no ha sido alterado, al estar firmado y cifrado con la clave privada de su emisor. Además, el único usuario que puede haber enviado el mensaje es el propio emisor, y este no puede negar haber enviado el mensaje ya que es el único que conoce cuál es su clave privada.

Formalizaremos con las siguientes definiciones el proceso anterior. Supongamos a partir de ahora que cada usuario u tiene un par de claves, $< P_u, S_u >$, que serán su clave pública y privada respectivamente. Recordemos que, por la definición de estas claves, al utilizar una función de cifrado e y otra de descifrado d, para cualquier mensaje m, se tiene que:

$$d_{S_u}(e_{P_u}(m)) = m = d_{P_u}(e_{S_u}(m)) \tag{3.2}$$

que significa que podemos descifrar con la clave privada un mensaje cifrado con la clave pública (comunicación segura), o descifrar con la clave pública un mensaje cifrado con la clave privada (firma digital).

Definición 3.2.5. Se conoce como *firma digital* al valor s, que obtiene un usuario al cifrar con su clave privada un mensaje cualquiera m

$$s = e_{S_u}(m).$$

A esta acción la llamaremos *protocolo de firma.*

Definición 3.2.6. El *protocolo de verificación* es el protocolo inverso del anterior. En este caso, cualquier usuario puede descifrar un mensaje enviado por el usuario u utilizando su clave pública

$$m' = d_{P_u}(s).$$

Si se tiene que $m = m'$, entonces el mensaje no ha sido modificado y se dice que la firma s es válida, en caso contrario, diremos que la firma s es una firma no válida.

El problema de esta forma de firmar es que el usuario u tiene que filtrar el mensaje m que ha firmado para que se pueda verificar que ha sido él la persona que lo ha enviado, lo que hace que se pierda la confidencialidad de la comunicación. Para que esto no ocurra, se suele combinar la firma digital con las funciones hash, y en lugar de firmar el mensaje m, se firma el resumen de ese mensaje, calculado con una función hash, $H(m)$. De esta forma, se mantiene el objetivo de la firma digital, ya que podemos comprobar el verdadero autor del mensaje, y además no se vulnera la confidencialidad del mismo, ya que distribuir el valor $H(m)$ no aporta ninguna información sobre el mensaje enviado. El protocolo de firma y verificación es análogo para este caso, con la diferencia de sustituir los valores de m por $H(m)$.

3.2.4. Algoritmo RSA

La propuesta de Diffie y Hellman de 1976 [10], introducía la teoría de la criptografía de clave pública, pero no daba ningún ejemplo práctico. Dos años más tarde, en 1978, Rivest, Shamir y Adleman propusieron el primer algoritmo de cifrado de clave pública, el RSA, propuesto en el artículo *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems* [24]. En la propuesta de este algoritmo se sugería la siguiente generación de claves

1. Cada usuario u del sistema debe elegir dos números primos, que llamaremos p y q, y calcular su producto para obtener el número $n = p \cdot q$.

2. Al elegir estos números, el usuario u ha elegido trabajar en el grupo multiplicativo $\mathbb{Z}_n^\times$, cuyo orden se calcula con la función *phi* de Euler y es

$$\varphi(n) = \varphi(p \cdot q) = (p-1)(q-1),$$

   por ser p y q números primos. Para el usuario u es sencillo calcular este orden, ya que conoce p y q.

3. Seguidamente, el usuario u debe elegir un entero positivo e, de forma que cumpla las condiciones: $1 \leq e \leq \varphi(n)$ y $\text{mcd}(e, \varphi(n)) = 1$. Nótese que esta segunda condición significa que e debe de ser coprimo con $\varphi(n)$.

4. El usuario debe calcular ahora el inverso de e en $\mathbb{Z}_{\varphi(n)}$, es decir, d tal que

$$e \cdot d \text{ mód } \varphi(n) = 1,$$

   que se calcula utilizando el algoritmo de Euclides extendido, buscando d tal que

$$d \equiv e^{-1} \text{ mód } \varphi(n).$$

5. Una vez tenemos todos estos valores, estamos en disposición de distribuir la clave pública, que será el par (n, e), y de almacenar de forma privada la clave privada, que en este caso es d. Cabe destacar que los valores p, q y $\varphi(n)$ también deben mantenerse en secreto para no facilitar la filtración de la clave pública.

La "trampa" de este procedimiento está en que para calcular d es necesario conocer $\varphi(n)$, y la forma sencilla de hallarlo es factorizar n, que como sabemos es un problema difícil computacionalmente hablando, siempre que n sea un número suficientemente grande. Actualmente, se recomienda elegir los números p y q de más de 200 dígitos para garantizar la seguridad del algoritmo.

Una vez disponemos de la clave pública y privada, podemos definir las funciones de cifrado y descifrado para el usuario u:

- **Función de cifrado:** Para el usuario u tenemos:

  $$f_u : \mathbb{Z}_n \longrightarrow \mathbb{Z}_n$$

  definida por: $c = f_u(m) = m^e$ mód n, donde m es el mensaje que se quiere enviar sin cifrar, c será el mensaje cifrado, y e, n son los valores obtenidos de la clave privada de u.

- **Función de descifrado:** Para el usuario u tenemos:

  $$g_u : \mathbb{Z}_n \longrightarrow \mathbb{Z}_n$$

  definida por: $m = g_u(c) = c^d$ mód n, donde c es el mensaje que se ha recibido y está cifrado, m será el mensaje que el emisor quería enviar, y d es el valor obtenido de la clave pública de u.

Para verificar que el criptosistema está bien definido, tenemos que garantizar que, con la notación anterior, se cumple la definición de criptosistema, vista en la definición 3.1.1. Lo único que nos falta probar de esa definición es la condición final que deben cumplir las funciones de cifrado y descifrado, para lo que utilizaremos el siguiente teorema.

Teorema 3.2.1. *Sea u un usuario que tiene como clave pública al par (n, e) y como clave privada al valor d. Además, sean f_u y g_u las funciones de cifrado y descifrado respectivamente, definidas como se ha hecho previamente. Entonces, se verifica que para todo mensaje $m \in \mathbb{Z}_n$*

$$g_u(f_u(m)) = m. \tag{3.3}$$

Demostración:
Por la definición de las funciones de cifrado y descifrado se dan las siguientes igualdades

$$g_u(f_u(m)) = (m^e)^d \text{ mód } n = m^{e\cdot d} \text{ mód } n,\ \forall m \in \mathbb{Z}_n.$$

Por tanto, nos bastaría probar que:

$$m \equiv m^{e\cdot d} \text{ mód } n,\ \forall m \in \mathbb{Z}_n. \tag{3.4}$$

Por la elección de claves que hemos hecho antes, se tenía que $e\cdot d$ mód $\varphi(n) = 1$, es decir, existe un entero s tal que

$$e \cdot d = 1 + s \cdot \varphi(n).$$

Entonces, podemos aplicar el teorema de Euler-Fermat, que dice que si $\text{mcd}(m, n) = 1$, entonces $m^{\varphi(n)} \equiv 1$ mód n. En este caso, no tenemos asegurado que $\text{mcd}(m, n) = 1$, pero bastaría con que o bien $\text{mcd}(m, p) = 1$ o $\text{mcd}(m, q) = 1$, y se razonaría de forma muy similar. Los casos en los que no se cumple ninguno de estos casos no se utilizan en RSA por falta de seguridad; por tanto, esta demostración nos sirve para los casos a los que aplicaremos el algoritmo. Aplicando el mencionado teorema, tenemos que

$$\begin{aligned} m^{e\cdot d} &\equiv m^{1+s\cdot\varphi(n)} \text{ mód } n \\ &\equiv m \cdot m^{s\cdot\varphi(n)} \text{ mód } n \\ &\equiv m \cdot (m^{\varphi(n)})^s \text{ mód } n \\ &\equiv m \cdot (1)^s \text{ mód } n \equiv m \text{ mód } n \end{aligned}$$

que es lo que queríamos probar. Con ello, queda asegurado que RSA tal y como lo hemos definido es un criptosistema válido.

□

Para finalizar este apartado, veamos un ejemplo de cómo aplicaría este sistema criptográfico a un envío de un mensaje real. Trabajaremos con números primos pequeños en este caso, no recomendados para comunicaciones reales, pero más sencillos para poder apreciar el funcionamiento del criptosistema.

Ejemplo 3.2.2. Supongamos que elegimos los números primos $p = 1223$ y $q = 1987$. Entonces tenemos que

$$n = p \cdot q = 1223 \cdot 1987 = 2430101$$

y calculando la función *phi* de Euler

$$\varphi(n) = (p - 1)(q - 1) = 2426892.$$

Elegimos como valor $e = 948047$, y entonces nuestra clave pública, en este caso, será el par $(n, e) = (2430101, 948047)$. Se puede comprobar con algún algoritmo de forma eficiente que

$$\text{mcd}(e, \varphi(n)) = \text{mcd}(2430101, 948047) = 1.$$

Recordemos que, como hemos explicado previamente, el siguiente paso es calcular el inverso de e módulo $\varphi(n)$, es decir, d tal que

$$948047 \cdot d \equiv 1 \text{ mód } 2426892.$$

También existen algoritmos eficientes para realizar dicho cálculo, obteniendo que en este caso $d = 1051235$, que sería nuestra clave pública.

Supongamos ahora que queremos enviar el mensaje "RSA", que correspondería al escribirlo en base 26 con:

$$18 \cdot 26^2 + 19 \cdot 26^1 + 1 \cdot 26^0 = 12663,$$

luego el mensaje cifrado sería

$$c \equiv 12663^{948047} \text{ mód } 2430101,$$

y realizando los cálculos, obtenemos $c = 1111168$, que alfabéticamente se correspondería con: BKESF, porque:

$$1111168 = 2 \cdot 26^4 + 11 \cdot 26^3 + 5 \cdot 26^2 + 19 \cdot 26^1 + 6 \cdot 26^0.$$

Para concluir, el descifrado lo podríamos hacer como hemos descrito previamente:

$$m \equiv 1111168^{1051235} \text{ mód } 2430101$$

y obtendríamos que $m = 12663$, que alfabéticamente se corresponde con RSA, que es exactamente el mensaje que hemos enviado.

Ejemplo 3.2.3. Firma digital en RSA.

En este ejemplo veremos como podríamos firmar digitalmente un mensaje utilizando el sistema criptográfico de clave pública RSA. Para ello, veamos primero como sería el protocolo de firma explicado en el apartado 3.2.3 aplicado al caso concreto de RSA.

Supongamos que tenemos dos usuarios, A y B, y que el usuario A quiere enviar un mensaje, no necesariamente cifrado, al usuario B. El usuario A tendrá su clave pública formada por el par (n_A, e_A), y una clave privada d_A. Para firmar digitalmente un mensaje m que quiera enviar, el usuario A debe calcular la firma utilizando RSA de la forma:

$$s = m^{d_A} \text{ mód } n_A.$$

Una vez obtenido s, el usuario A podrá publicar el mensaje firmado, para ello difundirá la pareja (m, s), de manera que cualquier usuario, y concretamente el B, al que debía llegar el mensaje, podrá leerlo y verificar que su autor es el usuario A.

Veamos lo anterior con un ejemplo concreto, obtenido de [14]. Supongamos que el usuario A tiene como clave pública al par $(n_A, e_A) = (34121, 15775)$, y que su clave privada es $d_A = 26623$. Supongamos que A quiere publicar el mensaje "YES" firmado para que B lo pueda ver y verificar que es suyo. Entonces, el usuario A codificará primero ese mensaje en base 26, obteniendo que $m = 16346$, y posteriormente calculará su firma utilizando la clave privada de la siguiente forma:

$$s = m^{d_A} \text{ mód } n_A = 16346^{26623} \text{ mód } 34121 = 20904.$$

Decodificaremos ese valor de s para expresarlo alfabéticamente, para lo pasamos el mensaje a base 26 y obtenemos la firma "BEYA". Entonces, el usuario A podrá publicar el mensaje (YES, BEYA) y todo el que quiera podrá decodificar "BEYA" utilizando la clave pública de A para asegurarse que ese mensaje ha sido escrito por ese usuario en concreto.

Supongamos ahora que un usuario B cualquiera quiere verificar dicha firma. Para ello, primero deberá buscar en el directorio de claves públicas la que corresponde al usuario A, del que quiere verificar la firma, obteniendo así el par $(n_A, e_A) = (34121, 15775)$. El siguiente paso que hará B será codificar en base 26 ambos mensajes, obteniendo el valor numérico de ambos para poder aplicar el descifrado de RSA. Obtendrá entonces que el mensaje "YES" se corresponde con el valor $m = 16346$, y el mensaje firmado que era "BEYA" se corresponde con $s = 20904$. Utilizando ahora la clave pública de A, se aplica el método de descifrado de RSA y se obtiene:

$$m' = s^{e_A} \text{ mód } n_A = 20904^{15775} \text{ mód } 34121 = 16346.$$

El usuario B podrá comprobar que $m = m'$, y, con ello, verificar que el mensaje que ha visto ha sido, con total seguridad, enviado por A.

3.2.5. Problema de los sistemas vigentes

Cuando se encuentra una buena solución a un problema, como puede ser el caso de RSA para enviar y firmar mensajes cifrados, también aparecen varias vías para intentar acabar con esta solución, buscando minimizar su efectividad. En el caso del algoritmo RSA, han surgido variados ataques que complican un poco el funcionamiento de este algoritmo, pero ninguno de ellos ha conseguido que el algoritmo quede obsoleto.

La mayoría de ataques que se hacen contra RSA son fácilmente contrarrestados realizando una buena elección de los primos p y q, ya que todo el algoritmo

se basa en la dificultad de hallar la factorización dado $n = p \cdot q$. Para ello, basta con evitar los siguientes casos particulares:

- p y q deben diferir en pocos dígitos, pero no deben ser cercanos entre sí, es decir, se deben alejar lo máximo posible de $\sqrt{n}$.
- Ni $p-1$ ni $q-1$ deben tener sus factores primos pequeños, para evitar el ataque $p-1$ de Pollard.
- Lo mismo debe de ocurrir con $p+1$ y $q+1$, no deben tener sus factores primos pequeños.
- Por último, se pide que por seguridad $\text{mcd}(p-1, q-1)$ debe de ser pequeño.

Concluimos con esto que, con una buena elección de p y q, podemos sortear casi todos los ataques que buscan las debilidades de estos dos factores. También hay otras vías de ataque, que intentan explotar debilidades en las funciones de encriptado o en el exponente de descifrado d, pero todos ellos son poco eficientes y no obtienen resultados que puedan preocupar a los usuarios.

Con los datos anteriores, parece que el criptosistema RSA es una solución segura y válida que no presenta ningún problema, ya que la suposición inicial es que no se puede calcular, con la computación actual, la factorización de un número de forma sencilla. Los algoritmos que se pueden utilizar para ello, como el Algoritmo de Shor, explicado anteriormente en el apartado 2.3, tienen partes que son computacionalmente muy costosas y prácticamente irresolubles con la tecnología conocida. Al menos así era hasta que apareció una nueva vía de ataque, la criptografía post-cuántica, en la que se plantea la posibilidad de realizar estos cálculos utilizando ordenadores cuánticos.

Como ya explicamos en el capítulo 2 de este trabajo, la computación cuántica surgió a finales del siglo XX, y su aparición brindó una gran cantidad de dudas sobre la seguridad de la criptografía tal y como se conocía hasta el momento. La computación cuántica abría la posibilidad de convertir problemas que antes eran considerados computacionalmente imposibles de resolver, en problemas que se podían abordar de forma eficiente. Para ello, simplemente era necesario construir un ordenador con una cantidad concreta de *qubits*, hecho que no ha ocurrido todavía pero que, como se explicaba en la introducción, podría suceder en los próximos años.

Concretamente, el problema de factorizar un número en el producto de dos primos se puede solucionar con el Algoritmo de Shor, y un ordenador con los *qubits* necesarios podría hacerlo en tiempo récord. Esto haría que la mayoría de sistemas criptográficos antiguos queden inutilizables por su vulnerabilidad, entre los que estaría, por ejemplo, RSA, algoritmo explicado previamente. Este cambio en el panorama de la criptografía hizo que surgieran nuevas corrientes de

estudio, con el fin de buscar criptosistemas que además fueran resistentes a los ordenadores cuánticos. En los próximos apartados daremos una visión general sobre como se plantea esta nueva rama de la criptografía, las diferentes vías de estudio que tiene, centrándonos en los capítulos siguientes en una de ellas (criptografía multivariante), y en concreto en un algoritmo propuesto como opción para llegar a ser el futuro de la criptografía.

3.3. Criptografía post-cuántica

La pregunta más obvia que nos surge después del problema planteado es qué criptosistemas se podrán usar una vez que se construyan ordenadores cuánticos. Además, cuando se encuentre un buen sistema de reemplazo, aún habrá problemas logísticos para cambiar todos los criptosistemas en uso, y llevará tiempo hacerlo, por lo que lo ideal sería hacerlo lo antes posible, para evitar filtraciones de información. Aparte de los criptosistemas, los datos más sensibles que se tengan almacenados deben mantenerse seguros incluso después de que se construyan los ordenadores cuánticos. Esto significa que cuando se disponga del nuevo criptosistema, tendremos que cifrar todos estos datos con el criptosistema resistente a ordenadores cuánticos, lo que supondrá todavía una mayor carga de trabajo.

Pero antes de pensar en todos estos cambios, lo ideal es elegir un criptosistema que sea resistente a ordenadores cuánticos y que nos ofrezca todas las ventajas de los criptosistemas utilizados actualmente como RSA. A continuación, presentaremos cuatro ramas de la criptografía post-cuántica que se han ido desarrollando en los últimos años con el fin de encontrar el criptosistema ideal. El objetivo de todas estas vías de estudio es encontrar un criptosistema que, obviamente, no se pueda romper con ningún algoritmo, y que además sea resistente a ordenadores cuánticos y eficiente, combinación que podemos adelantar que se antoja complicada de hallar.

3.3.1. Criptografía basada en funciones hash

Como ya explicamos en el apartado previo sobre firmas digitales, estas se han convertido en una tecnología clave para garantizar la seguridad de Internet y evitar la suplantación de identidad, entre otras ventajas que aportan. Las firmas digitales proporcionan principalmente autenticidad e integridad, siendo claves para cada vez más protocolos de identificación y autenticación. Por lo tanto, la existencia de algoritmos de firma digital seguros es crucial para mantener la seguridad informática.

Los algoritmos de firma digital que se utilizan en la práctica hoy en día

son RSA, DSA y ECDSA. Como ya hemos mencionado anteriormente, estos algoritmos no son seguros ante la aparición de la computación cuántica, ya que su seguridad depende de la dificultad de factorizar grandes números compuestos y de calcular logaritmos discretos. Como solución a esto, la criptografía basada en funciones hash es una de las formas más interesantes de construir firmas digitales resistentes a ataques cuánticos. Al igual que cualquier otro esquema de firma digital, estos esquemas utilizan una función hash criptográfica, y su seguridad se basa en la resistencia a colisiones de dicha función hash. De hecho, los esquemas de firma digital basados en hash, son seguros si y solo si la función hash utilizada es resistente a colisiones.

Los esquemas de firma basados en hash son los candidatos más importantes para la firma digital post-cuántica, ya que, aunque no existe una prueba formal de su resistencia a la computación cuántica, sus requisitos de seguridad son mínimos. Además, cada nueva función hash criptográfica genera un nuevo esquema de firma basado en hash. Por lo tanto, la construcción de esquemas de firma que sean seguros es independiente de problemas algebraicos difíciles, que son los problemas más afectados por la aparición de la computación cuántica. En el caso de las funciones hash, bastan las construcciones basadas en criptografía simétrica, lo que representa otra gran ventaja de los esquemas de firma basados en hash.

Estos esquemas de firma fueron inventados por Ralph Merkle [22]. Merkle partió del esquema de firma de un solo uso, en particular, de uno basado en los esquemas introducidos por Lamport y Diffie, que son los sistemas de firma más fundamentales que existen. La idea de Merkle fue intentar eliminar el principal problema que tenía este esquema, que era el límite de un solo uso por firma. Para ello, Merkle diseñó un árbol formado por muchas claves de la firma de un solo uso (las hojas del árbol), que se unen para formar una única clave pública (la raíz del árbol hash). De esta forma, la validez de la firma pasaría de estar limitada a un solo documento, a tener tantos usos como hojas del árbol dispongamos. La construcción inicial de Merkle no era suficientemente eficiente, especialmente en comparación con el esquema de firma RSA. Sin embargo, con el tiempo se han ido encontrado diversas mejoras, aunque sigue presentando algunos problemas como la limitación del número de firmas, que si se agotan hace que la clave de la raíz tenga que regenerarse. Aún así, las firmas basadas en hash son la alternativa más prometedora a los esquemas de firma RSA y de curvas elípticas.

3.3.2. Criptografía basada en códigos correctores

Otra de las ramas de la criptografía que ha acaparado gran parte de las investigaciones para encontrar sistemas resistentes a ordenadores cuánticos es la

criptografía basada en códigos correctores. Con este término, hacemos referencia a todos los criptosistemas basados en funciones unidireccionales que utilizan un código corrector de errores C. Estas funciones pueden consistir en añadir un error a una palabra de C o en calcular un síndrome en relación a una matriz de control de paridad de C.

El primer criptosistema de este tipo que surgió data de 1978, y todavía no se ha encontrado un ataque que pueda poner en riesgo el sistema, ni siquiera para ordenadores cuánticos. Este primer diseño es un esquema de cifrado de clave pública, propuesto por Robert J. McEliece [21], en el que la clave privada es un código de Goppa binario irreducible, generado aleatoriamente. La clave pública se construye a partir de dicho código, y es una matriz generadora aleatoria de una permutación del mencionado código de Goppa. El texto cifrado es un código al que se le han añadido algunos errores, que solo el propietario de la clave privada podrá eliminar fácilmente. Desde que apareció este criptosistema, se han incorporado algunas mejoras para aumentar su rendimiento, pero, como ya hemos mencionado, no ha aparecido ningún ataque que comprometa su seguridad.

El ejemplo previo de McEliece, al igual que todos los criptosistemas en general, busca un equilibrio entre la seguridad y la eficiencia. Los sistemas criptográficos basados en códigos correctores, y, en concreto, el caso del propuesto por McEliece, no tienen aplicaciones prácticas importantes hasta la fecha, posiblemente por los problemas que presentan en comparación con otros como RSA, o por la falta de necesidad de cambiar de criptosistema en el contexto en el que nos encontrábamos. En los próximos años, como hemos mencionado, puede que se dé la revolución en la criptografía, y podría ser la ocasión de estos criptosistemas para empezar a destacar como una de las principales opciones para sustituir los que se utilizan actualmente.

Como se ha mencionado en el apartado anterior, estos criptosistemas, como cualquier otro, presentan ciertos problemas. El más importante y que podría causar más inconvenientes es el tamaño de las claves públicas, que pueden llegar a ocupar desde 100 kilobytes a varios megabytes. Aún así, con la facilidad para procesar cada vez una mayor cantidad de información en menos tiempo, puede ser un problema que pierda importancia con los avances tecnológicos, mientras que las ventajas pueden hacerse cada vez más importantes.

Entre los puntos más destacados de estos criptosistemas está que son rápidos, ya que las funciones de cifrado y descifrado tienen una complejidad muy baja, y que las suposiciones sobre su seguridad son muy pocas, en concreto dos. La primera de ellas es que se supone que la dificultad de descifrar un código lineal aleatorio es alta; de hecho, es un problema bastante antiguo para el cual solo se han encontrado soluciones de complejidad exponencial. La segunda suposición está relacionada con los códigos de Goppa, usados para la clave privada, y es su

indistinguibilidad, problema no tan antiguo pero basado en la teoría algebraica de codificación, y supuestamente válido.

3.3.3. Criptografía basada en retículos

La criptografía basada en retículos (en inglés *lattice-based criptography*), es seguramente la que tiene un mayor potencial de las ramas de la criptografía post-cuántica mencionada. Entre las ventajas que nos puede ofrecer este tipo de criptosistemas, tenemos que tienen unas pruebas de seguridad muy sólidas, basadas en la dificultad de los peores casos; implementaciones relativamente eficientes y una escalabilidad excelente. Además, se considera que la criptografía basada en retículos es segura frente a ordenadores cuánticos.

Introduciremos de forma rápida y concisa este tipo de criptosistemas, y para ello, vamos a comprender rápidamente las bases sobre las que se sustenta. Lo primero que nos preguntamos es, ¿qué es un retículo?

Definición 3.3.1. Un *retículo* $\mathcal{L}$ de dimensión n es un conjunto de puntos en un espacio de n dimensiones con una estructura periódica. Formalmente, dados n vectores linealmente independientes $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^n$, el retículo generado por ellos es el conjunto de vectores:

$$\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_n) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z} \right\}.$$

donde los vectores $\mathbf{b}_1, \ldots, \mathbf{b}_n$ se conocen como una base del retículo.

La manera en que los retículos pueden utilizarse en criptografía no es en absoluto obvia y fue descubierta en un artículo revolucionario de Ajtai [1]. Gracias a este artículo, se ha armado una gran área de investigación alrededor de los retículos, centrando principalmente en el planteamiento de los problemas difíciles basados en retículos, base sobre la cual se construyen habitualmente estos criptosistemas.

El problema más básico en este ámbito es el SVP (*shortest vector problem*), en el que se da como entrada un retículo representado en una base arbitraria, y el objetivo es encontrar el menor vector distinto de cero en él. Por otro lado, el algoritmo más conocido y estudiado para problemas basados en retículos es el algoritmo LLL, que nos permite encontrar aproximaciones en tiempo polinomial para el problema de SVP.

Podríamos pensar que, al tener aproximaciones a soluciones para muchos de estos problemas en tiempo polinomial, esta no es una buena vía sobre la que seguir estudiando, pero nada más lejos de la realidad. Los algoritmos que proporcionan soluciones en tiempo polinomial dan resultados pobres y alejadas

de la realidad, y algoritmos que aproximan con mayor exactitud poseen una complejidad exponencial, lejos de poder ser calculados en un tiempo aceptable. Además, el Algoritmo de Shor no se puede aplicar en este caso, por tanto se consideran este tipo de problemas resistentes a ordenadores cuánticos, postulándose asi como unos candidatos idóneos sobre los que fundamentar el desarrollo de la criptografía post-cuántica.

3.3.4. Criptografía multivariante

La última de las posibilidades que destacamos de la criptografía post-cuántica es la criptografía multivariante. Esta es la rama en la que entraremos en más detalle, al ser el trabajo principalmente enfocado en explicar uno de los algoritmos más importantes de este tipo de criptografía. En este apartado daremos una comprensión general, a modo de introducción, profundizando más en este tema en los siguientes capítulos.

Un criptosistema de clave pública multivariante tiene un conjunto de polinomios cuadráticos sobre un cuerpo finito como clave pública. Este tipo de sistemas criptográficos están basados en la dificultad de resolver ecuaciones no lineales sobre un cuerpo finito, considerado un problema difícil. Este tipo de problemas no presenta ninguna solución fácil con la computación actual, y por estudios realizados parece ser una de las familias de criptosistemas que mejor resisten la evolución de los ordenadores cuánticos. Los avances en la criptografía multivariante han sido rápidos e intensos en las dos últimas décadas, a raíz, principalmente, de la aparición en escena de los ordenadores cuánticos. Tanta investigación ha hecho que se encuentren algunos algoritmos potencialmente resistentes, muchos de los cuales, con los años, se han ido rompiendo poco a poco, pero todavía quedan varios que soportan diferentes tipos de ataques.

Como ya hemos explicado previamente, los criptosistemas de clave pública se basan en la existencia de las ya mencionadas funciones "trampa". En el caso de la criptografía multivariante, diremos que es el estudio de todos los criptosistemas cuya función trampa toma la forma de una transformación basada en un sistema de ecuaciones cuadráticas en varias variables sobre un cuerpo finito. Normalmente, la clave pública se da con un conjunto de polinomios cuadráticos:

$$\mathcal{P} = (p_1(w_1, \dots w_n), \dots, p_m(w_1, \dots, w_n)),$$

donde cada p_k es un polinomio cuadrático no lineal en las variables $\mathbf{w} = (w_1, \dots, w_n)$. Entonces podemos definir:

$$z_k = p_k(\mathbf{w}) := \sum_i P_{ik} w_i + \sum_i Q_{ik} w_i^2 + \sum_{i>j} R_{ij} w_i w_j \tag{3.5}$$

donde todos los coeficientes (P_{ik}, Q_{ik}, R_{ik}) y las variables $\mathbf{w} = (w_1, \ldots, w_n)$ están en $\mathbb{K} = \mathbb{F}_q$, el cuerpo con q elementos. Tomando valores concretos para las variables y evaluando estos polinomios en dichos valores, encontraremos valores que se corresponden con el proceso de encriptado o de verificación. El problema a resolver sería, por tanto, la inversión del sistema cuadrático no lineal, que es equivalente a resolver un sistema de ecuaciones cuadráticas sobre un cuerpo finito. Este se conoce como el problema $\mathcal{MQ}$, que es un problema difícil y es el siguiente:

Problema $\mathcal{MQ}$: Resuelve el sistema $p_1(\mathbf{x}) = p_2(\mathbf{x}) = \cdots = p_m(\mathbf{x}) = 0$, donde cada p_k es cuadrático en $\mathbf{x} = (x_1, \ldots, x_n)$, con todos los coeficientes y variables en $\mathbb{K} = \mathbb{F}_q$, el cuerpo con q elementos.

La principal desventaja de los algoritmos basados en este tipo de criptografía es que, al realizar una elección aleatoria de un sistema de ecuaciones cuadráticas, lo más probable es que no tenga una función trampa asociada, y por tanto, ese sistema no pueda ser utilizado para construir un criptosistema. Por tanto, el planteamiento es diferente, ya que no estamos tratando con sistemas aleatorios o genéricos, sino sistemas específicos donde existen funciones trampa asociadas. Esto hace que la seguridad de los criptosistemas de este tipo no esté asegurada por la dificultad de resolver un problema difícil, sino que está enfocada en conseguir diseñar una función trampa que sea lo más segura posible, por lo que puede que existan ataques efectivos para cualquier función trampa escogida.

A modo de una primera toma de contacto, veamos como se construyen generalmente estos sistemas criptográficos. Como cualquier criptosistema, tienen una clave pública y privada, que en este caso serán aplicaciones. Denotaremos normalmente por $\mathcal{Q}$ a la "aplicación privada", y por $\mathcal{P}$ a la "aplicación pública". Para esconder la clave privada, normalmente se toman dos aplicaciones afines S, T, de forma que:

$$\mathcal{P} = T \circ \mathcal{Q} \circ S : \mathbb{K}^n \longrightarrow \mathbb{K}^m,$$

donde m sería el número de ecuaciones y n el número de variables utilizadas. La aplicación $\mathcal{Q}$ también se puede denominar como aplicación central, y normalmente pertenece a una clase de aplicaciones cuadráticas cuya inversa se puede calcular de forma relativamente sencilla, utilizando la computación clásica. Las aplicaciones S, T son afines y de rango completo, y en ciertas ocasiones concretas son lineales.

La clave de los criptosistemas multivariantes es la construcción de la aplicación central, pues, a partir de esta, eligiendo adecuadamente S y T, obtendremos $\mathcal{P}$, siendo los polinomios que definen esta aplicación los que llamaremos clave pública de nuestro criptosistema. Por otra parte, la clave privada será el conocimiento de S, T y $\mathcal{Q}$, y de sus inversas, que se deben poder calcular fácilmente. De esta forma, dado un mensaje cuya información se almacena en la variable $\mathbf{w} = (w_1, \ldots, w_n)$ (en este caso sin cifrar), se puede verificar la firma dada con

$\mathbf{z} = \mathcal{P}(\mathbf{w})$. Si por el contrario queremos descifrar o firmar un mensaje, depende del uso que queramos darle, tendremos que, conociendo $\mathbf{z} = (z_1, \ldots, z_m)$, calcular en orden: $\mathbf{y} = T^{-1}(\mathbf{z})$, $\mathbf{x} = \mathcal{Q}^{-1}(\mathbf{y})$ y por último $\mathbf{w} = S^{-1}(\mathbf{x})$. Cabe destacar que, en estas últimas transformaciones, el superíndice -1 no tiene porqué indicar la función inversa en el sentido estricto de la palabra, sino que basta con conocer una imagen inversa de todas las posibles.

Capítulo 4

Problema *MinRank*

Este cuarto capítulo va a contener la explicación e investigación de la dificultad de uno de los problemas más relevantes para la criptografía multivariante, el problema *MinRank*. El objetivo de este capítulo es comprender de qué trata este problema, la complejidad que tiene resolverlo y dos posibles formas de hacerlo. *MinRank* es un punto clave en la explicación de este trabajo, pues, como veremos en el último capítulo, es parte primordial del ataque que presentaremos contra el esquema de firma GeMSS.

Este problema tiene una alta complejidad y es considerado un problema NP-difícil, pero existen algunos algoritmos que permiten su resolución con alta probabilidad. Cabe destacar que los descubrimientos presentados en este capítulo son válidos para la computación tradicional, siendo bastante probable que, gracias a la computación cuántica, en los próximos años se encuentren técnicas más rápidas y eficientes para resolver dicho problema con ordenadores cuánticos.

4.1. Presentación del problema

El problema *MinRank* (MR) fue introducido originalmente en 2003 en [7] como una de las preguntas naturales en el álgebra lineal, donde los autores probaron su completitud NP. En los últimos años, MR se ha relacionado con varios sistemas criptográficos multivariantes de clave pública como HFE (*Hidden Field Equations*), y además es la base de un esquema de autenticación eficiente de conocimiento cero.

Antes de entrar en más detalles, presentaremos el planteamiento del problema de *MinRank* sobre un cuerpo $\mathbb{K}$.

Definición 4.1.1. Sean los enteros positivos $N, n, r, k \in \mathbb{N}$, y las matrices $M_0, M_1, \ldots, M_k \in \mathcal{M}_{N\times n}(\mathbb{K})$. El problema *MinRank* (MR) consiste en encon-

trar una tupla de k elementos de $\mathbb{K}$, $(\lambda_1, \ldots, \lambda_k)$ tales que:

$$\text{Rank}\left(\sum_{i=1}^{k} \lambda_i \cdot M_i - M_0\right) \leq r. \tag{4.1}$$

donde "Rank" hace referencia el rango de la matriz sobre la que se aplica.

En la práctica, consideraremos el problema *MinRank* de búsqueda. Esta variante es el caso particular en el que $N = n$, por lo que se obtiene una instancia "cuadrada" de MR, que llamaremos MR_s.

Antes de establecer la relación entre las variantes del problema *MinRank*, es útil recordar algunos conceptos fundamentales de la teoría de la complejidad computacional, en particular el concepto de *equivalencia many-one en tiempo polinomial*.

Definición 4.1.2. En términos generales, una *reducción many-one* es un procedimiento que permite transformar instancias de un problema de decisión A en instancias de otro problema B mediante una función computable, de forma que la instancia del problema reducido está en términos de B si y solo si la original está en términos de A. Si esta transformación puede realizarse en tiempo polinomial, hablamos de una *reducción many-one en tiempo polinomial*, también conocidas como reducciones de *Karp*.

Definición 4.1.3. En la teoría de la complejidad computacional, dos problemas son considerados *equivalentes many-one en tiempo polinomial* si existe una *reducción many-one en tiempo polinomial* entre ellos. Esto significa que uno de los problemas puede ser transformado en el otro en tiempo polinómico, conservando la solución.

Cuando existen reducciones de este tipo en ambos sentidos (de A a B y de B a A), se dice que los problemas son computacionalmente equivalentes en tiempo polinomial. Esto implica que, desde el punto de vista de la dificultad computacional, resolver uno de los problemas permite resolver el otro con un sobrecoste computacional polinómico. Este tipo de equivalencia es clave en la clasificación de problemas dentro de clases de complejidad como NP.

Con estas nociones en mente, podemos formalizar la relación entre la versión general del problema *MinRank* y su variante cuadrada:

Proposición 4.1.1. *La complejidad computacional de los problemas MR_s y MR es equivalente mediante una reducción en tiempo polinomial.*

Demostración:

$\mathrm{MR_s}$ es un subproblema de MR, en el sentido de que cualquier instancia de $\mathrm{MR_s}$ puede considerarse como una instancia de MR.

Ahora, sea g una función del conjunto de instancias de MR al conjunto de instancias de $\mathrm{MR_s}$, que asigna una matriz M de tamaño $N \times n$ a la matriz cuadrada de tamaño máx(N, n) obtenida a partir de M añadiendo $n - N$ filas (o, respectivamente, $N - n$ columnas) de ceros (dependiendo de si $N < n$ o no). Entonces, obviamente, $\mathrm{Rank}(g(M)) = \mathrm{Rank}(M)$, de modo que cualquier instancia afirmativa de MR se asigna a una instancia afirmativa de $\mathrm{MR_s}$ mediante g; y, recíprocamente, cualquier instancia de MR que, mediante g, se convierta en una instancia afirmativa de $\mathrm{MR_s}$, es efectivamente una instancia afirmativa de MR. □

La complejidad de este problema ha sido demostrada con varios resultados, vinculándolo con otros problemas que son presumiblemente difíciles. Uno de ellos consiste en una reducción muy simple de otro problema difícil propuesto años antes, en concreto el problema de *Maximum Likelihood Decoding*, lo que demuestra la NP-dificultad del problema de *MinRank*. Otro resultado asociado con la complejidad de MR, es su relación con otro problema importante de la teoría de Códigos, el conocido como *Rank Decoding* (RD), que también es considerado un problema NP-difícil. El resultado principal que se ha podido demostrar, y que relaciona estos problemas, es que RD es reducible en tiempo polinómico mediante reducción *many-one* a MR.

4.2. Técnicas de resolución

Existen dos técnicas no triviales generales para resolver el problema de *MinRank*. La primera de ellas, conocida como *kernel attack* [9], consiste en obtener una serie de vectores de un núcleo en concreto para posteriormente resolver el sistema lineal resultante. La segunda técnica es la de Kipnis-Shamir [18], que consiste en modelar el problema de *MinRank* como un problema de resolución de sistemas polinomiales, otro de los problemas más importantes en la criptografía multivariante.

Consideramos para este apartado una instancia del problema de búsqueda de *MinRank*, es decir, enteros positivos $n, r, k \in \mathbb{N}$ y matrices $M_0, M_1, \ldots, M_k \in \mathcal{M}_{n\times n}(\mathbb{K})$ que completan la instancia "cuadrada" del problema *MinRank*. En este apartado analizaremos dos métodos para resolver dicho problema. Antes de comenzar, observamos que cada búsqueda exhaustiva para encontrar un conjunto $(\lambda_1, \ldots, \lambda_k)$ de elementos de $\mathbb{K}$ necesita, a lo sumo, $(\#\mathbb{K})n^3$ operaciones elementales en matrices $n \times n$ sobre $\mathbb{K}$, donde $(\#\mathbb{K})$ representa el cardinal de $\mathbb{K}$.

4.2.1. *Kernel attack*

El funcionamiento de esta primera técnica de resolución es el siguiente. Primero, se eligen m vectores de forma aleatoria, que denotaremos por $\mathbf{x}^{(i)} \in \mathbb{F}_q^n$, con $1 \leq i \leq m$. Una vez elegidos, nos encargaremos de resolver el sistema de $m \cdot n$ ecuaciones para $(\mu_1, \ldots, \mu_k) \in \mathbb{F}_q^k$, dado por:

$$(M_0 - \sum_{j=1}^{k} \mu_j M_j)\mathbf{x}^{(i)} = \mathbf{0}_n, \quad \forall 1 \leq i \leq m.$$

Notemos que si $m = \lceil \frac{k}{n} \rceil$, entonces este sistema esencialmente tiene una única solución, que llamaremos $\lambda = (\lambda_1, \ldots, \lambda_k)$.

Ahora, una vez conocemos λ, definimos $E_\lambda = M_0 - \sum_{j=1}^{k} \lambda_j M_j$; y queremos que E_λ tenga rango $\leq r$. Si este fuera el caso, entonces $\dim(\ker E_\lambda) \geq n-r$ y, por lo tanto, para un $\mathbf{x} \in \mathbb{F}_q^n$ elegido al azar, tenemos las siguientes probabilidades:

$$\mathrm{p}[\mathbf{x} \in \ker E_\lambda] \geq q^{-r} \quad \text{y} \quad \mathrm{p}[\{\mathbf{x}^{(i)}, 1 \leq i \leq m\} \subseteq \ker E_\lambda] \geq q^{-mr}.$$

Por lo tanto, para encontrar un λ tal que E_λ tenga el rango deseado, debemos repetir el experimento anterior (es decir, repetir el procedimiento para obtener λ) un promedio q^{mr} veces. Tomando el valor de m como anteriormente, la complejidad de este ataque sería entonces

$$O(q^{\lceil \frac{k}{n} \rceil r} k^3).$$

4.2.2. Ataque de Kipnis-Shamir

Esta segunda técnica que vamos a presentar puede plantearse, de alguna manera, como el dual del visto en la subsección anterior. En este caso, el problema MR se modela como un problema MQ, es decir, uno en el que el propósito es resolver un sistema de ecuaciones cuadráticas.

La idea radica en intentar encontrar un conjunto de $n - r$ vectores independientes de una forma especial en el núcleo de:

$$E_\lambda = \sum_{i=1}^{k} \lambda_i \cdot M_i - M_0.$$

Al expresar estas restricciones como ecuaciones, obtenemos un sistema de ecuaciones cuadráticas cuyas incógnitas son algunas de las coordenadas de estos vectores, junto con el vector $\lambda = (\lambda_1, \ldots, \lambda_k)$.

Detalladamente, el procedimiento es el siguiente. Si λ es una solución de la instancia MR considerada, se tiene que $\mathrm{Rank}(E_\lambda) \leq r$. Queremos expresar esta

condición de rango de forma que tengamos una gran cantidad de ecuaciones en un número muy pequeño de variables, para facilitar y asegurarnos que se puede obtener una solución válida. Como $\dim(\ker(E_\lambda)) \geq n-r$, existen $n-r$ vectores linealmente independientes en $\ker(E_\lambda)$, que denotaremos por $x^{(1)}, \ldots, x^{(n-r)}$. Cabe destacar que, incluso si fijamos las primeras $n-r$ coordenadas de cada vector con valores elegidos arbitrariamente, podremos garantizar la obtención de $n-r$ vectores independientes en ciertas ocasiones.

Cada uno de los $x^{(i)}$ tiene, por lo tanto, la forma

$$x^{(i)} = (z_1, \ldots, z_{n-r}, x_1^{(i)}, \ldots, x_r^{(i)}),$$

donde los elementos z_i son elegidos arbitrariamente, mientras que los $x_j^{(i)}$ se definen como nuevas variables. Entonces, si planteamos las ecuaciones:

$$\left(\sum_{i=1}^{k} \lambda_i \cdot M_i - M_0\right) x^{(i)} = \mathbf{0}_n, \quad \text{para todo } i, 1 \leq i \leq n-r,$$

estas generan un sistema cuadrático de $(n-r) \cdot n$ ecuaciones con $r \cdot (n-r) + k$ incógnitas.

Para resolver el sistema de ecuaciones cuadráticas planteado, se puede utilizar técnicas de resolución de sistemas no lineales, por ejemplo, algoritmos basados en bases de Gröbner. Estos métodos permiten triangular el sistema y encontrar soluciones para las variables de manera eficiente.

Una vez resuelto el sistema para λ, se obtiene la combinación lineal de las matrices M_i que minimiza el rango de E_λ, proporcionando así la solución al problema de *MinRank*.

Capítulo 5

GeMSS: Great Multivariate Short Signature

Como hemos venido mencionando a lo largo de todo el trabajo, la criptografía es una parte indispensable en los sistemas de comunicación modernos. La seguridad de los algoritmos tradicionales ya hemos visto que está condicionada a la evolución de los ordenadores cuánticos, por lo que la investigación de nuevos criptosistemas es una de las principales preocupaciones en este ámbito hoy en día. Concretamente, nos vamos a centrar en la criptografía multivariante, uno de los principales candidatos para sustituir a los sistemas criptográficos tradicionales, especialmente en el área de las firmas digitales. GeMSS, que es una variante especial del esquema de firma HFEv-, es el algoritmo de firma digital que desarrollaremos más a fondo en este trabajo, centrándonos primero en el esquema del que surge y adentrándonos posteriormente en las principales características de este.

GeMSS es un algoritmo de firma digital basado en la criptografía multivariante, que está altamente relacionado con el sistema Quartz, siendo una versión más reciente, en la que se mejoran tanto la velocidad como la seguridad de Quartz. Ambos surgen a partir del criptosistema HFE (*Hidden Field Equations*) usando los modificadores *Vinegar* y *Minus*. Por ello, comenzaremos este capítulo con una explicación básica de HFEv-, para continuar después entrando en un mayor detalle para el caso concreto de GeMSS.

5.1. Introducción a HFEv-

HFEv- (*Hidden Field Equations minus*) es una variante mejorada del esquema HFE, diseñado para la firma digital mediante ecuaciones multivariantes. Su seguridad se basa en la dificultad de resolver sistemas de ecuaciones polinomia-

les no lineales sobre cuerpos finitos, lo que lo convierte en un candidato fuerte para la criptografía post-cuántica.

Como ya explicamos en el apartado de criptografía multivariante, estos sistemas criptográficos se basan en la construcción de una aplicación central, que es la clave privada. Pero antes de ello, tenemos que introducir cuáles serán los parámetros que pueden variar en este esquema de firma, que luego detallaremos y explicaremos más a fondo para el caso concreto de GeMSS. Definiremos los siguientes parámetros, además de una extensión de cuerpos con su isomorfismo asociado:

- Sean $n, v, D, \Delta \in \mathbb{N}$, y q un número primo cualquiera.
- Tomaremos como $\mathbb{F}_q$ el cuerpo finito con exactamente q elementos.
- Sea $\mu(X) \in \mathbb{F}_q[X]$ un polinomio irreducible de grado n. Definimos una extensión de cuerpo de grado n del cuerpo $\mathbb{F}_q$ como $\mathbb{F}_{q^n} = \mathbb{F}_q[X]/\mu(X)$.
- Definimos el isomorfismo $\phi : \mathbb{F}_{q^n} \longrightarrow \mathbb{F}_q^n$ entre el cuerpo $\mathbb{F}_{q^n}$ y el espacio vectorial $\mathbb{F}_q^n$ dado por:

$$\phi(a_0 + a_1 X + \cdots + a_{n-1} X^{n-1}) = (a_0, a_1, \ldots, a_{n-1})$$

En HFEv-, esta aplicación central $\mathcal{F} : \mathbb{F}_{q^n} \times \mathbb{F}_q^v \longrightarrow \mathbb{F}_{q^n}$ mencionada anteriormente, está formada por:

$$\mathcal{F}(X, v_1, \ldots, v_v) = \sum_{\substack{i,j \in \mathbb{N} \\ q^i + q^j \leq D}} A_{ij} X^{q^i + q^j} + \sum_{\substack{i \in \mathbb{N} \\ q^i \leq D}} \beta_i(v_1, \ldots, v_v) X^{q^i} + \gamma(v_1, \ldots, v_v), \tag{5.1}$$

donde $A_{i,j} \in \mathbb{F}_{q^n}$, $\beta_i : \mathbb{F}_q^v \to \mathbb{F}_{q^n}$ son aplicaciones lineales y $\gamma : \mathbb{F}_q^v \to \mathbb{F}_{q^n}$ es una aplicación cuadrática en las variables *Vinegar* $v_1, v_2, \ldots, v_v$. . Para ocultar la estructura de esta función en la clave pública, se aplican dos transformaciones afines privadas, $\mathcal{T}$ y $\mathcal{S}$, dando lugar a la clave pública:

$$\mathcal{P} = \mathcal{T} \circ \phi \circ \mathcal{F} \circ \psi \circ \mathcal{S}$$

donde ϕ es el isomorfismo definido previamente, y $\psi = \phi^{-1} \circ id_v$, con id_v la identidad en $\mathbb{F}_q^v$. En este caso, la clave privada constaría de tres aplicaciones: $\mathcal{T}$, $\mathcal{F}$ y $\mathcal{S}$, mientras que la clave pública solo de una: $\mathcal{P}$.

Recordemos que la función de este criptosistema es producir y verificar firmas digitales, y, una vez conocemos las claves pública y privada, podemos entender el proceso llevado a cabo para ello. En cuanto a la generación de firmas, se basa en los siguientes pasos:

1. Supongamos que el mensaje a firmar es: $\mathbf{y} = (y_1, y_2, \ldots, y_{n-a}) \in \mathbb{F}_q^{n-a}$. Entonces, tenemos que calcular una preimagen cualquiera $\bar{\mathbf{y}} = \mathcal{T}^{-1}(\mathbf{y}) \in \mathbb{F}_{q^n}^a$, donde hemos utilizado la aplicación afín $\mathcal{T}$ de la clave privada. Este valor $\bar{Y}$ debemos elevarlo a la extensión del cuerpo base que hemos definido anteriormente, obteniendo $\bar{Y} \in \mathbb{F}_{q^n}$.

2. Elegir valores aleatorios para las variables *Vinegar* $(v_1, \ldots, v_v) \in \mathbb{F}_q^v$ y sustituirlos en la aplicación $\mathcal{F}$, obtenida de la clave privada, para obtener una nueva $\mathcal{F}_V(X) : \mathbb{F}_{q^n} \to \mathbb{F}_{q^n}$.

3. Utilizar el algoritmo de Berlekamp (ver Apéndice A) para buscar una solución de
$$\mathcal{F}_V(X) = Y, \tag{5.2}$$
volviendo al paso 2 en caso de no encontrar ninguna. Una vez encontrada dicha solución que cumpla (5.2), calculamos $\phi(\hat{Y}) = (\tilde{y}_1, \ldots, \tilde{y}_n) \in \mathbb{F}_q^n$, donde $\hat{Y} \in \mathbb{F}_{q^n}$ es la solución encontrada, y ϕ el isomorfismo mencionado anteriormente. Además, obtendremos $\hat{\mathbf{y}} = (\tilde{y}_1, \ldots, \tilde{y}_n, v_1, \ldots, v_v) \in \mathbb{F}_q^{n+v}$, añadiendo las variables *vinegar* con las que hemos obtenido la solución a $\phi(\hat{Y})$.

4. Por último, buscando una imagen inversa de $\hat{\mathbf{y}}$ con la aplicación afín que nos faltaba de la clave privada, $\mathcal{S}$, obtenemos $\mathbf{z} = \mathcal{S}^{-1}(\hat{\mathbf{y}}) \in \mathbb{F}_q^{n+v}$, siendo entonces $\mathbf{z}$ la firma de $\mathbf{y}$.

Además de poder firmar mensajes, un sistema de firma digital tiene que permitir que el receptor compruebe que la firma es efectivamente del emisor que dice enviar el mensaje. El protocolo de verificación de firmas es el utilizado en este caso, y en HFEv- es relativamente sencillo. El receptor conoce la aplicación $\mathcal{P}$ de la clave pública, y su única acción debe ser evaluarla sobre la firma recibida y comprobar si coincide con el mensaje original, es decir, calcular $\mathbf{y}_1 = \mathcal{P}(\mathbf{z})$, y comprobar si $\mathbf{y} = \mathbf{y}_1$.

Este criptosistema ha visto como varias variantes más eficientes se utilizaban antes que él, ya que, aunque el planteamiento es bueno, presenta un problema a la hora de analizar su coste computacional. La parte más costosa del protocolo de firma es la resolución de $\mathcal{F}_V(X) = Y$, paso que además puede tener que repetirse en varias ocasiones dependiendo de la elección de las variables *Vinegar*. Por ello, el objetivo ha sido, a partir de la base que presenta este criptosistema, desarrollar ciertas variantes que mejoren la eficiencia y seguridad de este sistema, normalmente intentando reducir el grado del polinomio HFE, utilizado para la definición de la aplicación central $\mathcal{F}$.

Uno de los casos que podemos destacar de modificaciones de HFE es *Quartz*, un esquema de firma digital diseñado específicamente para lograr mayor eficien-

cia y seguridad. Quartz emplea una versión optimizada de HFEv- con parámetros cuidadosamente seleccionados para mejorar el rendimiento sin comprometer la seguridad.

5.1.1. Representación matricial de las claves de HFEv-

En este apartado, veremos cómo podemos representar la aplicación central $\mathcal{F}$ del esquema de firma HFEv- en forma matricial. Esta forma de expresar las claves definidas para HFEv- serán necesarias posteriormente en el capítulo 6 para poder completar la explicación del ataque. Esta explicación la vamos a hacer al nivel de los esquemas de firma basados en HFEv- al completo, y no centrándonos en GeMSS particularmente, pues el ataque lo plantearemos también de esta manera. Así, conseguimos ver un resultado más completo y que puede aplicar a varios ejemplos, además del caso particular de GeMSS, en el que nos centraremos más adelante.

Proposición 5.1.1. *Sea*

$$F^{*0} = \begin{pmatrix} \alpha_{00} & \alpha_{01} & \cdots & \alpha_{0,n-1} & \gamma_{00} & \gamma_{01} & \cdots & \gamma_{0,v-1} \\ \alpha_{10} & \alpha_{11} & \cdots & \alpha_{1,n-1} & \gamma_{10} & \gamma_{11} & \cdots & \gamma_{1,v-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{n-1,0} & \alpha_{n-1,1} & \cdots & \alpha_{n-1,n-1} & \gamma_{n-1,0} & \gamma_{n-1,1} & \cdots & \gamma_{n-1,v-1} \\ \beta_{00} & \beta_{01} & \cdots & \beta_{0,n-1} & \delta_{00} & \delta_{01} & \cdots & \delta_{0,v-1} \\ \beta_{10} & \beta_{11} & \cdots & \beta_{1,n-1} & \delta_{10} & \delta_{11} & \cdots & \delta_{1,v-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta_{v-1,0} & \beta_{v-1,1} & \cdots & \beta_{v-1,n-1} & \delta_{v-1,0} & \delta_{v-1,1} & \cdots & \delta_{v-1,v-1} \end{pmatrix}$$

una matriz de dimensión $(n+v)\times(n+v)$ *sobre el cuerpo* $\mathbb{F}_{q^n}$ *y*

$$\begin{aligned} &F(X, x_1, \ldots, x_v) = \\ &= (X, X^q, \ldots, X^{q^{n-1}}, x_1, \ldots, x_v) F^{*0} (X, X^q, \ldots, X^{q^{n-1}}, x_1, \ldots, x_v)^t \end{aligned}$$

un polinomio en el anillo cociente $\mathbb{F}_{q^n}[X, x_1, \ldots, x_v]/\langle x_1^q - x_1, \ldots, x_v^q - x_v\rangle$.

Entonces tenemos, para todo $0 \le k < n$,

$$\begin{aligned} &F^{q^k}(X, x_1, \ldots, x_v) = \\ &= (X, X^q, \ldots, X^{q^{n-1}}, x_1, \ldots, x_v) F^{*k} (X, X^q, \ldots, X^{q^{n-1}}, x_1, \ldots, x_v)^t, \end{aligned}$$

donde $F^{*k} \in \mathcal{M}_{(n+v)\times(n+v)}(\mathbb{F}_{q^n})$, *tiene las siguientes entradas:*

- *Para* $0 \le i, j, k < n$, *la entrada* (i, j) *de* F^{*k} *es* $\alpha_{i-k,j-k}^{q^k}$.

- *Para* $0 \leq i < v, 0 \leq j, k < n$, *e* $(i, n+j)$ *de* F^{*k} *es* $\gamma_{j-k,i}^{q^k}$.

- *Para* $0 \leq i < v, 0 \leq j, k < n$, *la entrada* $(n+i, j)$ *de* F^{*k} *es* $\beta_{i,j-k}^{q^k}$.

- *Para* $0 \leq i < v, 0 \leq j < v, 0 \leq k < n$, *la entrada* $(n+i, n+j)$ *de* F^{*k} *es* $\delta_{ij}^{q^k}$.

Demostración:
Si $k = 0$, entonces obviamente tenemos

$$F^{q^k}(X, x_1, \ldots, x_v) = F(X, x_1, \ldots, x_v).$$

Ahora consideremos el caso $1 \leq k < n$. Como $x_i^{q^K} = x_i$ para todo $1 \leq i \leq v$, tenemos

$$\begin{aligned} F^{q^k} &= \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} \alpha_{ij}^{q^k} X^{q^{i+k}+q^{j+k}} + \sum_{i=0}^{v-1}\sum_{j=0}^{n-1} (\beta_{ij}^{q^k} + \gamma_{ji}^{q^k}) x_i X^{q^{j+k}} + \sum_{i=0}^{v-1}\sum_{j=0}^{v-1} \delta_{ij}^{q^k} x_i x_j \\ &= \sum_{i=k}^{n-1+k}\sum_{j=k}^{n-1+k} \alpha_{i-k,j-k}^{q^k} X^{q^i+q^j} + \sum_{i=0}^{v-1}\sum_{j=k}^{n-1+k} (\beta_{i,j-k}^{q^k} + \gamma_{j-k,i}^{q^k}) x_i X^{q^j} \\ &+ \sum_{i=0}^{v-1}\sum_{j=0}^{v-1} \delta_{ij}^{q^k} x_i x_j. \end{aligned}$$

Esto se puede dividir como:

$$\begin{aligned} F^{q^k} =& \sum_{i=k}^{n-1} \left(\sum_{j=k}^{n-1+k} \alpha_{i-k,j-k}^{q^k} X^{q^i+q^j} \right) + \sum_{i=n}^{n-1+k} \left(\sum_{j=k}^{n-1+k} \alpha_{i-k,j-k}^{q^k} X^{q^i+q^j} \right) \\ &+ \sum_{i=0}^{v-1}\sum_{j=k}^{n-1} (\beta_{i,j-k}^{q^k} + \gamma_{j-k,i}^{q^k}) x_i X^{q^j} + \sum_{i=0}^{v-1}\sum_{j=n}^{n-1+k} (\beta_{i,j-k}^{q^k} + \gamma_{j-k,i}^{q^k}) x_i X^{q^j} \\ &+ \sum_{i=0}^{v-1}\sum_{j=0}^{v-1} \delta_{ij}^{q^k} x_i x_j. \end{aligned}$$

Que es lo mismo que escribir

$$\begin{aligned}F^{q^k} =&\sum_{i=k}^{n-1}\left(\sum_{j=k}^{n-1}\alpha_{i-k,j-k}^{q^k}X^{q^i+q^j}+\sum_{j=n}^{n-1+k}\alpha_{i-k,j-k}^{q^k}X^{q^i+q^j}\right)\\&+\sum_{i=n}^{n-1+k}\left(\sum_{j=k}^{n-1}\alpha_{i-k,j-k}^{q^k}X^{q^i+q^j}+\sum_{j=n}^{n-1+k}\alpha_{i-k,j-k}^{q^k}X^{q^i+q^j}\right)\\&+\sum_{i=0}^{v-1}\sum_{j=k}^{n-1}(\beta_{i,j-k}^{q^k}+\gamma_{j-k,i}^{q^k})x_iX^{q^j}+\sum_{i=0}^{v-1}\sum_{j=n}^{n-1+k}(\beta_{i,j-k}^{q^k}+\gamma_{j-k,i}^{q^k})x_iX^{q^j}\\&+\sum_{i=0}^{v-1}\sum_{j=0}^{v-1}\delta_{ij}^{q^k}x_ix_j.\end{aligned}$$

Por lo que tenemos

$$\begin{aligned}F^{q^k} =&\sum_{i=k}^{n-1}\left(\sum_{j=k}^{n-1}\alpha_{i-k,j-k}^{q^k}X^{q^i+q^j}+\sum_{j=0}^{k-1}\alpha_{i-k,j-k+n}^{q^k}X^{q^i+q^{j+n}}\right)\\&+\sum_{i=0}^{k-1}\left(\sum_{j=k}^{n-1}\alpha_{i-k+n,j-k}^{q^k}X^{q^{i+n}+q^j}+\sum_{j=0}^{k-1}\alpha_{i-k+n,j-k+n}^{q^k}X^{q^{i+n}+q^{j+n}}\right)\\&+\sum_{i=0}^{v-1}\sum_{j=k}^{n-1}(\beta_{i,j-k}^{q^k}+\gamma_{j-k,i}^{q^k})x_iX^{q^j}+\sum_{i=0}^{v-1}\sum_{j=0}^{k-1}(\beta_{i,j-k+n}^{q^k}+\gamma_{j-k+n,i}^{q^k})x_iX^{q^{j+n}}\\&+\sum_{i=0}^{v-1}\sum_{j=0}^{v-1}\delta_{ij}^{q^k}x_ix_j.\end{aligned}$$

Como $X^{q^n}=X$, reduciendo el índice de los coeficientes módulo n obtenemos:

$$\begin{aligned}F^{q^k} =&\sum_{i=k}^{n-1}\left(\sum_{j=k}^{n-1}\alpha_{i-k,j-k}^{q^k}X^{q^i+q^j}+\sum_{j=0}^{k-1}\alpha_{i-k,j-k}^{q^k}X^{q^i+q^j}\right)\\&+\sum_{i=0}^{k-1}\left(\sum_{j=k}^{n-1}\alpha_{i-k,j-k}^{q^k}X^{q^i+q^j}+\sum_{j=0}^{k-1}\alpha_{i-k,j-k}^{q^k}X^{q^i+q^j}\right)\\&+\sum_{i=0}^{v-1}\sum_{j=k}^{n-1}(\beta_{i,j-k}^{q^k}+\gamma_{j-k,i}^{q^k})x_iX^{q^j}+\sum_{i=0}^{v-1}\sum_{j=0}^{k-1}(\beta_{i,j-k}^{q^k}+\gamma_{j-k,i}^{q^k})x_iX^{q^j}\\&+\sum_{i=0}^{v-1}\sum_{j=0}^{v-1}\delta_{ij}^{q^k}x_ix_j.\end{aligned}$$

Agrupando nuevamente las sumas, se tiene ahora

$$\begin{aligned} F^{q^k} =& \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} \alpha_{i-k,j-k}^{q^k} X^{q^i+q^j} + \sum_{i=0}^{v-1}\sum_{j=0}^{n-1} (\beta_{i,j-k}^{q^k} + \gamma_{j-k,i}^{q^k}) x_i X^{q^j} + \sum_{i=0}^{v-1}\sum_{j=0}^{v-1} \delta_{ij}^{q^k} x_i x_j \\ &= (X, X^q, \ldots, X^{q^{n-1}}, x_1, \ldots, x_v) F^{*k} (X, X^q, \ldots, X^{q^{n-1}}, x_1, \ldots, x_v)^t. \end{aligned}$$

Donde $F^{*k} \in \mathcal{M}_{(n+v)\times(n+v)}(\mathbb{F}_{q^n})$, y se cumple que:

- Para $0 \leq i, j, k < n$, la entrada (i, j) de F^{*k} es $\alpha_{i-k,j-k}^{q^k}$.
- Para $0 \leq i < v, 0 \leq j, k < n$, e $(i, n+j)$ de F^{*k} es $\gamma_{j-k,i}^{q^k}$.
- Para $0 \leq i < v, 0 \leq j, k < n$, la entrada $(n+i, j)$ de F^{*k} es $\beta_{i,j-k}^{q^k}$.
- Para $0 \leq i < v, 0 \leq j < v, 0 \leq k < n$, la entrada $(n+i, n+j)$ de F^{*k} es $\delta_{ij}^{q^k}$.

Tal como queríamos probar.

□

Proposición 5.1.2. *Sea $(\theta_1, \theta_2, \ldots, \theta_n) \in \mathbb{F}_{q^n}^n$ una base vectorial de $\mathbb{F}_{q^n}$ sobre $\mathbb{F}_q$. Construimos la matriz M como*

$$M = \begin{pmatrix} \theta_1 & \theta_1^q & \ldots & \theta_1^{q^{n-1}} \\ \theta_2 & \theta_2^q & \ldots & \theta_2^{q^{n-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_n & \theta_n^q & \ldots & \theta_n^{q^{n-1}} \end{pmatrix},$$

que es la matriz cuyas columnas son las potencias de Frobenius de los elementos base. Podemos expresar el morfismo $\phi : \mathbb{F}_{q^n} \to \mathbb{F}_q^n$ definido en el apartado anterior como

$$V \mapsto (V, V^q, \ldots, V^{q^{n-1}}) M^{-1}.$$

Su inverso $\phi^{-1} : \mathbb{F}_q^n \to \mathbb{F}_{q^n}$ quedaría definido por

$$(v_1, v_2, \ldots, v_n) \mapsto V,$$

donde V es el primer componente del vector $(v_1, v_2, \ldots, v_n)M$. Más generalmente, tendríamos

$$(v_1, v_2, \ldots, v_n) \cdot M = (V, V^q, \ldots, V^{q^{n-1}}).$$

Demostración:

Sea $(v_1, \ldots, v_n) \in \mathbb{F}_q^n$ la descomposición de $V \in \mathbb{F}_{q^n}$ como vector en $\mathbb{F}_q^n$. Es decir, $V = \sum_{i=1}^n v_i\theta_i \in \mathbb{F}_{q^n}$. Por construcción:

$$\begin{aligned}(v_1, \ldots, v_n)\mathbf{M}_n &= \left(\sum_{i=1}^n v_i\theta_i^{q^0}, \ldots, \sum_{i=1}^n v_i\theta_i^{q^{n-1}}\right) \\ &= \left(\left(\sum_{i=1}^n v_i\theta_i\right)^{q^0}, \ldots, \left(\sum_{i=1}^n v_i\theta_i\right)^{q^{n-1}}\right) = (V^{q^0}, \ldots, V^{q^{n-1}}).\end{aligned} \tag{5.3}$$

Y como consecuencia:

$$\varphi_1^{-1}(v_1, \ldots, v_n) = ((v_1, \ldots, v_n)\mathbf{M}_n)[1] = V.$$

donde $((v_1, \ldots, v_n)\mathbf{M}_n)[1]$ representa la primera componente del vector indicado. Además, al ser $\mathbf{M}_n$ invertible, tenemos para φ_1:

$$(V^{q^0}, \ldots, V^{q^{n-1}}) = (v_1, \ldots, v_n)\mathbf{M}_n,$$

$$(V^{q^0}, \ldots, V^{q^{n-1}})\mathbf{M}_n^{-1} = (v_1, \ldots, v_n) = \varphi_1(V).$$

□

Para la teoría que estamos desarrollando, nos interesará elegir la matriz M de la siguiente forma

$$M = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ \theta & \theta^q & \ldots & \theta^{q^{n-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \theta^{n-1} & (\theta^{n-1})^q & \ldots & (\theta^{n-1})^{q^{n-1}} \end{pmatrix}, \tag{5.4}$$

donde θ es un generador de $\mathbb{F}_{q^n}$. Esta construcción la hacemos de esta forma para obtener posteriormente un beneficio cuando sea necesario que la utilicemos, en concreto, lo requeriremos en el capítulo 6 cuando expliquemos el ataque a HFEv-.

Definimos también la matriz

$$\widetilde{M} = \begin{pmatrix} M & 0 \\ 0 & I_v \end{pmatrix} \in \mathcal{M}_{(n+v)\times(n+v)}(\mathbb{F}_{q^n}), \tag{5.5}$$

donde I_v es la matriz identidad de dimensión $v \times v$.

De acuerdo con la Proposición 5.1.1, tenemos

$$(v_1, v_2, \ldots, v_n, x_1, \ldots, x_v)\widetilde{M} = (V, V^q, \ldots, V^{q^{n-1}}, x_1, \ldots, x_v),$$

donde $v_i, x_j \in \mathbb{F}_q$, para todo $1 \leq i \leq n$, y $1 \leq j \leq v$; y $V \in \mathbb{F}_{q^n}$.

Proposición 5.1.3. *Sean $p_i \in \mathbb{F}_q[x_1, x_2, \ldots, x_{n+v}]$ los polinomios de clave pública de HFEv- y P_i la matriz que representa la forma cuadrática de p_i, para cada $0 \leq i < n - \Delta$. Sea F la aplicación central de HFEv- dada por*

$$F = (X, X^q, \ldots, X^{q^{n-1}}, x_1, \ldots, x_v)F^{*0}(X, X^q, \ldots, X^{q^{n-1}}, x_1, \ldots, x_v)^t,$$

donde $F^{*0} \in \mathcal{M}_{(n+v)\times(n+v)}(\mathbb{F}_{q^n})$.

Sea ahora $S \in \mathcal{M}_{(n+v)\times(n+v)}(\mathbb{F}_q)$ y $T \in \mathcal{M}_{n\times(n-\Delta)}(\mathbb{F}_q)$ las matrices que representan las partes lineales de $\mathcal{S}$ y $\mathcal{T}$. Entonces, tenemos que

$$\begin{aligned}&\left(\widetilde{M}^{-1}S^{-1}P_0(S^{-1})^t(\widetilde{M}^{-1})^t, \ldots, \widetilde{M}^{-1}S^{-1}P_{n-\Delta-1}(S^{-1})^t(\widetilde{M}^{-1})^t\right) = \\ &= (F^{*0}, \ldots, F^{*n-1})M^{-1}T \end{aligned} \tag{5.6}$$

Demostración:
Se plantea de forma similar a la demostración del Lema 2 de [5]. □

Si denotamos ahora por $U = \widetilde{M}^{-1}S^{-1} \in \mathcal{M}_{(n+v)\times(n+v)}(\mathbb{F}_{q^n})$ y por $W = M^{-1}T \in \mathcal{M}_{n\times(n-\Delta)}(\mathbb{F}_q)$, entonces la ecuación 6.16 puede reescribirse como

$$(UP_0U^t, \ldots, UP_{n-1}U^t) = (F^{*0}, \ldots, F^{*n-1})W. \tag{5.7}$$

5.2. Clave pública y privada

Nos centraremos ahora en el esquema de firma que vamos a detallar a fondo, GeMSS. Antes de entrar con la definición de la clave pública y privada, vamos a definir una serie de parámetros que serán los que posteriormente podremos ajustar en función de la seguridad y eficiencia que requiramos. Los parámetros más importantes involucrados en el esquema de firma GeMSS son los siguientes:

- D, el grado de un polinomio secreto que formará parte de la clave privada. En concreto, será el grado del polinomio análogo al polinomio de HFE visto en el apartado anterior. Será un entero positivo tal que $D = 2^i$, con $i \geq 0$, o bien $D = 2^i + 2^j$, con $i \neq j$; $i, j \geq 0$.
- K, el tamaño de salida de la función hash en bits.

- λ, el nivel de seguridad de GeMSS. Puede ser 128, 192 o 256 en función de la elección del resto de parámetros.
- m, que representará el número de ecuaciones en la clave pública.
- $n_{ite} > 0$, número de iteraciones para el protocolo de firma y verificación.
- n, grado de un cuerpo que es extensión de de $\mathbb{F}_2$.
- v, número de variables *Vinegar*.
- Δ, número de ecuaciones *Minus*. Se puede obtener a partir de m y n, ya que necesariamente $m = n - \Delta$.

Conocidos los parámetros con los que trabajaremos, definamos ahora la clave pública y privada que se utilizarán en las distintas variaciones de GeMSS. Veremos que la clave pública será un conjunto de $m = n - \Delta$ ecuaciones cuadráticas en $n+v$ variables. Estas ecuaciones se obtienen a partir de la clave privada, dada por un polinomio en varias variables $F \in \mathbb{F}_2[X, v_1, \ldots, v_v]$, que describiremos a continuación, de forma que generar una firma sea equivalente a encontrar las raíces de F

Clave privada

Como hemos mencionado en el párrafo introductorio anterior, antes de definir la clave pública, necesitamos conocer la clave privada. En este caso, al igual que en HFEv- tendremos tres elementos que serán dos matrices invertibles $(\mathbf{S}, \mathbf{T}) \in \mathrm{GL}_{n+v}(\mathbb{F}_2) \times \mathrm{GL}_n(\mathbb{F}_2)$, y el polinomio $F \in \mathbb{F}_2[X, v_1, \ldots, v_v]$ antes mencionado, que tendrá la siguiente estructura:

$$\sum_{\substack{i,j\in\mathbb{N} \\ 2^i+2^j\leq D}} A_{ij} X^{2^i+2^j} + \sum_{\substack{i\in\mathbb{N} \\ 2^i\leq D}} \beta_i(v_1, \ldots, v_v) X^{q^i} + \gamma(v_1, \ldots, v_v), \tag{5.8}$$

donde se tienen que dar las siguientes condiciones:

- $A_{i,j} \in \mathbb{F}_2^n$,
- v_i y $0 \leq j < i < n$,
- cada $\beta_i : \mathbb{F}_2^v \to \mathbb{F}_2^n$ es lineal y
- $\gamma(v_1, \ldots, v_v) : \mathbb{F}_2^v \to \mathbb{F}_2^n$ es cuadrática.

Podemos observar aquí la similitud con la aplicación $\mathcal{F}$ definida en el apartado anterior en (5.1), viendo que la única diferencia la encontramos en que hemos elegido $q = 2$ para el caso de GeMSS. Diremos que este polinomio tiene forma HFEv.

Observación 5.2.1. La particularidad de un polinomio $F(X, v_1, \ldots, v_v)$ con forma HFEv es que, para cualquier elección de las variables *Vinegar*, el polinomio F se convierte en un polinomio HFE, que es un polinomio en una variable de la forma:

$$\sum_{\substack{i,j\in\mathbb{N}\\ 2^j+2^i\leq D}} A_{i,j}X^{2^i+2^j} + \sum_{\substack{i\in\mathbb{N}\\ 2^i\leq D}} B_iX^{2^i} + C \in \mathbb{F}_{2^n}[X].$$

con $A_{i,j}, B_i, C \in \mathbb{F}_{2^n}$, $\forall i, j$, con $0 \leq j < i < n$.

Por abuso de notación, llamaremos grado de F al grado máximo de sus polinomios HFE correspondientes. En este caso, el grado de F será D, tal como especificamos en los parámetros al inicio de esta sección.

Veamos como podemos simplificar levemente la forma en la que hemos definido la clave privada. Si observamos (5.8), vemos que su estructura es particular, ya que ha sido elegida de forma que su representación en varias variables sobre el cuerpo $\mathbb{F}_2$ esté compuesta por polinomios cuadráticos en $\mathbb{F}_2[x_1, \ldots, x_{n+v}]$. Esto se debe a la elección especial de X en $\mathbb{F}_{2^n}$, cuyas potencias tienen una descomposición binaria con un máximo de dos componentes no nulas, es decir, peso de Hamming igual a 2.

Si ahora suponemos que $(\theta_1, \ldots, \theta_n) \in (\mathbb{F}_{2^n})^n$ es una base de $\mathbb{F}_{2^n}$ sobre $\mathbb{F}_2$, podemos definir $\varphi : \mathbb{F}_{2^n} \longrightarrow \mathbb{F}_2^n$, y para cada $E = \sum_{k=1}^n e_k \cdot \theta_k \in \mathbb{F}_{2^n}$, $\varphi(E) = (e_1, \ldots, e_n)$.

Gracias a lo anterior, podemos definir ahora un conjunto de polinomios en varias variables $f = (f_1, \ldots, f_n) \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^n$ derivados de un polinomio HFEv $F \in \mathbb{F}_2[X, v_1, \ldots, v_v]$ mediante:

$$F\left(\sum_{k=1}^n \theta_k x_k, v_1, \ldots, v_v\right) = \sum_{k=1}^n \theta_k f_k.$$

A partir de ahora, con la finalidad de simplificar la notación, denotaremos las variables *Vinegar* $(v_1, \ldots, v_v)$ por $(x_{n+1}, \ldots, x_{n+v})$. Además, a los polinomios $f_1, \ldots, f_n \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]$ los conocemos como las componentes de F sobre $\mathbb{F}_2$.

Clave pública

Utilizaremos en este pequeño apartado la notación simplificada definida al final del anterior. La clave pública, como sabemos, se construye a partir de la clave privada. Suponemos, por tanto, que conocemos $(\mathbf{S}, \mathbf{T}) \in \mathrm{GL}_{n+v}(\mathbb{F}_2) \times \mathrm{GL}_n(\mathbb{F}_2)$, y también $F \in \mathbb{F}_2[X, v_1, \ldots, v_v]$. Si además hemos calculado las componentes de F sobre $\mathbb{F}_2$, es decir, conocemos $\mathbf{f} = (f_1, \ldots, f_n) \in \mathbb{F}_{\not=}[x_1, \ldots x_{n+v}]^n$, entonces

podemos obtener la clave pública calculando:

$$\left(f_1\Big((x_1,\ldots,x_{n+v})\mathbf{S}\Big),\ldots,f_n\Big((x_1,\ldots,x_{n+v})\mathbf{S}\Big)\right)\mathbf{T},$$

y reduciéndolo módulo $\langle x_1^2 - x_1, \ldots, x_{n+v}^2 - x_{n+v}\rangle$, tomando únicamente los primeros $m = n - \Delta$ polinomios. Resumiendo, el cálculo de la clave pública será el siguiente:

$$\begin{aligned}\mathbf{p} &= (p_1,\ldots,p_n) = \\ &= \left(f_1\Big((x_1,\ldots,x_{n+v})\mathbf{S}\Big),\ldots,f_n\Big((x_1,\ldots,x_{n+v})\mathbf{S}\Big)\right)\mathbf{T} \\ &\text{mód } \langle x_1^2 - x_1,\ldots,x_{n+v}^2 - x_{n+v}\rangle,\end{aligned} \tag{5.9}$$

de donde nos quedaremos con los primeros m polinomios. Entonces, la clave pública será $\mathbf{p_1} = (p_1,\ldots,p_m) \in \mathbb{F}_{\not 2}[x_1,\ldots x_{n+v}]^m$, ya que $m = n - \Delta \leq n$.

La generación de la clave pública y secreta se puede resumir con el siguiente algoritmo:

Una vez conocemos la definición de la clave pública y privada, tal como vimos en el apartado de firmas digitales (apartado 3.2.3), podemos definir los protocolos de firma y verificación, en los que se basa una firma digital.

5.3. Protocolo de firma

En los apartados anteriores hemos visto cómo un usuario que quiera utilizar el algoritmo GeMSS se puede generar una clave pública y privada propias. Para darle uso a estas claves, en un esquema de firma digital, podemos o bien firmar un mensaje, o verificar que una firma realizada con este algoritmo es válida. Explicaremos primero cómo firmar mensajes, viendo el funcionamiento del protocolo de firma asociado a este algoritmo, para después poder entender cómo puede funcionar correctamente la verificación de dicha firma.

Continuando con la notación anterior, tenemos que el principal paso del protocolo de firma requiere resolver:

$$p_1(x_1,\ldots,x_{n+v}) - d_1 = 0,\ldots,p_m(x_1,\ldots,x_{n+v}) - d_m = 0, \tag{5.10}$$

para $\mathbf{d} = (d_1,\ldots,d_m) \in \mathbb{F}_2^m$.

Para obtener dicha solución, comenzaremos seleccionando aleatoriamente $\mathbf{r} = (r_1,\ldots,r_{n-m}) \in \mathbb{F}_2^{n-m}$, y lo juntamos con $\mathbf{d}$, de forma que nos queda $\mathbf{d'} =$

Algorithm 1 Generación de clave pública (pk) y clave privada (sk) en GeMSS

1: **procedure** GeMSS_KeyGen(1^λ)

2: Elegir aleatoriamente $(\mathbf{S}, \mathbf{T}) \in \mathrm{GL}_{n+v}(\mathbb{F}_2) \times \mathrm{GL}_n(\mathbb{F}_2)$.

3: Elegir aleatoriamente $F \in \mathbb{F}_2[X, v_1, \ldots, v_v]$ con forma HFEv de grado D.

4: La clave privada será:
$\mathrm{sk} \leftarrow (F, \mathbf{S}, \mathbf{T}) \in \mathbb{F}_2[X, v_1, \ldots, v_v] \times \mathrm{GL}_{n+v}(\mathbb{F}_2) \times \mathrm{GL}_n(\mathbb{F}_2)$

5: Calcular $\mathbf{f} = (f_1, \ldots, f_n) \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^n$ tal que:

$$F\left(\sum_{k=1}^{n} \theta_k x_k, v_1, \ldots, v_v\right) = \sum_{k=1}^{n} \theta_k f_k.$$

6: Calculamos:

$$\begin{aligned}\mathbf{p} &= (p_1, \ldots, p_n) = \\ &= \left(f_1\Big((x_1, \ldots, x_{n+v})\mathbf{S}\Big), \ldots, f_n\Big((x_1, \ldots, x_{n+v})\mathbf{S}\Big)\right)\mathbf{T} \\ &\text{mód } \langle x_1^2 - x_1, \ldots, x_{n+v}^2 - x_{n+v}\rangle,\end{aligned}$$

7: Entonces la clave pública será:
$\mathrm{pk} \leftarrow p = (p_1, \ldots, p_m) \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^m$
▷ Tomar los primeros $m = n - \Delta$ polinomios obtenidos en el Paso 6

8: **return** pk, sk

9: **end procedure**

$(\mathbf{d}, \mathbf{r}) \in \mathbb{F}_2^n$. Después de ello, tendremos que calcular $D' = \varphi^{-1}(\mathbf{d}' \times \mathbf{T}^{-1}) \in \mathbb{F}_{2^n}$ y resolver la siguiente ecuación en varias variables:

$$F(Z, z_1, \ldots, z_v) - D' = 0. \tag{5.11}$$

Denotaremos la solución de la ecuación (5.11) por $(Z, z_1, \ldots, z_v) \in \mathbb{F}_{2^n} \times \mathbb{F}_2^v$. Para resolver esta ecuación, necesitaremos habernos aprovechado de la ventaja de su forma HFEv, es decir, elegir aleatoriamente $\mathbf{v} \in \mathbb{F}_2^v$ y considerar el polinomio en una variable $F(X, \mathbf{v}) \in \mathbb{F}_{2^n}[X]$. Con esto, y de acuerdo a la Observación 5.2.1, obtendremos un polinomio HFE y nos bastará con buscar las soluciones de la ecuación:

$$F(X, \mathbf{v}) - D' = 0. \tag{5.12}$$

Una manera de plantear la búsqueda de soluciones de la ecuación (5.12) es hallar las raíces de $F_{D'}(X) = F(X, \mathbf{v}) - D'$, para lo que podemos utilizar el algoritmo de Berlekamp, como está descrito en el apéndice, en el que se da una pequeña explicación de su funcionamiento y se detallan los pasos a seguir.

El algoritmo de Berlekamp (ver Apéndice A) tiene una complejidad asegurada por el siguiente resultado, cuya demostración la podemos encontrar en [16] (Corolario 14.16):

Teorema 5.3.1. *Sea $\mathbb{F}_q$ un cuerpo finito, y sea $M_q(D)$ el número de operaciones en $\mathbb{F}_q$ necesarias para multiplicar dos polinomios de grado $\leq D$. Dado un polinomio $f \in \mathbb{F}_q[x]$ de grado D, podemos encontrar todas sus raíces en $\mathbb{F}_q$ utilizando un número esperado de*

$$O\left(M_q(D) \log(D) \log(Dq)\right) \tag{5.13}$$

u $\tilde{O}(D \log(q))$ operaciones en $\mathbb{F}_q$.

En particular, tenemos que en este caso se cumple el teorema anterior para $q = 2^n$, y, para este valor en concreto, vemos que encontrar todas las raíces de un polinomio de grado D se puede hacer en un tiempo casi lineal:

$$\tilde{O}(nD).$$

Toda la explicación anterior es necesaria para realizar uno de los pasos de la inversión en GeMSS, que, a su vez, es uno de los pasos del protocolo de firma. Con la intención de resumir la explicación del cálculo de la inversa, se presenta el siguiente algoritmo que se encarga de calcular dicho valor inverso. Sean $\mathbf{d} \in \mathbb{F}_2^m$, $\mathrm{sk} = (F, S, T) \in \mathbb{F}_{2^n}[X, v_1, \ldots, v_v] \times \mathrm{GL}_{n+v}(\mathbb{F}_2) \times \mathrm{GL}_n(\mathbb{F}_2)$

Si ahora elegimos $\mathbf{d} \in \mathbb{F}_2^m$ cualquiera, como suponemos que cada usuario tiene su clave secreta sk, podemos calcular $\mathbf{s} \leftarrow \mathrm{Inv}_{\mathbf{p}}(\mathbf{d}, \mathrm{sk} = (F, S, T)) \in \mathbb{F}_2^{n+v}$, y por construcción, tendríamos que:

$$\mathbf{p}(\mathbf{s}) = \mathbf{d}$$

Algorithm 2 Inversión en GeMSS

function GeMSS.$\text{INV}_{\mathbf{p}}(\mathbf{d}, \text{sk})$
  repeat
    Se selecciona aleatoriamente $\mathbf{r} \in \mathbb{F}_2^{n-m}$
    $\mathbf{d}' \leftarrow (\mathbf{d}, \mathbf{r}) \in \mathbb{F}_2^n$
    $D' \leftarrow \varphi^{-1}(\mathbf{d}' \times T^{-1}) \in \mathbb{F}_{2^n}$
    Se elige aleatoriamente $\mathbf{v} \in \mathbb{F}_2^v$
    $F_{D'}(X) \leftarrow F(X, \mathbf{v}) - D'$
    sols $\leftarrow$ Berlekamp($F_{D'}$)
                ▷ Ver algoritmo Berlekamp en el Apéndice A.
  until sols $\neq \emptyset$
  Elegir aleatoriamente una de las raíces $Z \in$ sols
  return $(\varphi(Z), \mathbf{v}) \times S^{-1} \in \mathbb{F}_2^{n+v}$
end function

donde **p** es la clave pública asociada a sk. Este valor **s** podría utilizarse directamente como firma correspondiente al mensaje **d**, pero tiene un problema. En el caso de GeMSS, la elección de m normalmente es tan pequeña que un ataque muy simple, como el de la paradoja del cumpleaños, tendría una eficiencia demasiado alta. Este problema también se observó tanto en *Quartz* como en *Gui*, dos esquemas de firma anteriores a GeMSS y que presentan una estructura similar. La forma de solucionar este inconveniente es utilizar el esquema conocido como *Feistel-Patarin*.

El funcionamiento de este esquema se basa en iterar el algoritmo 2 tantas veces como indique el parámetro n_{ite}, uno de los que habíamos definido al inicio del apartado 5.2 y que todavía no habíamos utilizado. La elección del valor de este parámetro ha sido discutida en varias ocasiones, recomendando diferentes valores para diferentes propósitos, siendo $n_{ite} = 4$ el más recomendado, que es el mismo valor que se utiliza en Quartz.

Con todos estos conocimientos que hemos adquirido, estamos en disposición de mostrar el algoritmo de firma de GeMSS.

Para finalizar este apartado, cabe destacar que la firma que hemos obtenido la denotaremos por **sm**, y será de tamaño: $m+n_{ite}(n+v-m) = m+n_{ite}(\Delta+v)$, es decir, $\mathbf{sm} \in \mathbb{F}_2^{m+n_{ite}(\Delta+v)}$

5.4. Protocolo de verificación

Una vez sabemos cómo firmar un mensaje utilizando la clave privada que nos proporciona GeMSS, tenemos que dar también la opción de que cualquier usuario pueda verificar la validez de dicha firma. Para ello, utilizaremos la clave

Algorithm 3 Protocolo de firma en GeMSS

procedure GeMSS.Sign($\mathbf{M} \in \{0,1\}^*$, sk, GeMSS.Inv)
  $\mathbf{H} \leftarrow$ SHA3($\mathbf{M}$)
  $\mathbf{S}_0 \leftarrow \mathbf{0} \in \mathbb{F}_2^m$
  for i from 1 to n_{ite} **do**
    $\mathbf{D}_i \leftarrow$ primeros m bits de $\mathbf{H}$
    $(\mathbf{S}_i, \mathbf{X}_i) \leftarrow$ GeMSS.Inv($D_i \oplus S_{i-1}$)
      ▷ donde $\mathbf{S}_i \in \mathbb{F}_2^m$ y $\mathbf{X}_i \in \mathbb{F}_2^{n+v-m}$
      ▷ $\oplus$ hace referencia al operador XOR componente a componente
    $\mathbf{H} \leftarrow$ SHA3($\mathbf{H}$)
  end for
  return $(\mathbf{S}_{n_{ite}}, \mathbf{X}_{n_{ite}}, \ldots, \mathbf{X}_1)$
end procedure

pública $\mathbf{p}$ del usuario que ha firmado, y, utilizando las propiedades de claves públicas y privadas, así como la estructura del algoritmo de firma, se obtiene el siguiente algoritmo que se utiliza para la verificación de firmas.

Algorithm 4 Proceso de verificación en GeMSS

procedure GeMSS.Verif($\mathbf{M} \in \{0,1\}^*$, $n_{ite} > 0$, **sm**, pk $= \mathbf{p} \in \mathbb{F}_2[Z_1, \ldots, Z_{n+v}]$)
  $\mathbf{H} \leftarrow$ SHA3($\mathbf{M}$)
  $(\mathbf{S}_{n_{ite}}, \mathbf{X}_{n_{ite}}, \ldots, \mathbf{X}_1) \leftarrow$ **sm**
  for i from 1 to n_{ite} **do**
    $\mathbf{D}_i \leftarrow$ primeros m bits de $\mathbf{H}$
    $\mathbf{H} \leftarrow$ SHA3($\mathbf{H}$)
  end for
  for i from $n_{ite} - 1$ to 0 **do**
    $\mathbf{S}_i \leftarrow \mathbf{p}(\mathbf{S}_{i+1}, \mathbf{X}_{i+1}) \oplus \mathbf{D}_{i+1}$
  end for
  return VALID si $\mathbf{S}_0 = \mathbf{0}$ e INVALID en caso contrario.
end procedure

Dado un mensaje firmado, en este caso **sm**, y teniendo correctamente definidos todos los parámetros y funciones necesarias, podremos aplicar el algoritmo 4 y conocer si la firma es válida (el algoritmo devuelve VALID) o si por el contrario es una firma incorrecta (en ese caso devolverá INVALID).

Capítulo 6

Ataques

Como hemos mencionado en el capítulo anterior, GeMSS evoluciona del esquema de firma de HFE; por tanto, todo ataque que existe contra este esquema de firma podría aplicarse también en contra de la variante estudiada. Históricamente, los ataques más eficientes contra el esquema de firma HFE han sido los ataques de tipo directo y los ataques basados en el problema de *MinRank*, previamente explicado en el capítulo 4.

En el capítulo actual, nos centraremos en un ataque contra GeMSS, publicado en los últimos años, y que está basado en *MinRank*. Previamente, en el apartado 6.1 daremos una breve explicación del ataque directo y mencionaremos otros tipos de ataques más concretos de GeMSS que han ido apareciendo desde que surgió este esquema de firma digital.

6.1. Análisis de ataques conocidos

En este apartado vamos a centrarnos en los principales ataques a GeMSS, exceptuando los basados en *MinRank*, que trataremos en el apartado siguiente y sobre los que profundizaremos más. Primero, comentaremos el ataque directo, mencionado en la introducción, y que, históricamente, ha sido uno de los más eficientes contra esquemas de firma basados en HFE. Posteriormente, dedicaremos sendos apartados a ataques cuánticos y ataques basados en bases de Gröbner, dando una idea general sobre el funcionamiento de los mismos.

Ataque directo

Antes incluso de que surgiera GeMSS, algunos autores descubrieron que los sistemas de clave pública de HFE, e incluso algunas de sus variantes, podían ser resueltos de forma mucho más sencilla que simplemente resolviendo sistemas de forma aleatoria hasta encontrar la solución válida. Este fenómeno ha sido

analizado en varios artículos, en los que los autores encontraron que el grado de regularidad de estos sistemas tiene un límite superior que es:

$$\begin{cases} (q-1)(d+v+\Delta-1)+2, & \text{si } q \text{ es par y } d+\Delta \text{ es impar,} \\ (q-1)\frac{(d+v+\Delta)}{2}+2, & \text{en caso contrario.} \end{cases}$$

donde q, d, v, Δ son los valores previamente definidos en el apartado 5.1 como parámetros de HFEv-.

En el caso particular de GeMSS, los ataques directos están enfocados en falsificar una firma generada con este algoritmo. Recordemos que la clave pública de GeMSS está dada por un conjunto de ecuaciones no lineales $\mathbf{p} = (p_1, \ldots, p_m) \in \mathbb{F}_2[x_1, \ldots, x_{n+v}]^m$. Dado $(d_1, \ldots, d_m) \in \mathbb{F}_2^m$, el problema de falsificar una firma sería equivalente a resolver el siguiente sistema de ecuaciones no lineales:

$$\begin{cases} p_1(x_1, \ldots, x_{n+v}) - d_1 = 0, \\ \vdots \\ p_m(x_1, \ldots, x_{n+v}) - d_m = 0, \\ x_1^2 - x_1 = 0, \\ \vdots \\ x_{n+v}^2 - x_{n+v} = 0. \end{cases} \tag{6.1}$$

Dicho de otra forma, la tarea consistiría en invertir la función $\mathrm{G}e\mathrm{MSS.Inv}_{\mathbf{p}}$ (vista en el Algoritmo 2), aunque sin el conocimiento de la clave secreta sk.

En nuestro caso, como se tiene que $n+v > m$, podemos fijar $n+v-m$ variables $\mathbf{r} = (r_1, \ldots, r_{n+v-m}) \in \mathbb{F}_2^{n+v-m}$ en (6.1) e intentar resolver el sistema para las variables restantes. En este caso, tendríamos que el problema de falsificar una firma queda reducido a resolver un sistema de m ecuaciones cuadráticas en m variables sobre $\mathbb{F}_2$:

$$\begin{cases} p_1(x_1, \ldots, x_m, \mathbf{r}) - d_1 = 0, \\ \vdots \\ p_m(x_1, \ldots, x_m, \mathbf{r}) - d_m = 0, \\ x_1^2 - x_1 = 0, \\ \vdots \\ x_m^2 - x_m = 0. \end{cases} \tag{6.2}$$

En [6], podemos comprobar que para resolver este tipo de sistemas, existen métodos llamados "exhaustivos" que consiguen resolver estos sistemas binarios en aproximadamente $4 \cdot \log_2(m) \cdot 2^m$ operaciones.

Ataques cuánticos

Este tipo de ataques se basan en los ataques directos, pero con la diferencia de buscar una mejora en la resolución de un sistema de m ecuaciones cuadráticas en m variables sobre $\mathbb{F}_2$, para lo que se busca una solución basada en la computación cuántica. Estos algoritmos se basan el en algoritmo de Grover, que mencionamos en el capítulo 2 de este mismo trabajo. En [6] demostraron que podemos resolver un sistema de $m-1$ ecuaciones cuadráticas binarias en $n-1$ variables binarias utilizando $m+n+2$ *qubits* y evaluando un circuito de $2^{n/2}\left(2m(n^2+2n)+1\right)$ puertas cuánticas. Los autores describen además una segunda alternativa que usa menos *qubits*, en concreto, $3+n+\lceil \log_2(m) \rceil$, pero que requiere evaluar un circuito más grande de, aproximadamente, el doble de puertas cuánticas que el caso anterior. Como ya explicamos en los primeros capítulos, es necesario desarrollar ordenadores cuánticos más potentes que los que hay ahora mismo disponibles para poder aplicar este tipo de algoritmos, pero es una posibilidad real que se puedan aplicar en menos de 10 años.

Como mencionamos en la introducción de este trabajo, se hizo una competencia hace unos años en la que se buscaba el mejor algoritmo resistente a ataques cuánticos. En el caso de GeMSS, llegó a fases bastante avanzadas del concurso, pero se descartó porque la complejidad del ataque era asumible. En este caso, la complejidad de los ataques mencionados es de

$$O\left(((mn)^k)^{\omega \cdot D_{\text{reg}}}\right), \tag{6.3}$$

para algún valor constante k, donde m y n son parámetros del sistema, ω es la llamada constante de álgebra lineal, y D_{reg} es el grado de regularidad (es el mayor grado de los polinomios que aparecen durante la resolución del sistema antes de que se empiece a generar una base de Gröbner reducida).

Ataques basados en bases de Gröbner

Por último, mencionaremos que hay varios tipos de ataques basados en bases de Gröbner. No entraremos en detalles del funcionamiento de estos ataques, ya que sería necesario desarrollar una teoría demasiado extensa sobre esas bases, pero mencionaremos algunos ejemplos.

Gracias a las bases de Gröbner existen algoritmos como `BooleanSolve`, que es uno de los algoritmos asintóticos más rápidos que existen para solucionar ecuaciones no lineales binarias. Este algoritmo tiene también una versión cuántica, `QuantumBooleanSolve` que utiliza parte del ataque cuántico visto anteriormente, combinándolo con el uso de bases de Gröbner. En general, el uso de bases de Gröbner se suele combinar con otro tipo de ataque, como el directo o el cuántico visto en apartados anteriores, para obtener una complejidad más baja y factible para realizar dichos ataques.

6.2. Ataque de recuperación de claves

Hasta ahora conocemos algunos de los ataques que se pueden realizar contra los esquemas de firma digital basados en HFE, y, concretamente, los que se pueden aplicar contra GeMSS. Todos estos ataques sirven en la mayoría de ocasiones para falsificar firmas o solucionar parte de la complejidad que genera GeMSS, pero no obtienen directamente las claves pública y privada, y las transformaciones utilizadas para generarlas. En este apartado, presentamos un ataque de recuperación de claves basado en el problema de *MinRank*, que se puede aplicar a la mayoría de algoritmos que parten del esquema de firma HFEv-. Con este ataque se demuestra que la modificación *Minus* no aporta una mayor seguridad a este sistema, mientras que la modificación *Vinegar* solo incrementa la complejidad de este ataque en un factor polinomial, algo casi irrelevante en este punto. Cabe destacar que para poder seguir el desarrollo de este apartado se recomienda comprender los conceptos explicados en el capítulo 4, en el que se presenta la explicación del problema *MinRank*.

Para el resto de este capítulo, sean q, n, v, D, Δ los parámetros del esquema de firma HFEv- que hemos definido en el capítulo 5.2, y denotamos $d = \lceil \log_q(D) \rceil$. En el caso concreto de GeMSS, habíamos determinado que $q = 2$, pero realizaremos el desarrollo de forma general por ser más completo, teniendo simplemente que sustituir q por 2 para obtener una prueba concreta para GeMSS. Asumiremos además que $0 \leq \Delta \leq n - 2d - 1$, ya que, si se diera lo contrario ($\Delta \geq n - 2d - 1$), de la forma en la que hemos definido el esquema HFEv-, se tendría que el número de ecuaciones del sistema de clave pública $(n - \Delta)$ estaría acotado superiormente por $2d + 1$. Esto supondría que, para defenderse de ataques de fuerza bruta, tendríamos que definir valores muy altos de d, lo que haría que el esquema de firma fuera impracticable.

El ataque que vamos a presentar consta de dos pasos. El primero de ellos consiste en recuperar una transformación lineal equivalente $\mathcal{S}$ resolviendo el problema de *MinRank* sobre el cuerpo base $\mathbb{F}_q$. En el segundo paso, se utiliza la transformación lineal equivalente $\mathcal{S}$ para recuperar los mapas equivalentes $\mathcal{F}$ y $\mathcal{T}$. Con ello, lograremos obtener una clave privada equivalente a la real que nos permite generar firmas idénticas a las originales para cualquier mensaje.

6.2.1. Claves equivalentes

Antes de comenzar demostrando los resultados que nos permitirán obtener $\mathcal{S}$, $\mathcal{F}$ y $\mathcal{T}$, necesitamos introducir el concepto de claves equivalentes. En el contexto de los criptosistemas de clave pública multivariantes, se define el concepto de claves equivalentes como sigue.

Definición 6.2.1. Sea $((\mathcal{T}, \mathcal{F}, \mathcal{S}), \mathcal{P})$ un par con la clave privada y pública

respectivamente de un sistema criptográfico de clave pública multivariante. Se dice que la tupla $(\mathcal{T}', \mathcal{F}', \mathcal{S}')$ es una clave privada equivalente si y solo si se cumple que

$$\mathcal{P} = \mathcal{T} \circ \mathcal{F} \circ \mathcal{S} = \mathcal{T}' \circ \mathcal{F}' \circ \mathcal{S}', \tag{6.4}$$

y $\mathcal{F}'$ es una aplicación central válida para este criptosistema, es decir, $\mathcal{F}'$ tiene la misma estructura algebraica que $\mathcal{F}$.

Teorema 6.2.1. *Sea $\mathcal{P}$ la clave pública de un esquema de firma HFEv- sobre $\mathbb{F}_q$. Sea v el número de variabes Vinegar, Δ el número de ecuaciones Minus y n el grado del cuerpo de extensión. Entonces, existen exactamente*

$$nq^{\Delta+2n+vn}(q^n-1)^2, \prod_{i=0}^{v-1}(q^v - q^i) \prod_{i=n-\Delta-1}^{n-1} (q^n - q^i) \tag{6.5}$$

claves privadas equivalentes para la clave pública $\mathcal{P}$.

Se puede consultar una demostración para este teorema en [30], la cual no explicaremos en este trabajo al necesitar la demostración de varios teoremas previos que se escapan del objetivo principal de este trabajo.

El ataque que presentaremos a continuación, nos proporciona una de estas claves equivalentes a la clave privada, cuyo funcionamiento será prácticamente análogo al del usuario con la clave privada real, permitiendo la suplantación de identidad, entre otras cosas que se podrían conseguir con este ataque.

6.2.2. Primer paso: recuperar $\mathcal{S}$

Esta primera subsección se utilizará para presentar la técnica utilizada para completar el primero de los pasos mencionados en la introducción del ataque. Conseguiremos, con los resultados presentados a continuación, obtener un sistema de ecuaciones equivalente $\mathcal{S}$, que recordemos que era una de las partes de la construcción de la clave privada que utilizamos en el capítulo anterior.

Para ello, necesitaremos recuperar una ecuación vista en el apartado 5.1.1, en el que vimos la representación matricial de las claves de HFEv-. En concreto, la ecuación (5.7), que recordemos que era la siguiente:

$$(UP_0U^t, \ldots, UP_{n-1}U^t) = (F^{*0}, \ldots, F^{*n-1})W, \tag{6.6}$$

cuya construcción encontramos en el apartado previamente mencionado 5.1.1.

Lo primero que necesitamos hacer para obtener $\mathcal{S}$ es demostrar que el lado derecho de la igualdad (6.6) es una matriz de rango $\leq d$, para, posteriormente, poder demostrar cómo recuperar $\mathcal{S}$ resolviendo el problema *MinRank*.

Proposición 6.2.1. *Sean $F^{*0}, \ldots, F^{*n-1}$ y $W = [w_{ij}]$ las matrices de la ecuación* (6.6) *y a_i la primera fila de la matriz F^{*i}, para $i = 0, 1, \ldots, n-1$. Sea Q la matriz dada por*

$$Q = W^t \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \end{pmatrix}. \tag{6.7}$$

Entonces, el rango de Q es, a lo sumo, $d = \lceil \log_q(D) \rceil$.

Demostración:

Podemos escribir

$$Q = \begin{pmatrix} w_{11}\mathbf{a}_0 + w_{21}\mathbf{a}_1 + \cdots + w_{n1}\mathbf{a}_{n-1} \\ w_{12}\mathbf{a}_0 + w_{22}\mathbf{a}_1 + \cdots + w_{n2}\mathbf{a}_{n-1} \\ \vdots \\ w_{1,n-1}\mathbf{a}_0 + w_{2,n-1}\mathbf{a}_1 + \cdots + w_{n,n-1}\mathbf{a}_{n-1} \end{pmatrix} = W^t \cdot \begin{pmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_{n-1} \end{pmatrix}.$$

Por la construcción de las matrices F^{*i}, para cada $i = 0, 1, \ldots, n-1$, tenemos

$$\begin{pmatrix} \mathbf{a}a_0 \\ \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_{n-1} \end{pmatrix} = \begin{pmatrix} A_1 \\ 0 \\ A_2 \end{pmatrix},$$

donde A_1 es una matriz de $1 \times (n+v)$ y A_2 es una matriz de dimensión $(d-1) \times (n+v)$. Con esta construcción, vemos que entre $A1$ y A_2 pueden tener, a lo sumo, d filas no nulas, por lo que su rango es como máximo d. Esto, junto con la construcción de Q, nos permite concluir que el rango de Q es como máximo d.

□

Teorema 6.2.2. *Sean ahora $P_0, P_1, \ldots, P_{n-\Delta-1}$ y U las matrices restantes que no hemos considerado antes de la ecuación* (6.6)*. La primera fila de U la denotaremos por $\mathbf{u} = (u_0, u_1, \ldots, u_{n+v-1})$, y sea $\mathbf{b}_i = (u_0, u_1, \ldots, u_{n+v-1})P_i$, para cada $i = 0, 1, \ldots, n-\Delta$. Definimos $Z \in \mathcal{M}_{(n-\Delta)\times(n+v)}(\mathbb{F}_q)$ como la matriz cuyas filas son los vectores $\mathbf{b}_i$. Entonces, el rango de Z es, como máximo, d.*

Demostración:

Esta es una demostración sencilla. Podemos deducir directamente lo enunciado en el Teorema de la ecuación (6.6) y la Proposición 6.2.1. Para ello, por la ecuación y la Proposición mencionadas, tenemos que el rango de ZU^t será, a lo sumo, d. Conocido esto, es obvio que el rango de Z es como máximo d.

□

Proposición 6.2.2. *Sea $A = [a_{ij}]$ una matriz $n \times m$ sobre $\mathbb{F}_q$, y sea $B = M^{-1}A = [b_{ij}] \in \mathcal{M}_{n\times m}(\mathbb{F}_{q^n})$. Entonces,*

$$b_{ij} = b_{i-1,j}^q, \quad \text{para todo } i,j; \text{ con } 0 \leq i < n, 0 \leq j < m.$$

Es decir, cada fila se obtiene a partir de la anterior utilizando una aplicación de Frobenius. Por lo tanto,la matriz B quedaría definida al completo conociendo una cualquiera de sus filas.

Demostración:

Sea $(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n)$ una base dual de $(\theta_1, \theta_2, \ldots, \theta_n)$ de $\mathbb{F}_{q^n}$ sobre $\mathbb{F}_q$, entonces tenemos que la matriz M^{-1} queda definida como

$$M^{-1} = \begin{pmatrix} \varepsilon_1 & \varepsilon_2 & \cdots & \varepsilon_n \\ \varepsilon_1^q & \varepsilon_2^q & \cdots & \varepsilon_n^q \\ \vdots & \vdots & \ddots & \vdots \\ \varepsilon_1^{q^{n-1}} & \varepsilon_2^{q^{n-1}} & \cdots & \varepsilon_n^{q^{n-1}} \end{pmatrix}.$$

Entonces, por como hemos definido la matriz B, tenemos que:

$$b_{ij} = \sum_{k=0}^{n-1} a_{kj} \cdot \varepsilon_{k+1}^{q^j},$$

para todo i, j, con $0 \leq i < n$, y $0 \leq j < m$.

Para concluir, como $a_{ij} \in \mathbb{F}_q$, entonces $a_{ij}^q = a_{ij}$ y por la linealidad de Frobenius, tenemos

$$b_{i-1,j}^q = \left(\sum_{k=0}^{n-1} a_{kj} \cdot \varepsilon_{k+1}^{q^{i-1}}\right)^q = \sum_{k=0}^{n-1} a_{kj}^q \cdot (\varepsilon_{k+1}^{q^{i-1}})^q = \sum_{k=0}^{n-1} a_{kj} \cdot \varepsilon_{k+1}^{q^i} = b_{ij}.$$

para todo i, j tales que $0 \leq i < n$, $0 \leq j < m$. Y quedaría probada la proposición.

□

Recordemos que la matriz $\widetilde{M}$ la habíamos definido en el apartado 5.1.1 en la ecuación 5.5, como:

$$\widetilde{M} = \begin{pmatrix} M & 0 \\ 0 & I_v \end{pmatrix} \in \mathcal{M}_{(n+v)\times(n+v)}(\mathbb{F}_{q^n}). \tag{6.8}$$

La Proposición 5 implica que solo necesitamos encontrar una fila de la matriz $U = \widetilde{M}^{-1}S^{-1}$ para recuperar las primeras n filas de U. Llamemos

$u_0, u_1, \ldots, u_{n+v-1}$ a los valores que componen la primera fila de U, suponiendo que estos valores $u_0, u_1, \ldots, u_{n+v-1}$ son desconocidos. En HFEv- tenemos que hay varias claves privadas que son equivalentes y producen la misma firma digital, por lo que necesitaríamos encontrar solo una de esas claves privadas equivalentes. Para ello, podemos fijar $u_0 = 1$, y, dado que el rango de Z es como máximo d, podemos encontrar el resto de valores u_i, $i = 1, \ldots, n + v - 1$ resolviendo el problema *MinRank*. Para resolver este problema, se puede utilizar alguno de los métodos expuestos en el capítulo 4 referentes a solucionar los problemas *MinRank*. En resumen, podemos consultar uno a uno los pasos que hay que realizar para obtener $\mathcal{S}$ en el algoritmo 5.

Algorithm 5 Primera parte del ataque: recuperación de $\mathcal{S}$

Input: Parámetros HFEv- (q, n, v, D, Δ), matrices $(P_0, \ldots, P_{n-\Delta-1})$ que representan las formas cuadráticas de los polinomios de la clave pública, matriz $\widetilde{M}$ (ver ecuación (6.8)).
Output: Transformación lineal equivalente S.

1. Definir $\mathbf{b}_i = (1, u_1, \ldots, u_{n+v-1})P_i$, donde $0 \leq i < n-\Delta$, y $(u_1, \ldots, u_{n+v-1})$ son desconocidos.

2. Construir una matriz Z cuyas filas son los $\mathbf{b}_i$, para $0 \leq i < n - \Delta$. Según el Teorema 6.2.2, el rango de Z es a lo sumo d.

3. Resolver el problema de *MinRank* con la matriz Z usando un método de los mencionados en el capítulo 4. Denotar la solución como $u_0, u_1, \ldots, u_{n+v-1}$.

4. Una vez obtenida la solución, podemos definir la matriz U como:

$$U = \begin{pmatrix} u_0 & u_1 & \ldots & u_{n+v-1} \\ u_0^q & u_1^q & \ldots & u_{n+v-1}^q \\ \vdots & \vdots & \ddots & \vdots \\ u_0^{q^{n-1}} & u_1^{q^{n-1}} & \ldots & u_{n+v-1}^{q^{n-1}} \\ r_{00} & r_{01} & \ldots & r_{0,n+v-1} \\ \vdots & \vdots & \ddots & \vdots \\ r_{\gamma-1,0} & r_{01} & \ldots & r_{0,n+v-1} \end{pmatrix}$$

   donde r_{ij} con $0 \leq i < v,\ 0 \leq j < n + v$ son valores aleatorios en $\mathbb{F}_q$, elegidos de forma que nos aseguremos que U sea invertible.

5. Calcular $S' = (\widetilde{M}U)^{-1}$.

6. Devolvemos S', que será el elemento buscado.

6.2.3. Segundo paso: recuperar $\mathcal{F}$ y $\mathcal{T}$

Una vez hemos podido encontrar una transformación lineal equivalente $\mathcal{S}$, estamos en disposición de llevar a cabo la segunda parte del ataque. En esta subsección, explicaremos los resultados necesarios para comprender el flujo de esta segunda parte, en la que el objetivo es recuperar las transformaciones equivalentes $\mathcal{F}$ y $\mathcal{T}$, para lo que necesitaremos resolver varios sistemas de ecuaciones no lineales.

Algorithm 6 Segunda parte del ataque: recuperación de $\mathcal{F}$ y $\mathcal{T}$

Input: Parámetros HFEv- (q, n, v, D, Δ), matrices $(P_0, \ldots, P_{n-\Delta-1})$ que representan las formas cuadráticas de los polinomios de la clave pública, matriz M (ver apartado 5.1.1), Transformación lineal equivalente S previamente recuperada.
Output: Transformaciones lineales equivalentes F y T.

1. Sean $w_1, w_2, \ldots, w_{n-\Delta}$ desconocidos y tomemos $w_0 = 1$. Debemos obtener un sistema lineal con $d(n-\Delta)$ ecuaciones en las $n-\Delta-1$ variables w_i con $1 \leq i \leq n-\Delta$ de la ecuación matriz (6.12), como se muestra en la demostración de la Proposición 6.2.3. Resolviendo este sistema se obtiene una solución $w'_1, w'_2, \ldots, w'_{n-\Delta}$ con $w'_0 = 1$.

2. Sean $l_1, \ldots, l_\Delta$ y las entradas distintas de cero de F^{*0} desconocidas en la ecuación matricial (6.14). Se obtienen $(d+\Delta)\cdot(n+v)$ ecuaciones bilineales de las primeras $d+\Delta$ filas de (6.14) y $\binom{v+1}{2}$ ecuaciones polinomiales en una variable de las últimas v filas. Resolviendo tanto los sistemas lineales definidos como las ecuaciones mencionadas, conseguimos recuperar F^{*0}, tal como se puede ver en la Proposición 6.2.3. Entonces, podemos encontrar una aplicación central equivalente con

$$F' = (X, X^q, \ldots, X^{q^{n-1}}, x_1, \ldots, x_v)F^{*0}(X, X^q, \ldots, X^{q^{n-1}}, x_1, \ldots, x_v)^t. \tag{6.9}$$

3. Calcular F^{*k} , para $1 \leq k < n$, siguiendo lo definido en la Proposición 5.1.1 definida en el apartado 5.1.1.

4. Denotamos por $(t_{1k}, t_{2k}, \ldots, t_{nk})$ a las entradas de la k-ésima columna de T, que es desconocida, para $k = 1, 2, \ldots, n-r$. Bataría entonces con resolver $n-r$ sistemas lineales que se pueden obtener tal y como hemos definido en la Proposición 6.2.4. Con esta resolución, obtendríamos una aplicación equivalente T, tal como estábamos buscando.

5. Devolver F', T.

Proposición 6.2.3. *Supongamos que, al igual que en el apartado anterior, disponemos de las matrices de la ecuación* (6.6)*, y tenemos definidos todos los parámetros de HFEv- necesarios. Suponiendo que U es conocido, entonces F^{*0} se puede recuperar resolviendo un sistema lineal con $n-\Delta-1$ variables, $(d+\Delta)\cdot(n+v)$ ecuaciones lineales adicionales con, a lo sumo, $d+v$ variables y $\binom{v+1}{2}$ ecuaciones polinómicas cuadráticas de grado q^d.*

Demostración:

Comenzaremos fijándonos de nuevo en la ecuación (6.6). De ahí, obtenemos que $W = M^{-1}T \in \mathcal{M}_{n\times(n-\Delta)}(\mathbb{F}_q)$. Podemos expresar W como

$$W = \begin{pmatrix} W_1 \\ W_2 \end{pmatrix},$$

donde $W_1 \in \mathcal{M}_{\Delta\times(n-\Delta)}(\mathbb{F}_q)$ y $W_2 \in \mathcal{M}_{(n-\Delta)\times(n-\Delta)}(\mathbb{F}_q)$. Por como hemos construido estas matrices anteriormente, sabemos que M es invertible y los valores que componen T son elegidos aleatoriamente de $\mathbb{F}_q$, por lo que la probabilidad de que W_2 sea singular es

$$1 - \prod_{i=1}^{n-\Delta} \left(1 - \frac{1}{q^i}\right).$$

Según el Teorema 6.2.1, hay, al menos, q^n transformaciones equivalentes a T, por lo que la probabilidad de que todas las matrices W_2 asociadas a las matrices equivalentes T sean singulares es aproximadamente

$$\left(1 - \prod_{i=1}^{n-\Delta} \left(1 - \frac{1}{q^i}\right)\right)^{q^n}.$$

Por lo tanto, podemos encontrar una matriz invertible W_2 con una probabilidad muy alta, casi de forma segura. Multiplicamos ambos lados de la ecuación (6.6) por W_2^{-1}, obtenemos

$$(UP_0U^t, \ldots, UP_{n-\Delta-1}U^t)W_2^{-1} = (F^{*0}, \ldots, F^{*n-1}) \begin{pmatrix} W_1W_2^{-1} \\ I_{n-\Delta} \end{pmatrix}, \tag{6.10}$$

donde $I_{n-\Delta}$ es la matriz identidad de tamaño $(n-\Delta)\times(n-\Delta)$.

Llamaremos ahora $(\widetilde{w}_0, \widetilde{w}_1, \ldots, \widetilde{w}_{n-\Delta-1})$ a la primera columna de W_2^{-1} y $(\widetilde{l}_0, \widetilde{l}_1, \ldots, \widetilde{l}_{\Delta-1}, 1, 0, \ldots, 0)$ la primera columna de $\begin{pmatrix} W_1W_2^{-1} \\ I_{n-\Delta} \end{pmatrix}$.

Entonces, la ecuación (6.10) la podemos expresar como

$$\sum_{k=0}^{n-\Delta-1} \widetilde{w}_k UP_kU^t = \sum_{i=0}^{\Delta-1} \widetilde{l}_i F^{*i} + F^{*\Delta}. \tag{6.11}$$

Multiplicamos ahora ambos lados por $\widetilde{l}_0^{-1}$, obteniendo

$$\sum_{k=0}^{n-\Delta-1} \widetilde{l}_0^{-1}\widetilde{w}_k UP_kU^t = F^{*0} + \sum_{i=1}^{\Delta-1} \widetilde{l}_0^{-1}\widetilde{l}_i F^{*i} + \widetilde{l}_0^{-1}F^{*\Delta}.$$

Y denotando $w_k = \widetilde{l}_0^{-1}\widetilde{w}_k$ para $k = 0, 1, \ldots, n-\Delta-1$ y $l_i = \widetilde{l}_0^{-1}\widetilde{l}_i$ para $i = 1, 2, \ldots, \Delta-1$, y siendo $l_\Delta = \widetilde{l}_0^{-1}$, se tiene

$$\sum_{k=0}^{n-\Delta-1} w_k U P_k U^t = \sum_{i=1}^{\Delta} l_i F^{*i} + F^{*0}. \tag{6.12}$$

Nótese que, si expresamos el segundo término de la igualdad de (6.12) como una matriz, obtenemos

$$\sum_{i=1}^{\Delta} l_i F^{*i} + F^{*0} = \begin{pmatrix} F_0' & 0 & F_1' \\ 0 & 0 & 0 \\ F_1'^t & 0 & F_2' \end{pmatrix} \in \mathcal{M}_{(n+v)\times(n+v)}(\mathbb{F}_{q^n}), \tag{6.13}$$

donde $F_0' = [f_{ij}']$ es una matriz de banda diagonal simétrica de tamaño $(d+\Delta)\times(d+\Delta)$ con anchura de banda $2d-1$, es decir,

$$f_{ij}' = 0, \text{ si } |i-j| \geq d.$$

Además, $F_1' \in \mathcal{M}_{(d+\Delta)\times v}(\mathbb{F}_{q^n})$, con $F_1'^t \in \mathcal{M}_{v\times(d+\Delta)}(\mathbb{F}_{q^n})$ su transpuesta. Por último, de las submatrices definidas, también tenemos $F_2' \in \mathcal{M}_{v\times v}(\mathbb{F}_{q^n})$ que es una matriz simétrica.

Supongamos que $w_0, w_1, \ldots, w_{n-\Delta-1}$ son desconocidos. Dado que solo necesitamos encontrar una de las claves privadas equivalentes HFEv-, podemos fijar $w_0 = 1$. Debido al hecho de que U es conocido y a la estructura especial de la matriz definida en (6.13), obtenemos de la ecuación (6.12) $d(n-\Delta)-d$ ecuaciones lineales en las $n-\Delta-1$ variables $w_1, w_2, \ldots, w_{n-\Delta-1}$. Como $0 < \Delta < n-2d-1$, tenemos que

$$d(n-\Delta-d) \geq n-\Delta-1.$$

Si planteamos la resolución de estas ecuaciones lineales, podemos obtener una solución $(w_0', w_1', w_2', \ldots, w_{n-\Delta-1}')$ con $w_0' = 1$. Así, la ecuación (6.12) se puede reescribir como

$$\sum_{k=0}^{n-\Delta-1} w_k' U P_k U^t = \sum_{i=1}^{\Delta} l_i F^{*i} + F^{*0}. \tag{6.14}$$

El siguiente paso será encontrar $l_1, \ldots, l_\Delta$ y F^{*0} de la ecuación anterior. Sabemos que F^{*0} tiene la forma

$$F^{*0} = \begin{pmatrix} F_0 & 0 & F_1 \\ 0 & 0 & 0 \\ F_1^t & 0 & F_2 \end{pmatrix},$$

donde:

- $F_0 = [\alpha_{ij}] \in \mathcal{M}_{d\times d}(\mathbb{F}_{q^n})$ es una matriz simétrica,
- $F_1 = [\gamma_{ij}] \in \mathcal{M}_{d\times v}(\mathbb{F}_q)$,
- $F_1^t \in \mathcal{M}_{v\times d}(\mathbb{F}_q)$ es la transpuesta de F_1 y
- $F_2 = [\delta_{ij}] \in \mathcal{M}_{v\times v}(\mathbb{F}_q)$ es una matriz simétrica.

Según la Proposición 5.1.1, vista en el apartado 5.1.1, podemos representar F^{*k}, para todo $1 \leq k \leq n-1$ por los elementos de F^{*0}, por lo que nos bastaría con conocer los elementos de esta última matriz.

Para conseguir recuperar F^{*0}, supondremos que tanto $l_1, \ldots, l_\Delta$, como todos los elementos de las matrices F_0, F_1 y F_2, que son α_{ij} con $0 \leq i \leq j < d$, γ_{ij} con $0 \leq i < d$, $0 \leq j < v$, y δ_{ij} con $0 \leq i \leq j < v$, son desconocidos. Entonces, debemos seguir los siguientes pasos para poder recuperar F^{*0} tal como queremos:

1. A partir de la primera fila de la ecuación matricial (6.14), podemos encontrar un sistema lineal en las variables α_{0j} con $0 \leq j < d$, y γ_{0j} donde $0 \leq j < v$; de la forma:

$$\begin{cases} \alpha_{00} + \theta_{00} = 0, \\ \vdots \\ \alpha_{0,d-1} + \theta_{0,d-1} = 0, \\ \gamma_{00} + \theta_{0,d} = 0, \\ \vdots \\ \gamma_{0,v-1} + \theta_{0,d+v-1} = 0. \end{cases} \tag{6.15}$$

   Resolviendo este sistema lineal, podremos obtener la primera fila de F^{*0}, y poder empezar a construir la matriz que estamos buscando.

2. El segundo paso nos permitirá encontrar la segunda fila de F^{*0}, una vez hayamos obtenido los valores de la primera. Para ello, de forma similar a la primera, podemos utilizar de la segunda fila de (6.14) para obtener un sistema lineal en las variables l_1, α_{1j} con $1 \leq j < d$ y γ_{1j} donde en este caso $1 \leq j < v$. Al resolver este sistema lineal, podemos obtener la segunda fila de F^{*0} y, además, el valor de l_1.

3. Razonando de forma similar para todo $\Delta \leq d$, podemos obtener $l_1, \ldots, l_\Delta$, F_0 y F_1 a partir de las primeras d filas de la ecuación matricial (6.14). Si $\Delta > d$, podemos obtener $l_1, \ldots, l_d$, F_0 y F_1 usando las primeras d filas de la misma ecuación, y, por otro lado, cada l_{d+k} para $1 \leq k \leq \Delta - d$ a partir

de las $(d+k)$-ésima fila de (6.14). Con todo este proceso, conseguiremos obtener $l_1, \ldots, l_\Delta$, y además F_0 y F_1.

4. Una vez que conocemos $l_1, \ldots, l_\Delta$, F_0 y F_1, nos quedaremos en este caso con las últimas v filas de la ecuación matricial que venimos utilizando, que son $\binom{v+1}{n}$ ecuaciones polinomiales en una variable de la forma

$$\sum_{k=0}^{d} \lambda_{ijk} \delta_{ij}^{q^k} + \eta_{ij} = 0,$$

donde $\lambda_{ijk}, \eta_{ij} \in \mathbb{F}_{q^n}$, para $0 \leq i \leq j < v$. Al resolver estas ecuaciones, podemos obtener δ_{ij} y, con ello, recuperar por completo F^{*0} como estábamos buscando.

Una vez conocido F^{a0}, podemos obtener una aplicación central equivalente válida como:

$$\begin{aligned} &F'(X, x_1, x_2, \ldots, x_v) = \\ &= (X, X^q, \ldots, X^{q^{n-1}}, x_1, x_2, \ldots, x_v) F^{*0} (X, X^q, \ldots, X^{q^{n-1}}, x_1, x_2, \ldots, x_v)^t. \end{aligned}$$

Y quedará probado. □

Proposición 6.2.4. *En el contexto de la Proposición 5.1.3, supongamos que S, P_i con $0 \leq i < n - \Delta$, M, F^{*j} con $0 \leq j < n$ son conocidos. Entonces T puede recuperarse resolviendo $n - \Delta$ sistemas lineales en n variables.*

Demostración:

En la Proposición 5.1.3, vista en el apartado 5.1.1, teníamos que se cumplía la siguiente ecuación:

$$\begin{aligned} &\left(\widetilde{M}^{-1} S^{-1} P_0 (S^{-1})^t (\widetilde{M}^{-1})^t, \ldots, \widetilde{M}^{-1} S^{-1} P_{n-\Delta-1} (S^{-1})^t (\widetilde{M}^{-1})^t \right) = \\ &= (F^{*0}, \ldots, F^{*n-1}) M^{-1} T \end{aligned} \tag{6.16}$$

Podemos reescribir esta ecuación como:

$$(P_0, \ldots, P_{n-\Delta}) = (SMF^{*0} M^t S^t, \ldots, SMF^{*n-1} M^t S^t) M^{-1} T. \tag{6.17}$$

Sean $(t_{1k}, t_{2k}, \ldots, t_{nk})$ las entradas de la k-ésima columna de T, para $k = 1, 2, \ldots, n - \Delta$. Como S, P_i para $0 \leq i < n - \Delta$, M y F^{*j} con $0 \leq j < n$ son conocidos, obtenemos de (6.17) un sistema lineal con $\frac{n(n+1)}{2}$ ecuaciones en las n variables $(t_{1k}, t_{2k}, \ldots, t_{nk})$ para todo $k = 1, 2, \ldots, n - \Delta$. Finalmente, podríamos recuperar T resolviendo tan solo $n - \Delta$ de estos sistemas lineales.

□

Con este resultado, finalizamos el proceso de recuperación de la clave privada, al haber obtenido ya tanto la aplicación central $\mathcal{F}$ como $\mathcal{T}$, que era lo que estábamos buscando en esta segunda parte del ataque. Para resumirlo, se puede consultar el algoritmo 6, en el que encontramos los pasos necesarios para conseguir el objetivo de esta parte.

6.2.4. Complejidad del ataque

En este apartado, analizaremos la complejidad del ataque planteado en el apartado anterior. El paso más complejo de este ataque es el paso 3 del Algoritmo 5, que es el paso en el que se resuelve el problema *MinRank* en la matriz Z, que tiene rango a lo sumo d. Podemos resolverlo utilizando modelado de menores o modelado de soporte de menores.

Si usamos modelado de menores, el grado de regularidad para resolver el sistema público usando el algoritmo F4 es $d+1$. Por lo tanto, la complejidad de nuestro ataque usando modelado de menores es

$$\mathcal{O}\left(\binom{n+v+d+1}{d+1}^{\omega}\right), \tag{6.18}$$

donde $2 < \omega \leq 3$ es la constante del álgebra lineal.

6.3. Conclusiones

Ahora que hemos completado todo el desarrollo que queríamos hacer sobre este tema, podemos echar la vista atrás a la conclusión que hicimos al inicio del mismo, para dar unas pequeñas conclusiones acerca de cómo se plantea en el ámbito de la criptografía la aparición de los ordenadores cuánticos.

Como ya hemos mencionado en varias ocasiones, la criptografía post-cuántica emerge como una respuesta fundamental ante los avances vertiginosos de la computación cuántica, que pone en riesgo la seguridad de los sistemas tradicionales. Como vimos con el Algoritmo de Shor, la potencial ruptura de algoritmos ampliamente utilizados como RSA y ECDSA, hace que la investigación actual esté orientada a la creación de nuevos sistemas criptográficos que resistan los ataques de los ordenadores cuánticos.

En el contexto de la criptografía multivariante, hemos analizado la seguridad del esquema GeMSS que, como hemos visto, está totalmente amenazado por varios ataques que cuestionan su seguridad tal y como está planteado actualmente. Hemos visto que incluso existen ataques cuánticos sobre este esquema, lo que hace improbable que sea, actualmente, una opción de reemplazo de los

esquemas actuales. De hecho, este algoritmo superó 3 rondas en el concurso del NIST, mencionado en la introducción de este trabajo, pero fue retirado en la cuarta ronda por las amenazas a su seguridad que causan los ataques que se han descrito. Con esto, se quiere subrayar la necesidad de seguir investigando soluciones más robustas, poniendo de manifiesto que la criptografía multivariante aún enfrenta obstáculos significativos, y que la búsqueda de un cifrado multivariante eficiente e inmune a los ataques cuánticos sigue siendo un desafío abierto en la comunidad científica.

Con ello, concluimos que el rápido desarrollo de la computación cuántica indica que estamos al borde de una nueva era, en la que se hace indispensable una transición hacia sistemas criptográficos diseñados para ser resistentes a las capacidades de los ordenadores cuánticos. Sin embargo, la evolución de estos algoritmos aún está en sus primeras fases, y se necesita más investigación y pruebas para consolidar la seguridad a largo plazo de estos sistemas.

Apéndice A

Algoritmo de Berlekamp

El algoritmo de Berlekamp [2] es un método eficiente para factorizar polinomios sobre cuerpos finitos, particularmente $\mathbb{F}_q$, donde $q = p^m$ con p primo. Dado un polinomio $f(X) \in \mathbb{F}_q[X]$ de grado n, este algoritmo permite descomponerlo en factores irreducibles. Se basa en la estructura del anillo cociente $\mathbb{F}_q[X]/(f(X))$ y en propiedades del operador de Frobenius.

Sea $f(X) \in \mathbb{F}_q[X]$ un polinomio libre de cuadrados (sin factores repetidos) de grado n. El algoritmo construye el llamado espacio de Berlekamp, un subespacio de dimensión r del álgebra cociente $\mathbb{F}_q[X]/(f(X))$, y encuentra factorizaciones no triviales de $f(X)$ usando raíces del polinomio $x^q - x$ módulo $f(X)$. En cada paso se obtiene una descomposición $f = \gcd(f, g)$ y f/g para algún g, y el proceso se repite recursivamente sobre los factores no irreducibles.

Definición A.0.1 (Espacio de Berlekamp)**.** El espacio de Berlekamp asociado a $f(X)$ es:

$$\mathcal{B}_f := \{h(X) \in \mathbb{F}_q[X]/(f(X)) \mid h(X)^q \equiv h(X) \mod f(X)\}.$$

Es un subespacio vectorial de $\mathbb{F}_q^n$.

Cabe destacar que este algoritmo es aplicable cuando el polinomio $f(X)$ es libre de cuadrados. Si no lo es, primero se puede aplicar una factorización libre de cuadrados (podemos encontrar un ejemplo en [19]), fundamental en álgebra computacional. Este es un paso central en algoritmos de criptografía post-cuántica como GeMSS, donde se requiere encontrar raíces de un polinomio sobre $\mathbb{F}_{2^n}$.

Algorithm 7 Factorización de un polinomio mediante el algoritmo de Berlekamp

1: **function** BERLEKAMP($f(X)$)
Require: $f(X) \in \mathbb{F}_q[X]$, libre de cuadrados
Ensure: Lista de factores irreducibles de $f(X)$
2: Construir el espacio vectorial $\mathcal{B}_f$:
Sea V el conjunto de polinomios $h(X)$ tales que:
$h(X)^q \equiv h(X) \mod f(X)$
Esto equivale a resolver un sistema lineal sobre $\mathbb{F}_q$ con n incógnitas
3: Sea $\{h_1(X), \ldots, h_r(X)\}$ una base de $\mathcal{B}_f$ tal que $h_1(X) = 1$
4: Inicializar lista de factores: $\mathcal{F} \leftarrow \{f(X)\}$
5: **while** exista $g(X) \in \mathcal{F}$ no irreducible **do**
6: Elegir $g(X) \in \mathcal{F}$ no irreducible
7: Elegir al azar $a(X) \in \langle h_2(X), \ldots, h_r(X)\rangle$
8: Calcular $d(X) = \gcd(a(X), g(X))$
9: **if** $1 < \deg d(X) < \deg g(X)$ **then**
10: Reemplazar $g(X)$ en $\mathcal{F}$ por $d(X)$ y $g(X)/d(X)$
11: **end if**
12: **end while**
13: **return** $\mathcal{F}$ ▷ Todos los factores irreducibles de $f(X)$
14: **end function**

Bibliografía

[1] Miklós Ajtai. "Generating Hard Instances of Lattice Problems". En: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. ACM. 1996, págs. 99-108.

[2] Elwyn R. Berlekamp. "Factoring Polynomials Over Finite Fields". En: *Bell System Technical Journal* 46.8 (1967), págs. 1853-1859.

[3] Chris Bernhardt. *Quantum computing for everyone*. Mit Press, 2019.

[4] Daniel J Bernstein y Tanja Lange. "Post-Quantum Cryptography". En: *Nature* (2017).

[5] Luk Bettale, Jean-Charles Faugere y Ludovic Perret. "Cryptanalysis of HFE, Multi-HFE and Variants for Odd and Even Characteristic". En: *Designs, Codes and Cryptography* 69 (2013), págs. 1-52.

[6] Charles Bouillaguet, Hsieh-Chung Chen, Chen-Mou Cheng, Tung Chou, Ruben Niederhagen, Adi Shamir y Bo-Yin Yang. "Fast Exhaustive Search for Polynomial Systems in $\mathbb{F}_2$". En: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2010, págs. 203-218.

[7] Jonathan F Buss, Gudmund S Frandsen y Jeffrey O Shallit. "The Computational Complexity of Some Problems of Linear Algebra". En: *Journal of Computer and System Sciences* 58.3 (1999), págs. 572-596.

[8] Antoine Casanova, Jean-Charles Faugere, Gilles Macario-Rat, Jacques Patarin, Ludovic Perret y Jocelyn Ryckeghem. "GeMSS: A Great Multivariate Short Signature". Tesis doct. UPMC-Paris 6 Sorbonne Universités; INRIA Paris Research Centre, MAMBA Team ..., 2017.

[9] Nicolas T. Courtois, Matthieu Finiasz y Nicolas Sendrier. "How to Achieve a McEliece-Based Digital Signature Scheme". En: *Advances in Cryptology—ASIACRYPT 2001*. Springer, 2001, págs. 157-174.

[10] Whitfield Diffie y Martin E. Hellman. "New Directions in Cryptography". En: *IEEE Transactions on Information Theory* 22.6 (1976), págs. 644-654.

[11] Yumin Dong, Hengrui Liu, Yanying Fu y Xuanxuan Che. "Improving the Success Rate of Quantum Algorithm Attacking RSA Encryption System". En: *Journal of Applied Physics* 134.2 (2023). Incluye imagen esquemática del algoritmo de Shor.

[12] Jean-Charles Faugere. "A New Efficient Algorithm for Computing Gröbner Bases (F4)". En: *Journal of pure and applied algebra* 139.1-3 (1999), págs. 61-88.

[13] Josep Domingo Ferrer. *Criptosistemas de clave pública*. Apuntes del Módulo 5, Universitat Oberta de Catalunya. 2015.

[14] Josep Domingo Ferrer. *Firmas digitales*. Apuntes del Módulo 6, Universitat Oberta de Catalunya. 2015.

[15] Richard P. Feynman. "Simulating Physics with Computers". En: *International Journal of Theoretical Physics* 21.6-7 (1982), págs. 467-488.

[16] Joachim von zur Gathen y Jürgen Gerhard. *Modern Computer Algebra*. 3.ª ed. Cambridge: Cambridge University Press, 2013.

[17] Lov K Grover. "From Schrödinger's Equation to the Quantum Search Algorithm". En: *American Journal of Physics* 69.7 (2001), págs. 769-777.

[18] Aviad Kipnis y Adi Shamir. "Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization". En: *Advances in Cryptology—CRYPTO'99*. Springer, 1999, págs. 19-30.

[19] Rudolf Lidl y Harald Niederreiter. "Square-Free Factorization". En: *Finite Fields*. Cambridge university press, 1997. Cap. 8.

[20] José Ignacio Farrán Martín. *Apuntes de Criptografía*. Material de clase sobre clave pública y RSA. Universidad de Valladolid, 2024.

[21] Robert J McEliece. "A Public-Key Cryptosystem Based on Algebraic". En: *Coding Thv* 4244.1978 (1978), págs. 114-116.

[22] Ralph C. Merkle. "Secure Communications Over Insecure Channels". En: *Communications of the ACM* 21.4 (1978), págs. 294-299.

[23] Michael A Nielsen e Isaac L Chuang. *Quantum Computation and Quantum Information*. 10th. Cambridge university press, 2010.

[24] Ronald L. Rivest, Adi Shamir y Leonard M. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". En: *Communications of the ACM* 21.2 (1978), págs. 120-126.

[25] Kenneth H. Rosen. *Elementary Number Theory and Its Applications*. 5th. Ver capítulo sobre congruencias, sección del Teorema Chino del Resto. Addison-Wesley, 2005.

[26] Peter W Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". En: *SIAM review* 41.2 (1999), págs. 303-332.

[27] Chengdong Tao, Albrecht Petzoldt y Jintai Ding. "Efficient Key Recovery for All HFE Signature Variants". En: *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I 41*. Springer. 2021.

[28] Johan Van Benthem, Amitabha Gupta y Rohit Parikh. *Proof, Computation and Agency: Logic at the Crossroads*. Vol. 352. Springer Science & Business Media, 2011.

[29] Françoise Levy-dit Vehel, J-Ch Jean-Charles Faug'ere y Ludovic Perret. "Cryptanalysis of MinRank". En: *Advances in Cryptology-CRYPTO*. Citeseer. 2008.

[30] Christopher Wolf y Bart Preneel. "Equivalent Keys in Multivariate Quadratic Public Key Systems". En: *Journal of Mathematical Cryptology* 4.4 (2011), págs. 375-415.