Instituto Politécnico Nacional
Escuela Superior de Física y Matemáticas

Esquemas de Firma Digital Postcuánticos

Basados en NTRU

Tesis
que para obtener el título de
Licenciado en Física y Matemáticas

PRESENTA

Antonio Osorio Cruz

Asesor

Doctor Luis Carlos Coronado García

México D. F.

Febrero 2007

A Teofilo
por ser el padre más padre que he conocido
y
Martha,
a quienes debo más de lo que puedo decir.

A Lucciano Hadi Zubyn
y
a Mitzi Fiorile.

Agradecimientos

al Doktor Carlos Coronado por su apoyo y guía todos estos "días" y por animarme a hacer esta tésis, a su esposa Vero por su paciencia,

a mis hermanos Bernabel, Siria y Francisca, por creer en mi, pues sin su apoyo y comprensión no hubiera habido tésis,

a la Sra. Ana María Sánchez (tia) por darme de comer, además de casa durante muchos años, algo que nunca podré pagarle,

a Bertha por la ayuda en la captura y el apoyo, pero sobre todo por ser quien es,

a Hulda Sollano, siempre estaré en deuda contigo,

a todos mis amigos, gracias por estar ahí y hacer de este viaje algo agradable (no pongo nombres porque son muchos, pero cada uno de ustedes sabe el lugar que ocupa) y por compartir tantos desmadres,

a Lupita, que siempre ocupará un lugar especial en mi corazón; por apoyarme en más de un momento difícil de mi vida, y por supuesto por acompañarme en muchos muy felices,

al Ingeniero Químico Felipe García Silva, gracias, nunca olvidaré sus palabras.

Contenido

Introducción

La criptografía es una rama de las ciencias que se ocupa de la "escritura oculta". Un problema criptográfico es el referente a la privacidad: impedir la obtención no autorizada de información procedente de comunicación a través de un canal inseguro impera el uso de la criptografía para asegurar la privacidad. En criptografía se desarrollan y estudian esquemas que imposibiliten al máximo el decifrado de mensajes secretos a aquellos a quién no estén dirigidos, aún estando los mensajes en un medio abierto, pero cifrados. Un aspecto central en criptografía es la seguridad que otorga el esquema a emplear. Informalmente hablando, un esquema es seguro si para descifrar un mensaje secreto u obtener información importante de él, le cuesta a un "intruso" muchos recursos y, sobre todo, más tiempo del que se desea o necesita que permanezca en secreto el mensaje.

Para la comunicación entre dos partes se puede compartir una clave (secreta) que no sea conocida por nadie más, lo cual se puede hacer mediante el envío previo de la clave a través de un canal seguro. La comunicación entre las partes puede entonces realizarse utilizando un cifrado simétrico.

El uso de cifrado simétrico implica una administración de claves enorme. En un ámbito con n usuarios, es necesaria la utilización de $\binom{n}{2}$ claves simétricas. La estructura en internet hace necesario que el usuario pueda identificarse y que sus mensajes puedan ser tanto autenticados como confidenciales. Diffie y Hellman introdujeron en su artículo [DH76] una nueva forma de criptografía que es adecuada para su aplicación en ámbitos con muchos participantes. Un punto importante en esta corriente es la restricción en el poder de cómputo que se le atribuye al "intruso".

Criptografía, la ciencia de la comunicación secreta, está creciendo desmesuradamente con el desarrollo de redes de computadoras y transacciones electrónicas. Cuando información de seguridad personal, financiera, militar o nacional es transferida de un lugar a otro, es vulnerable a tácticas de espionaje lo cual puede tener consecuencias catastróficas.

Tales problemas pueden ser evadidos cifrando la información a fin de que

resulten ininteligibles a terceras partes. Esto puede ser logrado si la persona que envía el mensaje y la que lo recibe poseen una secuencia secreta aleatoria de bits (número binario), conocida como material "clave", el cual es usado para cifrar (quien envía el mensaje) y para descifrar (quien lo recibe).

El material clave es por tanto una fuente importante aunque no contenga información por sí mismo. De cualquier manera, el paso inicial de la "distribución de claves", por medio del cual las dos partes adquieren el material clave, debe poseer un alto grado de confidencialidad que terceras partes no puedan adquirir información parcial acerca de la secuencia aleatoria de bits. Con una comunicación convencional, que puede estar sujeta a monitoreo pasivo por parte de espías, es imposible crear una clave secreta certificable, y así se requieren medidas de seguridad física embarazosas para la generación, distribución, almacenamiento, y destrucción de claves después de ser usadas.

La Criptografía es usada para ocultar información. No es sólo usada por espías sino también en comunicaciones por telefóno, vía fax o por correo electrónico, transacciones bancarias, seguridad en las cuentas de banco, passwords y números de identificación. También se usa en las firmas digitales para verificar quien envía un mensaje.

Notación

$\mathbb{N}$ Naturales

$\mathbb{N}_0$ Enteros no negativos

$\mathbb{Z}$ Enteros

$\mathbb{Q}$ Racionales

$\mathbb{R}$ Reales

$\mathbb{C}$ Complejos

$\mathbb{Z}_m$ Enteros módulo m

$\mathbb{Z}[x]$ Anillo de polinomios con coeficientes en $\mathbb{Z}$

$\mathbb{Z}_m[x]$ Anillo de polinomios con coeficientes en $\mathbb{Z}_m$

$\mathbb{Z}_m^n[x]$ Anillo de polinomios con coeficientes en $\mathbb{Z}_m$ y de grado a lo más $n-1$

$\mathbb{Z}^n[x]$ Anillo de polinomios con coeficientes en $\mathbb{Z}$ y de grado a lo más $n-1$

$\boldsymbol{B}_{n\times n}(\mathcal{A})$ Matriz cuadrada de orden n con entradas en $\mathcal{A}$

p_m^{-1} Inverso multiplicativo de p módulo m (p polinomio o entero)

$\mathbb{F}$ Campo

$\mathbb{F}_{p^n}$ Campo finito con p^n elementos

D. I. P. Dominio de Ideales Principales

D. F. U. Dominio de Factorización Única

d. e. Dominio entero

D. E. Dominio Euclideano

$\sharp$ Número de elementos de un conjunto.

$PVMC_p$ Problema del vector más corto con respecto a la norma p.

$l.\ i.$ linealmente independientes.

TRF Transformada rápida de Fourier.

TCF Transformada cuántica de Fourier.

TDF Transformada discreta de Fourier.

$TCDF$ Transformada cuántica discreta de Fourier.

$\lfloor\ \rfloor$ Función parte entera.

$\lceil\ \rceil$ Función entero mayor o igual.

$\lfloor\ \rceil$ Función entero más cercano.

ord Orden de un elemento.

gra Grado de un polinomio.

$\boldsymbol{\mathcal{R}}$ Anillo de polinomios truncados de grado a lo más N.

Λ Retículo.

$\mathbb{E}^*$ Anillo o campo sin el cero.

φ Función fi (o totient) de Euler.

1

Bases Matemáticas

§1.1 Un Anillo de polinomios

Considere el Anillo de Polinomios Truncados de Grado N

$$\begin{aligned}\mathbb{Z}^N[x] &= \frac{\mathbb{Z}[x]}{\langle x^N-1\rangle}\\ &\cong \mathbb{Z}\times\frac{\mathbb{Z}[x]}{\langle x^{N-1}+x^{N-2}+\cdots+x+1\rangle}\\ &= \boldsymbol{\mathcal{R}}\end{aligned}$$

i. e., los polinomios de grado a lo más $N-1$ con coeficientes en $\mathbb{Z}$. Si $\boldsymbol{f}\in\mathbb{Z}^N[x]^*$, entonces $\boldsymbol{f}$ se escribe como un polinomio

$$\begin{aligned}\boldsymbol{f} &= \sum_{i=0}^{N-1} a_i x^i\\ &= a_0+a_1x+\cdots+a_{N-1}x^{N-1},\end{aligned}$$

o como un vector

$$\boldsymbol{f}=[a_0,a_1,\ldots,a_{N-1}].$$

donde algunos coeficientes pueden ser 0. Es decir, se identifica al $\mathbb{Z}$-módulo $\mathbb{Z}^N=\underbrace{\mathbb{Z}\times\cdots\times\mathbb{Z}}_{\text{N veces}}$ con el anillo de polinomios $\mathbb{Z}^N[x]$. Los polinomios en este anillo se suman de manera usual, coeficiente a coeficiente, si

$$\left.\begin{aligned}\boldsymbol{f} &= a_0+a_1x+\cdots+a_{N-1}x^{N-1}\\ \boldsymbol{g} &= b_0+b_1x+\cdots+b_{N-1}x^{N-1}\end{aligned}\right\}\in\boldsymbol{\mathcal{R}}$$

su suma será

$$\boldsymbol{f}+\boldsymbol{g}=(a_0+b_0)+(a_1+b_1)x+\cdots+(a_{N-1}+b_{N-1})x^{N-1}$$

y su producto (Convolución), denotado por $\circledast$, se define por

$$\begin{aligned}\boldsymbol{f}\circledast\boldsymbol{g} &= \boldsymbol{h}\\ &= \sum_{i=0}^{N-1} c_i x^i\end{aligned}$$

i. e.

$$\boldsymbol{h} = [c_0,c_1,\ldots,c_{N-1}]$$

donde cada c_k está dado por

$$\begin{aligned} c_k &= (\boldsymbol{f} \circledast \boldsymbol{g})_k \\ &= \sum_{i=0}^{k} a_i b_{k-i} + \sum_{i=k+1}^{N-1} a_i b_{N+k-i} \\ &= \sum_{i+j \,\equiv\, k \pmod N} a_i b_j \\ &= a_0 b_k + a_1 b_{k-1} + \cdots + a_{k-1} b_1 + a_k b_0 + \\ &\quad a_{k+1} b_{N-1} + a_{k+2} b_{N-2} + \cdots + a_{N-1} b_{k+1}. \end{aligned}$$

Este es el producto polinomial ordinario en $\mathbb{Z}^N[x]$.

Lo anterior también se puede escribir como

$$\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{pmatrix} = \begin{pmatrix} a_0 & a_1 & \cdots & a_{N-1} \\ a_1 & a_2 & \cdots & a_0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{N-1} & a_0 & \cdots & a_{N-2} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{N-1} \end{pmatrix}$$

Esta fórmula parece un poco complicada, pero realmente no lo es. El k-ésimo coeficiente c_k es simplemente el producto punto de los coeficientes de $\boldsymbol{f}$ con los coeficientes de $\boldsymbol{g}$, excepto que los coeficientes de $\boldsymbol{g}$ estan listados en orden inverso y recorridos cíclicamente k posiciones.

En otras palabras, la multiplicación aquí es casi la misma que la usual, excepto que después de hacer la multiplicación x^N debe ser reemplazado por 1, x^{N+1} debe ser reemplazado por x, x^{N+2} debe ser reemplazado por x^2, y así sucesivamente...

1.1.1 Aritmética modular y polinomios

Cuando hablamos del anillo $\mathbb{Z}^N[x]$ antes definido, combinado con la aritmética modular, queremos decir que se reducen los coeficientes de un polinomio $\boldsymbol{f} \in \boldsymbol{\mathcal{R}}$ módulo un entero q. El conjunto de polinomios de grado a lo más $N-1$ con coeficientes reducidos módulo un entero q es un subanillo de $\mathbb{Z}^N$, lo denotaremos por $\mathbb{Z}_q^N[x] \cong \dfrac{\mathbb{Z}_q[x]}{\langle x^N - 1 \rangle}$. Así la expresión

$$\boldsymbol{f} \pmod q$$

significa reducir los coeficientes de $\boldsymbol{f}$ módulo q, *i. e.* dividir cada coeficiente de $\boldsymbol{f}$ por q y tomar el residuo

$$\boldsymbol{f} \pmod q = \sum_{i=0}^{N-1} (a_i \bmod q)\, x^i.$$

Similarmente, la relación

$$\boldsymbol{f} \equiv \boldsymbol{g} \pmod{q}$$

significa que cada coeficiente de la diferencia $\boldsymbol{f} - \boldsymbol{g}$ es un múltiplo de q.

El inverso módulo q de un polinomio $\boldsymbol{f}$ es un polinomio $\boldsymbol{f}_q^{-1}$ con la propiedad de que

$$\begin{aligned} \boldsymbol{f} \circledast \boldsymbol{f}_q^{-1} &\equiv \mathbf{1} \\ &\equiv 1 \pmod{q}. \end{aligned}$$

No todo polinomio tiene un inverso módulo q, pero se puede determinar si $\boldsymbol{f}$ tiene un inverso y calcularlo si es que existe.

Proposición 1 *$\boldsymbol{f} \in \mathbb{Z}^N[x]$, tiene inverso, si y sólo si, $(\boldsymbol{f}, x^N - 1) = \mathbf{1}$.*

Ejemplo 1

$$\begin{aligned} N &= 7, \\ q &= 11, \\ \boldsymbol{f} &= 3 + 2x^2 - 3x^4 + x^6 \\ \boldsymbol{g} &= 1 - 3x + x^2 + 2x^5 - x^6 \end{aligned}$$

entonces...

$$\begin{aligned} \boldsymbol{f} \circledast \boldsymbol{g} &= (3 + 2x^2 - 3x^4 + x^6) \circledast (1 - 3x + x^2 + 2x^5 - x^6) \\ &= 3 - 9x + 5x^2 - 6x^3 - x^4 + 15x^5 - 5x^6 \\ &\quad -x^8 - 6x^9 + 3x^{10} + 2x^{11} - x^{12} \\ &= 3 - 9x + 5x^2 - 6x^3 - x^4 + 15x^5 - 5x^6 \\ &\quad -x - 6x^2 + 3x^3 + 2x^4 - x^5 \\ &= 4 - 10x - x^2 - 3x^3 + x^4 + 14x^5 - 5x^6 \\ &\equiv 4 + x + 10x^2 + 8x^3 + x^4 + 3x^5 + 6x^6 \pmod{11} \end{aligned}$$

y el inverso de $\boldsymbol{f}$ módulo 11 es

$$\boldsymbol{f}_{11}^{-1} = -2 + 4x + 2x^2 + 4x^3 - 4x^4 + 2x^5 - 2x^6$$

pues...

$$\begin{aligned} \boldsymbol{f} \circledast \boldsymbol{f}_{11}^{-1} &= (3 + 2x^2 - 3x^4 + x^6) \circledast (-2 + 4x + 2x^2 + 4x^3 - 4x^4 + 2x^5 - 2x^6) \\ &= -6 + 12x + 2x^2 + 20x^3 - 2x^4 + 2x^5 - 22x^6 \\ &\quad -4x^7 + 10x^8 - 2x^9 + 2x^{10} + 2x^{11} - 2x^{12} \\ &= -6 + 12x + 2x^2 + 20x^3 - 2x^4 + 2x^5 - 22x^6 \\ &\quad -4 + 10x - 2x^2 + 2x^3 + 2x^4 - 2x^5 \\ &= -10 + 22x + 22x^3 - 22x^6 \\ &\equiv 1 \pmod{11} \end{aligned}$$

§1.2 Inversión de polinomios en $\mathbb{Z}_N$

Sea $\boldsymbol{m} \in \mathbb{Z}_2[x]$. El " Algoritmo de Casi Inversos" ("Almost Inverse Algorithm") de Schroeppel, Orman, O'Malley y Spatscheck [SOOS95b] proporciona una manera eficiente de calcular el inverso de un polinomio $\boldsymbol{a} \in \frac{\mathbb{Z}_2[x]}{\langle \boldsymbol{m} \rangle}$ si $(\boldsymbol{a}, \boldsymbol{m}) = \mathbf{1}$ y $\boldsymbol{m}(0) = 1$.

Algoritmo de casi inversos para $\boldsymbol{m} = x^N - 1$

Inversión en $\frac{\mathbb{Z}_2[x]}{\langle \boldsymbol{m} \rangle}$
Entrada: $\boldsymbol{a}$
Salida: $\boldsymbol{b} \equiv \boldsymbol{a}^{-1}$ en $\frac{\mathbb{Z}_2[x]}{\langle \boldsymbol{m} \rangle}$
Paso 1: Inicialización: $k := 0,\ \boldsymbol{b} := 1,\ \boldsymbol{c} := 0,$
$\boldsymbol{f} := \boldsymbol{a},\ \boldsymbol{g} := \boldsymbol{m}$
Paso 2: Ciclo:
Paso 3: haz mientras $f_0 = 0$
Paso 4: $\boldsymbol{f} := \boldsymbol{f}/x,\ \boldsymbol{c} := \boldsymbol{c} \circledast x,\ k := k + 1$
Paso 5: si $\boldsymbol{f} = 1$ entonces
Paso 6: regresa $x^{N-k} \circledast \boldsymbol{b} \pmod{\boldsymbol{m}}$
Paso 7: si $\text{gra}(\boldsymbol{f}) < \text{gra}(\boldsymbol{g})$ entonces
Paso 8: intercambia $\boldsymbol{f}$ y $\boldsymbol{g}$ e intercambia $\boldsymbol{b}$ y $\boldsymbol{c}$
Paso 9: $\boldsymbol{f} := \boldsymbol{f} + \boldsymbol{g} \pmod 2$
Paso 10: $\boldsymbol{b} := \boldsymbol{b} + \boldsymbol{c} \pmod 2$
Paso 11: ve a Ciclo

El número f_0 en el Paso 3 es el coeficiente independiente de $\boldsymbol{f}$, y el valor regresado en el Paso 6 $x^{N-k} \circledast \boldsymbol{b} \pmod{(x^N - 1)}$ es $\boldsymbol{b}$ con sus coeficientes recorridos cíclicamente k posiciones.

Para crear un par de claves pública/privada del NTRU, se necesita calcular el inverso de un polinomio $\pmod p$, con $p \neq 2$ número primo.

La siguiente es una adaptación del Algoritmo de casi inversos para $p = 3$, dado que éste es otro valor requerido para el conjunto estándar de parámetros del NTRU.

Inversión en $\dfrac{\mathbb{Z}_3[x]}{\langle \boldsymbol{m} \rangle}$
Entrada: $\boldsymbol{a}$
Salida: $\boldsymbol{b} \equiv \boldsymbol{a}^{-1}$ en $\dfrac{\mathbb{Z}_3[x]}{\langle \boldsymbol{m} \rangle}$
Paso 1: Inicialización: $k := 0$, $\boldsymbol{b} := 1$, $\boldsymbol{c} := 0$, $\boldsymbol{f} := \boldsymbol{a}$, $\boldsymbol{g} := \boldsymbol{m}$
Paso 2: Ciclo:
Paso 3: haz mientras $f_0 = 0$
Paso 4: $\boldsymbol{f} := \boldsymbol{f}/x$, $\boldsymbol{c} := \boldsymbol{c} \circledast x$, $k := k + 1$
Paso 5: si $\boldsymbol{f} = \pm 1$ entonces
Paso 6: regresa $\pm x^{N-k} \circledast \boldsymbol{b} \pmod{\boldsymbol{m}}$
Paso 7: si $\text{gra}(\boldsymbol{f}) < \text{gra}(\boldsymbol{g})$ entonces
Paso 8: intercambia $\boldsymbol{f}$ y $\boldsymbol{g}$ e intercambia $\boldsymbol{b}$ y $\boldsymbol{c}$
Paso 9: si $f_0 = g_0$
Paso 10: $\boldsymbol{f} := \boldsymbol{f} - \boldsymbol{g} \pmod 3$
Paso 11: $\boldsymbol{b} := \boldsymbol{b} - \boldsymbol{c} \pmod 3$
Paso 12: else
Paso 13: $\boldsymbol{f} := \boldsymbol{f} + \boldsymbol{g} \pmod 3$
Paso 14: $\boldsymbol{b} := \boldsymbol{b} + \boldsymbol{c} \pmod 3$
Paso 15: ve a Ciclo

En esta rutina, todos los cálculos son hechos $\pmod 3$, así que todos los coeficientes son escogidos dentro del conjunto $\{-1, 0, 1\}$. También, los 2 ± 1's en los Pasos 5 y 6 son escogidos con el mismo signo.

Para la creación de un par de claves pública/privada del NTRU se requiere siempre encontrar el inverso de un polinomio $\boldsymbol{f}$ módulo no sólo un primo, si no también una potencia de un primo, en particular una potencia de 2. De cualquier modo, una vez que un inverso módulo un primo p se ha determinado, un método simple basado en iteraciones de Newton permite calcular rápidamente el inverso $\pmod{p^r}$. El siguiente algoritmo converge exponencialmente doble, en el sentido de que requiere solo aproximadamente $\log_2(r)$ pasos para encontrar el inverso de $\boldsymbol{a} \pmod{p^r}$, una vez que se conoce un inverso $\pmod p$.

Inversión en $\dfrac{\mathbb{Z}_{p^r}[x]}{\langle \boldsymbol{m} \rangle}$
Entrada: $\boldsymbol{a}$, p (un primo), r
$\boldsymbol{b} \equiv \boldsymbol{a}^{-1} \pmod p$
Salida: $\boldsymbol{b} \equiv \boldsymbol{a}^{-1} \pmod{p^r}$
Paso 1: $q = p$
Paso 2: haz mientras $q < p^r$
Paso 3: $q = q^2$
Paso 4: $\boldsymbol{b} := \boldsymbol{b} \circledast (2 - \boldsymbol{a} \circledast \boldsymbol{b}) \pmod q$

Finalmente, una versión del Algoritmo de casi inversos para un primo p arbitrario.

Inversión en $\dfrac{\mathbb{Z}_p[x]}{\langle \boldsymbol{m} \rangle}$

Entrada: $\boldsymbol{a}$, p (un primo)

Salida: $\boldsymbol{b} \equiv \boldsymbol{a}^{-1}$ en $\dfrac{\mathbb{Z}_p[x]}{\langle \boldsymbol{m} \rangle}$

Paso 1: Inicialización: $k := 0$, $\boldsymbol{b} := 1$, $\boldsymbol{c} := 0$, $\boldsymbol{f} := \boldsymbol{a}$, $\boldsymbol{g} := \boldsymbol{m}$

Paso 2: Ciclo:

Paso 3: haz mientras $f_0 = 0$

Paso 4: $\quad \boldsymbol{f} := \boldsymbol{f}/x$, $\boldsymbol{c} := \boldsymbol{c} \circledast x$, $k := k + 1$

Paso 5: si $\operatorname{gra}(\boldsymbol{f}) = 0$ entonces

Paso 6: $\quad \boldsymbol{b} := f_0^{-1}\boldsymbol{b} \pmod{p}$

Paso 7: $\quad$ regresa $x^{N-k} \circledast \boldsymbol{b} \pmod{\boldsymbol{m}}$

Paso 8: si $\operatorname{gra}(\boldsymbol{f}) < \operatorname{gra}(\boldsymbol{g})$ entonces

Paso 9: $\quad$ intercambia $\boldsymbol{f}$ y $\boldsymbol{g}$ e intercambia $\boldsymbol{b}$ y $\boldsymbol{c}$

Paso 10: $u := f_0 g_0^{-1} \pmod{p}$

Paso 11: $\boldsymbol{f} := \boldsymbol{f} - u\boldsymbol{g} \pmod{p}$

Paso 12: $\boldsymbol{b} := \boldsymbol{b} - u\boldsymbol{c} \pmod{p}$

Paso 13: ve a Ciclo

1.2.1 ¿Por qué funciona?

La idea es que se inicia con el vector $(\boldsymbol{f}, \boldsymbol{g}) = (\boldsymbol{a}, \boldsymbol{m})$. Luego multiplicamos (por la derecha) por las siguientes matrices

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad \boldsymbol{B} = \begin{pmatrix} x^{-1} & 0 \\ 0 & 1 \end{pmatrix}, \qquad \boldsymbol{C}_u = \begin{pmatrix} 1 & 0 \\ -u & 1 \end{pmatrix}.$$

El efecto de estas transformaciones es

$$(\boldsymbol{f}, \boldsymbol{g})\boldsymbol{A} = (\boldsymbol{g}, \boldsymbol{f}), \qquad (\boldsymbol{f}, \boldsymbol{g})\boldsymbol{B} = (x^{-1}\boldsymbol{f}, \boldsymbol{g}), \qquad (\boldsymbol{f}, \boldsymbol{g})\boldsymbol{C}_u = (\boldsymbol{f} - u\boldsymbol{g}, \boldsymbol{g}),$$

por lo que el Paso 4 es la matriz $\boldsymbol{B}$, el Paso 9 es la matriz $\boldsymbol{A}$, y el Paso 11 es la matriz $\boldsymbol{C}_u$. En el Paso 11, el valor de u es escogido de tal forma que $\boldsymbol{f} - u\boldsymbol{g}$ sea divisible por x (*i. e.*, que su término constante sea 0). Entonces en el Paso 4 dividimos $\boldsymbol{f}$ por x hasta que su término constante sea no cero. También, el Paso 9 asegura que $\operatorname{gra}(\boldsymbol{f}) \geq \operatorname{gra}(\boldsymbol{g})$. El efecto total es que cada vez que se repite el ciclo, el grado total $\operatorname{gra}(\boldsymbol{f}) + \operatorname{gra}(\boldsymbol{g})$ se reduce por lo menos en 1, así al final $\boldsymbol{f}$ llega a ser una constante (pues $(\boldsymbol{f}, \boldsymbol{q}) = 1$). Por lo que el algoritmo termina en a lo más en $\operatorname{gra}(\boldsymbol{a}) + \operatorname{gra}(\boldsymbol{m})$ iteraciones.

De esta manera el algoritmo produce una secuencia de transformaciones $\boldsymbol{D}_1, \boldsymbol{D}_2, \ldots, \boldsymbol{D}_r$, donde cada $\boldsymbol{D}_i$ es una de las matrices $\boldsymbol{A}$, $\boldsymbol{B}$, o $\boldsymbol{C}_u$, por lo que

$$(\boldsymbol{a}, \boldsymbol{m})\boldsymbol{D}_1\boldsymbol{D}_2\boldsymbol{D}_3 \cdots \boldsymbol{D}_{r-1}\boldsymbol{D}_r = (\boldsymbol{\alpha}, *),$$

donde $\boldsymbol{\alpha} \not\equiv 0 \pmod{p}$. Desafortunadamente, los coeficientes del producto $\boldsymbol{D}_1\boldsymbol{D}_2 \cdots \boldsymbol{D}_r$ no son polinomios, pues la matriz $\boldsymbol{B}$ tiene x^{-1} como una de sus

entradas. Sea k el número de veces que $\boldsymbol{B}$ aparece en el producto $\boldsymbol{D}_1\boldsymbol{D}_2\cdots\boldsymbol{D}_r$. (Se ve fácilmente que este es el valor de k que se calcula en el algoritmo.) Entonces $x^k\boldsymbol{D}_1\boldsymbol{D}_2\cdots\boldsymbol{D}_r$ tiene coeficientes que son polinomios, digamos

$$x^k\boldsymbol{D}_1\boldsymbol{D}_2\cdots\boldsymbol{D}_r = \begin{pmatrix} \boldsymbol{a}' & * \\ \boldsymbol{m}' & * \end{pmatrix}.$$

Ahora multiplicando por la izquierda por (a, m) se tiene

$$\begin{aligned} (\boldsymbol{a}\boldsymbol{a}' + \boldsymbol{m}\boldsymbol{m}', *) &= (\boldsymbol{a}, \boldsymbol{m}) \begin{pmatrix} \boldsymbol{a}' & * \\ \boldsymbol{m}' & * \end{pmatrix} \\ &= (\boldsymbol{a}, \boldsymbol{m})x^k\boldsymbol{D}_1\boldsymbol{D}_2\cdots\boldsymbol{D}_r \\ &= x^k(\boldsymbol{\alpha}, *), \end{aligned}$$

así tenemos

$$\boldsymbol{a}\boldsymbol{a}' \equiv \boldsymbol{\alpha}x^k \pmod{m}.$$

Ahora la pregunta es

¿Cómo construye el Algoritmo de Casi Inversos a este valor $\boldsymbol{a}'$?

La respuesta es que mientras se estan aplicando las transformaciones

$$\boldsymbol{D}_1,\ \boldsymbol{D}_2, \ldots, \boldsymbol{D}_r$$

comenzando con $(\boldsymbol{a}, \boldsymbol{m})$, se estan aplicando también las mismas transformaciones comenzando con $(\boldsymbol{b}, \boldsymbol{c}) = (\boldsymbol{1}, \boldsymbol{0})$, excepto que en lugar de $\boldsymbol{B} = \begin{pmatrix} x^{-1} & 0 \\ 0 & 1 \end{pmatrix}$, se aplica $x\boldsymbol{B} = \begin{pmatrix} 1 & 0 \\ 0 & x \end{pmatrix}$. Como $\boldsymbol{B}$ ha sido usado k veces, al final del algoritmo el valor de $(\boldsymbol{b}, \boldsymbol{c})$ es

$$(\boldsymbol{b}, \boldsymbol{c}) = (1, 0)x^k\boldsymbol{D}_1\boldsymbol{D}_2\cdots\boldsymbol{D}_r = (1, 0)\begin{pmatrix} \boldsymbol{a}' & * \\ \boldsymbol{m}' & * \end{pmatrix} = (\boldsymbol{a}', *).$$

En otras palabras, al final del algoritmo, $\boldsymbol{b}$ tiene un valor que satisface

$$\boldsymbol{a}\boldsymbol{b} \equiv \boldsymbol{\alpha}x^k \pmod{m}.$$

Dado que el valor de $\boldsymbol{\alpha}$ es simplemente f_0 (el término constante de $\boldsymbol{f}$, el cual de hecho es igual a $\boldsymbol{f}$ a estas alturas del algoritmo), se tiene que $\boldsymbol{a}^{-1} = f_0^{-1}x^{N-k}\boldsymbol{b}$. (Note que $x^{-k} = x^{N-k} \pmod{x^N - 1}$.)

§1.3 Complejidad de algoritmos

Sean $n \in \mathbb{N}$ y $\ell(n)$ el número de dígitos de n cuando se escribe en base b (constante). A $\ell(n)$ se le llama la LONGITUD de n en base b. Considerando que

todos los logaritmos difieren por una constante[1], por ejemplo $\frac{\log_{10} n}{\log_2 n} = c$, se tiene que

$$\text{longitud de } n = \ell(n) = 1 + \lfloor \log_b(n) \rceil = 1 + \left\lfloor \frac{\ln(n)}{\ln(b)} \right\rceil,$$

como b es constante, y por tanto $\ln(b)$. A grosso modo, la función número de dígitos se comporta como el logaritmo.

Para cálculos en cómputo se usa n en base 2 ($0'$s y $1'$s), así como toda la aritmética, por lo que la longitud de un entero (expansión binaria o número de bits) está dada por

$$\ell(n) = 1 + \lfloor \log_2(n) \rceil = 1 + \left\lfloor \frac{\ln(n)}{\ln(2)} \right\rceil \approx \log_2(n).$$

Definición 1 *Sean $f,\ g\colon \mathbb{N} \longrightarrow \mathbb{R}^+$. Se dice $f(n) = O(g(n))$; o simplemente que $f = O(g)$; si existe $C \in \mathbb{R}$ y $n_0 \in \mathbb{N}$ tal que $f(n) < Cg(n)\ \forall\ n > n_0$. Se dice en este caso que f es O* GRANDE *de g.*

Informalmente estamos diciendo que f no crece más rápido que g. El n_0 indica que no se toma en cuenta cómo son f y g para valores pequeños de n. Como ejemplo tenemos que $\ell(n) = O(\ln(n))$, donde $\ell(n)$ es la función definida al inicio de esta sección.

Definición 2 *Se dice que $f(n) = o(g(n))$, o simplemente que $f = o(g)$ si $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$. Se dice en este caso que f es O* PEQUEÑA *de g.*

Informalmente estamos diciendo que f crece mucho más lento que g ($f \ll g$) cuando n es grande. Si $\lim_{n\to\infty} \frac{f(n)}{g(n)} = k$ constante, entonces $f = O(g)$ por lo que o proporciona información más precisa que O.

La notación $g = \Omega(f)$ significa exactamente lo mismo que $f = O(g)$, en otras palabras, es la negación de o. Se dice en este caso que g es OMEGA de f.

Definición 3 *Se dice que $f(n) = \Theta(g(n))$, o simplemente que $f = \Theta(g)$ si existen $c_1,\ c_2 \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$ tales que $\forall\ n > n_0$ se cumple que $c_1 g(n) < f(n) < c_2 g(n)$. Es decir, $f = O(g)$ y $f = \Omega(g)$. Se dice en este caso que f es* TETA *de g.*

Podemos decir entonces que f y g tienen la misma tasa de crecimiento, sólo las constantes multiplicativas son desconocidas. Θ es más precisa que o.

[1] $\log_a(n) = c \log_b(n)$, es decir $\frac{\log_a(n)}{\log_b(n)} = \log_a(b)$

Definición 4 *Se dice que f es* ASINTÓTICAMENTE *igual a g para n grande, y se denota por $f \asymp g$ o por $f \sim g$, si* $\lim_{n\to\infty} \dfrac{f(n)}{g(n)} = 1$.

$\asymp$ es aún más precisa que Θ por lo que da una relación más exacta entre f y g; por ejemplo decir que cuando n crece el porcentaje de error al usar g en vez de f se va a cero.

Proposición 2 *Sean* $f,\ g : \mathbb{N}_0 \longrightarrow \mathbb{R}^+$.

i) *Si f es un polinomio de grado d en n entonces* $f(n) = O(n^d)$

ii) $O(kf(n)) = Of(n)$

iii) *Si* $g(n) \leq f(n)$ *entonces* $Of(n) + Og(n) = O(f(n) + g(n)) = Of(n)$

iv) $f(n)Og(n) = O(f(n)g(n))$

v) $O(\log(f(n))) = O(\log(n))$ *si $f(n)$ es un polinomio en n*

vi) $\log(n) = O(n^\epsilon)\ \forall\ \epsilon > 0,\ \epsilon \in \mathbb{R}^+$.

Definición 5 *Se dice que un algoritmo es de* TIEMPO POLINOMIAL *si existe $d \in \mathbb{N}$ tal que el número de operaciones bit requerido para desarrollar el algoritmo en una entrada de longitud a lo más k es $O(k^d)$.*

Definición 6 *Se dice que un algoritmo es de* TIEMPO EXPONENCIAL *si existe $c \in \mathbb{R}^+$ tal que el número de operaciones bit requerido para desarrollar el algoritmo en una entrada de longitud a lo más k es $O\left(e^{ck}\right)$.*

Definición 7 *Sean* $n \in \mathbb{N},\ 0 \leq \gamma \leq 1,\ c \in \mathbb{R}^+$ *y*

$$L_n(\gamma; c) = O\left(e^{c\left((\ln n)^\gamma (\ln\ln n)^{1-\gamma}\right)}\right).$$

Se dice que un algoritmo es de TIEMPO $L(\gamma)$ *si su tiempo de corrida estimado es de la forma $L_n(\gamma; c)$ cuando se aplica a una entrada de longitud a lo más n.*

En particular

$$L_n(0; c) = O\left(e^{c(\ln\ln n)}\right) = O\left((\ln n)^c\right),$$

$$L_n(1; c) = O\left(e^{c\ln n}\right) = O\left(n^c\right)$$

son algoritmos polinomiales y exponenciales respectivamente. Si $0 < \gamma < 1$ se dice que el algoritmo es de TIEMPO SUBEXPONENCIAL.

Definición 8 *Una función $f(x)$ es de crecimiento* EXPONENCIAL *si* $\exists\ \ c > 1$ *tal que* $f(x) = \Omega(c^x)$ *y* $\exists\ \ d$ *tal que* $f(x) = O(d^x)$.

Un Problema de Decisión es aquel cuya solución (o salida) es un *sí* o un *no*. Si se requiere de un ejemplo que dé respuesta, entonces se llama Problema de Búsqueda. En muchos casos el problema de decisión y de búsqueda son esencialmente equivalentes, esto significa que un algoritmo que resuelva uno puede convertirse *fácilmente* en un algoritmo que resuelva el otro, por lo que desde el punto de vista computacional no hay pérdida de generalidad al trabajar con problemas de decisión en vez de problemas de búsqueda.

Aquí el término Problema se refiere a una descripción general de una tarea, y una Instancia de un Problema, o simplemente Instancia, es un caso particular de la tarea.

El conjunto de problemas cuyos algoritmos solución tienen tiempo polinomial es llamado $\boldsymbol{\mathcal{P}}$. Sin embargo existe un conjunto mayor de problemas para los cuales no se conoce algoritmos solución de tiempo polinomial, éste es llamado $\boldsymbol{\mathcal{NP}}$. Se sabe que en términos de tiempo de corrida, $\boldsymbol{\mathcal{P}} \leq \boldsymbol{\mathcal{NP}} \leq$ exponencial.

Definición 9 *Un problema de decisión* $\boldsymbol{P}$ *está en la* Clase $\boldsymbol{\mathcal{P}}$ *de problemas de tiempo polinomial si existe un polinomio* $p(n)$ *y un algoritmo tal que para cada instancia de* $\boldsymbol{P}$ *que tiene una entrada de longitud a lo más* n*, entonces el algoritmo da solución al problema en un tiempo a lo más* $p(n)$*.*

Aquí se supone que tenemos una computadora fija para implementar el algoritmo, y tiempo se refiere al tiempo de corrida en esta computadora, o alternativamente, se puede definir el tiempo como el número de operaciones bit requeridas para ejecutar el algoritmo. Alternativamente contamos con la siguiente definición

Definición 10 *Un problema de decisión* $\boldsymbol{P}$ *está en* $\boldsymbol{\mathcal{P}}$ *si existe* $c \in \mathbb{R}$ *y un algoritmo tal que si una instancia de* $\boldsymbol{P}$ *tiene entrada de longitud a lo más* n*, entonces el algoritmo responde la pregunta en un tiempo* $O(n^c)$*.*

Es decir un problema de decisión está en $\boldsymbol{\mathcal{P}}$ si existe un algoritmo de tiempo polinomial que lo resuelve.

$\boldsymbol{\mathcal{P}}$ es la clase de problemas que pueden ser resueltos rápida o fácilmente en la práctica, es decir existe un algoritmo para el cual su tiempo de corrida está acotado por una potencia de la longitud de la entrada. Algunas veces un problema que está en $\boldsymbol{\mathcal{P}}$ o que se cree que está en $\boldsymbol{\mathcal{P}}$ tiene un algoritmo práctico eficiente que no es de tiempo polinomial o que no se presta a un riguroso análisis de su tiempo de corrida.

Definición 11 *Un problema de decisión* $\boldsymbol{P}$ *está en la clase* Clase $\boldsymbol{\mathcal{NP}}$*, si, dada una instancia de* $\boldsymbol{P}$ *una persona con poder de cómputo ilimitado no solo puede responder a la pregunta, sino que en el caso de que la respuesta sea* sí*, puede dar evidencia que otra persona puede usar para verificar la veracidad de la respuesta en tiempo polinomial. La demostración de que su respuesta* sí *es correcta se llama* Certificado*, más precisamente,* Certificado de Tiempo Polinomial*.*

Un problema de decisión $\boldsymbol{P}$ se dice que está en la CLASE $co-\boldsymbol{\mathcal{NP}}$ si la condición anterior es válida para un *no*, es decir, para un caso teniendo *no* como respuesta debe existir un certificado de tiempo polinomial que valide la veracidad del *no*.

Si un problema está en la clase $\boldsymbol{\mathcal{P}}$ entonces trivialmente está en la clase $\boldsymbol{\mathcal{NP}}$, es decir $\boldsymbol{\mathcal{P}} \subseteq \boldsymbol{\mathcal{NP}}$. Es casi cierto que $\boldsymbol{\mathcal{NP}}$ es una clase mucho más grande que $\boldsymbol{\mathcal{P}}$, pero esto no ha sido probado. La aseveración de que $\boldsymbol{\mathcal{P}} \neq \boldsymbol{\mathcal{NP}}$ es la conjetura más famosa en la teoría de computación.

Definición 12 *Sean $\boldsymbol{P}_1$, $\boldsymbol{P}_2$ problemas de decisión. Decimos que $\boldsymbol{P}_1$ se* REDUCE *a $\boldsymbol{P}_2$ si existe un algoritmo que es de tiempo polinomial como una función de la longitud de la entrada de $\boldsymbol{P}_1$ y que dado cualquier instancia p_1 de $\boldsymbol{P}_1$ existe una instancia p_2 de $\boldsymbol{P}_2$ tal que la solución para p_1 es la misma para p_2.*

Si se tiene un algoritmo eficiente para $\boldsymbol{P}_2$ y si $\boldsymbol{P}_1$ se reduce a $\boldsymbol{P}_2$, entonces se puede usar el algoritmo de $\boldsymbol{P}_2$ para resolver $\boldsymbol{P}_1$ también. A saber, dado una instancia p_1 de $\boldsymbol{P}_1$, en tiempo polinomial se encuentra una instancia correspondiente p_2 de $\boldsymbol{P}_2$ usando el algoritmo de la definición (12). Entonces si se aplica el algoritmo de $\boldsymbol{P}_2$ a p_2, la respuesta obtenida será también la respuesta a la pregunta $\boldsymbol{P}_1$ original, es decir, un algoritmo para $\boldsymbol{P}_2$ da automáticamente un algoritmo para $\boldsymbol{P}_1$. Si el algoritmo para $\boldsymbol{P}_2$ es de tiempo polinomial, también lo será el resultante para $\boldsymbol{P}_1$.

Recíprocamente, si se sabe (o se cree) que el problema $\boldsymbol{P}_1$ es muy difícil, es decir es "casi cierto" que no existe un algoritmo eficiente para él, y si $\boldsymbol{P}_1$ se reduce a $\boldsymbol{P}_2$, entonces tampoco hay un algoritmo eficiente para $\boldsymbol{P}_2$.

La definición (12) es un poco restrictiva, por lo que vale la pena tener una definición más amplia de reducción polinomial de $\boldsymbol{P}_1$ a $\boldsymbol{P}_2$ que permita usar varias instancias distintas de $\boldsymbol{P}_2$ para resolver una sola instancia de $\boldsymbol{P}_1$.

Definición 13 *Sea $\boldsymbol{P}_2$ un problema de búsqueda o decisión. Al describir un algoritmo para algún otro problema $\boldsymbol{P}_1$, cuando se dice llamada a un* ORÁCULO *$\boldsymbol{P}_2$, significa que nuestro algoritmo ha creado una instancia de $\boldsymbol{P}_2$ y se supone que algún otro algoritmo entonces da la correspondiente salida de $\boldsymbol{P}_2$. El tiempo tomado por este algoritmo para $\boldsymbol{P}_2$ no se incluye en el tiempo total de corrida del algoritmo para $\boldsymbol{P}_1$, en otras palabras, se pretende que el algoritmo para $\boldsymbol{P}_2$ es una caja negra que trabaja instantáneamente.*

En términos de programación, se piensa en un oráculo como una subrutina que se puede usar a voluntad (o llamar) sin incluir su tiempo de corrida en la estimación del tiempo de corrida total del programa principal.

Definición 14 *Sean $\boldsymbol{P}_1$ y $\boldsymbol{P}_2$ dos problemas. Se dice que $\boldsymbol{P}_1$ se* REDUCE *a $\boldsymbol{P}_2$ en tiempo polinomial se existe un algoritmo de tiempo polinomial para $\boldsymbol{P}_1$ que usa a lo más una cantidad polinomial de llamadas a un oráculo $\boldsymbol{P}_2$.*

Se dice entonces que $\boldsymbol{P}_1$ *y* $\boldsymbol{P}_2$ *son* EQUIVALENTES EN TIEMPO POLINOMIAL *o que son* $\boldsymbol{\mathcal{NP}}-$EQUIVALENTES.

Definición 15 *Un problema de decisión* $\boldsymbol{P} \in \boldsymbol{\mathcal{NP}}$ *se dice* $\boldsymbol{\mathcal{NP}}-$COMPLETO *si cualquier otro problema* $\boldsymbol{Q} \in \boldsymbol{\mathcal{NP}}$ *puede reducirse a* $\boldsymbol{P}$ *en tiempo polinomial.*

Así, si se tuviera un algoritmo de tiempo polinomial para un problema $\boldsymbol{P}$ $\boldsymbol{\mathcal{NP}}-$completo, entonces se tendrían algoritmos de tiempo polinomial para todos los demás problemas $\boldsymbol{Q} \in \boldsymbol{\mathcal{NP}}$. Lo cual implica que $\boldsymbol{\mathcal{P}} = \boldsymbol{\mathcal{NP}}$, y la conjetura de que $\boldsymbol{\mathcal{P}} \neq \boldsymbol{\mathcal{NP}}$ sería falsa. Por esta razón, no es probable que alguien encuentre un algoritmo de tiempo polinomial para un problema $\boldsymbol{\mathcal{NP}}-$completo. En un sentido, los problemas $\boldsymbol{\mathcal{NP}}-$completo son los problemas más difíciles en la clase $\boldsymbol{\mathcal{NP}}$, y los tiempos de corrida de los algoritmos aquí crecen exponencialmente con la longitud de la entrada. Sin embargo, es posible que un problema $\boldsymbol{P}$ se reduzca a un problema $\boldsymbol{\mathcal{NP}}$ aún cuando $\boldsymbol{P}$ mismo probablemente no esté en $\boldsymbol{\mathcal{NP}}$.

Definición 16 *Un problema* $\boldsymbol{P}$ *se dice* $\boldsymbol{\mathcal{NP}}-$DIFÍCIL *si cualquier otro problema* $\boldsymbol{\mathcal{NP}}$ *se reduce a* $\boldsymbol{P}$.

Debido a la transitividad de la reducción, para probar que un problema es $\boldsymbol{\mathcal{NP}}-$difícil es suficiente encontrar un solo problema $\boldsymbol{\mathcal{NP}}-$completo que se reduzca a él.

1.3.1 Funciones de un solo sentido

Definición 17 *Sea* $f : X \to Y$*, decimos que* f *es una* (t, ϵ)-FUNCIÓN DE UN SOLO SENTIDO *si para todo algoritmo probabilístico* A *que corra en tiempo a lo más* t*, la probabilidad de obtener una preimagen es menor que* ϵ*, es decir,*

$$Pr[f(x') = y | x \overset{R}{\leftarrow} X; y = f(x); x' \leftarrow A(y))] < \epsilon$$

Se le llama FUNCIÓN DE ESCOTILLÓN *si es factible calcular* $f^{-1}(y)$ *cuando y solamente cuando se posea información adicional (lo que sería información confidencial).*

Intuitivamente hablando tenemos que $f(x)$ se calcula rápidamente, pero el cálculo $f^{-1}(y)$ no es factible, aún con los algoritmos más eficaces y las computadoras más avanzadas.

Definición 18 *Sea* $f : \{0,1\}^m \to \{0,1\}^s$*, decimos que* f *se una* FUNCIÓN DE DIGESTIÓN *o de* RESUMEN *si* $m > s$.

Definición 19 *Sea* $f : \{0,1\}^m \to \{0,1\}^s$ *una función de digestión, decimos que* f *es* (t, ϵ)-RESISTENTE A COLISIONES *si para todo algoritmo probabilístico* A *que corra a lo más en tiempo* t*, la probabilidad de obtener alguna colisión es menor que* ϵ*, es decir,*

$$Pr[f(x) = f(x') | (x, x') \leftarrow A \text{ con } x \neq x'] < \epsilon$$

§1.4 Curvas elípticas

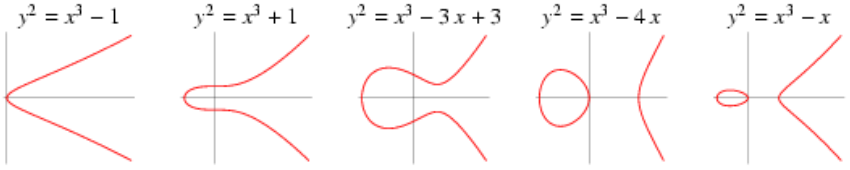
Definición 20 *Una* Curva Elíptica $\mathcal{C}$ *sobre un campo* $\mathbb{K}$ *se define como el conjunto de puntos* (x, y) *que satisfacen una ecuación de la forma*

$$y^2 = x^3 + a\,x + b,^2 \quad a,\ b \in \mathbb{K}^3,$$

donde cada par $a,\ b$ genera una curva elíptica distinta.

Si $\mathcal{C} = x^3 + ax + b$ no contiene factores repetidos, "es suave" (equivalentemente $4a^3 + 27b^2 \neq 0$ y $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$ para el caso $\mathbb{F}_p$), $\mathcal{C}$ puede ser usada para formar un grupo abeliano.

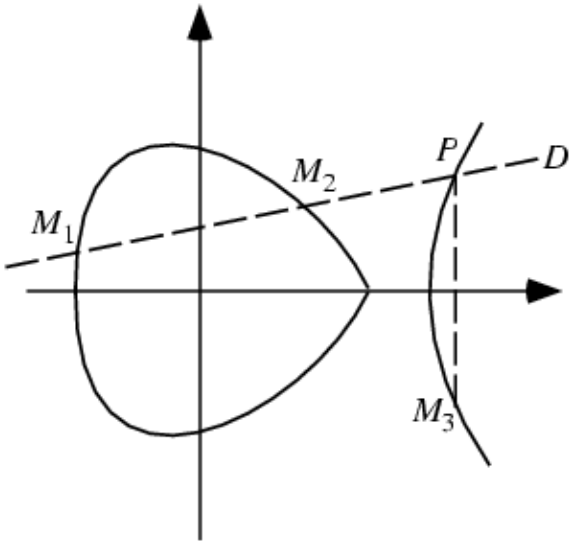
Definición 21 *El* Grupo E de una Curva Elíptica $\mathcal{C}$ *sobre* $\mathbb{K}$ *es el conjunto de puntos de la correspondiente curva elíptica además de un punto especial* O *llamado* Identidad *o* el Punto al Infinito. *En otras palabras, tenemos una regla de adición o suma sobre los puntos de la curva elíptica.*



§1.5 Adición geométrica en curvas elípticas

Si $\mathbb{K} = \mathbb{R}$ es el campo de los números reales, esta regla tiene una interpretación geométrica sencilla.

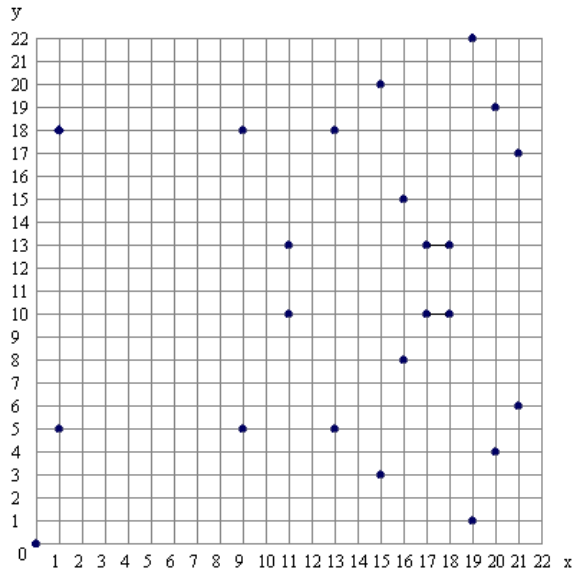
[2] se usa una ecuación diferente en el caso de característica 2 ó 3.
[3] aquí $\mathbb{K}$ puede ser $\mathbb{R}$, $\mathbb{C}$, $\mathbb{F}_p$ o $\mathbb{F}_{q^m}$



§1.6 Grupos de curvas elípticas sobre $\mathbb{F}_p$

Los cálculos sobre $\mathbb{R}$ son lentos e inexactos debido al error de redondeo; para las aplicaciones criptográficas se requiere aritmética rápida y precisa; por lo que en la práctica se usan grupos de curvas elípticas sobre campos finitos: $\mathbb{F}_p$ y $\mathbb{F}_{2^m}$.

Los grupos de una curva elíptica sobre $\mathbb{F}_p$ son finitos, la cual es una propiedad deseable para fines criptográficos. Como estas curvas consisten de puntos discretos, no es claro como conectar sus puntos para que su gráfica luzca como una curva. Tampoco es claro como se pueden aplicar relaciones geométricas. Como resultado, la geometría usada en grupos de curvas elípticas sobre $\mathbb{R}$ no es útil para grupos de curvas elípticas sobre $\mathbb{F}_p$. De cualquier forma las reglas algebraicas para la aritmética pueden ser adaptadas para curvas elípticas sobre $\mathbb{F}_p$ y a diferencia de las curvas elípticas sobre $\mathbb{R}$, los cálculos sobre $\mathbb{F}_p$ no involucran error de redondeo –una propiedad esencial requerida para un criptosistema.



§1.7 Grupos de curvas elípticas sobre $\mathbb{F}_{2^m}$

Los elementos de el campo $\mathbb{F}_{2^m}$ son cadenas de bits de longitud m. Las reglas para la aritmética en $\mathbb{F}_{2^m}$ pueden ser definidas ya sea por representación polinomial o por representación de bases normales óptimas. Dado que $\mathbb{F}_{2^m}$ opera sobre cadenas de bits, su aritmética no tiene errores de redondeo. Esto combinado con la naturaleza binaria del campo $\mathbb{F}_{2^m}$ da como resultado que la aritmética pueda ser desarrollada muy eficientemente por una computadora.

Para una curva elíptica $\mathcal{C}$ sobre $\mathbb{F}_{2^m}$ se requiere que $b \neq 0$ y como resultado de que el campo $\mathbb{F}_{2^m}$ tenga característica 2, la ecuación de $\mathcal{C}$ se ajusta ligeramente para su representación binaria

$$y^2 + x\ y = x^3 + a\,x^2 + b.$$

La suma con cadenas de bits es controlada por una función XOR.

Usualmente se utilizan criptosistemas de log discreto con campos de tipo $\mathbb{F}_{2^d}$ en vez de los de tipo $\mathbb{F}_p$ donde $p \approx 2^d \approx 10^{100}$. Sobre $\mathbb{F}_p$ son más seguros; sobre $\mathbb{F}_{2^d}$ son más fáciles de implementar en una computadora.

1.7.1 Ejemplo

Considere el campo $\mathbb{F}_{2^4} \cong \dfrac{\mathbb{F}_2}{\langle x^4 + x + 1\rangle}$, donde $f(x) = x^4 + x + 1 \ \in \mathbb{F}_2[x]$ es un polinomio irreducible en $\mathbb{F}_2$.

El elemento $g = (0010)$ es un generador del campo.

$$\begin{array}{cccc} g^0 = (0001) & g^1 = (0010) & g^2 = (0100) & g^3 = (1000) \\ g^4 = (0011) & g^5 = (0110) & g^6 = (1100) & g^7 = (1011) \\ g^8 = (0101) & g^9 = (1010) & g^{10} = (0111) & g^{11} = (1110) \\ g^{12} = (1111) & g^{13} = (1101) & g^{14} = (1001) & g^{15} = (0001) \end{array}$$

En una aplicación criptográfica real, el parámetro m debe ser suficientemente grande para impedir la generación eficiente de una tabla semejante, de otra manera el criptosistema puede ser roto. En la práctica actual $m = 160$ es una elección adecuada. La tabla permite el uso de notación de un generador (g^e), en vez de una cadena de bits como notación. También el uso de notación de un generador permite la multiplicación sin hacer referencia al polinomio irreducible

$$f(x) = x^4 + x + 1.$$

Considere la curva elíptica

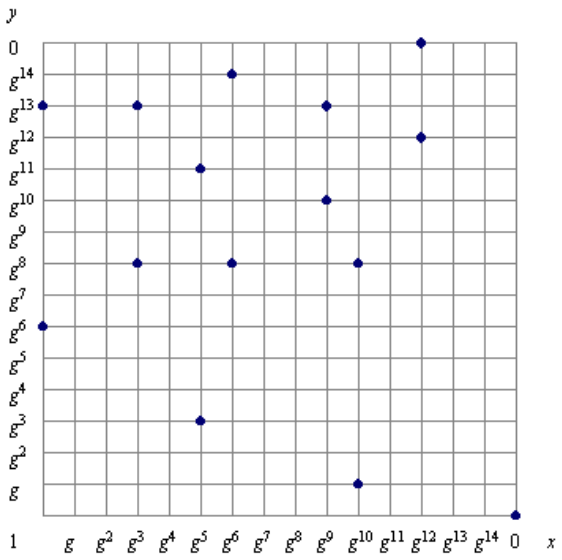
$$y^2 + xy = x^3 + g^4x^2 + 1, \quad a = g^4, \quad \text{y} \quad b = g^0 = 1.$$

El punto (g^5, g^3) satisface esta ecuación sobre $\mathbb{F}_{2^m}$

$$\begin{array}{rcl} y^2 + xy & = & x^3 + g^4x^2 + 1 \\ (g^3)^2 + g^5g^3 & = & (g^5)^3 + g^4g^{10} + 1 \\ g^6 + g^8 & = & g^{15} + g^{14} + 1 \\ (1100) + (0101) & = & (0001) + (1001) + (0001) \\ (1001) & = & (1001). \end{array}$$

Los 15 puntos que satisfacen esta ecuación son

$$\begin{array}{ccccc} (1, g^{13}) & (g^3, g^{13}) & (g^5, g^{11}) & (g^6, g^{14}) & (g^9, g^{13}) \\ (g^{10}, g^8) & (g^{12}, g^{12}) & (1, g^6) & (g^3, g^8) & (g^5, g^3) \\ (g^6, g^8) & (g^9, g^{10}) & (g^{10}, g) & (g^{12}, 0) & (0, 1) \end{array}$$


y
0
g^{14}
g^{13}
g^{12}
g^{11}
g^{10}
g^{9}
g^{8}
g^{7}
g^{6}
g^{5}
g^{4}
g^{3}
g^{2}
g
1
g g^2 g^3 g^4 g^5 g^6 g^7 g^8 g^9 g^{10} g^{11} g^{12} g^{13} g^{14} 0
x

2

Bases Criptográficas

Suponga que un emisor $\mathscr{E}$ desea enviar un mensaje $\mathbf{m}$[1] a un receptor $\mathscr{R}$. Sobre todo, $\mathscr{E}$ desea enviar el mensaje de forma segura; *i. e.*, desea asegurarse que un intruso $\mathscr{I}$ no pueda leer el mensaje.

Un mensaje es TEXTO LLANO (llamado también TEXTO CLARO o PLANO). Al proceso de disfrazar un mensaje de tal manera que ocultemos su substancia es llamado CIFRADO, por lo que un mensaje cifrado se llama TEXTO CIFRADO; y al proceso de recuperar el texto plano se llama DESCIFRADO.

A el arte y ciencia de mantener los mensajes seguros o en secreto se le llama CRIPTOGRAFÍA (CRYPTO = *secreto y* GRAFOS = *escritura*) y es practicada por CRIPTÓGRAFOS. CRIPTOANALISTAS son quienes practican el CRIPTOANÁLISIS. La rama de las matemáticas que abarca a la criptografía y al criptoanálisis se llama CRIPTOLOGÍA y la practican los CRIPTÓLOGOS.

Los criptosistemas antiguos tienen una historia de al menos 4000 años, entre ellos los egipcios cifraron algunos de sus jeroglifícos en los monumentos, los hebreos cifraron ciertas palabras en las escrituras bíblicas.

Hace 2000 años Julio César usó un sistema de simple sustitución, ahora conocido como CIFRADO DE CÉSAR. En el año 1200 d. C. Roger Bacon describió varios métodos de cifrado.

Geoffrey Chaucer incluyó varios "cifrados" en su trabajo.

León Alberti diseñó una rueda de cifrado, y describió los principios del análisis de frecuencias en 1460's.

Blaise de Vigenére publicó un libro sobre criptología en 1585 y describió el cifrado de sustitución polialfabético.

[1] puede ser flujo de bits, un archivo de texto, voz digitalizada, imagen de video digital, etc., etc.

§2.1 Nociones básicas

En la actualidad la CRIPTOGRAFÍA es el estudio de técnicas matemáticas relacionadas con aspectos de seguridad en la transferencia de información, tales como

- Confidencialidad o privacidad. Mantener la información secreta fuera del alcance de entidades no autorizadas.
- Integridad de datos. Debe ser posible para el receptor de un mensaje verificar que no ha sido modificado en el trayecto; un intruso no debe ser capaz de sustituir un mensaje legítimo por uno falso.
- Autenticidad. Identificar cada una de las partes que intervienen en el esquema. Debe ser posible para un receptor de un mensaje determinar su origen; un intruso no debe ser capaz de suplantar a nadie.
- Autenticidad del origen de la información. Autentificar es asegurar o garantizar que una entidad es quien dice ser, o que la información enviada no ha sido manipulada por partes no autorizadas. Las técnicas de "autentificación" del origen de datos o mensajes suministran a $\mathscr{R}$ la seguridad de la identidad de $\mathscr{E}$.
- Control de acceso. Restringir el acceso a los medios a entidades privilegiadas.
- No repudiación de origen. Un emisor no debe ser capaz de negar después falsamente que él envió un mensaje.

Las herramientas criptográficas básicas se evaluan con respecto a varios criterios, tales como

-- Nivel de seguridad. Esto es difícil de cuantificar, se da en términos del número de operaciones requeridas (usando los mejores métodos conocidos actualmente) para alcanzar el objetivo buscado. Se define también por una cota superior en la cantidad de trabajo, factor de trabajo, necesario para lograr un objetivo.

-- Funcionalidad. Se refiere a la interacción entre las primitivas al combinarse para obtener varios objetivos en la seguridad de la información. Las primitivas más efectivas para cierto objetivo se determinan por sus propiedades básicas.

-- Métodos de operación. Cuando se aplican primitivas de maneras distintas y con diferentes entradas exhiben distintas características, así una primitiva muestra funcionalidades muy distintas dependiendo del modo de operación o uso.

-- Desempeño. Se refiere a la eficiencia de una primitiva en un modo particular de operación.

– – Facilidad de implementación. Se refiere a la dificultad de poner marcha una primitiva en una instancia práctica. Esto puede incluir la complejidad de implementar la primitiva ya sea en un ambiente de software o hardware.

2.1.1 Vocabulario

TEXTO LLANO Mensaje que se desea enviar.

TEXTO CIFRADO Mensaje "disfrazado" con ayuda de una CLAVE y que es factible de que se le quite el "disfraz" con ayuda de una CLAVE, no necesariamente distinta.

CIFRAR Convertir el texto llano a texto cifrado.

DESCIFRAR Convertir de regreso el texto cifrado a texto llano.

CODIFICAR Convertir el texto llano a una cantidad (secuencia) de números.

DECODIFICAR Convertir de regreso la cantidad (secuencia) de números a texto llano. No hay nada secreto en codificar o decodificar.

CIFRADO DE FLUJO Opera sobre un mensaje símbolo a símbolo o bit a bit.

CIFRADO DE BLOQUE Opera sobre bloques de símbolos.

CIFRADO DE TRANSPOSICIÓN Rearregla o permuta símbolos, letras o bits.

CIFRADO DE SUSTITUCIÓN Reemplaza símbolos, letras o bits por otros, sin cambiar el orden.

CIFRADO DE PRODUCTO Alterna transposición y sustitución. El concepto de cifrado de flujo contra cifrado de bloque realmente se aplica solamente al cifrado de sustitución y producto, pero no al de transposición.

CRIPTOSISTEMA Conjunto de algoritmos necesarios para implementar una forma particular de cifrado y descifrado. Comúnmente consiste de 3 algoritmos: 1 para la generación de claves, 1 para cifrar y 1 para descifrar.

CRIPTOSISTEMA DE CLAVE PRIVADA También llamado SIMÉTRICO. Requiere de una clave, la cual ha sido acordada previamente por los 2 usuarios $\mathscr{E}$ y $\mathscr{R}$.

CRIPTOSISTEMA DE CLAVE PÚBLICA También llamado ASIMÉTRICO. Cada usuario tiene una clave de cifrado la cual es dada a conocer y una clave de descifrado que mantiene en secreto.

CRIPTANÁLISIS Es el proceso por el cual un enemigo trata de convertir el texto cifrado a texto llano.

Criptoanálisis (del griego "KRYPTÓS" *oculto*, y "ANALÝEIN" *aflojar* o *desatar* es el estudio de métodos para obtener el significado de información

cifrada, sin acceso a la información secreta. Típicamente, esto conlleva a encontrar la clave secreta[2].

Este término se usa también para referirse a cualquier intento de evadir la seguridad de otros tipos de algoritmos criptográficos y protocolos en general, y no solo cifrado, excluyendo ataques cuyo objetivo es debilitar en la criptografía actual; el soborno, coacción física, robo de claves y demás, aunque estos últimos son importantes en seguridad computacional, y estan llegando a ser más efectivos que el criptoanálisis tradicional.

Aún cuando el objetivo ha sido el mismo, los métodos y técnicas del criptoanálisis han cambiado drásticamente a través de la historia de la criptografía, adaptándose a la creciente complejidad criptográfica, con métodos que van desde lápiz y papel en el pasado, pasando por máquinas como Enigma en la 2a. guerra mundial, hasta esquemas actuales basados en computadoras. Los resultados del criptoanálisis han cambiado también, ya no es posible tener éxito ilimitado en la ruptura de códigos y existe una clasificación gerárquica de lo que constituye un ataque práctico raro. A mediados de los 70's, una nueva clase de criptografía fue introducida, Criptografía Asimétrica. Los métodos típicos de ruptura para estos criptosistemas son diametralmente opuestos a los de antes, y usualmente involucran la resolución de problemas construidos cuidadosamente en matemáticas puras, como por ejemplo el bien conocido de factorización entera.

Esquema de Cifrado o Criptosistema consta de

- $\mathcal{A}$ Conjunto finito llamado Alfabeto de Definición, $\mathcal{A} = \{0, 1\}$.
- $\mathcal{M}$ Conjunto llamado Espacio de Mensajes, el cual consiste de cadenas de símbolos de $\mathcal{A}$. Un elemento $\mathbf{m}$ de $\mathcal{M}$ se llama Texto Llano.
- $\mathcal{K}$ Conjunto llamado Espacio de Claves. Un elemento $\mathbf{k}$ de $\mathcal{K}$ se llama Clave.
- $\mathcal{K}_{\mathcal{S}}$ conjunto de claves privadas, secretas o de firmado.
- $\mathcal{K}_{\mathcal{V}}$ conjunto de claves públicas o de verificación.
- $\mathcal{C}$ Conjunto llamado Espacio de Texto Cifrado, el cual consiste de cadenas de símbolos de un alfabeto de definición (que puede diferir del alfabeto de definición para $\mathcal{M}$). Un elemento $\mathbf{c}$ de $\mathcal{C}$ se llama Texto Cifrado.

i) Cada elemento $\mathbf{e} \in \mathcal{K}$ determina de manera única una biyección $E_{\mathbf{e}} : \mathcal{M} \longrightarrow \mathcal{C}$ llamada Función de Cifrado o Transformación de Cifrado .

[2] En lenguaje no técnico, esta es la práctica de romper códigos, aunque tal expresión tenga un significado técnico especializado.

- El proceso de aplicar la trasformación $E_{\mathbf{e}}$ a un mensaje $\mathbf{m} \in \mathcal{M}$ se denomina CIFRADO DE $\mathbf{m}$.

ii) Cada elemento $\mathbf{d} \in \mathcal{K}$ determina de manera única una biyección $D_{\mathbf{d}} : \mathcal{C} \longrightarrow \mathcal{M}$ llamada FUNCIÓN DE DESCIFRADO o TRANSFORMACIÓN DE DESCIFRADO, con la propiedad de que para cada $\mathbf{e} \in \mathcal{K}$ existe una única clave $\mathbf{d} \in \mathcal{K}$ tal que $D_{\mathbf{d}} = E_{\mathbf{e}}^{-1}$.

- El proceso de aplicar la trasformación $D_{\mathbf{d}}$ a un texto cifrado $\mathbf{c} \in \mathcal{C}$ se denomina DESCIFRADO DE $\mathbf{c}$.

2.1.2 Nociones de seguridad

Si la seguridad de un algoritmo se basa en guardar en secreto la forma en que ese algoritmo funciona, se llama ALGORITMO RESTRINGIDO. Los algoritmos restringidos tienen interés histórico, pero son lamentablemente inadecuados para los estándares actuales. Un grupo cambiante o grande de usuarios no puede usarlos, porque cada vez que un usuario deja el grupo todos los demás deben cambiar a un algoritmo distinto. Si alguien accidentalmente revela el secreto, todos deben cambiar su algoritmo.

Aún peor, los algoritmos restringidos no permiten control de calidad o estandarización. Cada grupo de usuarios debe tener su propio algoritmo único. Tal grupo no puede usar productos de hardware o software almacenado; un intruso puede comprar el mismo producto y aprender el algoritmo; por lo que tendría que escribir su propio algoritmo e implementación. Si ninguno en el grupo es un buen criptógrafo, entonces no podrán saber si tienen un algoritmo seguro. Debido a estos inconvenientes principales, los algoritmos restringidos son enormemente populares para aplicaciones de baja seguridad.

La seguridad de todos los algoritmos de clave pública se basa en las claves; ninguno en los detalles del algoritmo. Esto significa que el algoritmo puede ser publicado o analizado. Productos que usan estos algoritmos pueden producirse en masa. No importa si un espía conoce el algoritmo, si no conoce su clave particular no puede leer el mensaje.

Se dice que un criptosistema tiene seguridad incondicional cuando, sin importar el poder de cómputo, éste no puede ser roto. Se dice que tiene seguridad computacional cuando con recursos limitados de cómputo el criptosistema no puede ser roto.

ATAQUE Distinguimos 2 tipos básicos de ataques.

- ATAQUE SÓLO SOBRE LA CLAVE. En este ataque el enemigo conoce únicamente la clave pública de $\mathscr{E}$.
- ATAQUE CON MENSAJES. Aquí al enemigo se le permite examinar algunos textos firmados correspondientes con sus mensajes, ya sean

sólo conocidos o escogidos, antes de intentar romper el esquema. Este ataque se subdivide en 4 tipos

1. Ataque con Mensajes Conocidos. El enemigo tiene acceso a textos firmados de un conjunto de mensajes $\mathbf{m}_1, \ldots, \mathbf{m}_t$ los cuales conoce, mas no los escoge.
2. Ataque Genérico con Mensajes Escogidos. Al enemigo se le permite escoger firmas válidas de $\mathscr{E}$ de una lista de mensajes $\mathbf{m}_1, \ldots, \mathbf{m}_t$ antes de que *él* intente romper el esquema de firma de $\mathscr{E}$. Estos mensajes los escoge el enemigo pero se fijan y son independientes de la clave pública de $\mathscr{E}$. Este ataque no es adaptativo: la lista completa de mensajes se construye antes de producir cualquier firma. Este ataque es genérico ya que no depende de la clave pública de $\mathscr{E}$.
3. Ataque Dirigido con Mensajes Escogidos. Es similar al ataque genérico con mensajes escogidos, excepto que la lista de mensajes a firmar puede crearse después de conocer la clave pública de $\mathscr{E}$ pero antes de producirse cualquier firma. Este ataque es dirigido en contra de un usuario $\mathscr{E}$ en particular.
4. Ataque Adaptativo con Mensajes Escogidos. Al enemigo se le permite utilizar al usuario $\mathscr{E}$ como un oráculo; no sólo se puede solicitar firmas de $\mathscr{E}$ de mensajes que dependan de la clave pública de $\mathscr{E}$, sino también de mensajes que adicionalmente dependan de firmas previamente obtenidas.

2.1.3 Ruptura de esquemas

Ruptura Total Obtener la información secreta de $\mathscr{E}$.

Falsificación Universal Encontrar un algoritmo eficiente de firma que sea funcionalmente equivalente al de $\mathscr{E}$.

Falsificación Selectiva Falsificar una firma de un mensaje particular escogido *a priori* por el enemigo $\mathscr{I}$.

Falsificación Existencial Falsificar una firma de al menos un mensaje. El enemigo no tiene control sobre el mensaje de la firma que obtiene.

§2.2 Criptografía de clave privada

2.2.1 Criptografía de bloque

El Cifrado de Bloque separa el texto llano en bloques de longitud fija, para cifrarlos uno por uno. Usualmente, con el cifrado de bloque se cifran textos planos de una longitud dada, en bloques texto de la misma longitud.

DES

DES (Data Encryption Standard) es un esquema de cifrado y descifrado de bloque, fue desarrollado por IBM y NSA en 1974 en respuesta a una invitación pública del gobierno federal para algoritmos de cifrado de datos. En 1977 DES fue publicado como un estándar federal, FIPS PUB 46.

DES usa 16 rondas del CIFRADO DE FEISTEL con bloques de 64 bits.

El cifrado de Feistel (Horst Feistel) [Fei73] es un criptosistema de bloque iterativo, es también un método general de transformar cualquier función (llamada $F-$función) en una permutación que separa la mitad del bloque de bits para hacer que esta opere sobre la otra mitad; la $F-$función, usa una de las mitades del bloque de datos y la clave para crear un flujo de bits pseudo-aleatorios que se usa para cifrar o descifrar la otra mitad del bloque. Por lo que para cifrar o descifrar ambas mitades se requieren 2 iteraciones del algoritmo de Feistel.

Cifrado de Feistel
Entrada: $\mathbf{m}$ ($2t$ bits de texto llano),
$k_1,\ k_2, \ldots,\ k_r$ (ciclos clave),
f (cifrado de bloque con bloques de longitud t)
Salida: $\mathbf{c}$ ($2t$ bits de texto cifrado)
Inicia: $(L_0, R_0) = \mathbf{m}$ (divide $\mathbf{m}$ en 2 partes L_0 y R_0 de t bits cada una)
Paso 1: $(L_1, R_1) = (R_0, L_0 \oplus f(R_0, k_1))$
Paso 2: $(L_2, R_2) = (R_1, L_1 \oplus f(R_1, k_2))$
Paso 3: $(L_3, R_3) = (R_2, L_2 \oplus f(R_2, k_3))$
Paso 4: ...
Paso 5: $(L_{r-1}, R_{r-1}) = (R_{r-2}, L_{r-2} \oplus f(R_{r-2}, k_{r-1}))$
Paso 6: $(L_r, R_r) = (R_{r-1}, L_{r-1} \oplus f(R_{r-1}, k_r))$
Paso 7: $\mathbf{c} = (R_r, L_r)$ (intercambia las 2 partes)

$\oplus$ es la operación XOR.

El algoritmo de descifrado para un cifrado de bloque debe ser idéntico al de cifrado, paso a paso pero en orden inverso. Pero para DES, el algoritmo de cifrado está tan bien diseñado que el algoritmo de descifrado es idéntico al de cifrado paso por paso en el mismo orden, solo se aplica a las subclaves en orden inverso.

DES se usa como base para el programa de PASSWORD de UNIX y ha sido suplantado por RSA como un hecho de los estandáres industriales debido a la longitud de las claves y seguridad. Sin embargo triple DES es mucho más fuerte.

El Código de Vigenère

El criptosistema de Vigenère –y todos los otros sistemas que existían antes de la década de 1970– son criptosistemas de Clave Privada.

Para cifrar un mensaje se escoge una PALABRA-CLAVE de k letras, la cual se traduce en una secuencia de k números enteros –vector fijo $\vec{a} \in \mathbb{Z}_n^k$– a través de la correspondencia dada entre el alfabeto y el grupo $\mathbb{Z}_n$, donde n es el número de símbolos en el alfabeto.

Luego se divide al mensaje en bloques de k letras –un k-vector $\in \mathbb{Z}_n$–.

Se cifra trasladando cada bloque por el vector fijo $\vec{a}$, *i. e.*, trasladamos cada letra por el número de la letra de la palabra-clave que ocupa el lugar correspondiente; después de la $k-$ésima letra del texto hay que comenzar de nuevo desde el principio de la palabra-clave.

Suponga que un mensaje dado ha sido cifrado a través de una palabra-clave de k letras, pero no se sabe cuál es esta palabra.

Si tenemos un texto largo, entonces podemos utilizar un análisis de frecuencia para hallar la palabra-clave y descifrar el código. Para ello se determina la $j-$ésima letra de la clave ($1 \leq j \leq k$) usando únicamente las letras del texto cifrado ubicadas en los lugares $j,\ j+k,\ j+2k,\ j+3k, \ldots$. La letra que ocurre con más frecuencia seguramente es el cifrado de la letra e (aunque en algunos casos la e no corresponde a la letra que ocurre con más frecuencia sino, digamos, a la segunda letra más común). Esta información es suficiente para determinar la $j-$ésima letra de la palabra-clave.

Aunque el criptosistema de Vigenère no tiene un alto nivel de seguridad, en el límite cuando $k \longrightarrow \infty$ (*i. e.*, cuando la clave tiene la misma longitud que el mensaje) este sistema se vuelve muy seguro — de hecho, éste es el único criptosistema con seguridad perfecta; en inglés se le llama ONE-TIME PAD.

§2.3 Criptografía de flujo

En criptografía, se llama cifrado de flujo al proceso de cifrar texto plano dígito por dígito, además de ser simétrico, y en el cual la transformación de dígitos sucesivos varía durante el cifrado. También se llama cifrado de estado, pues el cifrado de cada dígito depende de su estado actual. De hecho, los dígitos son comúnmente un solo bit o un solo byte.

El cifrado de flujo se ejecuta normalmente más rápido que el de bloque y tiene menor complejidad de hardware, sin embargo puede ser suceptible a serios problemas de seguridad si se usa incorrectamente.

2.3.1 RC2

RC2 es un criptosistema de bloque diseñado por Ron Rivest en 1987. "RC" viene de "Ron's Code" or "Rivest Cipher"; entre otros diseñados por Rivest están RC4, RC5 y RC6.

RC2 es un criptosistema de bloques de 64 bits con una clave de longitud variable. Sus 18 ciclos están arreglados como una red Feistel, con 16 ciclos de un tipo (MIXING) el cual se vale de otros 2 ciclos de otro tipo (MASHING). Un ciclo MIXING consiste de 4 aplicaciones de una transformación llamada MIX. RC2 es vulnerable a ataques relacionados con la clave usando 2^{34} textos llanos escogidos.

2.3.2 RC4

Por el camino de Vietnam llamado RC4, "Route Coloniale 4". RC4 es el software de cifrado de flujo más ampliamente usado en protocolos populares tales como SSL (Secure Sockets Layer) para la protección del tráfico en internet y WEP (para la seguridad en redes inalámbricas). Aún cuando es notoria su simplicidad, RC4 falla rápidamente en los altos estándares de seguridad establecidos por los criptógrafos, y de alguna manera el usar RC4 lleva a criptosistemas muy inseguros (incluyendo WEP). No se recomienda para uso en nuevos sistemas. Sin embargo, algunos sistemas basados en RC4 son suficientemente seguros para uso práctico.

RC4 fue diseado por Ron Rivest de RSA Security en 1987; aunque es oficialmente llamado "Rivest Cipher 4".

RC4 genera un flujo pseudoaleatorio de bits ("flujo clave") el cual, para el cifrado, se combina con el texto plano usando el XOR como con el cifrado de Vernam [Ver26]; el descifrado se hace de la misma manera. Para generar el flujo clave, el cifrado hace uso de un estado interno secreto el cual consiste de 2 partes

i) Una permutación de todos los 256 bytes posibles.

ii) 2 apuntadores índice de 8 bits.

La permutación es inicializada con una clave de longitud variable, típicamente entre 40 y 256 bits, usando el algoritmo de programación de claves KSA (Key-Scheduling Algorithm). Una vez que esto se ha completado, el flujo de bits es generado usando el algoritmo de generación pseudoaleatoria.

Varios cifrados de flujo se basan en retroalimentación lineal de registros de corrimiento LFSRs (Linear Feedback Shift Registers), y, aunque son eficientes en hardware, son muy lentos en software. El diseño de RC4 es totalmente diferente, y es ideal para implementaciones de software, y solo requiere manipulaciones en la longitud en bytes. Usa 256 bytes de memoria para el arreglo de estado, k

bytes de memoria para la clave, y variables enteras. El desarrollo módulo 256 se puede hacer con un AND bit a bit con 255 (o sobre algunas plataformas, con simple adición de bytes ignorando el desbordamiento (overflow)).

§2.4 Criptografía de clave pública

La criptografía de Clave Pública fue propuesta por primera vez en 1976 por WHITFIELD DIFFIE y MARTIN HELLMAN. Aquí la idea central es que la función de cifrado sea accesible a cualquier persona, usando la clave pública; el concepto fundamental de la criptografía de clave pública

"FUNCIÓN DE UN SOLO SENTIDO".

Sin embargo, la función inversa (de descifrado) puede ser calculada solamente por la persona que posee otra clave – la secreta.

Además del cifrado de mensajes secretos, la criptografía de clave pública tiene muchas otras aplicaciones, tales como las firmas digitales.

Definición 22 *Un esquema de cifrado se dice de* CLAVE PÚBLICA *si para cada pareja de claves de cifrado-descifrado asociadas, la clave* $\mathbf{e}$ *de* CIFRADO *se hace disponible (* CLAVE PÚBLICA*), mientras que la de* DESCIFRADO $\mathbf{d}$ *se mantiene en secreto (* CLAVE PRIVADA*).*

Cifrado y descifrado deben ser computacionalmente fáciles para usuarios propios y el descifrado debe ser computacionalmente difícil para espías y enemigos. Además se asume que el enemigo conoce la naturaleza del criptosistema pero no la clave privada.

Para que el esquema sea seguro, conociendo $E_{\mathbf{e}}$ debe ser computacionalmente inviable, dado un texto cifrado $\mathbf{c}$, encontrar el mensaje $\mathbf{m}$ tal que $E_{\mathbf{e}}(\mathbf{m}) = \mathbf{c}$, lo cual implica que conociendo $\mathbf{e}$, es computacionalmente inviable determinar $\mathbf{d}$. $E_{\mathbf{e}}$ es considerada una función de digestión o de resumen.

Para transmitir un mensaje $\mathbf{m}$, $\mathscr{E}$ utiliza la clave pública $\mathbf{e}_{\mathscr{R}}$ de $\mathscr{R}$ para cifrarlo, luego lo transmite. El mensaje cifrado (texto cifrado) será

$$\mathbf{c} = E_{\mathbf{e}_{\mathscr{R}}}(\mathbf{m}),$$

$\mathscr{R}$ utiliza su clave privada $\mathbf{d}_{\mathscr{R}}$ para descifrar el mesaje y leerlo

$$\mathbf{m} = D_{\mathbf{d}_{\mathscr{R}}}(\mathbf{c})$$

Este sistema debe satisfacer

- $D_{\mathbf{d}_{\mathscr{R}}}(E_{\mathbf{e}_{\mathscr{R}}}(\mathbf{m})) = D_{\mathbf{d}_{\mathscr{R}}}(\mathbf{c}) = \mathbf{m}, \forall\, \mathbf{m} \in \mathcal{M}$ (Propiedad fundamental).
- Obtener $\mathbf{e}$ a partir de $\mathbf{d}$ debe de ser tan difícil como leer $\mathbf{m}$ (Seguridad).

- Tanto **e** como **d** son fáciles de calcular (Factibilidad de uso).

La seguridad de diversos criptosistemas de clave pública basan su seguridad en problemas matemáticos difíciles de resolver, tales como: logaritmo discreto, factorización en números primos, encontrar un vector corto o uno muy cercano a uno dado en un retículo, entre otros.

2.4.1 Esquemas de firma digital

Un criptograma fundamental para verificar la autenticidad, autorización y no rechazo de un mensaje, es la firma digital. El propósito de una firma digital es proporcionar un medio para vincular a una entidad con cierta información. El proceso de firmar conlleva la transformación del mensaje y de cierta información secreta sujeta por la identidad a una etiqueta llamada FIRMA DIGITAL.

Definamos algunos términos.

- $\mathcal{M}$ es el conjunto de todos los mensajes que pueden ser firmados.
- $\mathcal{S}$ es el conjunto de todos los mensajes firmados o firmas (posiblemente cadenas binarias de longitud fija).
- $S_{\mathscr{E}}$: $\mathcal{M} \longrightarrow \mathcal{S}$ es la transformación de firmado para una entidad $\mathscr{E}$, quien la mantiene en secreto y la usa para crear firmas de mensajes.
- $V_{\mathscr{E}}$: $\mathcal{M} \times \mathcal{S} \longrightarrow \{falso, verdadero\}$ es la transformacion de verificación para las firmas de $\mathscr{E}$, es públicamente conocida y es usada por otras entidades para verificar las firmas creadas por $\mathscr{E}$.

$S_{\mathscr{E}}$ y $V_{\mathscr{E}}$ definen un ESQUEMA o MECANISMO DE FIRMA DIGITAL para $\mathscr{E}$.

Definición 23 *Un* ESQUEMA DE FIRMA *es una terna* (Gen, Sig, Ver) *de algoritmos probabilísticos de tiempo polinomial tales que*

i) Gen *es un* ALGORITMO GENERADOR DE CLAVES, *con entrada* 1^s*;* PARÁMETRO DE SEGURIDAD*; y posiblemente alguna otra información adicional* I*, y que da como salida un par* $(\mathbf{k}_{\mathcal{S}}, \mathbf{k}_{\mathcal{V}}) \in \mathcal{K}_{\mathcal{S}} \times \mathcal{K}_{\mathcal{V}}$ *llamados* CLAVE PRIVADA, SECRETA *o* DE FIRMADO *y* CLAVE PÚBLICA *o* DE VERIFICACIÓN *respectivamente.*

ii) Sig *es un* ALGORITMO DE FIRMADO, *con entrada* $(\mathbf{m}, \mathbf{k}_{\mathcal{S}}) \in \mathcal{M} \times \mathcal{K}_{\mathcal{S}}$ *y que da como salida un elemento* $\sigma \in \mathcal{S}$*, denominado* FIRMA *(del mensaje* $\mathbf{m}$ *con la clave* $\mathbf{k}_{\mathcal{S}}$*).*

iii) Ver *es un* ALGORITMO DE VERIFICACIÓN, *con entrada* $(\mathbf{m}, \sigma, \mathbf{k}_{\mathcal{V}})$ *en* $\mathcal{M} \times \mathcal{S} \times \mathcal{K}_{\mathcal{V}}$ *y que da como salida un elemento en* $\{verdadero, falso\}$ *y* $Ver(\mathbf{m}, Sig(\mathbf{m}, \mathbf{k}_{\mathcal{S}}), \mathbf{k}_{\mathcal{V}}) = verdadero \ \forall \ (\mathbf{k}_{\mathcal{S}}, \mathbf{k}_{\mathcal{V}})$ *obtenido de* Gen *y todo* $\mathbf{m} \in \mathcal{M}$*.*

La seguridad deseada para un esquema de firma digital es aquella que está basada en la imposibilidad computacional de obtener falsificaciones existenciales por medio de los ataques mencionados en la sección 2.1.2.

Para crear una firma $\mathscr{E}$ calcula $\mathbf{s} = Sig(\mathbf{m})$ y luego transmite el par $(\mathbf{m}, \mathbf{s})$ el cual es llamado FIRMA PARA EL MENSAJE $\mathbf{m}$.

Para verificar la firma $(\mathbf{m}, \mathbf{s})$, $\mathscr{R}$ primero obtiene la función Ver y calcula $u = Ver(\mathbf{m}, \mathbf{s})$, y acepta el mensaje si, y sólo si $u = verdadero$.

Sig y Ver se caracterizan por una clave, es decir, existe una clase de algoritmos de verificación y de firmado públicamente conocidos, y cada algoritmo es identificado por una clave. El algoritmo Sig de $\mathscr{E}$ está determinado por la clave $\mathbf{k}_{\mathscr{E}}$, por lo que $\mathscr{E}$ requiere mantener en secreto $\mathbf{k}_{\mathscr{E}}$ solamente. Similarmente el algoritmo de verificación Ver de $\mathscr{E}$ está determinado por la clave $\mathbf{k}_{\mathscr{E}}$ la cual es pública[3].

Sig y Ver son tales que

- $\mathbf{s}$ es una firma válida sobre el mensaje $\mathbf{m} \in \mathcal{M}$ de $\mathscr{E}$ si, y sólo si,

$$Ver(\mathbf{m}, \mathbf{s}) = verdadero.$$

- Es computacionalmente inviable para una entidad distinta de $\mathscr{E}$ encontrar dado un mensaje $\mathbf{m} \in \mathcal{M}$ un $\mathbf{s} \in \mathcal{S}$ tal que $Ver(\mathbf{m}, \mathbf{s}) = verdadero$.

2.4.2 Funciones de resumen y firmas digitales

El uso más común de una función de resumen es en la creación de firmas digitales y en la verificación de integridad de la información.

Un valor de resumen es un representante compacto de una cadena de bits entrada, para cifrar la función de resumen se escoge de tal manera que sea computacionalmente inviable encontrar 2 entradas distintas con valores de resumen iguales. Un mensaje largo es resumido y solo se firma el valor de resumen.

El receptor $\mathscr{R}$ digiere el mensaje y verifica que la firma recibida sea la correcta para este valor de resumen. Esto ahorra tiempo y espacio comparado con firmar un mensaje directamente, lo cual implica la partición de éste en bloques de longitud apropiada y el firmar cada bloque individualmente. Además de que esto contribuye con el requerimiento de seguridad, pues no permite encontrar 2 mensajes con el mismo valor de resumen, de otra manera la firma de un mensaje sería igual a la de otro permitiendo a un emisor $\mathscr{E}$ firmar un mensaje y después negarlo o decir que firmó otro mensaje.

[3] La clave pública $\mathbf{e}$ se usa para verificar y la clave privada $\mathbf{d}$ se usa para firmar.

Para la verificación de integridad en la información se calcula el valor de resumen a determinada hora y se protege la integridad de este valor. Pasado un tiempo, para verificar que la entrada de datos no haya sido alterada, se recalcula este valor y se compara con el valor previo. Aplicaciones específicas de verificación de integridad de datos incluyen la protección de virus y la distribución de software. En este caso las funciones de resumen se conocen como Códigos de Autentificación de Mensajes (MACs Message Authentication Code).

§2.5 RSA

El primer criptosistema de clave pública que fue usado ampliamente (y todavía el sistema más popular)

$$RSA \quad (Rivest,\ Shamir,\ Adleman).$$

Está basado en la aplicación de algoritmos aritméticos sobre enteros muy grandes; específicamente en el Problema de la Factorización en Números Primos

Dado un $n \in \mathbb{N}$ número compuesto, encontrar un factor; $p \in \mathbb{Z}$ número primo; de n, *i. e.*, $p|n$.

2.5.1 Creación de claves

1. Se generan 2 números primos grandes distintos p y q (en la práctica cada uno de estos es de aproximadamente 100 dígitos además de que deben ser no cercanos).

2. Se calcula $n = pq$ (Módulo de Cifrado) y $\phi(n) = (p-1)(q-1)$.

3. Se selecciona aleatoriamente $e \in \mathbb{Z}$ grande (Exponente de Cifrado), $1 < e < \phi(n)$, tal que $(e, \phi(n)) = 1$, *i. e.* $(e, \phi(p)) = (e, \phi(q)) = 1$.

4. Se calcula el único $d \in \mathbb{Z},\ 1 < d < \phi(n)$ (Exponente de Descifrado), que cumple $ed \equiv 1 \pmod{\phi(n)}$ ($ed \equiv 1 \pmod{\phi(p)}$ y $ed \equiv 1 \pmod{\phi(q)}$); para esto se usa el Algoritmo de Euclides.

La clave pública es la pareja (n, e) y la privada es d (aunque p y q se mantienen en secreto también).

2.5.2 Cifrado

$\mathscr{E}$ cifra un mensaje $\mathbf{m}$, el cual debe descifrar $\mathscr{R}$.

1. $\mathscr{E}$ utiliza la clave pública de $\mathscr{R}$, $(n, e) = \mathbf{e}_{\mathscr{R}}$.

2. Representa el mensaje $\mathbf{m}$ como un entero en el intervalo $[1, n-1]$ (en la práctica el mensaje $\mathbf{m}$ se divide en números menores a n, por ejemplo tomando cada vez $\log n$ bits de la cadena binaria correspondiente a su codificación por carácteres).

3. Para cifrar $\mathbf{m}$, se calcula $\mathbf{c} \equiv \mathbf{m}^e \pmod{n}$ (para cifrar una parte de $\mathbf{m}$, se eleva cada uno de estos números de forma independiente a la potencia $e \pmod{n}$).

4. $\mathscr{E}$ envía a $\mathscr{R}$ el texto cifrado $\mathbf{c}$.

La función de un solo sentido de RSA es

$$x \longmapsto x^e \pmod{n}.$$

2.5.3 Descifrado

Para recuperar el mensaje original $\mathbf{m}$ a partir de $\mathbf{c}$, $\mathscr{R}$ calcula $\mathbf{m} \equiv \mathbf{c}^d \pmod{n}$.

La función inversa aquí es

$$x \longmapsto x^d \pmod{n}.$$

Para romper la clave secreta de RSA, hay que encontrar la factorización de n. En la actualidad existe un algoritmo de la clase $L(1/3)$ para hacer eso.

A la fecha de manera comercial se utiliza el algoritmo RSA para cifrar y descifrar mensajes, existen para ello estándares[4].

- PKCS[5], para el manejo de claves RSA y firmas con estas claves, PKCS #1[6], PKCS #6[7] y PKCS #10[8].
- FIPS PUB[9] 186, DSS[10] para firmas digitales.
- RFC[11] 1321, MD5 algoritmos de resumen.
- **PGP** *Pretty Good Privacy* es un sistema de clave pública para el cifrado de correo electrónico que utiliza RSA para cifrar. Cifra el sistema SWISS IDEA con una clave generada aleatoriamente. Luego cifra la clave usando la clave pública del receptor. Cuando el receptor recibe el mensaje, PGP

[4] http://www.rsasecurity.com/rsalabs/pkcs,
http://www.nist.gov/(National Institute of Standards and Technology),
http://www.ansi.org/(American National Standards Institute)
[5] PKCS: Public Key Cryptography Standard
[6] RSA Cryptogrphy Standard
[7] Extended-Certificate Syntax Standard
[8] Certificate Request Syntax Standard
[9] FIPS PUB: Federal Information Processing Standard PUBlication
[10] Digital Signature Standard
[11] RFC: Request for Comments

usa su clave privada RSA para descrifrar la clave IDEA y usa entonces la clave IDEA para descrifrar el mensaje. Esto fue ideado por Zimmermann [McL06, Gar94].

- **DES** vea 2.2.1.
- **IDEA** Desarrollado por Dr. X. Lai y el profesor J. Massey en Suiza a principios de los 90's para reemplazar el estandar DES. Es un sistema de clave simétrica de bloque, opera sobre 8 bytes a la vez, como DES, pero con una clave de 128 bits. Esta longitud de la clave lo hace imposible de romper por fuerza bruta, sin embargo no se conoce otro ataque.

2.5.4 Ejemplo didáctico

Sean

$$p = 47, \quad q = 79, \quad n = pq = 3713, \quad \phi(n) = (p-1)(q-1) = 3588, \quad e = 97.$$

Usando el Algoritmo de Euclides encontramos

$$d = 37, \quad ed \equiv 1 \pmod{\phi(n)},$$

el único entero que multiplicado por 97 da resto 1 cuando se divide por 3588.

Clave Pública

$$(n, e) = (3713, 97)$$

Clave Privada

$$d = 37.$$

Con la codificación estándar (A=01, B=02, etc...), el mensaje

ATAQUE AL AMANECER

corresponde al número de 36 cifras

$$\mathbf{m} = 012001172105000112000113011405030518$$

Dividimos el mensaje en paquetes de cuatro cifras que se elevan a la potencia $e \pmod{n}$, lo cual nos da el Mensaje Cifrado

$$\mathbf{c} = 140403340803000108231215181505271657$$

es decir

$$\begin{aligned} 0120^{97} &\equiv 1404 \pmod{3713}, \\ 0117^{97} &\equiv 0334 \pmod{3713}, \\ 2105^{97} &\equiv 0803 \pmod{3713} \end{aligned}$$

y esto es lo que se envía.

El proceso de descifrado es el mismo pero utilizando d en lugar de e. Así retrocediendo se encuentra el mensaje original pues

$$\begin{aligned} 1404^{37} &\equiv 0120 \pmod{3713}, \\ 0334^{37} &\equiv 0117 \pmod{3713}, \\ 0803^{37} &\equiv 2105 \pmod{3713}. \end{aligned}$$

§2.6 Criptografía de curvas elípticas (CCE)

Suponga que tenemos manera de convertir un mensaje o unidades de mensaje (un bloque de texto) a un entero grande, y luego a un punto $\mathbf{m} \in \mathcal{C}$ el cual queremos enviar.

La clave pública es $\mathcal{C}/\mathbb{F}_q$ y un punto fijo (punto base) $\mathbf{P} \in \mathcal{C}$.

La clave secreta de $\mathscr{R}$ es un número $k \in \mathbb{Z}$ aleatorio y su clave pública es el punto $k\mathbf{P} = \mathbf{Q}$.

El $\mathscr{E}$ quiere enviar a $\mathscr{R}$ el mensaje $\mathbf{m} \in \mathcal{C}$. Para esto escoge aleatoriamente $t \in \mathbb{Z}$, y calcula los 2 puntos $t\mathbf{P}$ y $t\mathbf{Q}$ y envía a $\mathscr{R}$

$$(t\mathbf{P}, t\mathbf{Q} + \mathbf{m});$$

es decir, cifra el mensaje al adherir el punto $t\mathbf{Q}$; pero este punto también es igual a $t(k\mathbf{P})$, y $\mathscr{R}$ puede descifrar multiplicando el primer punto por su clave secreta y sustrayendo el resultado del segundo punto

$$t\mathbf{Q} + \mathbf{m} - t(k\mathbf{P}) = \mathbf{m}.$$

¿Qué tendría que hacer un espía para encontrar el mensaje $\mathbf{m}$? Resolver el "problema del logaritmo discreto" (PLD) en el grupo E.

2.6.1 Firmas digitales

Sea $\mathcal{C}$ una curva elíptica sobre $\mathbb{F}_q$ tal que $\sharp\mathcal{C}(\mathbb{F}_q)$ es igual a un primo ℓ de al menos 160 bits (o un múltiplo entero pequeño de ℓ.)

Sea $\mathbf{P}$ un $\mathbb{F}_q-$punto de $\mathcal{C}$ de orden ℓ.

Sea $f_{\mathcal{C}} \colon \mathcal{C}(\mathbb{F}_q) \longrightarrow \mathbb{F}_\ell$ una función fija, fácilmente calculable que dispersa los puntos sobre $\mathbb{F}_\ell$ uniformemente. Por ejemplo, si q es un primo p, $f_{\mathcal{C}}$ podría ser el residuo $\bmod(\ell)$ de la coordenada x de un punto (considerada como un entero entre 1 y $p-1$).

Sea $H \colon \mathcal{M} \longrightarrow \mathbb{F}_\ell$ una función de digestión. H debe tener ciertas propiedades de aleatoriedad y de intratabilidad para regresarse.

Clave secreta de $\mathscr{E}$: $x \in \mathbb{Z}$ aleatorio en el rango $1 < x < \ell$.

Clave pública de $\mathscr{E}$: El punto $\mathbf{Q} = x\mathbf{P} \in \mathcal{C}(\mathbb{F}_q)$.

Para firmar un mensaje $\mathbf{m}$, $\mathscr{E}$ hace lo siguiente

1. Selecciona aleatoriamente $k \in \mathbb{Z}$ en el rango $1 < k < \ell$
2. Calcula $k\mathbf{P}$ y $r = f_{\mathcal{C}}(k\mathbf{P})$
3. Calcula $k^{-1} \in \mathbb{F}_\ell$ y $s = k^{-1}(H(\mathbf{m}) + xr) \in \mathbb{F}_\ell$

Su firma para el mensaje $\mathbf{m}$ es el par (r, s).

Para verificar la firma, $\mathscr{R}$ calcula

$$u_1 = s^{-1}H(\mathbf{m}) \in \mathbb{F}_\ell, \quad u_2 = s^{-1}r \in \mathbb{F}_\ell,$$

y por último

$$u_1\mathbf{P} + u_2\mathbf{Q} \in \mathcal{C}(\mathbb{F}_q).$$

Si $f_{\mathcal{C}}(u_1\mathbf{P} + u_2\mathbf{Q}) = r$, $\mathscr{R}$ acepta la firma.

2.6.2 Problema del logaritmo discreto

En la fundación de cada criptosistema está un problema matemático "difícil" de resolver, es decir computacionalmente inviable de resolver. El problema del logaritmo discreto (PLD) es la base para la seguridad de varios criptosistemas incluyendo el criptosistema de curvas elípticas. Específicamente, la CCE se basa en la dificultad del problema del logaritmo discreto en curvas elípticas (PLDCE).

La determinación de un punto nP (suma de P, n veces) se conoce como MULTIPLICACIÓN ESCALAR. El PLDCE se basa en la intratabilidad de productos de multiplicación escalar. Con notación multiplicativa, esta operación consiste en multiplicar k copias del punto P, obteniendo

$$\underbrace{\mathbf{P} \cdot \mathbf{P} \cdots \mathbf{P} \cdot \mathbf{P}}_{k \text{ veces}} = \mathbf{P}^k.$$

Definición 24 *El PLD en el grupo E para la base $g \in E$ es el problema, dado $y \in E$, de encontrar $x \in \mathbb{Z}$ tal que $g^x = y$ (o $xg = y$ si la operacion en el grupo es la suma), si tal x existe (es decir si y está en el subgrupo generado por g.)*

El PLD en el grupo multiplicativo $\mathbb{F}_q^*$ es lo que se necesita resolver para invertir la función

$$x \longmapsto g^x,$$

donde $g \in \mathbb{F}_q^*$ es un elemento fijo y $x \in \mathbb{Z}$ (módulo el orden de g). Esta función es la función de un solo sentido usada en los criptosistemas de Diffie - Hellman.

Similarmente, si $\mathcal{C}$ es una curva elíptica sobre $\mathbb{F}_q$, se tiene la función en el grupo de curva elíptica E

$$x \longmapsto x\mathbf{P}$$

donde $\mathbf{P} \in \mathcal{C}$ es un punto fijo y $x \in \mathbb{Z}$. En este caso, el problema del log discreto consiste en encontrar, dado $\mathbf{Q} \in \mathcal{C}(\mathbb{F}_q)$, $x \in \mathbb{Z}$ tal que $\mathbf{Q} = x\mathbf{P}$ si tal entero existe.

La seguridad de todos los criptosistemas de curvas elípticas depende de la presumible intratabilidad del PLDCE.

El PLD en el grupo multiplicativo de un campo finito $\mathbb{F}_q$ bien escogido es difícil, de hecho parece que requiere aproximadamente el mismo tiempo que la factorización de un entero N de aproximadamente el mismo tamaño del campo finito $\sharp\mathbb{F}_q = q$, es decir $N \approx q$.

Teorema 1 (Hasse) *Sea $\mathcal{C}$ una curva elíptica sobre $\mathbb{F}_q$, entonces para el número $\sharp\mathcal{C}(\mathbb{F}_q)$ de $\mathbb{F}_q$-puntos de la curva $\mathcal{C}$ existe una estimación*

$$|\sharp\mathcal{C}(\mathbb{F}_q) - \sharp\mathbb{P}(\mathbb{F}_q)| \leq 2\sqrt{q}$$

donde $\sharp\mathbb{P}(\mathbb{F}_q) = q + 1$ es el número de puntos sobre la línea proyectiva $\mathbb{P}(\mathbb{F}_q)$ sobre $\mathbb{F}_q$.

Si $\mathbb{K}$ es un campo finito, entonces E es un grupo finito. Según el Teorema de Hasse, el número de elementos en E satisface

$$|\sharp E - (q+1)| \leq 2\sqrt{q}. \tag{2.1}$$

Es decir, $\sharp E$ es de la misma magnitud que $\sharp\mathbb{K} = q$.

De cualquier manera, como en el caso del problema de factorización, existen muchos algoritmos de tiempo subexponencial, es decir el número de pasos es asintóticamente menor que $\exp((\log N)^{\gamma})$, con $0 < \gamma < 1$ y N el número a factorizar; para el problema del logaritmo discreto en el grupo multiplicativo de $\mathbb{F}_q$.

La mayoría tienen tiempos de corrida de la forma

$$\exp\left((\log N)^{\frac{1}{2}+\epsilon}\right).$$

En la última década con la criba de campos numéricos y la función de la criba de campos, el tiempo asintótico de corrida ha sido reducido a

$$\exp\left((\log N)^{\frac{1}{3}+\epsilon}\right).$$

Sin embargo, no se conoce ningún algoritmo subexponencial para curvas elípticas, salvo en los siguientes casos especiales

- no hay un primo grande ℓ que divida a $\sharp\mathcal{C}(\mathbb{F}_q)$, es decir, $\#\mathcal{C}\ (\mathbb{F}_q)$ se factoriza como producto de primos relativamente pequeños
- existe un primo grande ℓ, pero que tambien divide a $q^k - 1$ para un k muy pequeño (en tal caso Menezes-Okamoto-Vanstone [MVO91] demostraron como se puede imbuir el grupo de la curva elíptica en $\mathbb{F}^*_{q^k}$).
- $\mathbb{F}_q$ es un campo primo $\mathbb{F}_p$ y $\sharp\mathcal{C}(\mathbb{F}_p) = p$ (en este caso Araki-Satoh-Semaev-Smart [SA98, Sem98b, Sem98a, Sma99b] mostraron como se puede imbuir el grupo de la curva elíptica en $\mathbb{Z}^*_p$).

En criptografía, lo importante es que exista un primo muy grande que divida a $\sharp\mathcal{C}\ (\mathbb{F}_q)$ el orden del grupo de la curva elíptica. Por ejemplo, estaría muy bien si el mismo $\sharp\mathcal{C}\ (\mathbb{F}_q)$ fuera primo, o igual al producto de un primo y un factor pequeño.

El método ρ de Pollard [Pol75] es el mejor conocido para para resolver el PLDCE. Requiere más o menos $\sqrt{\ell}$ operaciones, donde ℓ es el factor primo grande de $\sharp\mathcal{C}(\mathbb{F}_q)$.

2.6.3 Estándares

A la fecha se utilizan criptosistemas de curvas elípticas de manera comercial, `http:\\www.certicom.com` aquí se puede ver los productos que existen actualmente. Aquí también se puede ver los estandáres para su implementación.

IEEE P1363 - CCE está incluida en **IEEE P**1363, aprobado en Febrero del 2000. Este documento incluye una cobertura comprensible de CCE, DL y RSA. Los investigadores de Certicom fueron los primeros autores de la parte de CCE de este estándard.

ANSI X9F - CCE está incorporada a ANSI, en versiones preliminares de la ASC (Accredited Standards Committee) y de X9F (Servicios Financieros). Otros estándares ANSI hacen referencia a algoritmos y esquemas contenidos en estas 2 versiones. Instituciones Financieras, Gobierno y Cías. de Tecnología envían representantes a los cuerpos de Estándares ANSI X9. Este es un foro crítico para que un algoritmo pueda ser considerado como aceptable para uso en la liberación de servicios financieros. Certicom es autor de estos estándares.

ANSI X9.62 - ECDSA (Eliptic Curve Digital Signature Algorithm) es un análogo de el NIST DSA (Digital Signature Algorithm) que usa curvas elípticas. X9.62 cuenta con los requerimientos de seguridad estrictos usuales en la industria de servicios financieros. Publicado en Enero de 1999.

ANSI X9.63 - Este es un estándar para acuerdo y manejo de Claves con Curvas Elípticas, aceptado en el año 2000.

FIPS (Federal Information Processing Standard) 186-2. NIST del gobierno de los EU anunció en Febrero del 2000 la extensión del DSS (Digital Signature Standard) para incluir el ECDSA como se específica en **ANSI X9.62.** El estándar revisado es **FIPS 186-2**. Este estándar es un hito histórico en la aceptación comercial de ECC dado que agencias gubernamentales pueden comprar ahora productos de seguridad que contengan ECC sin tener que obtener una aprobación especial. NIST incluye también especificaciones para ECC en MISPC (Minimum Interoperability Specification).

ISO/IEC - ECC esta incorporada en varios versiones preliminares de ISO/IEC

- ISO/IEC 14888: Firma Digital con Apéndice Parte 3: Mecanismos Basados en Certificados
- ISO/IEC 9796-4: Firma Digital con Recuperación de Mensajes, Mecanismos Basados en Logaritmos Discretos
- ISO/IEC 14946: Técnicas Criptográficas basadas en Curvas Elípticas

Estándares de mercado vertical

Los estándares de Mercado Vertical o de Aplicación, se refieren usualmente a estándares establecidos sobre temas centrales de criptografía. Existes varias iniciativas para desarrollar protocolos que usen certificados de clave pública y otros tipos de sistemas con manejo de claves públicas. La mayoría de estos protocolos son independientes del algoritmo, esto permite la implementación de algoritmos de clave pública comunes; como es el caso de ECC; en ambientes donde otros tipos de sistemas de clave pública serían imprácticos, especialmente cuando el tamaño de la clave crece.

ATM Asynchronous Transport Mode — este es un foro.

WAP Wireless Application Protocol — V 1.0 (liberado en Mayo de 1998) proporciona acceso seguro a Internet, a servicios avanzados para teléfonos celulares digitales y a terminales inalámbricas. ECC está incorporada en la capa de seguridad WAP a través de WTLS (Wireless Transport Layer Security).

ANSI X12 y UN/EDIFACT - ECC está incorporada en el Estándar EDI (Electronic Data Interchange). Nada previene el uso de ECC para EDI, nuevos elementos de información simplemente necesitan ser definidos.

FSTC Financial Services Technology Consortium — especificación de chequeo electrónico.

OTP 0.9 Open Trading Protocol — esquema para el encapsulamiento de protocolos de pago.

IETF - SSL/TLS, IPSEC, PKIX, S/MIME CDPD (Cellular Digital Packet Data) — ECC para el acuerdo de claves.

ReFLEX - Estándar de pago de 2 vias de Motorola.

DH *Diffie-Hellman* Técnica de acuerdo DH, Intercambio de claves Diffie-Hellman es el trabajo del Dr. Whitfield Diffie & Martin Hellman de la U. de Stanford en 1976. Esta fue la 1^{er} técnica criptográfica de clave pública divulgada (publicada). Se usa principlamente para el intercambio público de claves para el uso de algún otro criptosistema del tipo de clave privada. Su seguridad se basa en la dificultad de calcular *log*'s en aritmética modular.

$\mathscr{E}$ y $\mathscr{R}$ van a establecer una clave

$$\mathscr{E} \longrightarrow\ g \bmod (m), \quad h_1 = g\, e_1 \bmod (m)\ \longrightarrow \mathscr{R},$$

con e_1 número grande.

$$\mathscr{R} \longrightarrow\ h_2 = g\, e_2 \bmod (m)\ \longrightarrow \mathscr{E}.$$

Ahora cada uno de ellos usa el número

$$k = g\,(e_1 \cdot e_2) = h_1\, e_2 = h_2\, e_1 \bmod (m)$$

como clave privada.

Cualquier enemigo debe calcular e_1 a partir de $g,\ m,\ h_1$ o e_2 de $g,\ m,\ h_2$. Se sabe que esto es muy difícil para valores suficientemente grandes de m y g.

El Gamal Crypto Public basado en el mismo problema que DH. Para usarlo, el receptor publica $g,\ m$ y h_1 y el emisor selecciona un exponente aleatorio e_2 y envía h_2 junto con el mensaje cifrado usando la clave privada y la clave k. Aquí no es fácil firmar mensajes. También depende de que A y B seleccionen e_1 y e_2 de manera especial.

3

Cómputo Cuántico

§3.1 Preliminares

La observación de que varias definiciones diferentes, aparentemente, de lo que significa que una función sea calculable condujo al mismo conjunto de funciones calculables a la proposición de la Tesis de Church

todos los dispositivos calculables pueden ser simulados por una máquina de Turing

la cual simplifica en gran parte el estudio de la computación reduciendo el campo de estudio de dispositivos de cálculo con potencial infinito a máquinas de Turing.

Tesis Cuantitativa de Church

Cualquier dispositivo físico (que pueda ser construido y hecho funcionar) de cálculo puede ser simulado por una máquina de Turing en un número polinomial de pasos en la entrada (espacio, tiempo, precisión, etc.) usada por el dispositivo de cálculo.

Si se hace que la precisión de una computadora cuántica crezca polinomialmente en el tamaño de la entrada (tal que el número de bits de precisión crezca logarítmicamente en el tamaño de la entrada) parecería que se obtiene una computadora más poderosa. Permitiendo el mismo crecimiento polinomial en la precisión no parece conferir poder de cálculo extra a la mecánica clásica.

Los científicos de la computación han llegado a convencerse de la verdad de la tesis cuantitativa de Church a través de la falla de todos los contraejemplos propuestos. La mayoría de estos contraejemplos propuestos han sido basados en las leyes de la mecánica clásica; de cualquier manera, el universo es en realidad mecánico cuántico. Parece plausible que el poder de cómputo de la mecánica clásica corresponda al de las máquinas de Turing, mientras que el poder natural de cómputo de la mecánica cuántica podría ser mayor.

Benioff demostró [Ben80, Ben82a, Ben82b] que la evolución reversible unitaria era suficiente para darse cuenta del poder de cómputo de una máquina de Turing, mostrando así que la mecánica cuántica es computacionalmente al menos tan poderosa como las computadoras clásicas. Feynman [Fey82, Fey84, Fey86]

sugirió que la mecánica cuántica podría ser computacionalmente más poderosa que las máquinas de Turing. Para explotar esto, Deutsch [Deu85] definió en 1985 y 1989 las máquinas cuánticas de Turing y los circuitos cuánticos.

A la fecha no se sabe cómo construir una computadora cuántica. Los mayores obstáculos parecen ser la decoherencia de la superposición cuántica a través de la interacción de la computadora con el medio ambiente (que pueden ser evadidos con correctores de error cuánticos), y la implementación de transformaciones de estado con suficiente precisión. Ambos obstáculos se hacen más difíciles a medida que el tamaño de la computadora crece.

Aún con la imposibilidad de construir computadoras cuánticas útiles, cualquier método general para simular la mecánica cuántica con a lo más lentitud polinomial llevaría a un algoritmo de tiempo polinomial para factorizar.

La clase de complejidad $\mathcal{BPP}$ (*Bounded Polynomial Probability*) es vista como la clase de problemas resolubles eficientemente, la cual requiere de la ayuda de generadores de números aleatorios y que permitan una probabilidad de error pequeña. Si sólo se les permite una probabilidad de error pequeña, las máquinas de Turing cuánticas y los arreglos de compuertas cuánticas pueden calcular las mismas funciones en tiempo polinomial. La clase de funciones calculables en tiempo polinomial cuántico con una pequeña probabilidad de error es llamada, por analogía con la clase clásica $\mathcal{BPP}$, como $\mathcal{BQP}$ (*Bounded error probability Quantum Polynomial time*).

3.1.1 Qubits

Una diferencia importante entre las computadoras clásicas y las cuánticas es el proceso de lectura. En el caso clásico la salida es una cadena de bits la cual se obtiene de manera determinística, es decir, la repetición del cálculo llevará a la misma salida otra vez. Esto es diferente para una computadora cuántica debido a la probabilidad estadística de la mecánica cuántica.

Los elementos de información básica en una computadora cuántica son los Bits Cuánticos o Qubits (del inglés Quantum Bits). Un qubit es vector unitario en un $\mathbb{C}$-espacio vectorial 2-dimensional (¡de Hilbert!) con una base $\{|0\rangle, |1\rangle\}$ fija. Ésta puede corresponder por ejemplo a la polarización de fotones o al giro hacia arriba y hacia abajo de un electrón.

A diferencia de los bits clásicos, los cuales toman el valor 0 ó 1, los qubits pueden estar en superposición, es decir, los qubits pueden estar en cada una de las posiciones de los vectores $|0\rangle$ y $|1\rangle$. Esto significa que el vector que describe el estado puro de un qubit puede ser cualquier combinacion lineal

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad \alpha,\ \beta, \in \mathbb{C} \tag{3.1}$$

donde $\alpha,\ \beta$ son tales que $|\alpha|^2 + |\beta|^2 = 1$.

Físicamente, los qubits corresponden a sistemas efectivos de 2 niveles como el estado base y el estado excitado de un átomo, el grado de libertad en la polarización de la luz o la orientación hacia arriba y hacia bajo de una partícula con spin $\frac{1}{2}$. Tal sistema físico puede estar en cualquier ESTADO PURO que puede ser representado por un vector normalizado de la forma anterior.

Si la salida de un cálculo es por ejemplo el vector de estado $|\psi\rangle$ en la ecuación (3.1), α y β no pueden ser determinados por una sola medida sobre un solo especimen. De hecho, $|\alpha|^2$ y $|\beta|^2$ son las probabilidades para que el sistema se encuentre en $|0\rangle$ y $|1\rangle$ respectivamente. De aquí que, los valores absolutos de estos coeficientes puedan ser determinados repitiendo el cálculo, midiendo en la base y después contando las frecuencias relativas. La salida de cada medida unitaria está por lo tanto completamente indeterminada.

De la misma manera un sistema de n qubits puede estar en una superposición de todos los estados posibles clásicos

$$|0,0,\ldots,0\rangle + |1,0,\ldots,0\rangle + \cdots + |1,1,\ldots,1\rangle. \tag{3.2}$$

La base $\{|0,0,\ldots,0\rangle, |1,0,\ldots,0\rangle, \ldots, |1,1,\ldots,1\rangle\}$ que corresponde a una palabra binaria de longitud n en un sistema cuántico de n qubits es llamada una BASE COMPUTACIONAL.

También, el estado de un sistema cuántico de n qubits puede ser medido en la base computacional, lo cual significa que la salida correspondiente a la misma palabra binaria ocurre con la probabilidad dada por el cuadrado de los valores absolutos de los coeficientes respectivos.

En física clásica, los estados posibles de un sistema de n partículas, cuyos estados individuales pueden ser descritos por un vector en un espacio vectorial 2-dimensional, forman un espacio vectorial de dimensión $2n$. Mientras que en la física clásica, una descripción completa del estado de este sistema requiere sólo n bits, en la física cuántica, una descripción completa de el estado de este sistema requiere $2^n - 1$ números complejos. En un sistema cuántico el espacio de estados resultante de un sistema de n qubits tiene dimensión 2^n. La dimensión del estado en la física clásica corresponde al producto cartesiano de los espacios vectoriales individuales para cada partícula

$$\dim(X \times Y) = \dim(X) + \dim(Y),$$

pero en un sistema cuántico corresponde al producto tensorial

$$\dim(X \otimes Y) = \dim(X) \times \dim(Y).$$

Por ejemplo, la base para un sistema de 3 qubits se denota como

$$\{|000\rangle,\ |001\rangle,\ |010\rangle,\ |011\rangle,\ |100\rangle,\ |101\rangle,\ |110\rangle,\ |111\rangle\}$$

donde $|001\rangle$ es la abreviación de $|0\rangle \otimes |0\rangle \otimes |1\rangle$.

3.1.2 Estados entrelazados

Un estado puro de un sistema cuántico compuesto que no es un producto con respecto a los elementos de la base es llamado un ESTADO PURO ENTRELAZADO.

El estado $|00\rangle + |11\rangle$ es un ejemplo de un estado cuántico que no puede ser descrito separadamente en términos del estado de cada uno de sus componentes (qubits). En otras palabras, no existen $\vec{a}_1, \vec{a}_2, \vec{b}_1, \vec{b}_2 \in \mathbb{C}$ tales que $(\vec{a}_1|0\rangle + \vec{b}_1|1\rangle) \otimes (\vec{a}_2|0\rangle + \vec{b}_2|1\rangle) = |00\rangle + |11\rangle$. Los estados que no pueden ser descompuestos de esta manera son llamados ESTADOS ENTRELAZADOS. Estos estados no tienen una contraparte clásica y proporcionan el crecimiento exponencial de los espacios de estado de acuerdo con el número de partículas. La medida proporciona otra manera de ver las partículas entrelazadas, las partículas no están entrelazadas si la medida de una no tiene efecto sobre otra.

3.1.3 Paralelismo Cuántico

Cualquier función arbitraria clásica

$$f\colon \mathbb{R}^m \longrightarrow \{0,1\}^k$$

puede ser implementada en una computadora cuántica. Se puede también construir un arreglo de compuertas cuánticas $\mathcal{U}_f$ definidas como una transformación lineal (¡unitaria!) de los estados para calcular $f(x)$ como

$$\mathcal{U}_f|x,0\rangle = |x, f(x)\rangle.$$

Si se aplica $\mathcal{U}_f$ a una superposición, entonces, por la linealidad de $\mathcal{U}_f$, esta se aplica a todos los vectores base en la superposición simultáneamente y se generará una superposición de resultados. De esta manera es posible calcular $f(x)$ para N valores de x en una sola aplicación de $\mathcal{U}_f$. Este efecto es llamado PARALELISMO CUÁNTICO. En otras palabras, el usar la superposición de la ecuación (3.2) como entrada para un algoritmo es equivalente de alguna manera a correr el cálculo sobre todos los estados posibles clásicos de entrada al mismo tiempo. Este es uno de los fundamentos del poder computacional de una computadora cuántica.

La estructura matemática tras la composición de un sistema cuántico es la del producto tensorial. De aquí que, vectores como $|\underbrace{0,0,\ldots,0}_{N}\rangle$ deben de ser entendidos como $\underbrace{|0\rangle \otimes \cdots \otimes |0\rangle}_{N} = |0\rangle^{\otimes N}$. Esto implica que la dimensión del estado que caracteriza al sistema crece exponencialmente con el número de qubits.

La naturaleza probabilística del proceso de lectura por una parte y la posibilidad de explotar el paralelismo cuántico por otro lado son aspectos que compiten al comparar el poder computacional de las computadoras cuánticas y las clásicas.

§3.2 Computadoras cuánticas

Nielsen y Chuang [NC97] probaron que las computadoras cuánticas no pueden ser arreglos universales de compuertas. Aún si el programa está dado en sí mismo en forma de un estado cuántico requeriría un registro de programa de longitud infinita para desarrollar una operación arbitraria (unitaria) en una cantidad finita de qubits –fue probado que la universalidad es posible solamente de manera probabilística–. En este sentido, las computadoras cuánticas no serán del tipo de dispositivos de todo propósito como son las clásicas. Sin embargo, cualquier conjunto finito de programas cuánticos puede correr en una computadora cuántica con un registro de programa finito.

Una computadora clásica es programable. Es un dispositivo capaz de realizar diferentes operaciones dependiendo del programa que le ha sido dado: procesamiento de palabras, transformaciones algebraicas, proyección de películas, etc. En palabras más abstractas una computadora clásica es un arreglo universal de compuertas, podemos programar cada función posible con $n-$bits de entrada y $n-$bits de salida especificando un programa de longitud $n2^n$. Esto es, un circuito fijo con $n(1+2^n)$ bits de entrada puede ser usado para calcular cualquier función de los primeros n bits en el registro.

Una computadora clásica digital opera sobre una cadena de bits de entrada y regresa una cadena de bits de salida. La función en medio puede ser descrita como un circuito lógico construido de muchas operaciones lógicas elementales. Esto es, el cálculo completo puede ser descompuesto en un arreglo de operaciones mas pequeñas –compuertas– actuando solo sobre 1 o 2 bits como las operaciones AND, OR y NOT. De hecho, estas 3 compuertas junto con la operación COPY (o FANOUT) forman un conjunto universal de compuertas en las cuales toda función bien definida entrada–salida puede ser descompuesta. La complejidad de un algoritmo es entonces esencialmente el número de compuertas elementales requeridas, respectivamente su crecimiento asintótico con el tamaño de la entrada.

El modelo del circuito para una computadora cuántica es de hecho muy parecido al modelo del circuito clásico, reemplazando la función de entrada–salida por una operación cuántica que mapee estados cuánticos en estados cuánticos. Es suficiente para ello considerar operaciones unitarias, lo cual significa que el cálculo tomado puede ser lógicamente reversible. En cambio, cualquier operación unitaria puede ser descompuesta en compuertas elementales que actúan solamente sobre 1 o 2 qubits. Un conjunto de compuertas elementales que permi-

ten la realización de cualquier operación unitaria a una aproximación arbitraria es también referido como universal. Un imporante ejemplo de un conjunto de compuertas universales es en este caso cualquier rotación de un qubit escogida al azar junto con la operación CNOT (Controlled NOT), la cual actúa como

$$|x, y\rangle \longmapsto |x, y \oplus x\rangle, \tag{3.3}$$

donde $\oplus$ significa adición (mod 2). Como en el caso clásico existen infinidad de compuertas universales. Notablemente, también cualquier compuerta genérica (es decir, escogida al azar) de 2 bits es en sí misma un conjunto universal (junto con la posibilidad de intercambio entre qubits), muy similar a lo que la compuerta NAND es para la computación clásica. Notablemente, cualquier circuito cuántico que hace uso de cierto conjunto de compuertas universales puede ser simulado por un circuito cuántico diferente basado en otro conjunto universal de compuertas.

Una compuerta de un solo bit particularmente usual es la de HADAMARD

$$\begin{aligned} |0\rangle \longmapsto H|0\rangle &= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \\ |1\rangle \longmapsto H|1\rangle &= \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \end{aligned} \tag{3.4}$$

por lo que H crea una superposición de estados, y cuando se aplica a N bits se llama de WALSH o de WALSH-HADAMARD y se puede definir recursivamente como

$$W_1 = H, \quad W_{N+1} = H \otimes W_N,$$

cuando se aplica a N bits actúa en cada uno de ellos para crear una superposición de 2^N estados.

Una COMPUERTA DE FASE no hace nada más que multiplicar uno de los vectores base por una fase

$$|0\rangle \longmapsto |0\rangle, \quad |1\rangle \longmapsto i|1\rangle, \tag{3.5}$$

y una COMPUERTA DE PAULI corresponde a una de las 3 MATRICES UNITARIAS DE PAULI[1]. Las compuertas CNOT, de Hadamard, de fase y de Pauli son las compuertas de mayor importancia para muchos algoritmos cuánticos, por lo que se pensaría que solo con estos (junto con medidas de los operadores de Pauli), se podrían construir poderosos algoritmos cuánticos que desarrollen el mejor algoritmo clásico para un problema. Sin embargo el TEOREMA DE GOTTESMAN-KNILL [NC00, AB06], establece que cualquier circuito cuántico consistente de estos operadores solo puede ser simulado eficientemente en una

[1] $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$

computadora clásica.

Una de las diferencias cruciales entre circuitos clásicos y cuánticos es que en el caso cuántico la operacion COPY no es posible. De hecho, la linealidad de la mecánica cuántica prohibe un dispositivo que pueda copiar un estado cuántico desconocido –esto es conocido como el Teorema de No Clonación–[2]. Una de sus mayores consecuencias es la posibilidad de aplicarlo en la criptografía cuántica.

El poder de los algoritmos cuánticos viene de que tienen como ventaja el paralelismo cuántico y el entrelazamiento. Así que la mayoría de los algoritmos cuánticos comienzan calculando una función de interés sobre una superposición de todos los valores como sigue.

Se aplica W_N a un estado de $N-$qubits $|00\ldots0\rangle$ para obtener

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

la cual debe de ser vista como la superposición de todos los enteros $0 \leq j < 2^N$. Luego se suma un registro de $k-$bits $|0\rangle$, entonces por linealidad

$$\begin{aligned}\mathcal{U}_f\left(\frac{1}{\sqrt{2^N}}\sum_{j=0}^{2^N-1}|j,0\rangle\right) &= \frac{1}{\sqrt{2^N}}\sum_{j=0}^{2^N-1}\mathcal{U}_f|j,0\rangle \\ &= \frac{1}{\sqrt{2^N}}\sum_{j=0}^{2^N-1}|j,f(j)\rangle\end{aligned}$$

cuando se mide el estado se obtiene $|j_0, f(j_0)\rangle$ para algún j_0 tomado aleatoriamente. La parte medular de cualquier algoritmo cuántico es su manera manipular el paralelismo cuántico, por lo que los resultados deseados serán medidos con alta probabilidad. Una manera es amplificar los valores de salida de interés, es decir transformar el estado de tal manera que los valores de interés tengan una amplitud mayor y por lo tanto tengan una probabilidad más grande de ser medidos. Otra manera es encontrar propiedades de todos los valores de $f(j)$. Esta idea es explotada en el algoritmo de Shor el cual usa la transformada cuántica de Fourier (TCF) para obtener el período de f, es decir, después de calcular la superposición de todos los valores, se calcula la TCF de la función, la cual como la transformada clásica de Fourier, pone toda la amplitud de la función en múltiplos del recíproco del período, el cual se usa para factorizar un entero n. La mayor complicación es que la TCF se basa en la transformada

[2]Si se conoce que un vector estado es ó $|0\rangle$ ó $|1\rangle$ entonces una máquina de clonación es perfectamente consistente con las reglas de la mecánica cuántica. De cualquier forma, el producir clones perfectos de un estado cuántico arbitrario como el dado por la ecuación (3.1) está prohibido, Wootters, Zurek y Dieks [WZ82].

rápida de Fourier (TRF) y da, por lo tanto, en la mayoría de los casos resultados aproximados solamente.

3.2.1 Criptografía cuántica

El objetivo es establecer claves secretas sobre canales no autentificados sin la necesidad de la criptogafía de clave pública. La mayoría de los algoritmos usan una codificación basada en 2 observables no conmutativos (es decir un enemigo no puede obtener los valores distintos de los 2 observables), por ejemplo polarización rectilínea $\{|-\!\!\rightarrow\rangle, |\uparrow\rangle\}$ y diagonal $\{|\nearrow\rangle, |\nwarrow\rangle\}$ de fotones. Si un fotón es polarizado como $|-\!\!\rightarrow\rangle$, la probabilidad de ser leído en la base $\{|\nearrow\rangle, |\nwarrow\rangle\}$ es $\frac{1}{2}$ para cada base.

El emisor envía fotones con una de las 4 polarizaciones (0 o 1 en forma rectilínea o diagonal), con una base elegida al azar. El receptor elige al azar el tipo de medición pero la mantiene en secreto. En seguida el receptor anuncia el tipo de medida y el emisor le dice al receptor cuales medidas fueron del tipo correcto. Un enemigo introduce error a esta transmisión debido a que él no conoce de antemano el tipo de polarización de cada fotón. Los 2 usuarios legales del canal cuántico prueban la intromisión a este revelando un subconjunto de los bits clave y checando en público la tasa de error.

Aún si las computadoras cuánticas llegan a ser realidad a gran escala, no se afectaría a esquemas teóricos de información como "one time pad". Ni aún todos los criptosistemas de clave pública son amenazados por el cómputo cuántico. Se ha discutido que podría haber una función de un solo sentido fuerte que pueda ser calculada eficientemente con computadoras clásicas y díficil de invertir aún con computadoras cuánticas. Esto es suficiente para lograr generación pseudo-aleatoria segura computacionalmente, algunos esquemas de acuerdo y protocolos de 0 conocimiento, todos $\boldsymbol{\mathcal{NP}}$. Información teórica segura (es decir incondicionalmente), de algunos acuerdos cuánticos es imposible.

§3.3 Algunos algoritmos criptográficos cuánticos

3.3.1 Transformada cuántica de Fourier

La transformada discreta de Fourier (TDF) de una función (con una muestra de tamaño L y período r) es una función concentrada cerca de los múltiplos de $\frac{L}{r}$. Si $r|L$ de manera par, el resultado es una función que tiene valores distintos de 0 sólo en múltiplos de $\frac{L}{r}$. De otra manera el resultado se aproxima a este comportamiento y el mayor de los valores de la TDF ocurrirá en los enteros cercanos a múltiplos de $\frac{L}{r}$.

La TRF es la TDF con $L = 2^N$ para algún N. La TCF es esencialmente la TRF adaptada para una computadora cuántica. Está definida como

$$\mathcal{U}_{TCF}|j\rangle = \frac{1}{\sqrt{2^N}} \sum_{k=0}^{2^N-1} \exp\left(\frac{2\pi ikj}{2^N}\right)|k\rangle.$$

La TCF opera sobre la amplitud del estado cuántico

$$\sum_{j=0}^{2^N-1} g(j)|j\rangle \longmapsto \sum_{k=0}^{2^N-1} G(k)|k\rangle$$

donde $G(k)$ es la TDF de $g(j)$. Si el estado es medido después de que TCF sea aplicada, la probabilidad que el resultado sea $|j\rangle$ es $|G(j)|^2$. Shor prueba que la TCF es eficientemente calculable usando solamente $\dfrac{N(N+1)}{2}$ compuertas.

Mientras que la manera intuitiva de calcular la TDF

$$\mathcal{U}_{TDF}(f(j)) = \frac{1}{\sqrt{2^N}} \sum_{k=0}^{2^N-1} G(k)\exp\left(\frac{2\pi ikj}{2^N}\right)$$

por multiplicación matricial toma $O\left(\left(2^N\right)^2\right)$ pasos, la TRF requiere $O\left(2^N \log(2^N)\right)$. La TCF es de hecho una generación cuántica directa de la TRF, la cual, de cualquier manera puede ser implementada usando solo $O\left(\left(\log 2^N\right)^2\right)$ operaciones elementales –una aceleración exponencial.

Si se caracteriza la base computacional de estados de qubits

$$|x\rangle = |x_0, \ldots, x_{N-1}\rangle.$$

por la representación binaria de números

$$x = \sum_{j=0}^{N-1} x_j 2^j$$

x denota un número natural o 0 y no una palabra binaria. Entonces para $n = 2^N$ la TCF actúa sobre un vector de estado general de N qubits como

$$\sum_{k=0}^{2^N-1} G(k)|k\rangle \longmapsto \sum_{j=0}^{2^N-1} \mathcal{U}_{TDF}(f(j))|j\rangle.$$

Esta transformación puede ser implementada usando la compuerta H y las Compuertas Condicionales de Fase P_d

$$|j,k\rangle \longmapsto |j,k\rangle \exp\frac{\delta_{j+k,2}\,\pi i}{2^d}$$

el cual rota la fase relativa condicionalmente un angulo de $\pi 2^{-d}$, donde d es la "distancia" entre los 2 qubits involucrados.

3.3.2 Factorización en primos y logaritmos discretos

En 1994 Peter Shor [Sho94] dió el primer problema computacional práctico que las computadoras cuánticas podían resolver más rápido que las clásicas y específicamente que afecta a la criptografía. Antes de su resultado nadie estaba seguro de cómo usar los efectos cuánticos para acelerar el cómputo, aunque existía la especulación meramente teórica, dada por Deutsch-Jozsa [Joz96], 3.3.5, de que el cómputo podía hacerse más eficiente; aunque no existe prueba de que la factorización no esté en $\mathcal{P}$ en el sentido clásico.

Factorización

El método de FACTORIZACIÓN DE EXPONENTES asume que se tiene un exponente $r > 0$ y $a \in \mathbb{Z}$ tal que $a^r = 1 \bmod (N)$. Con alta probabilidad el algoritmo tendrá éxito al calcular los factores $p,\ q$ de N si $p = \left(a^{\frac{r}{2}} + 1, N\right)$ y $q = \left(a^{\frac{r}{2}} - 1, N\right)$.

Dado N se selecciona un $a \in \{2,\ 3, \ldots,\ N-1\}$ aleatoriamente (si a es 0 o 1, el algoritmo de factorización de exponentes falla) y se considera la sucesión $1,\ a,\ a^2,\ a^3,\ \ldots \bmod (N)$. Si $a^r = 1 \bmod (N)$ esta sucesión se repetirá cada r términos, por lo que calculando el período r (o un múltiplo) de la función $f_N(x) = a^x \bmod (N)$ se obtiene un r tal que $a^r = 1 \bmod (N)$. Clásicamente calcular r es tan difícil como tratar de factorizar N. Las computadoras cuánticas pueden encontrar potencialmente r en un tiempo que crece sólo como una función cuadrática del número de dígitos de N.

Logaritmos discretos

Por simplicidad se presenta el caso suponiendo que se puede calcular una TDF (para evitar las aproximaciones debidas a la TRF). Sea a un generador de un grupo G de orden q. Se considera la función $f(j,k) = a^j b^k$, esta tiene 2 períodos independientes

$$\begin{aligned} f(j+q,k) &= a^q a^j b^k = f(j,k) \\ f(j+d,k-1) &= a^d a^j b^k b^{-1} = f(j,k) \end{aligned}$$

por lo que se puede calcular, con paralelismo cuántico, la función anterior para obtener

$$\frac{1}{q} \sum_{k=0}^{q-1} \sum_{j=0}^{q-1} |j,k,a^j b^k\rangle$$

así que al medir el último registro se obtiene un valor a^{j_0} lo cual implica que $j = (j_0 - dk) \bmod (q)$. El estado es entonces

$$\frac{1}{q} \sum_{k=0}^{q} |j_0 - dk, k\rangle.$$

Tomando la TDF se obtiene después de simplificar

$$\frac{1}{\sqrt{q}} \sum_{l=0}^{q-1} \exp\left(\frac{2\pi i j_0 l}{q}\right) |l, idl \bmod (q)\rangle.$$

De donde se infiere que de una medida de los estados se puede deducir que $d = il^{-1} \bmod (q)$ si $(l, q) = 1$.

3.3.3 Algoritmo de Shor

Factorización de $M = 21$ con el algoritmo de Shor.

1. *Aplicación de paralelismo cuántico:* Sea $a \in \mathbb{Z}$ arbitrario, con $(a, n) = 1$; de otra manera tenemos un factor de n. Sea m la solución a $N^2 \leq 2^m \leq 2N^2$. Esta elección se hace de tal manera que la aproximación para las funciones, cuyo período no es una potencia de 2, sea suficientemente buena para que el resto del algoritmo funcione. Se usa el paralelismo cuántico para calcular $f_N(j)$ para todos los enteros de 0 a $2^m - 1$. La función es por lo tanto codificada en el estado cuántico

$$\frac{1}{\sqrt{2^m}} \sum_{j=0}^{2^m-1} |j, f_N(j)\rangle.$$

   Si $a = 11$ entonces $m = 9$. Si se mide la segunda parte, la probabilidad para obtener una medida de 1 sería $\frac{43}{2^9} \approx \frac{1}{6}$, como $f_N(j)$ se repite cada 6 términos, y la aproximación viene del hecho de que $6 \not| 2^9$ por la TRF. Se infiere que para primos grandes el período de $f_N(j)$ es mucho mayor y la probabilidad de medir 1 decrecerá exponencialmente de manera inmediata.

2. *Medición del estado de $f_N(j)$:* Se miden los bits del estado que codifica a $f_N(j)$ (los últimos $\lceil \log_2 M \rceil$ qubits) obteniendo un valor aleatorio u. El valor u no es de interés en sí mismo; sólo es de interés el efecto que tiene medir sobre el conjunto de superposiciones. Esta medición proyecta el espacio de estados sobre el espacio compatible con el valor medido, así que el estado es después de ser medido

$$K \sum_{j=0}^{2^m-1} g(j)|j, u\rangle$$

   donde

$$g(j) = \begin{cases} 1 & \text{si } f_N(j) = u \\ \\ 0 & \text{en otro caso} \end{cases}.$$

   Los j que aparecen en la suma difieren por múltiplos del período, así que $g(j)$ es la función buscada, de hecho, $\frac{g(j)}{K}$ es la función de probabilidad de

masa de medir j después de medir u. En el ejemplo anterior con $a = 11$, si se mide el valor 2, se reduce el estado a

$$\frac{1}{\sqrt{85}}(|5,2\rangle + |11,2\rangle + |17,2\rangle + \ldots + |503,2\rangle + |509,2\rangle)$$

si se pudieran medir 2 elementos j y k en la suma anterior se tendría el período (o un múltiplo de este) dado que $11^j = 11^k \pmod{21}$, y así $r = k - j$. De cualquier manera las leyes de la física cuántica dicen que la segunda medida dará la misma respuesta que la primera.

3. *Aplicación TCF:* La parte $|u\rangle$ del estado no será usada, así que no se escribirá más. Se aplica la TCF al estado obtenido anteriormente

$$\mathcal{U}_{TCF}: \sum_{j=0}^{2^m-1} g(j)|j\rangle \longmapsto \sum_{k=0}^{2^m-1} G(k)|k\rangle$$

donde la mayoría de las amplitudes de $G(k)$ estaran cerca de $t\dfrac{2^m}{r}$ para algún $t \in \mathbb{Z}$.

4. *Extracción del período:* El hacer una observación del estado dará un resultado v. Shor demostró que con alta probabilidad $\left(\text{de al menos } \dfrac{1}{3r^2}\right)$, v está dentro de $\dfrac{1}{2}$ de algún $t\dfrac{2^m}{r}$, es decir

$$\left| v - t\frac{2^m}{r} \right| < \frac{1}{2}.$$

Dado que $2^m > N$ entonces con alta probabilidad $\dfrac{t}{r}$ está dentro de $\dfrac{1}{2N^2}$ de $\dfrac{v}{2^m}$, es decir

$$\left| \frac{v}{2^m} - \frac{t}{r} \right| < \frac{1}{2N^2}.$$

Más aún, 2 números racionales distintos $\dfrac{t}{r}$ y $\dfrac{t'}{r'}$ con $0 < r < N$ y $0 < r' < N$ están separados por más de $\dfrac{1}{N^2}$, es decir

$$\left| \frac{t}{r} - \frac{t'}{r'} \right| > \frac{1}{N^2}$$

así que se puede usar el algoritmo de fracciones continuas (eficiente) para encontrar un número racional $\dfrac{p}{q}$ tal que $0 < q < N$ (dentro de $\dfrac{1}{2N^2}$ de $\dfrac{v}{2^m}$, de manera optimista) y con alta probabilidad de que el número

racional obtenido sea $\frac{t}{r}$. Se toma el denominador q de la fracción obtenida como supuesto para el período, el cual funcionará cuando $(t, r) = 1$ (así q sería un factor del período, y no el período mismo). Shor muestra que por lo menos con probabilidad $\frac{\varphi(r)}{3r}$ (φ función totient o φ de Euler) se obtiene r. Se sabe que $\frac{\varphi(r)}{r} > \frac{\delta}{\log\log r}$ para alguna constante δ, por lo que repitiendo el experimento $O(\log\log r)$ se asegura una probabilidad alta de éxito. Se puede tratar heurísticamente con v's cercanos o probar múltiplos de q, para evitar repeticiones de las evaluaciones cuánticas.

5. *Un factor de N:* Dado que

$$\left(a^{\frac{r}{2}} - 1\right)\left(a^{\frac{r}{2}} + 1\right) = a^r - 1 = 0 \bmod (N),$$

los números $\left(a^{\frac{r}{2}} - 1, N\right)$ y $\left(a^{\frac{r}{2}} + 1, N\right)$ serán factores de N. Este procedimiento falla sólo si r es impar, o si $a^{\frac{r}{2}} = -1 \bmod (N)$, en tal caso el procedimiento obtiene los factores triviales 1 y N. La forma algorítmica es llamada Método de Factorización de Exponentes. Se escribe r como $2^k m$ con m impar para algún k. Sea $b_0 = a^m \bmod (N)$. Elevando al cuadrado sucesivamente b_0 para obtener b_1, b_2, ..., hasta lograr $1 \bmod (N)$. Si b_u es el último $b_i \neq 1 \bmod (N)$, se calcula $(b_u - 1, N)$ para obtener un factor de N.

6. *Repetición del algoritmo:* Algo pudo haber salido mal, así que el proceso no da un factor de M: el valor de v no está dentro de $\frac{1}{2}$ de $\frac{2^m}{r}$; el período r y el multiplicador t pueden haber tenido un factor común o el método de factorización de exponentes falla. De cualquier manera es muy probable que, en unos cuantos intentos, se encuentre una factorización de N.

3.3.4 Algoritmo de Deutsch

Con este algoritmo es posible decidir si una función tiene cierta propiedad con una sola llamada, en vez de 2 que son necesarias clásicamente. Sea

$$f\colon \{0,1\} \longrightarrow \{0,1\}$$

una función. Esta función puede ser Constante o Balanceada, lo que significa que $f(0) \oplus f(1) = 0$ ó $f(0) \oplus f(1) = 1$, respectivamente. El problema es decidirlo con el mínimo número de llamadas, equivalentemente, cual es el valor de $f(0) \oplus f(1)$. En términos coloquiales, el problema en consideración puede ser descrito como una manera de probar si una moneda es fraudulenta o genuina.

Una forma de calcular la función f en una computadora cuántica es transformar el vector estado de 2 qubits de acuerdo con

$$|j, k\rangle \longmapsto \mathcal{U}_f|j, k\rangle = |j, f(j) \oplus k\rangle. \tag{3.6}$$

De esta manera, la evaluación puede ser realizada de manera unitaria. La función anterior es a lo que se le conoce como un ORÁCULO CUÁNTICO ESTÁNDAR (como un opuesto a un ORÁCULO CUÁNTICO MÍNIMO, el cual es de la forma

$$|j\rangle \ \longmapsto |f(j)\rangle).$$

La afirmación ahora es que usando tal oráculo, una sola llamada a la función es suficiente para la evaluación de $f(0) \oplus f(1)$. Para mostrar esto, se asume que se han preparado 2 qubits en el estado con el vector de estado

$$|\Psi\rangle = (H \otimes H)|0,1\rangle.$$

Ahora se aplica $\mathcal{U}_f$ a éste y luego H al primer qubit para obtener el vector estado

$$|\Psi'\rangle = (H \otimes \mathbf{1})\mathcal{U}_f(H \otimes H)|0,1\rangle.$$

Un pequeño cálculo muestra que $|\Psi'\rangle$ puede ser evaluado como

$$|\Psi'\rangle = \pm|f(0) \oplus f(1)\rangle H|1\rangle.$$

El segundo qubit está en el estado correspondiente al vector $H|1\rangle$, el cual no es de relevancia para el problema. El estado del primer qubit, de cualquier manera, es totalmente remarcable, codificado es $|f(0) \oplus f(1)\rangle$ y ambas alternativas son decidibles con probabilidad 1 en una medida y la base computacional, dado que los 2 vectores son ortogonales. Esto es, con una sola medida del estado, y notablemente, con una sola llamada a la función f, del primer qubit se puede decidir si f es constante o balanceada.

3.3.5 Algoritmo de Deutsch-Jozsa

El algoritmo de Deutsch no tiene todavía ninguna implicación acerca de la superioridad de una computadora cuántica sobre una clásica. Después de todo, es solamente una llamada a la función en vez de 2. La situación es diferente en el caso de la extensión del algoritmo de Deutsch conocida como algoritmo de Deutsch-Jozsa. Aquí la tarea es otra vez encontrar si una función es constante o balanceada, pero f es ahora una función

$$f\colon \{0,1\}^N \ \longrightarrow \ \{0,1\},$$

donde $N \in \mathbb{N}$. Se promete que la función es ya sea constante, lo que ahora significa que $f(i) = 0$ ó balanceada $f(i) = 1$ para toda $i = 0, \ldots, \ 2^N - 1$. Se dice que es balanceada si la imagen bajo f toma tantas veces el valor 1 como el valor 0. La propiedad de ser balanceada o constante se puede decir que es una propiedad global de muchos valores de la función. Es una propiedad prometida de la función, pues el algoritmo de Deutsch-Jozsa es un algoritmo de promesa. Existen solamente 2 cajas negras posibles a disposición, y la tarea es encontrar cual es usada.

En una computadora clásica el escenario del peor caso es que después de 2^{N-1} llamadas a la función, la respuesta ha sido siempre 0 o siempre 1. De aquí que, $2^{N-1}+1$ llamadas a la función son requeridas para saber con certeza si la función es balanceada o constante (un resultado que puede ser significantemente mejorado si se permiten algoritmos probabilísticos). En mecánica cuántica, otra vez, una llamada a la función es suficiente. Similamente a la situacion anterior, uno puede preparar $N+1$ qubits en el estado con el vector de estado

$$|\Psi\rangle = H^{\otimes(N+1)}|0,\ldots,0,1\rangle,$$

y aplicarle $\mathcal{U}_f$ como en la ecuación (3.6), actuando como un oráculo, y aplicar $H^{\otimes N}\otimes\mathbf{1}$ al estado resultante para obtener

$$|\Psi'\rangle = (H^{\otimes N}\otimes\mathbf{1})\mathcal{U}_f H^{\otimes(N+1)}|0,\ldots,0,1\rangle.$$

En el último paso, se desarrolla una medida sobre los primeros N qubits en la base computacional. En efecto, se observa que exactamente si la función f es constante, se obtiene la medida de salida correspondiente a $|0,0,\ldots,0\rangle$ con certidumbre. Para cualquier otra salida la función será balanceada. Una vez más, la prueba para la propiedad prometida puede ser llevada a cabo con una sola consulta, en vez de las $2^{N-1}+1$ clásicas.

3.3.6 Algoritmo de Bernstein-Vazirani

Existe un número de problemas relacionados que presentan características muy similares. En el algoritmo de Bernstein-Vazirani una vez más se da la función

$$f\colon \{0,1\}^N \longrightarrow \{0,1\}$$

de la forma

$$f(j) = aj$$

para $a,\ j \in \{0,1\}^N$ y para algún $N\in\mathbb{N}$. aj denota el producto escalar estándar $aj = a_0j_0 + \ldots + a_{2^N-1}j_{2^N-1}$. ¿Cuántas medidas se requieren para encontrar el vector a? Clásicamente, se deben desarrollar medidas para todos los argumentos posibles, y al final resolver sistemas de ecuaciones lineales. Con el oráculo estándar $|j,k\rangle \longmapsto |j,f(j)\oplus k\rangle$ a la mano, en su versión del algoritmo de Bernstein-Vazirani solamente una simple llamada al oráculo es requerida.

3.3.7 Algoritmo de Simon

El problema de Simon es un caso de un problema de oráculo el cual es clásicamente difícil, aún para algoritmos probabilísticos, pero tratable para computadoras cuánticas. El reto es encontrar el período p de la función

$$f : \{0,1\}^N \longrightarrow \{0,1\}^N,$$

la cual se presume ser 2 a 1 con $f(j) = f(k)$ si, y sólo si, $k = j \oplus p$. Aquí j y k denotan palabras binarias de longitud N, y $\oplus$ es la suma bit a bit (mod 2). El problema también puede ser establecido como un problema de decisión y el objetivo sería entonces decidir si existe un periodo o no, es decir si f es 2 a 1 o 1 a 1.

Clásicamente el problema es difícil, dado que la probabilidad de haber encontrado 2 elementos idénticos j y k después de $2^{\frac{N}{4}}$ consultas sigue siendo menor que $2^{\frac{-N}{2}}$. La solución cuántica de Simon. Se comienza con el vector de estado $(H|0\rangle)^{\otimes N}|0\rangle^{\otimes N}$ y se corre el oráculo una vez, produciéndose el vector de estado

$$\frac{1}{\sqrt{2^N}} \sum_{j=0}^{2^N-1} |j\rangle|f(j)\rangle.$$

Luego se mide el segundo registro. Si la medida obtenida es $f(j_0)$, entonces el vector de estado del primer registro será

$$\frac{1}{\sqrt{2}}(|j_0\rangle + |j_0 \oplus p\rangle).$$

Aplicando H a cada uno de los N qubits restantes

$$\frac{1}{\sqrt{2^{N+1}}} \sum_{k=0}^{2^{N+1}-1} ((-1)^{j_0 \cdot k} + (-1)^{(j_0 \oplus p)\cdot k})|k\rangle = \frac{1}{\sqrt{2^{N-1}}} \sum_{p \cdot k = 0}^{2^{N-1}-1} (-1)^{j_0 \cdot k}|k\rangle.$$

Si finalmente se mide el primer registro en la base computacional, se obtiene un valor k el cual es tal que $k \cdot p = 0$ (mod 2). Repitiendo este proceso para obtener $N-1$ vectores l. i. $k_1, \ldots, k_{N-1}$ se puede determinar p del conjunto de ecuaciones $k_i \cdot p = 0$. Para este fin se tiene que consultar el oráculo $O(N)$ veces. De aquí se obtiene una aceleración exponencial comparada con cualquier algoritmo clásico. Y en contraste con el algoritmo de Deutsch-Jozsa, este salto exponencial se mantiene en algoritmos probabilísticos clásicos. Tanto el algoritmo de Simon como el algoritmo de Shor tratan de encontrar el período de una función, el de Simon en $(\mathbb{Z}_2)^N$ y el de Shor en $\mathbb{Z}_{2^N}$, y ambos generan una aceleración exponencial y hacen uso de algoritmos clásicos en un proceso posterior a cada paso.

3.3.8 Algoritmo de Grover

Si se tiene una lista no ordenada y se quiere conocer un elemento con ciertas propiedades o saber si tal elemento existe y/o en caso de existir saber el número de tales elementos –todos estos son problemas comunes o subrutinas necesarias para programas más complejos y debido al algoritmo de Grover todos estos problemas en principio admiten una aceleración cuadrática comparada con soluciones clásicas. Tal mejora en la operación puede no parecer muy espectacular, sin embargo, los problemas a los cuales es aplicable son numerosos, el problema del agente de viajes, el ciclo de Hamilton y ciertos problemas sobre coloración, y

el progreso de la transformada de Fourier ordinaria a la transformada rápida de Fourier demostró ya cómo una aceleración cuadrática en una rutina elemental puede tener muchas aplicaciones.

Considere el problema de buscar un elemento marcado $j_0 \in \{1, \ldots, 2^N\}$ dentro de una base de datos no ordenada. En el caso clásico se tiene que consultar la base de datos $O(2^N)$ veces para identificar el elemento buscado, el algoritmo de Grover requiere solamente $O(\sqrt{2^N})$ intentos. Si la base de datos esta representada por un operador unitario

$$\mathcal{U}_{j_0} = \mathbf{1} - 2|j_0\rangle\langle j_0|,$$

donde $\mathcal{U}_{j_0}|j\rangle = (-1)^{\delta_{j,j_0}}|j\rangle$, el cual regresa el signo de $|j_0\rangle$ pero preserva todos los vectores ortogonales a $|j_0\rangle$. El primer paso del algoritmo es preparar una superposición igualmente pesada de todos los estados base

$$|\psi\rangle = \frac{1}{\sqrt{2^N}} \sum_{j=0}^{2^N-1} |j\rangle.$$

Esto se puede lograr aplicando W_{2^N} al vector estado $|0\rangle$. En seguida, se aplica el operador de Grover

$$\mathcal{G} = \mathcal{U}_\psi \mathcal{U}_{j_0}, \quad \mathcal{U}_\psi = 2|\psi\rangle\langle\psi| - 1$$

a $|\psi\rangle$. Geométricamente, la acción de $\mathcal{G}$ es rotar $|\psi\rangle$ hacia $|j_0\rangle$ un ángulo 2φ donde $\operatorname{sen}\varphi = |\langle\psi|j_0\rangle| = \dfrac{1}{\sqrt{2^N}}$. Ahora la idea es iterar esta rotación k veces hasta que el estado inicial esté cercano a $|j_0\rangle$, es decir,

$$\mathcal{G}^k|\psi\rangle \approx |j_0\rangle.$$

Al medir el sistema (en la base computacional) se obtendrá entonces el valor de j_0 con alta probabilidad.

Ahora la pregunta es: ¿Cuántas iteraciones se necesitan? cada paso es una rotación de 2φ y el ángulo inicial entre $|\psi\rangle$ y $|j_0\rangle$ es $\dfrac{\pi}{2} - \varphi$. Usando esto para 2^N grande $\operatorname{sen}\varphi \approx \varphi$ se ve que $k \approx \pi\dfrac{\sqrt{2^N}}{4}$ rotaciones harán el trabajo y la probabilidad de obtener una medida de salida distinta de j_0 decrecerá como $O\left(\dfrac{1}{2^N}\right)$. Dado que cada paso en la iteración de Grover consulta a la base de datos una vez, se necesitan solamente $O\left(\sqrt{2^N}\right)$ intentos comparados con $O(2^N)$ en algoritmos clásicos. Para explotar esta aceleración es necesaria una implementación eficiente no solamente del oráculo de la base de datos $\mathcal{U}_{j_0}$, sino también del operador unitario $\mathcal{U}_\psi$. Afortunadamente, este último puede ser construido en

$O(\log(2^N))$ compuertas elementales.

¿Qué pasa si hay más de un elemento marcado, digamos M? usando la superposición igualmente pesada de todos los estados respectivos en vez de $|j_0\rangle$ se puede repetir esencialmente el argumento anterior y obtener que $O\left(\sqrt{\frac{2^N}{M}}\right)$ consultas son requeridas para encontrar uno de los M elementos con alta probabilidad. De cualquier manera, desarrollar más iteraciones de Grover sería ir más allá de la marca: se rotaría el estado inicial más allá del objetivo buscado y la probabilidad de encontrar un elemento marcado decrecería otra vez rápidamente. Si inicialmente no se conoce el número M de elementos marcados, esto es de cualquier forma un problema no serio. Tanto como $M \ll 2^N$ se puede seguir ganando una aceleración cuadrática simplemente escogiendo de manera aleatoria el número de iteraciones entre 0 y $\pi\sqrt{\frac{2^N}{4}}$. La probabilidad de encontrar un elemento marcado será entonces cercana a $\frac{1}{2}$ para cada M. Notablemente, el algoritmo de Grover es óptimo en el sentido de cualquier algoritmo cuántico para este problema requerirá necesariamente $O\left(\sqrt{\frac{2^N}{M}}\right)$ consultas.

3.3.9 Comentarios sobre el algoritmo de Shor

El problema del algoritmo de Shor trata acerca de la factorización, un típico problema $\boldsymbol{\mathcal{NP}}$. En pocas palabras la idea del algoritmo de factorización de Shor es como sigue.

i) *Parte clásica.* Usando algo de teoría de números elemental se puede probar que el problema de encontrar un factor de un entero dado es esencialmente equivalente a determinar el período de cierta función.

ii) *TCF para encontrar el período.* Se implementa la función del paso *i*) en un circuito cuántico y se aplica a la superposición de todos los estados de entrada clásicos. Luego se desarrolla una transformada cuántica discreta de Fourier (TCDF) y se mide la salida. La medida de salida estará probabilísticamente distribuida de acuerdo con la inversa del período buscado. Este último puede ser determinado así (con cierta probabilidad), repitiendo el procedimiento.

iii) *Implementación eficiente.* El punto crucial del algoritmo es que la TCF también como la función del paso *i*) puedan ser implementadas eficientemente, *i. e.*, el número de operaciones elementales requeridas crece solo polinomialmente con el tamaño de la entrada. Más aún, la probabilidad de éxito del algoritmo puede hacerse arbitrariamente cercana a 1 sin que el esfuerzo crezca exponencialmente.

Claramente, la parte medular del algoritmo es una implementación eficiente de la TCF.

Parte clásica

Sea N impar y se quiere factorizar un $a < N$ tal que $(N, a) = 1$. Esto último se puede checar eficientemente con el algoritmo de Euclides en un tiempo de $O((\log N)^3)$. Un factor de N puede ser encontrado entonces indirectamente determinando el período p de la función $f : \mathbb{Z} \longrightarrow \mathbb{Z}_N$ definida como

$$f(x) = a^x \bmod N. \tag{3.7}$$

Se busca una solución a $a^p - 1 = 0 \bmod N$. Si p es par podemos descomponer

$$a^p - 1 = (a^{\frac{p}{2}} + 1)(a^{\frac{p}{2}} - 1) = 0 \bmod N,$$

y por lo tanto uno o ambos términos $(a^{\frac{p}{2}} \pm 1)$ deben tener un factor común con N. Cualquier divisor común no trivial de N con $(a^{\frac{p}{2}} \pm 1)$, otra vez calculado con el algoritmo de Euclides, da un factor no trivial de N.

Obviamente, el procedimiento descrito es exitoso solamente si p es par y el factor final es uno no trivial. Afortunadamente, si escogemos a aleatoriamente, este caso ocurre con probabilidad mayor que $\frac{1}{2}$ a menos que N sea una potencia de un primo.[3]

Esto último puede ser checado eficientemente por un algoritmo clásico conocido, el cual regresa el valor del primo. Además un algoritmo de tiempo polinomial para determinar el período de la función en la ecuación (3.7) lleva a un algoritmo probabilístico de tiempo polinomial el cual regresa ya sea un factor de N o nos dice si N es primo.

Juntando las piezas

¿Cómo se usa la TCF para calcular eficientemente el período p de la función en la ecuación (3.7)? Considere 2 registros de qubits cada uno, donde $2^q = n \geq N^2$ y todos los qubits están en el vector estado $|0\rangle$ inicialmente. Aplicando H a cada qubit en el primer registro se tiene $\frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} |j, 0\rangle$. Si se ha implementado la función en la ecuación (3.7) en un circuito cuántico el cual actúa como $|j, 0\rangle \longmapsto |j, f(j)\rangle$, donde $j \in \mathbb{Z}_n$. Aplicando esto al vector estado y desarrollando una TCF sobre el primer registro se obtiene

$$\frac{1}{n} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} \exp\left(\frac{2\pi i j k}{n}\right) |k, f(j)\rangle.$$

[3] Se puede mostrar que la probabilidad para esto es mayor que $\frac{1}{\log N}$. Por lo que la probabilidad total de éxito es por lo menos $\frac{1}{2 \log N}$.

¿Cómo se verá la distribución de las medidas de salida si se mide ahora el primer registro en la base computacional? A *groso modo* la suma sobre j llevará a una interferencia constructiva siempre que $\frac{k}{n}$ esté cerca de un múltiplo del inverso del período p de f y esto da interferencia destructiva de otro modo. De aquí que, la distribución de probabilidad para medir k claramente alrededor de múltiplos de $\frac{n}{p}$ y p misma puede ser determinada repitiendo el procedimiento completo $O(\log N)$ veces. Al mismo tiempo la probabilidad de éxito se puede hacer arbitrariamente cercana a 1. Al final se puede de cualquier modo verificar fácilmente si el resultado, el factor obtenido de N es válido o no.

Lo que resta es probar si la función

$$|j,0\rangle \longmapsto |j,f(j)\rangle, \quad f(j) = a^j \pmod N$$

puede ser implementada eficientemente. Esto se puede hacer elevando al cuadrado repetidamente para obtener a^{2^j} y luego multiplicando un subconjunto de estos números de acuerdo con la expansión binaria de j. Esto requiere $O(\log N)$ veces elevar al cuadrado y multiplicaciones de números de $\log N$ bits. Para cada multiplicación el algoritmo elemental de escuela requiere $O((\log N)^2)$ pasos. De aquí que, implementando este algoritmo clásico simple en nuestra computadora cuántica podemos calcular $f(j)$ con $O((\log N)^3)$ operaciones elementales. De hecho, esta parte de desarrollar un algoritmo clásico estándar de multiplicación en una computadora cuántica es el cuello de botella en la parte cuántica del algoritmo de Shor. Si hubiera un algoritmo más refinado de exponenciación modular cuántica se mejoraría el desarrollo asintótico del algoritmo, de hecho la exponenciación modular se puede hacer en un tiempo de $O((\log N)^2 \log\log N \log\log\log N)$ utilizando el algoritmo de Schönhagen-Strassen [Zal98] para la multiplicación. De cualquier manera, este es un algoritmo clásico, primero hecho reversible y luego corrido en una computadora cuántica. Si existe un algoritmo cuántico más rápido sería posible romper codigos RSA en una computadora cuántica aún más rápido que el cifrado en una computadora clásica.

En conjunto, la parte cuántica del algoritmo de factorización de Shor requiere del orden de $((\log N)^3)$ pasos elementales, *i. e.*, el tamaño del circuito es cúbico en la longitud de la entrada. Como se describió anteriormente, es necesario de un preprocesamiento y un posprocesamiento clásico adicional para obtener un factor de N. El tiempo requerido para esta parte clásica del algoritmo es, de cualquier forma, polinomial en $\log N$ también, de tal suerte que el algoritmo completo realiza el trabajo en tiempo polinomial. En contraste a esto, el tiempo de corrida de la criba de campos númericos, el cual es actualmente el mejor algoritmo clásico de factorización, es $\exp[O((\log N)^{\frac{1}{3}}(\log\log N)^{\frac{2}{3}})]$. Más aún, se cree ampliamente que la factorización es un problema clásico difícil, en el sentido de que no existe un algoritmo clásico de tiempo polinomial. Sin embargo, se cree también que probar esta última conjetura (si es verdadera) es extremadamente

difícil dado que ello resolvería el problema $\boldsymbol{\mathcal{P}} \stackrel{?}{=} \boldsymbol{\mathcal{NP}}$.

§3.4 Ruptura de RSA

Lo que aquí se busca es una manera de calcular $P \bmod (N)$ dados P^e, e y N, esto es un método de encontrar las raices e-ésimas en el grupo multiplicativo de los enteros módulo N (este grupo es denotado por $\mathbb{Z}_N^*$ y contiene los enteros coprimos a N). Sigue siendo una pregunta abierta si una solución a este problema necesariamente da un algoritmo aleatorio de tiempo polinomial para factorizar. De cualquier manera la factorización da un algoritmo de tiempo polinomial para encontrar raices e-ésimas para cualquier e primo relativo a $\phi(N)$ y así romper RSA. Conociendo la factorización en primos de N, digamos $\prod_{i=1}^{k} p_i^{a_i}$, se puede fácilmente calcular $\phi(N) = N \prod_{i=1}^{n} \left(1 - \frac{1}{p_i}\right)$. Y entonces calcular d tal que $ed \equiv 1 \bmod (\phi(N))$, lo cual implica que $P^{ed} \equiv P \bmod (N)$.

De cualquier manera, para romper un caso particular de RSA, es suficiente encontrar un entero d tal que $ed \equiv 1 \bmod (\mathrm{ord}(P))$, esto es $ed = \mathrm{ord}(P)k + 1$ para algún $k \in \mathbb{Z}$. Se tendría entonces $C^d \equiv P^{ed} \equiv P^{\mathrm{ord}(P)k+1} \equiv P \bmod (N)$.

Dado que $(e,\ \phi(N)) = 1$ es fácil ver que $\mathrm{ord}(P) = \mathrm{ord}(P^e) = \mathrm{ord}(C)$. Así, dado $C = P^e$ se puede calcular $\mathrm{ord}(P)$ usando el algoritmo de Shor y luego calculando d que satisfaga $de \equiv 1 \bmod (\mathrm{ord}(P))$ usando el algoritmo extendido de Euclides. Por lo que son necesarias varias repeticiones del algoritmo de Shor para encontrar el orden de a para varios a aleatorios; solo encontrar el orden de C y resolver para P sin importar que esto permita o no factorizar N.

4

Retículos

Los retículos son objetos geométricos que han sido usados para resolver varios problemas matemáticos y computacionales. En esta parte se considerarán desde el punto de vista algorítmico motivado por su aplicación a la criptología; cabe hacer una observación acerca del término retículo. Existen 2 estructuras matemáticas comúnmente llamadas retículos, una de ellas tiene que ver con conjuntos parcialmente ordenados y la otra se puede describir "pictóricamente" como los puntos de intersección de una rejilla regular $n-$dimensional, no necesariamente ortogonal. En esta tésis se considera la segunda, además de que se consideran conocidos los términos "norma," "producto escalar," "$\mathbb{K}$-espacio vectorial" así como sus propiedades básicas.

§4.1 Conceptos básicos

El conjunto de vectores enteros $\mathbb{Z}^n$ es cerrado bajo la adición y bajo la multiplicación por un entero. Sin embargo $\mathbb{Z}^n$ no es un espacio vectorial pues $\mathbb{Z}$ no es un campo, así $\mathbb{Z}^n$ es el ejemplo más simple de la estructura que a continuación se define.

Definición 25 *Dado un conjunto* $\{\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n\} = \boldsymbol{\mathcal{B}} \subset \mathbb{R}^m$; $\mathbb{R}^m$ *espacio euclideano* $m-$*dimensional y* $n \leq m$; *de vectores l. i., se define el* RETÍCULO $\Lambda(\boldsymbol{\mathcal{B}})$ *generado por* $\boldsymbol{\mathcal{B}}$ *como*

$$\begin{aligned} \Lambda(\boldsymbol{\mathcal{B}}) &= \left\{ \sum_{i=1}^{n} r_i \vec{b}_i \;\middle|\; r_i \in \mathbb{Z}, \quad 1 \leq i \leq n \right\} \\ &= \sum_{i=1}^{n} \mathbb{Z}\vec{b}_i. \end{aligned}$$

Se llama a $\boldsymbol{\mathcal{B}}$ *una* BASE, *a* m DIMENSIÓN *y a* n RANGO *del retículo* $\Lambda(\boldsymbol{\mathcal{B}})$.

La notación $\Lambda(\boldsymbol{\mathcal{B}})$ tiene sentido aún cuando los vectores en $\boldsymbol{\mathcal{B}}$ no sean l. i.

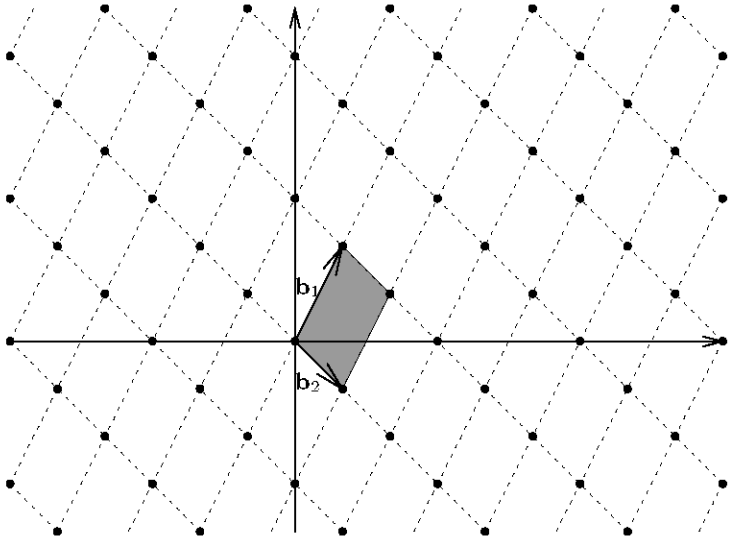
Definición 26 *Sea* $\emptyset \neq \Lambda \subset \mathbb{R}^m$, *de dice que* Λ *es un retículo si*

i) es cerrado bajo la sustracción: si $\vec{x}$, $\vec{y} \in \Lambda$, *entonces* $\vec{x} - \vec{y} \in \Lambda$

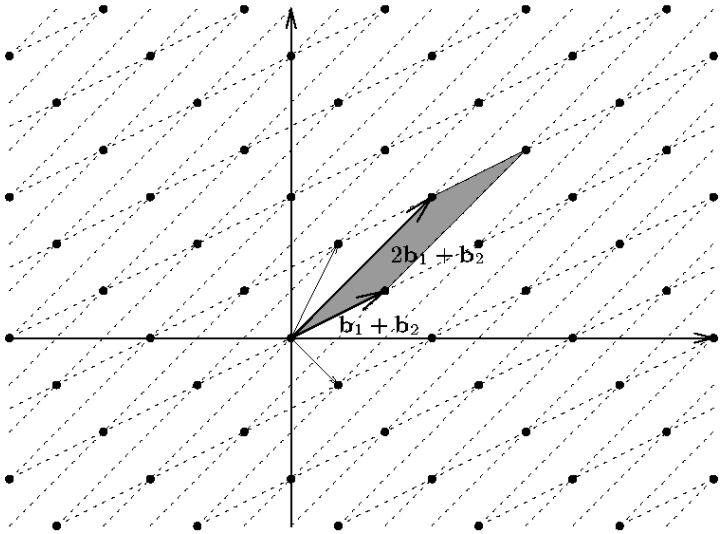
ii) es discreto: existe $\lambda \in \mathbb{R}^+$ *tal que* $\lambda \leq \|\vec{\boldsymbol{x}} - \vec{\boldsymbol{y}}\| \ \forall \ \vec{\boldsymbol{x}}, \ \vec{\boldsymbol{y}} \in \Lambda$.

Observe que $\vec{\boldsymbol{x}} - \vec{\boldsymbol{x}} = \vec{\mathbf{0}} \in \Lambda$ siempre, que es cerrado bajo complementos, si $\vec{\boldsymbol{x}} \in \Lambda$, entonces $-\vec{\boldsymbol{x}} = \vec{\mathbf{0}} - \vec{\boldsymbol{x}} \in \Lambda$; y bajo la adición, si $\vec{\boldsymbol{x}}, \ \vec{\boldsymbol{y}} \in \Lambda$, entonces $\vec{\boldsymbol{x}} + \vec{\boldsymbol{y}} \in \Lambda$. En otras palabras Λ es un SUBGRUPO ADITIVO de $\mathbb{R}^m$.

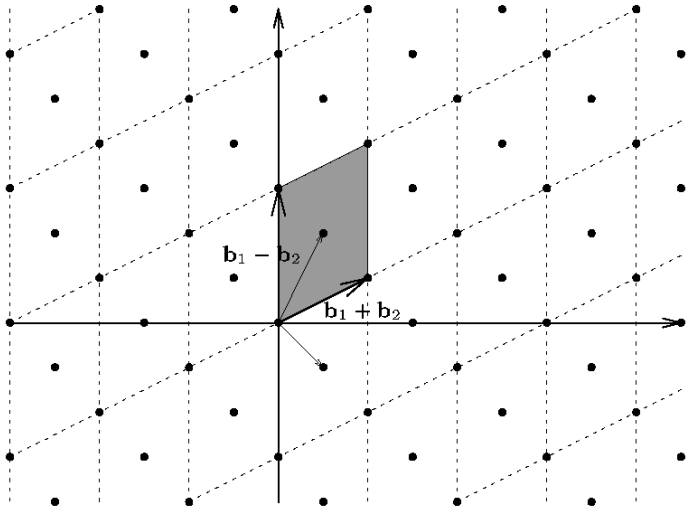
Ejemplo 2 *Sean* $\vec{b}_1 = (1, 2), \ \vec{b}_2 = (1, -1)$ *, entonces*



este retículo también es generado por $\vec{b}_1 + \vec{b}_2 = (2, 1)$ *y* $2\vec{b}_1 + \vec{b}_2 = (3, 3)$



sin embargo $\vec{\boldsymbol{b}}_1 + \vec{\boldsymbol{b}}_2$ *y* $\vec{\boldsymbol{b}}_1 - \vec{\boldsymbol{b}}_2$ *no generan al retículo completo.*



Ortogonalización de Gram-Schmidt

Sean $\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n \in \mathbb{R}^m$ l. i. Los vectores $\vec{b}_i^\star$ $(1 \leq i \leq n)$ y los números reales μ_{ij} $(1 \leq j \leq i \leq n)$ se definen inductivamente por

$$\begin{aligned}
\vec{b}_1^\star &= \vec{b}_1 \\
\vec{b}_2^\star &= \vec{b}_2 - \mu_{2\,1}\vec{b}_1^\star, \quad \text{donde} \quad \mu_{2\,1} = \frac{\langle \vec{b}_2, \vec{b}_1^\star \rangle}{\langle \vec{b}_1^\star, \vec{b}_1^\star \rangle}, \\
&\vdots \\
\vec{b}_i^\star &= \vec{b}_i - \sum_{j=1}^{i-1} \mu_{ij}\vec{b}_j^\star, \quad \text{donde} \qquad (4.1) \\
\mu_{ij} &= \frac{\langle \vec{b}_i, \vec{b}_j^\star \rangle}{\langle \vec{b}_j^\star, \vec{b}_j^\star \rangle}, \qquad (4.2)
\end{aligned}$$

donde $\langle \cdot, \cdot \rangle$ es el producto interno usual en $\mathbb{R}^m$, $\vec{b}_i^\star$ es la proyección de $\vec{b}_i$ sobre el complemento ortogonal de $\sum_{j=1}^{i-1} \mathbb{R}\vec{b}_j$ y $\sum_{j=1}^{i-1} \mathbb{R}\vec{b}_j = \sum_{j=1}^{i-1} \mathbb{R}\vec{b}_j^\star$ para $1 \leq i \leq n$. Se sigue que $\boldsymbol{\mathcal{B}}^\star$ es una base ortogonal de $\mathbb{R}^n$ y de $\mathcal{L}(\boldsymbol{\mathcal{B}}^\star)$ donde $\boldsymbol{\mathcal{B}}^\star = \left\{ \vec{b}_1^\star, \vec{b}_2^\star, \ldots, \vec{b}_n^\star \right\}$.

Ejemplo 3 *El conjunto de vectores unitarios*

$$\boldsymbol{\mathcal{B}} = \left\{ \vec{e}_i \in \mathbb{R}^m \;\middle|\; \vec{e}_i = (\underbrace{0, \ldots, 0}_{i-1}, 1, \underbrace{0, \ldots, 0}_{m-i}),\ 1 \leq i \leq n \right\}$$

forma una base para $\mathbb{Z}^n$.

Definición 27 *Se dice que una matriz* $\boldsymbol{A} \in \mathcal{M}_{n\times n}(\mathbb{K})$ *es* No Singular *si*

$$\det(\boldsymbol{A}) \neq 0.$$

La base $\boldsymbol{\mathcal{B}}$ de un retículo en $\mathbb{R}^m$ se puede representar como una matriz $\boldsymbol{B} \in \mathcal{M}_{m\times n}(\mathbb{R})$ no singular cuyas columnas son los vectores base

$$\boldsymbol{B} = \left[\vec{b}_1 \middle| \vec{b}_2 \middle| \ldots \middle| \vec{b}_n \right] \in \mathbb{R}^{m\times n}.$$

Se llama a $\boldsymbol{B}$ la Matriz Asociada a $\boldsymbol{\mathcal{B}}$.

De esta manera se tiene

$$\Lambda(\boldsymbol{\mathcal{B}}) = \left\{ \boldsymbol{B}\vec{b} \;\middle|\; \vec{b} \in \mathbb{Z}^n \right\}.$$

Esto difiere del espacio vectorial generado por $\boldsymbol{\mathcal{B}}$

$$\begin{aligned}\mathcal{L}(\boldsymbol{\mathcal{B}}) &= \left\{ \sum_{i=1}^{n} k_i \vec{\boldsymbol{b}}_i \;\middle|\; k_i \in \mathbb{R},\ \ 1 \leq i \leq n \right\} \\ &= \sum_{i=1}^{n} \mathbb{R}\vec{\boldsymbol{b}}_i \\ &= \left\{ \boldsymbol{B}\vec{\boldsymbol{b}} \;\middle|\; \vec{\boldsymbol{b}} \in \mathbb{R}^n \right\}.\end{aligned}$$

Observe que en general $n \leq m$. Si $m = n$ se dice que el retículo es de Rango Completo o de Dimensión Total.

Proposición 3 *$\Lambda(\boldsymbol{\mathcal{B}})$ es de rango total si, y sólo si $\{\boldsymbol{B}\vec{\boldsymbol{x}} \mid \vec{\boldsymbol{x}} \in \mathbb{R}^n\} = \mathbb{R}^m$.*

Toda base $\boldsymbol{\mathcal{B}}$ de $\Lambda(\boldsymbol{\mathcal{B}})$ es base de $\mathcal{L}(\boldsymbol{\mathcal{B}})$ pero el recíproco no se cumple en general. Por ejemplo si aplicamos Gram-Schmidt a $\boldsymbol{B}$ para obtener $\boldsymbol{B}^\star$, con $\boldsymbol{B}$ matriz asociada a $\boldsymbol{\mathcal{B}}$ base de $\Lambda(\boldsymbol{\mathcal{B}})$, entonces $\Lambda(\boldsymbol{\mathcal{B}}^\star)$ no necesariamente es un retículo, pues puede suceder que

$$\Lambda(\boldsymbol{\mathcal{B}}) \neq \Lambda(\boldsymbol{\mathcal{B}}^\star),$$

aunque

$$\mathcal{L}(\boldsymbol{\mathcal{B}}) = \mathcal{L}(\boldsymbol{\mathcal{B}}^\star)$$

¡siempre!

Ejemplo 4 *Sean $m = 1$ y $\boldsymbol{\mathcal{B}} = \{1, \sqrt{2}\}$, entonces $\Lambda(\boldsymbol{\mathcal{B}}) = \{i + j\sqrt{2} \mid i,\ j \in \mathbb{Z}\}$. Sin embargo, debido a la irracionalidad de $\sqrt{2}$ no existe k tal que*

$$\Lambda(\boldsymbol{\mathcal{B}}) = \Lambda([k]) = k\mathbb{Z}.$$

Se sigue de esto que cuando $\boldsymbol{\mathcal{B}} \subset \mathbb{Q}^n$ siempre se puede encontrar un conjunto l. i. que genere al mismo retículo Λ.

Teorema 2 *Para toda $\boldsymbol{B} \in \mathcal{M}_{m\times n}(\mathbb{Q})$, matriz asociada a $\boldsymbol{\mathcal{B}}$, $\Lambda(\boldsymbol{\mathcal{B}})$ es un retículo.*

La demostración de esta proposición se basa en el hecho de que existe un algoritmo que tiene como entrada una matriz racional $\boldsymbol{B}$ para calcular una matriz $\boldsymbol{B}'$ con la propiedad de que las columnas no cero de $\boldsymbol{B}'$ son l. i. y tal que $\Lambda(\boldsymbol{\mathcal{B}}) = \Lambda(\boldsymbol{\mathcal{B}}')$.

Definición 28 *Dada una $\boldsymbol{B} \in \mathcal{M}_{m\times n}(\mathbb{Z})$, se llama* Operación Elemental *(*Columna*) a*

i) Intercambiar 2 columnas

ii) Multiplicar una columna por una unidad

iii) Sumar un múltiplo entero de una columna a otra columna

$$\vec{b}_i \longleftarrow \vec{b}_i + a\vec{b}_j \quad con \quad i \neq j \quad y \quad a \in \mathbb{Z}.$$

Si $\boldsymbol{B}'$ es el resultado de aplicar una secuencia de operaciones elementales a $\boldsymbol{B}$, entonces

$$\Lambda(\mathcal{B}) = \Lambda(\mathcal{B}').$$

Para probar que $\Lambda(\mathcal{B}) = \Lambda(\mathcal{B}')$ en general, es suficiente probar que $\mathcal{B} \subseteq \Lambda(\mathcal{B}')$ y que $\boldsymbol{B}' \subseteq \Lambda(\mathcal{B})$.

Definición 29 *Sea $\boldsymbol{B} \in \mathcal{M}_{n\times n}(\mathbb{R})$ una matriz no singular, digamos*

$$\boldsymbol{B} = \left[\vec{b}_1 \middle| \vec{b}_2 \middle| \ldots \middle| \vec{b}_n\right].$$

Se dice que $\boldsymbol{B}$ está en la Forma Normal de Hermite *(FNH) si*

i) $\boldsymbol{B}$ es triangular inferior ($b_{i\,j} \neq 0$ implica que $i \geq j$).

ii) Para todo $i > j$, $0 \leq b_{i\,j} < b_{i\,i}$ (los elementos de la diagonal son los más grandes en la fila y todas las entradas son no negativas).

Definición 30 *Sea $\boldsymbol{B} \in \mathcal{M}_{m\times n}(\mathbb{R})$ una matriz no singular, digamos*

$$\boldsymbol{B} = \left[\vec{b}_1 \middle| \vec{b}_2 \middle| \ldots \middle| \vec{b}_n\right].$$

Se dice que $\boldsymbol{B}$ está en la Forma Normal de Hermite *(FNH) si*

i) Existen $1 \leq i_1 < \ldots < i_n$ tales que $b_{i\,j} \neq 0$ implica que $i \geq i_j$ (Estrictamente decreciente en la altura de la columna).

ii) Para todo $k > j$, $0 \leq b_{i_j\,k} < b_{i_j\,j}$ (el elemento no cero de más arriba de cada columna es el más grande en la fila y todas las entradas son no negativas).

El índice i_j anterior se define como la fila de elemento no 0 de más arriba de la columna j. Debido a que estos términos son estrictamente crecientes, cada columna contiene filas que ninguna de las últimas columnas tiene. Por lo que se puede garantizar que las columnas no 0 de una matriz FNH son l. i.

Proposición 4 *La matriz $\boldsymbol{A}$ FNH de una matriz $\boldsymbol{A}$ es única, es decir si dos matrices $\boldsymbol{A}$ FNH y $\boldsymbol{B}$ FNH generan el mismo retículo*

$$\Lambda(\mathcal{A}) = \Lambda(\mathcal{B}),$$

entonces $\boldsymbol{A} = \boldsymbol{B}$.

Cálculo de FNH

El siguiente algoritmo calcula la FNH de una matriz en $\mathbb{Z}^{m\times n}$. Se desconoce su tiempo de corrida pero se sabe que termina

FNH en $\mathbb{Z}^{m\times n}$
Entrada: $\boldsymbol{B}$
Salida: $\boldsymbol{B}$ FNH
Paso 1: Inicialización: $fila = 1$, $col = 1$,
Paso 2: Repite
Paso 3: Multiplica las columnas para hacer a todos los elementos $b[fila, col], \ldots, b[fila, n]$ positivos
Paso 4: para $k = col + 1$ hasta n haz
Paso 5: Columna-MCD($\boldsymbol{B}, fila, k, col$)
Paso 6: si $b[fila, col] = 0$
Paso 7: $fila = fila + 1$
Paso 8: en otro caso
Paso 9: para $k = 1$ hasta $col - 1$ haz
Paso 10: $$c = \left\lfloor \frac{b[fila, k]}{b[fila, col]} \right\rfloor$$
Paso 11: $b[\cdot, k] = b[\cdot, k] - cb[\cdot, col]$
Paso 12: $fila = fila + 1$
Paso 13: $col = col + 1$
Paso 14: hasta que $fila > m$ o $col > n$

Columna-MCD
Entrada: $\boldsymbol{B}, fila, k, col,$
Paso 1: mientras $[fila, k] \neq 0$ haz
Paso 2: $$c = \left\lfloor \frac{b[fila, k]}{b[fila, col]} \right\rfloor$$
Paso 3: $b[\cdot, k] = b[\cdot, k] - cb[\cdot, col]$
Paso 4: intercambia $(b[\cdot, k], b[\cdot, col])$
Paso 5: ciclo

Este último algoritmo calcula el mcd del elemento $col-$ésimo en una fila y el $k-$ésimo elemento solo aplicando operaciones elementales. Una vez que el algoritmo termina el $col-$ésimo elemento es el mcd y el $k-$ésimo es 0. Esto hace que todos los elementos anteriores al $col-$ésimo en esa columna sean 0. Como $fila$ siempre crece y col nunca decrece esto satisface la primera condición de FNH. La segunda se satisface en el segundo ciclo del algoritmo FNH. No se garantiza que este algoritmo corra en tiempo polinomial, pues las entradas de la matriz pueden crecer exponencialmente.

FNH en $\mathbb{Q}^{m\times n}$

Para calcular FNH de una matriz $\boldsymbol{B} \in \mathbb{Q}^{m\times n}$ se multiplica por el mcd l de los denominadores de cada una de sus entradas para obtener $l\boldsymbol{B} = \boldsymbol{C} \in \mathbb{Z}^{m\times n}$,

digamos. Ahora se aplica el algoritmo FNH a $\boldsymbol{C}$ para obtener $\boldsymbol{C}'$. Así $\frac{1}{l}\boldsymbol{C}'$ es $\boldsymbol{B}$ FNH.

DEMOSTRACIÓN:
(Teorema 2)

Ahora se sabe que dada $\boldsymbol{B} \in \mathbb{Q}^{m\times n}$, se puede encontrar $\boldsymbol{B}'$ tal que $\Lambda(\boldsymbol{\mathcal{B}}) = \Lambda(\boldsymbol{\mathcal{B}}')$ con $\boldsymbol{B}'$ FNH. Como se observó, los vectores no 0 en $\boldsymbol{B}'$ son l. i., luego entonces $\Lambda(\boldsymbol{\mathcal{B}}) = \Lambda(\boldsymbol{\mathcal{B}}')$ es un retículo.

Definición 31 *Sea $\Lambda(\boldsymbol{\mathcal{B}})$ un retículo de dimensión completa. Se define el* DETERMINANTE DE $\Lambda(\boldsymbol{\mathcal{B}})$ *como*

$$\begin{aligned} \mathrm{d}(\Lambda) &= \det(\Lambda(\boldsymbol{\mathcal{B}})) \\ &= \det(\boldsymbol{B}). \end{aligned}$$

Este es un número real que no depende de la elección de la base.

Teorema 3 *Si $\boldsymbol{\mathcal{B}}$ y $\boldsymbol{\mathcal{B}'}$ son 2 bases para el mismo retículo $\Lambda(\boldsymbol{\mathcal{B}}) = \Lambda(\boldsymbol{\mathcal{B}'})$, entonces* $\det(\boldsymbol{B}) = \pm\det(\boldsymbol{B}')$.

DEMOSTRACIÓN:
Como $\Lambda(\boldsymbol{\mathcal{B}}) = \Lambda(\boldsymbol{\mathcal{B}'})$, se tiene que $\boldsymbol{B} = \boldsymbol{B}'\boldsymbol{U}$ y $\boldsymbol{B}' = \boldsymbol{B}\boldsymbol{V}$ con $\boldsymbol{U},\ \boldsymbol{V} \in \mathbb{Z}^{n\times n}$.

Entonces $\det(\boldsymbol{B}) = \det(\boldsymbol{B}'\boldsymbol{U}) = \det(\boldsymbol{B}\boldsymbol{V}\boldsymbol{U}) = \det(\boldsymbol{B})\det(\boldsymbol{V}\boldsymbol{U})$. Pero $\det(\boldsymbol{V}\boldsymbol{U}) \in \mathbb{Z}$ de donde se sigue que $1 = \det(\boldsymbol{V}\boldsymbol{U}) = \det(\boldsymbol{V})\det(\boldsymbol{U})$, luego $\det(\boldsymbol{V}) = \det(\boldsymbol{U}) = \pm 1$. Por lo tanto

$$\det(\boldsymbol{B}) = \pm\det(\boldsymbol{B}').$$

Dado un conjunto arbitrario de vectores $\boldsymbol{\mathcal{B}}$ ¿es $\Lambda(\boldsymbol{\mathcal{B}})$ siempre un retículo? es decir, ¿existe un conjunto más pequeño $\boldsymbol{\mathcal{B}'}$ de vectores l. i tal que $\Lambda(\boldsymbol{\mathcal{B}}) = \Lambda(\boldsymbol{\mathcal{B}'})$?

Definición 32 *Dado $\Lambda(\boldsymbol{\mathcal{B}})$, se define el* PARALELEPÍPEDO FUNDAMENTAL *asociado a $\boldsymbol{\mathcal{B}}$ como*

$$\mathcal{P}(\boldsymbol{\mathcal{B}}) = \left\{ \sum_{i=1}^{n} r_i \vec{\boldsymbol{b}}_i \;\middle|\; r_i \in [0,1) \right\}.$$

$\mathcal{P}(\boldsymbol{\mathcal{B}})$ es semiabierto, por lo que los transladados $\mathcal{P}(\boldsymbol{\mathcal{B}}) + \vec{\boldsymbol{v}},\ \vec{\boldsymbol{v}} \in \Lambda(\boldsymbol{\mathcal{B}})$ forman una partición de $\mathbb{R}^m$.

Para todo $\vec{\boldsymbol{x}} \in \mathbb{R}^m$ existe un único $\vec{\boldsymbol{v}} \in \Lambda(\boldsymbol{\mathcal{B}})$ tal que $\vec{\boldsymbol{x}} \in (\vec{\boldsymbol{v}} + \mathcal{P}(\boldsymbol{\mathcal{B}}))$.

Definición 33 *Dado $\Lambda(\boldsymbol{\mathcal{B}})$, se define el* DETERMINANTE DE $\Lambda(\boldsymbol{\mathcal{B}})$ *como el volumen del paralelepipedo fundamental $n-$dimensional asociado a $\boldsymbol{\mathcal{B}}$, es decir*

$$\mathrm{d}(\Lambda) = \mathrm{Vol}(\mathcal{P}(\boldsymbol{\mathcal{B}})).$$

Cuando $\boldsymbol{B} \in \mathcal{M}_{n\times n}(\mathbb{R})$ es no singular, entonces $\mathrm{d}(\Lambda) = |\det(\boldsymbol{B})|$ y para un retículo que no sea de dimensión total se tiene que el volumen del paralelepipedo fundamental no depende de la elección de la base

$$\mathrm{Vol}(\mathcal{P}(\boldsymbol{\mathcal{B}})) = \mathrm{Vol}(\mathcal{P}(\boldsymbol{\mathcal{B}'}))$$

Proposición 5 *Dado* $\boldsymbol{\mathcal{B}} = \{\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{b}}_2, \ldots, \vec{\boldsymbol{b}}_n\} \in \Lambda$ *l. i., entonces* $\boldsymbol{\mathcal{B}}$ *es una base para* Λ *si, y sólo si,*

$$\mathcal{P}(\boldsymbol{\mathcal{B}}) = \{\vec{\boldsymbol{0}}\}.$$

Proposición 6 *Dos bases* $\boldsymbol{\mathcal{B}} \subset \mathbb{R}^{m\times n}$ *y* $\boldsymbol{\mathcal{B}'} \subset \mathbb{R}^{m\times k}$ *generan al mismo retículo* Λ *si, y sólo si,* $n = k$ *y existe* $\boldsymbol{U} \in \mathbb{Z}^{n\times n}$ *matriz unimodular tal que* $\boldsymbol{B'} = \boldsymbol{BU}$.

De lo anterior se infiere que la dimensión de un retículo no depende de la elección de la base.

Proposición 7 *Sea* $\Lambda(\boldsymbol{\mathcal{B}})$ *un retículo, entonces*

$$\mathrm{d}(\Lambda) = \prod_{i=1}^{n} \left\| \vec{\boldsymbol{b}}_i^{\star} \right\|$$

y

$$\mathrm{d}(\Lambda) = \sqrt{\left|\det\left(\boldsymbol{B}^T\boldsymbol{B}\right)\right|}$$

donde $\boldsymbol{B}^T\boldsymbol{B}$ *es la matriz de Gram definida por*

$$\boldsymbol{B}^T\boldsymbol{B} = \begin{pmatrix} \left\langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{b}}_1 \right\rangle & \cdots & \left\langle \vec{\boldsymbol{b}}_1, \vec{\boldsymbol{b}}_n \right\rangle \\ \vdots & \ddots & \vdots \\ \left\langle \vec{\boldsymbol{b}}_n, \vec{\boldsymbol{b}}_1 \right\rangle & \cdots & \left\langle \vec{\boldsymbol{b}}_n, \vec{\boldsymbol{b}}_n \right\rangle \end{pmatrix}.$$

Proposición 8 *Sea* $\boldsymbol{B} \in \mathbb{Z}^{n\times n}$ *no singular, entonces*

$$|\det(\boldsymbol{B})|\mathbb{Z}^n \subseteq \Lambda(\boldsymbol{\mathcal{B}}).$$

La demostración de la proposición anterior puede consultarse en [MG02]. También, a manera de comentario, existen algoritmos de tiempo polinomial para encontrar la FNH de retículos de dimensión total y de dimensión no total. Para esto usamos el hecho de que

$$\Lambda(\boldsymbol{\mathcal{B}}) = \Lambda([\boldsymbol{\mathcal{B}}||\det(\boldsymbol{B})|\boldsymbol{\mathcal{I}}])$$

y se encuentra $[\boldsymbol{\mathcal{B}}||\det(\boldsymbol{B})|\boldsymbol{\mathcal{I}}]$ FNH. De hecho se aplican operaciones elementales de columna a la parte derecha de la matriz, para mantener acotadas las entradas en la parte izquierda.

Proposición 9 (Desigualdad de Hadamard)

$$\mathrm{Vol}(\mathcal{P}(\boldsymbol{\mathcal{B}})) \leq \prod_{i=1}^{n} \left\| \vec{\boldsymbol{b}}_i \right\|$$

de donde

$$|\det(\boldsymbol{B})| \leq \prod_{i=1}^{n} \left\| \vec{\boldsymbol{b}}_i \right\|.$$

§4.2 Problemas del vector más corto y más cercano

El problema del vector más corto (PVMC) no tiene solución polinomial, de hecho es $\boldsymbol{\mathcal{NP}}-$difícil.

Definición 34 *Sea $\|\cdot\|$ una norma. El* Problema del Vector más Corto *(PVMC), con respecto a la norma $\|\cdot\|$, es, dado un retículo $\Lambda(\boldsymbol{\mathcal{B}})$, encontrar un vector $\vec{\boldsymbol{v}}_0 \in \Lambda(\boldsymbol{\mathcal{B}}) \setminus \{\vec{\mathbf{0}}\}$ de norma mínima. En otras palabras, dado $\Lambda(\boldsymbol{\mathcal{B}})$, encontrar $\vec{\boldsymbol{v}}_0 \in \Lambda(\boldsymbol{\mathcal{B}}) \setminus \{\vec{\mathbf{0}}\}$ tal que $\|\vec{\boldsymbol{v}}_0\| \leq \|\vec{\boldsymbol{w}}\|$ para cualquier $\vec{\boldsymbol{w}} \in \Lambda(\boldsymbol{\mathcal{B}}) \setminus \{\vec{\mathbf{0}}, \vec{\boldsymbol{v}}_0\}$.*

La solución al PVMC depende de la norma; por ejemplo si

$$\Lambda(\{\vec{\boldsymbol{e}}_1 + \vec{\boldsymbol{e}}_2, 2\vec{\boldsymbol{e}}_2\}) = \{\vec{\boldsymbol{v}} \in \mathbb{Z}^2 \mid v_1 + v_2 \equiv 0 \pmod 2,\ \vec{\boldsymbol{v}} = (v_1, v_2)\},$$

entonces $2\vec{\boldsymbol{e}}_2 = (0,2)^T$ es un vector más corto con respecto a la norma ℓ_1, pero no con respecto a la norma ℓ_2 o ℓ_∞ porque en estas normas el vector $\vec{\boldsymbol{e}}_1 + \vec{\boldsymbol{e}}_2 = (1,1)^T$ es estrictamente más corto.

¿Está bien definido el PVMC?, es decir, ¿siempre existe un vector en $\Lambda(\boldsymbol{\mathcal{B}})$ cuya norma sea mínima?

Considere las longitudes de todos los vectores no cero en un retículo $\Lambda(\boldsymbol{\mathcal{B}})$ y defina

$$\lambda = \inf\{\|\vec{\boldsymbol{v}}\| \mid \vec{\boldsymbol{v}} \in \Lambda(\boldsymbol{\mathcal{B}}) \setminus \{\vec{\mathbf{0}}\}\}$$

se afirma que siempre existe un vector $\vec{\boldsymbol{v}} \in \Lambda(\boldsymbol{\mathcal{B}}) \setminus \{\vec{\mathbf{0}}\}$ tal que

$$\lambda = \|\vec{\boldsymbol{v}}\|.$$

Proposición 10 *Dado $\Lambda(\boldsymbol{\mathcal{B}})$ se tiene*

$$\lambda = \inf\left\{ \|\vec{\boldsymbol{v}}\| \;\middle|\; \vec{\boldsymbol{v}} \in \Lambda(\boldsymbol{\mathcal{B}}) \setminus \{\vec{\mathbf{0}}\} \right\} \geq \min\left\{ \left\| \vec{\boldsymbol{b}}_i^{\star} \right\| \;\middle|\; \vec{\boldsymbol{b}}_i^{\star} \in \boldsymbol{\mathcal{B}}^{\star} \right\}.$$

Corolario 1 *Sean $\vec{\boldsymbol{v}},\ \vec{\boldsymbol{w}} \in \Lambda(\boldsymbol{\mathcal{B}})$ tales que*

$$\|\vec{\boldsymbol{v}} - \vec{\boldsymbol{w}}\| < \min\left\{ \left\| \vec{\boldsymbol{b}}_i^{\star} \right\| \;\middle|\; \vec{\boldsymbol{b}}_i^{\star} \in \boldsymbol{\mathcal{B}}^{\star} \right\},$$

entonces $\vec{\boldsymbol{v}} = \vec{\boldsymbol{w}}$.

Con esto se puede probar que siempre existe $\vec{\boldsymbol{v}}_0 \in \Lambda(\boldsymbol{\mathcal{B}})$ tal que $\|\vec{\boldsymbol{v}}_0\| = \lambda$.

Teorema 4 (de Blichfeldt) *Dado un retículo $\Lambda(\boldsymbol{\mathcal{B}})$ y un subconjunto $\boldsymbol{S} \subseteq \mathbb{R}^m$ si*

$$\mathrm{Vol}(\boldsymbol{S}) > \det(\boldsymbol{B})$$

o

$$\mathrm{Vol}(\boldsymbol{S}) = \det(\boldsymbol{B}) \quad y\ \boldsymbol{S}\ es\ compacto,$$

entonces existen $\vec{\boldsymbol{v}}_1,\ \vec{\boldsymbol{v}}_2 \in \boldsymbol{S}$ tales que $\vec{\boldsymbol{v}}_1 - \vec{\boldsymbol{v}}_2 \in \Lambda(\boldsymbol{\mathcal{B}})$

Corolario 2 (Minkowski) *Si $\boldsymbol{S}$ es un conjunto convexo simétrico tal que*

$$\mathrm{Vol}(\boldsymbol{S}) > 2^m \det(\boldsymbol{B})$$

o

$$\mathrm{Vol}(\boldsymbol{S}) = \det(\boldsymbol{B}) \quad y\ \boldsymbol{S}\ es\ compacto,$$

entonces existe $\vec{\boldsymbol{v}} \in \Lambda(\boldsymbol{\mathcal{B}}) \setminus \{\vec{\boldsymbol{0}}\}$ tal que $\vec{\boldsymbol{v}} \in \boldsymbol{S}$.

El teorema de Minkowski y acotar la longitud del vector más corto en un retículo tienen mucho que ver. Considere

$$\boldsymbol{S} = C(\vec{\boldsymbol{0}}, \det(\boldsymbol{B})) = \left\{ \vec{\boldsymbol{x}} \in \Lambda(\boldsymbol{\mathcal{B}}) \;\middle|\; \|\vec{\boldsymbol{x}}\|_\infty \leq \sqrt[m]{\det(\boldsymbol{B})} \right\}$$

cuerpo simétrico convexo con la norma ℓ_∞. Observe que

$$\mathrm{Vol}(\boldsymbol{S}) = 2^m \det(\boldsymbol{B}).$$

Por el teorema de Minkowski existe un vector $\vec{\boldsymbol{v}} \in \Lambda(\boldsymbol{\mathcal{B}}) \setminus \{\vec{\boldsymbol{0}}\}$ tal que $\vec{\boldsymbol{v}} \in \boldsymbol{S}$, es decir

$$\|\vec{\boldsymbol{v}}\|_\infty \leq \sqrt[m]{\det(\boldsymbol{B})}.$$

Corolario 3 *Sea $\Lambda(\boldsymbol{\mathcal{B}})$ un retículo de rango completo, entonces existe $\vec{\boldsymbol{v}} \in \Lambda(\boldsymbol{\mathcal{B}}) \setminus \{\vec{\boldsymbol{0}}\}$ tal que*

$$\|\vec{\boldsymbol{v}}\|_\infty \leq \sqrt[m]{\det(\boldsymbol{B})}.$$

Corolario 4 *Sea $\Lambda(\boldsymbol{\mathcal{B}})$ un retículo de rango completo, entonces existe $\vec{\boldsymbol{v}} \in \Lambda(\boldsymbol{\mathcal{B}}) \setminus \{\vec{\boldsymbol{0}}\}$ tal que*

$$\|\vec{\boldsymbol{v}}\|_2 < \sqrt{m}\ \sqrt[m]{\det(\boldsymbol{B})}.$$

Reformulando

$$PVMC_\infty \leq \sqrt[m]{\det(\boldsymbol{B})}$$

y

$$PVMC_2 < \sqrt{m}\ \sqrt[m]{\det(\boldsymbol{B})}.$$

Para el caso euclideano el rango completo no es necesario.

Definición 35 *Se define el* $i-$ÉSIMO MÍNIMO $\lambda_i(\Lambda)$ *de un retículo* Λ *como el radio de la esfera más pequeña centrada en el origen que contiene* i *vectores l. i. de* Λ*, y se llama a* $\lambda_1(\Lambda), \lambda_2(\Lambda), \ldots, \lambda_n(\Lambda)$ MÍNIMOS SUCESIVOS *de* Λ*.*

En particular

$$\lambda_1(\Lambda) = \min\left\{\|\vec{\boldsymbol{x}} - \vec{\boldsymbol{y}}\| \;\middle|\; \vec{\boldsymbol{x}},\ \vec{\boldsymbol{y}} \in \Lambda,\ \vec{\boldsymbol{x}} \neq \vec{\boldsymbol{y}}\right\} = \min\left\{\|\vec{\boldsymbol{x}}\| \;\middle|\; \vec{\boldsymbol{x}} \in \Lambda \setminus \{\vec{\mathbf{0}}\}\right\}$$

Otro problema para el cual no se conoce solución de tiempo polinomial es el siguiente.

Definición 36 *Sea* $\|\cdot\|$ *una norma. El* PROBLEMA DEL VECTOR MÁS CERCANO *(PVMCe), con respecto a la norma* $\|\cdot\|$*, es, dado un retículo* $\Lambda(\boldsymbol{\mathcal{B}})$*, encontrar un vector* $\vec{\boldsymbol{v}}_0 \in \Lambda(\boldsymbol{\mathcal{B}}) \setminus \{\vec{\boldsymbol{v}}\}$ *de distancia mínima a* $\vec{\boldsymbol{v}}$*. En otras palabras, dado* $\Lambda(\boldsymbol{\mathcal{B}})$*, encontrar* $\vec{\boldsymbol{v}}_0 \in \Lambda(\boldsymbol{\mathcal{B}}) \setminus \{\vec{\boldsymbol{v}}\}$ *tal que* $\|\vec{\boldsymbol{v}} - \vec{\boldsymbol{v}}_0\| \leq \|\vec{\boldsymbol{w}}\|$ *para cualquier* $\vec{\boldsymbol{w}} \in \Lambda(\boldsymbol{\mathcal{B}})$*.*

Se dice que un algoritmo resuelve el PVMC aproximadamente dentro de un factor c (que posiblemente depende de la dimensión del retículo) si encuentra un vector $\vec{\boldsymbol{x}} \in \Lambda \setminus \{\vec{\mathbf{0}}\}$ de longitud a lo más c veces la longitud del vector más corto en $\Lambda \setminus \{\vec{\mathbf{0}}\}$ y analógamente un algoritmo resuelve el PVMCe aproximadamente dentro de un factor c si encuentra un vector $\vec{\boldsymbol{x}} \in \Lambda \setminus \{\vec{\mathbf{0}}\}$ tal que $\|\vec{\boldsymbol{x}} - \vec{\boldsymbol{y}}\|$ es a lo más c veces la distancia de $\vec{\boldsymbol{y}}$ a Λ.

Aunque un retículo Λ puede contener vectores mucho más cortos que $\sqrt{n}\sqrt[n]{\mathrm{d}(\Lambda)}$, se ha probado que aproximar el vector más corto dentro de un factor polinomial (en n) se puede reducir a encontrar un vector de longitud dentro de un factor polinomial de $\sqrt[n]{\mathrm{d}(\Lambda)}$.

Con el desarrollo del algoritmo LLL para la reducción de bases fue posible aproximar el PVMC en tiempo polinomial dentro de un factor $2^{\frac{n}{2}}$. Esta aproximación fue mejorada a $2^{O\left(\frac{n(\ln\ln n)^2}{\ln n}\right)}$ por Schnorr [Sch87] usando una modificación de LLL. El algoritmo LLL y sus variantes también se puede usar para encontrar en tiempo polinomial soluciones exactas al PVMC en cualquier número fijo de dimensiones. La dependencia del tiempo de corrida en la dimensión es 2^{n^2}, aunque existen algoritmos para resolver el PVMC exactamente con una dependencia del tiempo de corrida en la dimensión dada por $2^{n \ln n}$ dado por Kannan [Kan].

Aunque en la práctica el algoritmo LLL y sus variantes funcionan mucho mejor que la cota inferior teórica para el peor caso, a la fecha no se conoce un algoritmo de tiempo polinomial en la dimensión del retículo para aproximar el PVMC dentro de un factor polinomial en la dimensión del retículo.

La relación entre el PVMC y el PVMCe también se ha considerado, y la analogía con algunos problemas diofantinos además del rápido avance en las pruebas de la dificultad del PVMCe y la mayor facilidad en la aproximación

del PVMC sugieren que éste puede ser más fácil que el PMVCe. Aunque la completitud $\mathcal{NP}$ del PVMCe implica que la versión decisional del PVMC puede ser reducida en tiempo polinomial al PVMCe, no existe una reducción directa y obvia entre estos 2 problemas, y encontrar la relación entre sus versiones de aproximación ha sido un problema abierto hasta ahora. Observe que una instancia del PVMC en Λ no es equivalente a una instancia del PVMCe en $(\Lambda, \vec{\mathbf{0}})$ pues el vector más cercano al origen es el origen mismo (en el PVMCe no se requiere que la solución sea necesariamente distinta de $\vec{\mathbf{0}}$). Goldreich [GMSS] encontró una reducción directa muy eficiente del PVMC al PVMCe que preserva el factor de aproximación, el rango del retículo y se mantiene para cualquier norma, formalizando la intuición de que el PVMC no es más dificil que el PVMCe.

§4.3 Algoritmo LLL

El algoritmo LLL [LLL82] sirve para resolver en tiempo polinomial el siguiente **Problema:**

dado un polinomio $\boldsymbol{f} \in \mathbb{Q}[x]^*$, encontrar la descomposición de $\boldsymbol{f}$ en factores irreducibles en $\mathbb{Q}[x]$.

Esto es equivalente a factorizar polinomios primitivos $\boldsymbol{f} \in \mathbb{Z}[x]$ en factores irreducibles en $\mathbb{Z}[x]$.

Este algoritmo funciona bien en la práctica. Su tiempo de corrida, medido en operaciones de bit, es $O(n^{12} + n^9(\log|\boldsymbol{f}|)^3)$. Aquí $\boldsymbol{f} \in \mathbb{Z}[x]$ es el polinomio a ser factorizado, $n = \mathrm{gra}(\boldsymbol{f})$ es el grado de $\boldsymbol{f}$, y

$$\left|\sum_{i=0}^{n-1} a_i x^i\right| = \left(\sum_{i=0}^{n-1} a_i^2\right)^{\frac{1}{2}}$$

para un polinomio en $\mathbb{R}[x]$.

Un esbozo del algoritmo es como sigue. Primero se encuentra, para un número primo p pequeño adecuado, un factor irreducible $p-$ádico $\boldsymbol{h}$ de $\boldsymbol{f}$, para una cierta precisión. Esto es hecho con el algoritmo de Berlekamp [Ber68] para factorizar polinomios sobre campos finitos pequeños, combinado con el lema de Hensel [Eis99]. En seguida se busca un factor irreducible $\boldsymbol{h}_0 \in \mathbb{Z}[x]$ de $\boldsymbol{f}$ que es divisible por $\boldsymbol{h}$. La condición que $\boldsymbol{h}|\boldsymbol{h}_0$ significa que $\boldsymbol{h}_0$ pertenece a cierto retículo, y la condición de que $\boldsymbol{h}_0|\boldsymbol{f}$ implica que los coeficientes de $\boldsymbol{h}_0$ son relativamente pequeños. Se sigue que se debe buscar un elemento pequeño en el retículo, y esto es hecho por medio del algoritmo de reducción de bases. Lo cual permite determinar $\boldsymbol{h}_0$. El algoritmo se repite hasta que todos los factores reducibles de $\boldsymbol{f}$ han sido encontrados.

Al proceso de resolver el problema de encontrar vectores muy cortos en retículos de dimensión grande se le llama REDUCCIÓN RETICULAR y al estudio general de vectores pequeños en retículos es llamado GEOMETRÍA DE NÚMEROS.

4.3.1 Bases reducidas para un retículo

La demostración de las proposiciones presentadas en el resto de este capítulo pueden consultarse en [LLL82].

Definición 37 *Se dice que una base* $\{\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n\}$ *de un retículo* Λ *es* $\delta - LLL$ $\left(\frac{1}{4} \leq \delta < 1\right)$ *reducida si*

$$|\mu_{ij}| \leq \frac{1}{2} \qquad para \qquad 1 \leq j < i \leq n \tag{4.3}$$

y

$$\delta \left|\vec{b}^{\star}_{i-1}\right|^2 \leq \left|\mu_{ii-1}\vec{b}^{\star}_{i-1} + \vec{b}^{\star}_{i}\right|^2 \qquad para \qquad 1 < i \leq n, \tag{4.4}$$

donde $|\cdot|$ *denota la norma euclideana ordinaria.*

Observe que los vectores $\vec{b}^{\star}_i + \mu_{ii-1}\vec{b}^{\star}_{i-1}$ y $\vec{b}^{\star}_{i-1}$ que aparecen en (4.4) son las proyecciones de $\vec{b}_i$ y $\vec{b}_{i-1}$ sobre el complemento ortogonal de $\sum_{j=1}^{i-2} \mathbb{R}\vec{b}_j$.

Proposición 11 *Sean* $\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n$ *una base reducida para un retículo* Λ *en* $\mathbb{R}^n$ *y* $\vec{b}^{\star}_1, \vec{b}^{\star}_2, \ldots, \vec{b}^{\star}_n$ *definida como antes. Entonces se tiene*

$$\left|\vec{b}_j\right|^2 \leq 2^{i-1}\left|\vec{b}^{\star}_i\right|^2 \quad para \;\; 1 \leq j \leq i \leq n, \tag{4.5}$$

$$\begin{aligned} \mathrm{d}(\Lambda) &\leq \prod_{i=1}^{n}\left|\vec{b}_i\right| \\ &\leq 2^{\frac{n(n-1)}{4}}\,\mathrm{d}(\Lambda), \qquad (4.6) \\ \left|\vec{b}_1\right| &\leq 2^{\frac{n-1}{4}}\,\mathrm{d}(\Lambda)^{\frac{1}{n}} \qquad (4.7) \end{aligned}$$

Observación 1 *Si* δ *en (4.4) es reemplazada por* y*, con* $\frac{1}{4} < y < 1$*, entonces las potencias de 2 que aparecen en (4.5), (4.6) y (4.7) deben de ser reemplazadas por las mismas potencias de* $\frac{4}{4y-1}$.

Proposición 12 *Sea* $\Lambda \subset \mathbb{R}^n$ *un retículo con una base reducida* $\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n$. *Entonces*

$$\left|\vec{b}_1\right|^2 \leq 2^{n-1}|\vec{x}|^2, \quad \forall\, \vec{x} \in \Lambda \setminus \{\vec{0}\}.$$

Proposición 13 *Sean $\Lambda \subset \mathbb{R}^n$ un retículo con base reducida $\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n$ y $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_t\} \subset \Lambda$ un conjunto l. i. Entonces*

$$\left|\vec{b}_j\right|^2 \leq 2^{n-1} \cdot \max\left\{ |x_1|^2, |x_2|^2, \ldots, |x_t|^2 \;\middle|\; j = 1, 2, \ldots, t \right\}.$$

Observación 2 *Si $\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n$ es una base reducida para Λ. Entonces (4.5) y (13) implican que*

$$2^{1-i}\lambda_i(\Lambda) \leq \left|\vec{b}_i\right|^2 \leq 2^{n-1}\lambda_i(\Lambda) \quad \textit{para} \quad 1 \leq i \leq n$$

por lo que $\left|\vec{b}_i\right|^2$ es una aproximación razonable de $\lambda_i(\Lambda)$.

Observación 3 *Note que el número 2^{n-1} puede ser reemplazado en (12) por* $\max\left\{ \dfrac{\left|\vec{b}_1\right|^2}{\left|\vec{b}_i^{\star}\right|^2} \;\middle|\; 1 \leq i \leq n \right\}$ *y en (13)* $\max\left\{ \dfrac{\left|\vec{b}_j\right|^2}{\left|\vec{b}_i^{\star}\right|^2} \;\middle|\; 1 \leq j \leq i \leq n \right\}$.

Se describe ahora un algoritmo que transforma una base dada $\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n$ para un retículo Λ en una reducida. Esta descripción incorpora una optimización adicional debida a J. J. M. Cuppen, que reduce el tiempo estimado de corrida por un factor n.

Para inicializar el algoritmo se calcula $\vec{b}_i^{\star}$ para $1 \leq i \leq n$ y μ_{ij} para $1 \leq j < i \leq n$ usando (4.1) y (4.2). A lo largo del algoritmo los vectores $\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n$ cambiaran varias veces, pero siempre de tal manera que formen una base para Λ. Después de cada cambio de los $\vec{b}_i$ se cambian también $\vec{b}_i^{\star}$ y μ_{ij} de tal manera que (4.1) y (4.2) siguen siendo válidas.

En cada paso del algoritmo se tiene un subíndice $k \in \{1, 2, \ldots, n+1\}$. Se empieza con $k = 2$ y se itera despues una secuencia de pasos por la cual empezar, y a la cual regresar, en una situación en la cual se satisfagan las siguientes condiciones

$$|\mu_{ij}| \leq \frac{1}{2} \quad \text{para} \quad 1 \leq j < i < k \tag{4.8}$$

$$\delta \left|\vec{b}_{i-1}^{\star}\right|^2 \leq \left|\mu_{ii-1}\vec{b}_{i-1}^{\star} + \vec{b}_i^{\star}\right|^2 \quad \text{para} \quad 1 < i < k. \tag{4.9}$$

Estas condiciones se satisfacen trivialmente si $k = 2$.

Con la situación anterior se prodece como sigue. Si $k = n + 1$ entonces la base está reducida, y el algoritmo termina. Suponga ahora que $k \leq n$. Entonces primero se obtiene que

$$|\mu_{kk-1}| \leq \frac{1}{2} \quad \text{si} \quad k > 1. \tag{4.10}$$

Si esto no es válido, sea $r \in \mathbb{Z}$ el más cercano a μ_{kk-1}, y se reemplaza $\vec{b}_k$ por $\vec{b}_k - r\vec{b}_{k-1}$. Los números μ_{kj} con $j < k-1$ son entonces reemplazados por $\mu_{kj} - r\mu_{k-1j}$, y μ_{kk-1} por $\mu_{kk-1} - r$. Los otros μ_{ij} y todos los $\vec{b}_i^\star$ no cambian. Después de este cambio (4.10) sigue siendo válida.

En seguida se distinguen 2 casos.

i) Suponga que $k \geq 2$ y

$$\left|\mu_{kk-1}\vec{b}_{k-1}^\star + \vec{b}_k^\star\right|^2 < \delta \left|\vec{b}_{k-1}^\star\right|^2. \tag{4.11}$$

Entonces se intercambian $\vec{b}_{k-1}$ y $\vec{b}_k$, y se dejan los otros $\vec{b}_i$ sin cambiar. Los vectores $\vec{b}_{k-1}^\star$ y $\vec{b}_k^\star$ y los números μ_{kk-1}, μ_{k-1j}, μ_{kj}, μ_{ik-1}, μ_{ik}, para $j < k-1$ y para $i > k$, tienen que ser reemplazados. Esto se hace con la forma que se da más abajo. El cambio más importante de éstos es que $\vec{b}_{k-1}^\star$ es reemplazado por $\mu_{kk-1}\vec{b}_{k-1}^\star + \vec{b}_k^\star$; así que el nuevo valor $\left|\vec{b}_{k-1}^\star\right|^2 < \delta$ veces el anterior. Estos cambios siendo hechos, se reemplaza k por $k-1$. Entonces se está en la situación descrita por (4.8) y (4.9), y se procede con el algoritmo de ahí.

ii) Suponga que $k = 1$ o

$$\delta \left|\vec{b}_{k-1}^\star\right|^2 \leq \left|\mu_{kk-1}\vec{b}_{k-1}^\star + \vec{b}_k^\star\right|^2. \tag{4.12}$$

En este caso primero se logra que

$$|\mu_{kj}| \leq \frac{1}{2} \quad \text{para} \quad 1 \leq j \leq k-1. \tag{4.13}$$

Para $j = k-1$ esto es ya verdadero, por (4.10). Si (4.13) no se cumple, sea l el mayor índice $< k$ con $|\mu_{kl}| > \frac{1}{2}$, sea $r \in \mathbb{Z}$ el más cercano a μ_{kl}, y se reemplaza $\vec{b}_k$ por $\vec{b}_k - r\vec{b}_l$. Los números μ_{kj} con $j < l$ son entonces reemplazados por $\mu_{kj} - r\mu_{lj}$, y μ_{kl} por $\mu_{kl} - r$; los otros μ_{ij} y todos los $\vec{b}_i^\star$ no cambian. Se está ahora en la situación descrita por (4.8) y (4.9), y se procede con el algoritmo a partir de aquí.

Para el caso $k = 1$ no se ha hecho mas que reemplazar k por 2.

Para completar se da la fórmula necesaria en el caso *i*). Sea $\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n$ la base actual y $\vec{b}_i^\star$, μ_{ij} como en (4.1) y (4.2). Sea k el subíndice actual para el cual (4.8), (4.9), (4.10) y (4.11) son válidas. Por $\vec{c}_i$, $\vec{c}_i^\star$ y ν_{ij} se denota a los vectores y números que reemplazarán a $\vec{b}_i$, $\vec{b}_i^\star$ y μ_{ij} respectivamente. La nueva base $\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_n$ está dada por

$$\vec{c}_{k-1} = \vec{b}_k, \quad \vec{c}_k = \vec{b}_{k-1}, \quad \vec{c}_i = \vec{b}_i \quad \text{para} \quad i \neq k-1, k.$$

Dado que $\vec{c}^{\star}_{k-1}$ es la proyección $\vec{b}_k$ sobre el complemento ortogonal de $\sum_{j=1}^{k-2} \mathbb{R}\vec{b}_j$ se tiene, como se afirmó (Obs. 1)

$$\vec{c}^{\star}_{k-1} = \mu_{kk-1}\vec{b}^{\star}_{k-1} + \vec{b}^{\star}_{k}.$$

Para obtener $\vec{c}^{\star}_{k}$ se debe proyectar $\vec{b}^{\star}_{k-1}$ sobre el complemento ortogonal de $\mathbb{R}\vec{c}^{\star}_{k-1}$. Esto lleva a

$$\begin{aligned} \nu_{kk-1} &= \frac{\left\langle \vec{b}^{\star}_{k-1}, \vec{c}^{\star}_{k-1} \right\rangle}{\left\langle \vec{c}^{\star}_{k-1}, \vec{c}^{\star}_{k-1} \right\rangle} \\ &= \mu_{kk-1} \frac{\left|\vec{b}^{\star}_{k-1}\right|^2}{\left|\vec{c}^{\star}_{k-1}\right|^2}, \\ \vec{c}^{\star}_{k} &= \vec{b}^{\star}_{k-1} - \nu_{kk-1}\vec{c}^{\star}_{k-1}. \end{aligned}$$

Para $i \neq k-1,\ k$ se tiene $\vec{c}^{\star}_{i} = \vec{b}^{\star}_{i}$. Sea ahora $i > k$. Para encontrar ν_{ik-1} y ν_{ik} se sustituye

$$\begin{aligned} \vec{b}^{\star}_{k-1} &= \nu_{kk-1}\vec{c}^{\star}_{k-1} + \vec{c}^{\star}_{k} \\ \vec{b}^{\star}_{k} &= (1 - \mu_{kk-1}\nu_{kk-1})\vec{c}^{\star}_{k-1} - \mu_{kk-1}\vec{c}^{\star}_{k} \\ &= \left(\frac{\left|\vec{b}^{\star}_{k}\right|^2}{\left|\vec{c}^{\star}_{k-1}\right|^2} \right) \cdot \vec{c}^{\star}_{k-1} - \mu_{kk-1}\vec{c}^{\star}_{k} \end{aligned}$$

en $\vec{b}_i = \vec{b}^{\star}_{i} + \sum_{j=1}^{i-1} \mu_{ij}\vec{b}^{\star}_{i}$. Esto lleva a

$$\begin{aligned} \nu_{ik-1} &= \mu_{ik-1}\nu_{kk-1} + \mu_{ik} \frac{\left|\vec{b}^{\star}_{k}\right|^2}{\left|\vec{c}^{\star}_{k-1}\right|^2}, \\ \nu_{ik} &= \mu_{ik-1} - \mu_{ik}\mu_{kk-1}. \end{aligned}$$

Finalmente, se tiene

$$\nu_{k-1j} = \mu_{kj}, \qquad \nu_{kj} = \mu_{k-1j}$$

para $1 \leq j < k-1$, y $\nu_{ij} = \mu_{ij}$ si $1 \leq j < i \leq n,\ \{i,j\} \cap \{k-1,k\} = \emptyset$.

Se observa que después de la etapa de la inicialización del algoritmo no es necesario no perder de vista a los vectores $\vec{b}^{\star}_{i}$. Es suficiente con llevar cuenta de los números $\left|\vec{b}^{\star}_{i}\right|^2$, además de μ_{ij} y de los vectores $\vec{b}_i$. Note que

$\left|\vec{c}_k^{\star}\right|^2 = \left|\vec{b}_{k-1}^{\star}\right|^2 \cdot \dfrac{\left|\vec{b}_k^{\star}\right|^2}{\left|\vec{c}_{k-1}^{\star}\right|^2}$ en lo anterior, y que el lado izquierdo de (4.11) y (4.12) es igual a $\left|\vec{b}_k^{\star}\right|^2 + \mu_{kk-1}^2 \left|\vec{b}_{k-1}^{\star}\right|^2$.

Para probar que el algoritmo termina se introducen las cantidades

$$d_i = \det\left(\left\langle \vec{b}_j, \vec{b}_l \right\rangle\right)_{1 \le j,l \ge i} \qquad \text{para } \ 0 \le i \le n. \tag{4.14}$$

Se checa que

$$d_i = \prod_{j=1}^{i} \left|\vec{b}_j^{\star}\right|^2 \qquad \text{para } \ 0 \le i \le n. \tag{4.15}$$

De aquí que los $d_i \in \mathbb{R}^+$. Note que $d_0 = 1$ y $d_n = \mathrm{d}(\Lambda)^2$. Sea $D = \prod_{i=1}^{n-1} d_i$. Por (4.15), el número D cambia solamente si alguna de los $\vec{b}_i^{\star}$ es cambiado, lo cual sólo ocurre en el caso i). En el caso i), el número $d_k - 1$ es reducido por un factor $< \delta$, por (4.15), mientras que los otros d_i no cambian, por (4.14); de aquí que D esté reducida por un factor $< \delta$. Abajo se prueba que existe una cota positiva inferior para d_i que sólo depende de Λ. Se sigue que existe también una cota positiva inferior para D, y por lo tanto una cota superior para el número de veces que pasamos por el caso i).

En el caso i), el valor de k es decrementado en 1, y en el caso ii) es incrementado en 1. Inicialmente se tiene $k = 2$ y $k \le n + 1$ a través del algoritmo. Por lo tanto el número de veces que se pasa por el caso ii) es a lo más $n - 1$ más que el número de veces que se pasa por el caso i), y consecuentemente está acotado. Esto implica que el algoritmo termina.

Para probar que d_i tiene una cota inferior se pone

$$m(\Lambda) = \min\{|\vec{x}|^2 \mid \vec{x} \in \Lambda, \ \vec{x} \neq 0\}.$$

Este es un número real positivo. Para $i > 0$, se interpreta d_i como el cuadrado del determinante del retículo de rango i generado por $\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_i$ en el espacio vectorial $\sum_{j=1}^{i} \mathbb{R}\vec{b}_j$. Este retículo contiene un vector $\vec{x} \neq \vec{0}$ con $|\vec{x}|^2 \le \left(\dfrac{1}{\delta}\right)^{\frac{i-1}{2}} d_i^{\frac{1}{i}}$. Por lo tanto $\delta^{\frac{i(i-1)}{2}} m(\Lambda)^i \le d_i$, como se requería.

Proposición 14 *Sean $\Lambda \subset \mathbb{Z}^n$ un retículo con base $\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n$, $\vec{b}_i \in \mathbb{Z}^n$ para $1 \le i \le n$; y $B \in \mathbb{R}$, $B \ge 2$, tal que $|\vec{b}_i|^2 \le B$ para $1 \le i \le n$. Entonces el número de operaciones aritméticas necesarias para el algoritmo de reducción de bases es $O(n^4 \log B)$, y los enteros en los cuales se aplican estas operaciones tienen cada uno longitud $O(n \log B)$.*

Observación 4 *Usando algoritmos clásicos para las operaciones aritméticas se encuentra que el número de operaciones bit necesarias para el algoritmo de reducción de bases es* $O(n^6(\log B)^3)$. *Esto puede ser reducido a* $O(n^{5+\varepsilon}(\log B)^{2+\varepsilon})$, $\forall\ \varepsilon > 0$, *si se emplean técnicas de multiplicación rápida.*

Observación 5 *Sea* $1 \leq t \leq n$. *Si* k, *en las situaciones descritas por (4.8) y (4.9), es por primera vez igual a* $t+1$, *entonces los primeros* t *vectores* $\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{b}}_2, \ldots, \vec{\boldsymbol{b}}_t$ *forman una base reducida para el retículo de rango* t *generado por los primeros* t *vectores de la base dada inicialmente.*

Observación 6 *Se verifica que como aparte de algunos cambios menores, el análisis de este algoritmo permanece válido si la condición* $\Lambda \subset \mathbb{Z}^n$ *es reemplazada por la condición que* $\langle \vec{\boldsymbol{x}}, \vec{\boldsymbol{y}} \rangle \in \mathbb{Z}\ \forall\ \vec{\boldsymbol{x}},\ \vec{\boldsymbol{y}} \in \Lambda$; *o, equivalentemente,* $\langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{b}}_j \rangle \in \mathbb{Z}$ *para* $1 \leq i, j \leq n$. *La condición más débil* $\langle \vec{\boldsymbol{b}}_i, \vec{\boldsymbol{b}}_j \rangle \in \mathbb{Q}$, *para* $1 \leq i, j \leq n$, *es también suficiente.*

Algoritmo de Reducción

Entrada: $\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_n$

Salida: Una base reducida.

Paso 1: Inicialización: $\vec{b}_i^{\star} := \vec{b}_i$; $B_i = \left|\vec{b}_i^{\star}\right|^2$;

Paso 2: Para $i = 1, 2, \ldots, n$

Paso 3: Para $j = 1, 2, \ldots, i-1$

Paso 4: $$\mu_{ij} := \frac{\left\langle \vec{b}_i, \vec{b}_j^{\star} \right\rangle}{B_j};$$

Paso 5: $\vec{b}_i^{\star} := \vec{b}_i^{\star} - \mu_{ij}\vec{b}_j^{\star}$;

Paso 6: $B_i := \left\langle \vec{b}_i^{\star}, \vec{b}_i^{\star} \right\rangle$;

Paso 7: $k := 2$;

Paso 8: Haz 26 con $l = k - 1$;

Paso 9: Si $B_k < \left(\frac{3}{4} - \mu_{kk-1}^2\right) B_{k-1}$

Paso 10: ve a 16;

Paso 11: Haz 26 con $l = k-2,\ k-3,\ \ldots,\ 1$;

Paso 12: Si $k = n$

Paso 13: Termina;

Paso 14: $k := k + 1$;

Paso 15: Ve a 8;

Paso 16: $\mu := \mu_{kk-1}$;
$B := B_k + \mu^2 \vec{b}_{k-1}$;
$\mu_{kk-1} := \mu \frac{B_{k-1}}{B}$;

Paso 17: $B_k := B_{k-1} \frac{B_k}{B}$;
$B_{k-1} := B$;

Paso 18: $$\begin{pmatrix} \vec{b}_{k-1} \\ \vec{b}_k \end{pmatrix} := \begin{pmatrix} \vec{b}_k \\ \vec{b}_{k-1} \end{pmatrix};$$

Paso 19: Para $j = 1, 2, \ldots, k-2$

Paso 20: $$\begin{pmatrix} \mu_{k-1j} \\ \mu_{kj} \end{pmatrix} := \begin{pmatrix} \mu_{kj} \\ \mu_{k-1j} \end{pmatrix};$$

Paso 21: Para $i = k+1, k+2, \ldots, n$

Paso 22: $$\begin{pmatrix} \mu_{ik-1} \\ \mu_{ik} \end{pmatrix} := \begin{pmatrix} 1 & \mu_{kk-1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -\mu \end{pmatrix} \begin{pmatrix} \mu_{ik-1} \\ \mu_{ik} \end{pmatrix};$$

Paso 23: Si $k > 2$

Paso 24: $k := k - 1$;

Paso 25: Ve a 8

Paso 26: Si $|\mu_{kl}| > \frac{1}{2}$

Paso 27: $r :=$ el entero mas cercano a μ_{kl};
$\vec{b}_k := \vec{b}_k - r\vec{b}_l$;

Paso 28: Para $j = 1, 2, \ldots, l-1$

Paso 29: $\mu_{kj} := \mu_{kj} - r\mu_{lj}$;

Paso 30: $\mu_{kl} := \mu_{kl} - r$;

5 *NTRU*

§5.1 Introducción

La colección original de criptosistemas rotos por reducción reticular es la de los basados en variaciones del PROBLEMA DE LA MOCHILA (KNAPSACK).

Dados $\{v_1, v_2, \ldots, v_k\} \subset \mathbb{N}$ y $s \in \mathbb{Z}$, encontrar $n \in \mathbb{Z}$ de k bits

$$n = b_{k-1}b_{k-2}\ldots b_1 b_0, \quad b_i \in \{0, 1\},$$

tal que

$$s = \sum_{i=0}^{k-1} b_i v_i,$$

si tal n existe.

Cada que aparecía una variación, los autores esperanzados hacían todo lo posible para evitar transformaciones potenciales de sus "problemas difíciles" a uno que pudiera ser atacado por reducción reticular. En un artículo, Odlyzko [CLOS91] describe que no importa que tan oscuro sea el problema original, siempre es posible transformarlo a uno vulnerable a métodos de reducción reticular. Uno de estos criptosistemas es el de Chor-Rivest [CR85], roto por Schnorr [SH94].

Entre otros, uno de los problemas fundamentales que enfrentaron estos autores fue el hecho de que el tamaño de la clave de su sistema crecía con el cuadrado de la dimensión del retículo usado para un ataque. Debido a esto, era imposible hacer la dimensión de un ataque reticular mayor que un par de cientos sin simultáneamente hacer el criptosistema impráctico. Desarrollos en tecnología LLL por Schnorr, Euchner y otros [SE91a], permiten ahora el análisis de retículos de dimensión 200 o menor en un lapso corto, y aún ciertos retículos de dimensión mayor a 300 pueden ser ocasionalmente rotos. Esto es suficiente para disponer de todos los sistemas knapsack prácticos, entre otros el propuesto en 1997 por Goldreich, Golwasser & Halevi [GGH97]. Este sistema también tenía una clave que crecía con el cuadrado de la dimensión del retículo, por lo que el retículo era muy pequeño para ser seguro o el tamaño de la clave muy

grande para ser práctico.

Con la llegada de computadoras modernas, la atención se enfocó en el problema práctico de encontrar vectores cortos en retículos. Siempre se puede encontrar "el" vector más corto por búsqueda exahustiva, pero el espacio de búsqueda crece exponencialmente en la dimensión del retículo.

El avance moderno más importante en teoría algorítmica de reducción retícular es el método descubierto por Lenstra, Lenstra, Lovasz (LLL) [LLL82].

5.1.1 Descripción

El NTRU[1] (Criptosistema de Clave Pública) fue presentado por primera vez en **CRYPTO 96**, y desarrollado conjuntamente por los profesores Jeffrey Hoffstein, Jill Pipher & Joseph H. Silverman [JHS98] y se tiene patentado por una PCT de los EE. UU.

Las operaciones básicas de NTRU involucran números menores que 255. Un riguroso análisis matemático muestra que para llaves de N bits NTRU requiere sólo de alrededor de N^2 $(N\log(N))$ operaciones para cifrar o descrifrar un mensaje.

En el NTRU los tamaños de las llaves son lineales en la dimensión por lo que son prácticas y basan su seguridad en la reducción reticular, la clave privada corresponde a un vector en un retículo de dimensión n (500 a 1000) y las técnicas actuales son completamente incapaces de tratar con problemas tipo de este tamaño, si además se usan los parámetros sugeridos. A grandes rasgos, si el vector corto, o uno de no más de 2 o 3 veces su longitud se puede encontrar, entonces el correspondiente por clave pública-privada se puede romper. Para establecer un nivel de seguridad de, digamos 2^t es necesario escoger parámetros (digamos la clave privada de t bits) de tal manera que el tiempo estimado para encontrar un vecto corto en el correpondiente retículo sea del orden de 2^t.

Para refinar el análisis de seguridad de NTRU, se han corrido pruebas usando las mejores implementaciones del algoritmo LLL, hechas por Victor Shoup[2], logrando obtener un tiempo de ruptura en al menos tiempo exponencial en la dimensión. Mientras que la criba numérica es un método de factorización muy sofisticado que factoriza en tiempo subexponencial.

Finalmente, se puede incrementar un poco la complejidad del problema subyacente al NTRU incrementando la dimensión del retículo problema asociado. Esto permite un fortalecimiento considerable de seguridad a un costo de sólo incrementos muy pequeños en la complejidad computacional.

[1] Number Theory Research Unit
[2] http://www.shoup.net/

§5.2 Parámetros

Los objetos básicos usados por el NTRU son los elementos del anillo $\boldsymbol{\mathcal{R}}$.

Además del número N que específica el grado de los polinomios truncados el NTRU requiere que el usuario proporcione algunos otros parámetros.

N Los polinomios en el anillo $\mathbb{Z}^N[x]$ tienen grado a lo más $N-1$. Además $N = q - 1$ para firmas digitales.

q Módulo mayor.

p Módulo menor.

$d_{\boldsymbol{f}}$ La clave privada $\boldsymbol{f}$ tiene $d_{\boldsymbol{f}}$ de sus coeficientes iguales a 1, $d_{\boldsymbol{f}} - 1$ iguales a -1 y el resto a 0. Satisface $d_{\boldsymbol{f}} < \left[\frac{N}{2}\right]$.

$d_{\boldsymbol{g}}$ La clave privada $\boldsymbol{g}$ tiene $d_{\boldsymbol{g}}$ de sus coeficientes iguales a 1, $d_{\boldsymbol{g}}$ iguales a -1 y el resto a 0. Satisface $d_{\boldsymbol{g}} < \left[\frac{N}{2}\right]$.

$d_{\boldsymbol{r}}$ El polinomio aleatorio de cifrado $\boldsymbol{r}$ tiene $d_{\boldsymbol{r}}$ de sus coeficientes iguales a 1, $d_{\boldsymbol{r}}$ iguales a -1 y el resto a 0. Satisface $d_{\boldsymbol{r}} < \left[\frac{N}{2}\right]$.

$d_{\boldsymbol{c}}$ Los polinomios de intercambio $\boldsymbol{c}_{ij}$ tienen $d_{\boldsymbol{c}}$ de sus coeficientes iguales a 1, $d_{\boldsymbol{c}}$ iguales a -1 y el resto a 0.

$\mathcal{L}(d_1, d_2)$ Conjunto de polinomios con d_1 coeficientes iguales a 1 y d_2 coeficientes iguales a -1, y el resto a 0

$\mathcal{L}_{\boldsymbol{f}}$ Conjunto de polinomios que satisfacen la condición $d_{\boldsymbol{f}}$, $\mathcal{L}_{\boldsymbol{f}} \subset \mathbb{Z}^N$. $\mathcal{L}(d_{\boldsymbol{f}}, d_{\boldsymbol{f}} - 1)$, no se pone $\mathcal{L}_{\boldsymbol{f}} = \mathcal{L}(d_{\boldsymbol{f}}, d_{\boldsymbol{f}})$ porque se requiere que $\boldsymbol{f}$ sea invertible y un polinomio que satisface $\boldsymbol{f}(1) = 0$ no puede ser invertible, además $\boldsymbol{f} \in \mathcal{L}_{\boldsymbol{f}}$.

$\mathcal{L}_{\boldsymbol{g}}$ Conjunto de polinomios que satisfacen la condición $d_{\boldsymbol{g}}$, $\mathcal{L}_{\boldsymbol{g}} \subset \mathbb{Z}^N$. $\mathcal{L}(d_{\boldsymbol{g}}, d_{\boldsymbol{g}})$, además $\boldsymbol{g} \in \mathcal{L}_{\boldsymbol{g}}$.

$\mathcal{L}_{\boldsymbol{r}}$ Conjunto de polinomios que satisfacen la condición $d_{\boldsymbol{r}}$, $\mathcal{L}_{\boldsymbol{r}} \subset \mathbb{Z}^N$. $\mathcal{L}(d_{\boldsymbol{r}}, d_{\boldsymbol{r}})$, además $\boldsymbol{r} \in \mathcal{L}_{\boldsymbol{r}}$.

$\mathcal{L}_{\boldsymbol{c}}$ Conjunto de polinomios que satisfacen la condición $d_{\boldsymbol{c}}$.

$\mathcal{S}$ Conjunto de valores $\{a_1, a_2, \ldots, a_{\frac{N}{2}}\} \pmod q$, multiplicativamente cerrado respecto a los inversos.

$\mathcal{S}_{\boldsymbol{m}} = \{0, 1, \ldots, p-1\}^N$ conjunto de mensajes $\boldsymbol{m}$ permisibles.

$\mathbf{B}_{\boldsymbol{h}}$ Una cota para la norma del polinomio clave pública $\boldsymbol{h}$.

Además de $s,\ t,\ t_1,\ t_2 \in \mathbb{Z}$ los cuales no necesariamente se mantienen en secreto.

Se puede tener $\mathcal{L}_{\boldsymbol{f}} = \mathcal{L}_{\boldsymbol{g}} = \mathcal{L}_{\boldsymbol{r}}$ y en tal caso tendríamos la colección de todos los $\binom{N}{d}$ $N-$vectores, y para aumentar la seguridad se requiere que $(p, q) = 1$.

El espacio de mensajes $\mathcal{L}_{\boldsymbol{m}}$ consiste de todos los polinomios $(\mathrm{mod}\,(m))$. Suponiendo m impar, es más conveniente tomar

$$\mathcal{L}_{\boldsymbol{m}} = \left\{ \boldsymbol{m} \in \boldsymbol{\mathcal{R}} \;\middle|\; \boldsymbol{m} \text{ tiene coeficientes en } \left[-\frac{m-1}{2}, \frac{m-1}{2} \right] \right\}.$$

La tabla siguiente proporciona un conjunto de parámetros de NTRU a varios niveles de seguridad, esto da una idea de las cantidades usadas en aplicaciones comerciales.

	N	q	p	$d_{\boldsymbol{f}}$	$d_{\boldsymbol{g}}$	$d_{\boldsymbol{r}}$
Seguridad Muy Alta	503	256	3	216	72	55
Seguridad Alta	347	128	3	61	20	18
Seguridad Media	251	128	3	61	20	18
Seguridad Moderada	167	128	3	61	20	18
Seguridad Baja	107	64	3	15	12	5
Ejemplo	11	32	3	4	3	3

Se define el Ancho de un elemento $\boldsymbol{f} \in \boldsymbol{\mathcal{R}}$ como

$$\|\boldsymbol{f}\|_\infty = \max\{a_i \mid 0 \leq i \leq N-1\} - \min\{a_i \mid 0 \leq i \leq N-1\}$$

Como sugiere la notación, este es un tipo de norma sobre $\boldsymbol{\mathcal{R}}$. Similarmente se define la Norma Centrada sobre $\boldsymbol{\mathcal{R}}$ por

$$\begin{aligned} \|\boldsymbol{f}\| &= \left(\sum_{i=0}^{N-1} (a_i - \overline{\boldsymbol{f}})^2 \right)^{\frac{1}{2}} \\ &= \left(\sum_{i=0}^{N-1} (a_i)^2 - \frac{1}{N} (\overline{\boldsymbol{f}})^2 \right)^{\frac{1}{2}} \end{aligned}$$

donde

$$\overline{\boldsymbol{f}} = \frac{1}{N} \sum_{i=0}^{N-1} a_i.$$

Equivalentemente, $\dfrac{\|\boldsymbol{f}\|}{\sqrt{N}}$ es la desviación estándar de los coeficientes de $\boldsymbol{f}$.

La norma es casi multiplicativa, *i. e.*, dados $\boldsymbol{f},\ \boldsymbol{g} \in \boldsymbol{\mathcal{R}}$,

$$||\boldsymbol{f} \circledast \boldsymbol{g}|| \approx ||\boldsymbol{f}||\ ||\boldsymbol{g}||.$$

Se tiene

$$\|\boldsymbol{f}\| = \sqrt{2d_{\boldsymbol{f}} - 1 - N^{-1}}, \quad \|\boldsymbol{g}\| = \sqrt{2d_{\boldsymbol{g}}}, \quad \|\boldsymbol{r}\| = \sqrt{2d_{\boldsymbol{r}}}.$$

Proposición 15 *Para todo $\varepsilon > 0$ existen constantes $\gamma_1,\ \gamma_2 > 0$ dependientes de ε y N, tales que para cualesquiera polinomios $\boldsymbol{f},\ \boldsymbol{g} \in \boldsymbol{\mathcal{R}}$, la probabilidad de que satisfagan*

$$\gamma_1 \|\boldsymbol{f}\|\ \|\boldsymbol{g}\| \leq \|\boldsymbol{f} \circledast \boldsymbol{g}\|_\infty \leq \gamma_2 \|\boldsymbol{f}\|\ \|\boldsymbol{g}\|$$

es mayor que $1 - \varepsilon$.

Esta proposición sería inservible desde el punto de vista práctico si la razón $\frac{\gamma_2}{\gamma_1}$ fuera muy grande para ε pequeños. Sin embargo para valores moderadamente grandes de N y valores muy pequeños de ε, las constantes $\gamma_1,\ \gamma_2$ no están en el extremo.

§5.3 Algoritmo

5.3.1 Creación de claves

Para la creación de la clave pública, $\mathscr{R}$ escoge dos polinomios $\boldsymbol{f} \in \mathcal{L}_{\boldsymbol{f}}$, $\boldsymbol{g} \in \mathcal{L}_{\boldsymbol{g}}$, $\boldsymbol{g}$ no necesariamente invertible y con $\boldsymbol{f}$ tal que tenga inversos $\pmod p$ y $\pmod q$, *i. e.*, que existan $\boldsymbol{F}_p,\ \boldsymbol{F}_q \in \boldsymbol{\mathcal{R}}$ tales que

$$\begin{aligned} \boldsymbol{f} \circledast \boldsymbol{F}_q &\equiv \mathbf{1} \pmod q \\ &\equiv (1, 0, \ldots, 0)^T \pmod q, \\ \boldsymbol{f} \circledast \boldsymbol{F}_p &\equiv \mathbf{1} \pmod p \\ &\equiv (1, 0, \ldots, 0)^T \pmod p. \end{aligned}$$

Estos productos corresponden al polinomio $\mathbf{1}$. Si por alguna razón los inversos no existieran, $\mathscr{R}$ tendría que regresarse y escoger otro $\boldsymbol{f}$.

Luego calcula

$$\boldsymbol{h} \equiv\ p\boldsymbol{F}_q \circledast \boldsymbol{g} \pmod q\ \in \mathbb{Z}_q^N[x].$$

La clave pública de $\mathscr{R}$ es $\boldsymbol{h}$ y su clave privada es $\boldsymbol{f}$ (en la práctica también se mantienen en secreto $\boldsymbol{F}_p$ y $\boldsymbol{g}$, pues si alguien llegara a conocer alguno de ellos, sería capaz de descifrar un mensaje enviado a $\mathscr{R}$).

Note que de la última ecuación

$$\begin{aligned} \boldsymbol{f} \circledast \boldsymbol{h} &\equiv \boldsymbol{f} \circledast\ p\boldsymbol{F}_q \circledast \boldsymbol{g} \\ &\equiv p\boldsymbol{g} \pmod q, \end{aligned}$$

i. e. $\boldsymbol{f}$ y $\boldsymbol{g}$ satisfacen

$$p\boldsymbol{g} \equiv\ \boldsymbol{f} \circledast \boldsymbol{h} \pmod q. \tag{5.1}$$

5.3.2 Cifrado

Si $\mathscr{E}$ desea enviar un mensaje $\mathbf{M}$ a $\mathscr{R}$, pone primero su mensaje en forma de un polinomio $\boldsymbol{m}$ con coeficientes $\pmod p$, digamos en $\left[-\frac{p}{2}, \frac{p}{2}\right]$, ($\boldsymbol{m}$ es un polinomio pequeño mod$q$).

Para cifrarlo toma aleatoriamente un polinomio $\boldsymbol{r} \in \mathcal{L}_r$ y usa la clave pública $\boldsymbol{h}$ de $\mathscr{R}$, para calcular

$$\boldsymbol{e} \equiv \boldsymbol{r} \circledast \boldsymbol{h} + \boldsymbol{m} \pmod q.$$

El polinomio $\boldsymbol{e}$ es el mensaje cifrado que $\mathscr{E}$ envia a $\mathscr{R}$. Se debe seleccionar un $\boldsymbol{r}$ aleatorio por cada mensaje.

5.3.3 Descifrado

Ahora $\mathscr{R}$ ha recibido el mensaje cifrado $\boldsymbol{e}$ de $\mathscr{E}$ y desea descifrarlo, calcula primero

$$\begin{aligned} \boldsymbol{a} &\equiv \boldsymbol{f} \circledast \boldsymbol{e} \pmod q \\ &\equiv \boldsymbol{f} \circledast (\boldsymbol{r} \circledast \boldsymbol{h} + \boldsymbol{m}) \pmod q \\ &\equiv (\boldsymbol{f} \circledast \boldsymbol{h}) \circledast \boldsymbol{r} + \boldsymbol{f} \circledast \boldsymbol{m} \pmod q, \end{aligned}$$

como

$$(\boldsymbol{f} \circledast \boldsymbol{h}) \circledast \boldsymbol{r} = p\boldsymbol{g} \circledast \boldsymbol{r} \pmod q$$

i. e.

$$\boldsymbol{a} \equiv p\boldsymbol{g} \circledast \boldsymbol{r} + \boldsymbol{f} \circledast \boldsymbol{m} \pmod q,$$

donde $\boldsymbol{a}$ es un polinomio con coeficientes en $\left[-\frac{q}{2}, \frac{q}{2}\right]$. Luego ajusta las entradas de $\boldsymbol{a}$ por

$$b_k = a_k + t - \begin{cases} 0 & \text{si } a_k < s \\ q & \text{si } a_k \geq s \end{cases}.$$

Los parámetros se escogen de tal manera que ambos

$$t_1\mathbf{1} + p\boldsymbol{g} \circledast \boldsymbol{r}, \quad t_2\mathbf{1} + \boldsymbol{f} \circledast \boldsymbol{m}$$

sean suficientemente pequeños; garantizar que las entradas de la expresión no modular

$$\boldsymbol{b} = t\mathbf{1} + p\boldsymbol{g} \circledast \boldsymbol{r} + \boldsymbol{f} \circledast \boldsymbol{m}$$

esten en $\left[-\frac{q}{2}, \frac{q}{2}\right]$ la mayoría de las veces. De hecho todas las entradas estan en ese rango. Es muy importante que $\mathscr{R}$ haga esto antes de continuar con el siguiente paso. En seguida $\mathscr{R}$ calcula

$$\boldsymbol{b} = \boldsymbol{a} \pmod p,$$

es decir, reduce cada uno de los coeficientes de $\boldsymbol{a}$ (mod p). Finalmente $\mathscr{R}$ usa su otro polinomio privado $\boldsymbol{F}_p$ para calcular

$$\boldsymbol{c} = \boldsymbol{F}_p \circledast \boldsymbol{b} \pmod{p}.$$

Esto elimina la dependencia sobre el $\boldsymbol{r}$ desconocido, y el polinomio $\boldsymbol{c}$ será el mensaje original $\boldsymbol{m}$ de $\mathscr{E}$.

5.3.4 Ejemplo

Creación de claves

$\mathscr{R}$ necesita escoger un polinomio $\boldsymbol{f}$ de grado 10 con cuatro 1's y tres -1's, y necesita escoger un polinomio $\boldsymbol{g}$ de grado 10 con tres 1's y tres -1's, digamos

$$\begin{aligned} \boldsymbol{f} &= -1 + x + x^2 - x^4 + x^6 + x^9 - x^{10}, \\ \boldsymbol{g} &= -1 + x^2 + x^3 + x^5 - x^8 - x^{10}. \end{aligned}$$

En seguida $\mathscr{R}$ calcula $\boldsymbol{F}_p$ y $\boldsymbol{F}_q$,

$$\begin{aligned} \boldsymbol{F}_p &= \boldsymbol{F}_3 \\ &= 1 + 2x + 2x^3 + 2x^4 + x^5 + 2x^7 + x^8 + 2x^9, \\ \boldsymbol{F}_q &= \boldsymbol{F}_{32} \\ &= 5 + 9x + 6x^2 + 16x^3 + 4x^4 + 15x^5 + \\ &\quad 16x^6 + 22x^7 + 20x^8 + 18x^9 + 30x^{10}. \end{aligned}$$

Por último calcula

$$\begin{aligned} \boldsymbol{h} &= p\boldsymbol{F}_q \circledast \boldsymbol{g} \\ &= 3\boldsymbol{F}_{32} \circledast \boldsymbol{g} \\ &= 8 + 25x + 22x^2 + 20x^3 + 12x^4 + 24x^5 + \\ &\quad 15x^6 + 19x^7 + 12x^8 + 19x^9 + 16x^{10} \pmod{32}. \end{aligned}$$

Cifrado

Si $\mathscr{E}$ desea enviar el mensaje

$$\boldsymbol{m} = -1 + x^3 - x^4 - x^8 + x^9 + x^{10}$$

a $\mathscr{R}$ usando la clave pública $\boldsymbol{h}$ de éste, primero escoge aleatoriamente un polinomio $\boldsymbol{r}$ de grado 10 con tres 1's y tres -1's, digamos

$$\boldsymbol{r} = -1 + x^2 + x^3 + x^4 - x^5 - x^7.$$

El mensaje cifrado $\boldsymbol{e}$ es

$$\begin{aligned} \boldsymbol{e} &= \boldsymbol{r} \circledast \boldsymbol{h} + \boldsymbol{m} \\ &= 14 + 11x + 26x^2 + 24x^3 + 14x^4 + 16x^5 + \\ &\quad 30x^6 + 7x^7 + 25x^8 + 6x^9 + 19x^{10} \pmod{32} \end{aligned}$$

que es el que envía a $\mathscr{R}$.

Descifrado

$\mathscr{R}$ ha recibido el mensaje cifrado $\boldsymbol{e}$ de $\mathscr{R}$. Ahora usa su clave privada $\boldsymbol{f}$ para calcular

$$\begin{aligned} \boldsymbol{a} &= \boldsymbol{f} \circledast \boldsymbol{e} \pmod q \\ &= 3 - 7x - 10x^2 - 11x^3 + 10x^4 + 7x^5 + \\ &\quad 6x^6 + 7x^7 + 5x^8 - 3x^9 - 7x^{10} \pmod{32}. \end{aligned}$$

Cuando $\mathscr{R}$ reduce los coeficientes de $\boldsymbol{f} \circledast \boldsymbol{e} \pmod{32}$, escoge valores entre -15 y 16, y no entre 0 y 31. Es muy importante que escoja los coeficientes de esta manera.

En seguida $\mathscr{R}$ reducen los coeficientes de $\boldsymbol{a} \pmod 3$

$$\begin{aligned} \boldsymbol{b} &= \boldsymbol{a} \pmod 3 \\ &= -x - x^2 + x^3 + x^4 + x^5 + x^7 - x^8 - x^{10} \pmod 3. \end{aligned}$$

Finalmente usa $\boldsymbol{F}_p = \boldsymbol{F}_3$, la otra parte de su clave privada, para calcular

$$\begin{aligned} \boldsymbol{c} &= \boldsymbol{F}_3 \circledast \boldsymbol{b} \pmod 3 \\ &= -1 + x^3 - x^4 - x^8 + x^9 + x^{10} \pmod 3. \end{aligned}$$

El polinomio $\boldsymbol{c}$ es el mensaje original $\boldsymbol{m}$ de $\mathscr{E}$, por lo que $\mathscr{R}$ ha descifrado el mensaje $\boldsymbol{e}$ de $\mathscr{E}$ exitosamente.

5.3.5 ¿Por qué funciona?

El mensaje cifrado $\boldsymbol{e}$ de $\mathscr{E}$ tiene la forma

$$\boldsymbol{e} = \boldsymbol{r} \circledast \boldsymbol{h} + \boldsymbol{m} \pmod q,$$

pero, claro $\mathscr{R}$ inicialmente no conoce los valores de $\boldsymbol{r}$ y de $\boldsymbol{m}$. Así que el primer paso de $\mathscr{R}$ es calcular $\boldsymbol{f} \circledast \boldsymbol{e}$ y reducir los coeficientes $\pmod q$. Recuerde que la clave pública $\boldsymbol{h}$ de $\mathscr{R}$ se obtuvo al multiplicar $p\boldsymbol{F}_q \circledast \boldsymbol{g}$ y después reducir sus coeficientes $\pmod q$. Así que aunque $\mathscr{R}$ no conozca $\boldsymbol{r}$ y $\boldsymbol{m}$, cuando calcula $\boldsymbol{a} = \boldsymbol{f} \circledast \boldsymbol{e} \pmod q$, desarrolla de hecho el siguiente cálculo

$$\begin{aligned} & & \boldsymbol{a} &= \boldsymbol{f} \circledast \boldsymbol{e} \pmod q \\ \text{sustituyendo} & & \boldsymbol{e} &= \boldsymbol{r} \circledast \boldsymbol{h} + \boldsymbol{m} \pmod q \\ & & &= \boldsymbol{f} \circledast (\boldsymbol{r} \circledast \boldsymbol{h} + \boldsymbol{m}) \pmod q \\ \text{como} & & \boldsymbol{h} &= p\,\boldsymbol{F}_q \circledast \boldsymbol{g} \pmod q \\ & & &= \boldsymbol{f} \circledast (\boldsymbol{r} \circledast p\boldsymbol{F}_q \circledast \boldsymbol{g} + \boldsymbol{m}) \pmod q \\ & & &= p\,\boldsymbol{r} \circledast \boldsymbol{g} + \boldsymbol{f} \circledast \boldsymbol{m} \pmod q \\ \text{pues} & \boldsymbol{f} \circledast \boldsymbol{F}_q &= \boldsymbol{1} \pmod q. \end{aligned}$$

Los polinomios $\boldsymbol{r}$, $\boldsymbol{g}$, $\boldsymbol{f}$ y $\boldsymbol{m}$ tienen coeficientes suficientemente pequeños (0's, 1's y -1's). Esto significa que los coeficientes de los productos $\boldsymbol{r} \circledast \boldsymbol{g}$ y $\boldsymbol{f} \circledast \boldsymbol{m}$ serán también suficientemente pequeños, por lo menos en comparación con q. Similarmente, el primo p es también pequeño comparado con q. Lo que reditúa en que, suponiendo que los parámetros han sido escogidos de manera apropiada, los coeficientes del polinomio $p\boldsymbol{r} \circledast \boldsymbol{g} + \boldsymbol{f} \circledast \boldsymbol{m}$ estarán en $\left[-\frac{q}{2}, \frac{q}{2}\right]$, así que reducir los coeficientes $\pmod q$ no tendrá efecto en nada.

En otras palabras, cuando $\mathscr{R}$ calcula $\boldsymbol{f} \circledast \boldsymbol{e} \pmod q$ (reduciendo los coeficientes dentro del intervalo $\left[-\frac{q}{2}, \frac{q}{2}\right]$), el polinomio $\boldsymbol{a}$ con el que termina es exactamente igual a el polinomio

$$\boldsymbol{a} = p\,\boldsymbol{r} \circledast \boldsymbol{g} + \boldsymbol{f} \circledast \boldsymbol{m} \in \mathbb{Z}^N[x].$$

Por lo que, cuando $\mathscr{R}$ en seguida reduce los coeficientes de $\boldsymbol{a} \pmod p$ para formar el polinomio $\boldsymbol{b}$, realmente está reduciendo los coeficientes de $p\,\boldsymbol{r} \circledast \boldsymbol{g} + \boldsymbol{f} \circledast \boldsymbol{m} \pmod p$, así que el $\boldsymbol{b}$ que obtiene es igual a

$$\boldsymbol{b} = \boldsymbol{f} \circledast \boldsymbol{m} \pmod p.$$

No olvide que $\mathscr{R}$ aún no conoce el valor de $\boldsymbol{m}$, pero conoce el valor de $\boldsymbol{b}$. Por lo que el paso final es multiplicar $\boldsymbol{b}$ por $\boldsymbol{F}_p$ y usar el hecho de que $\boldsymbol{F}_p \circledast \boldsymbol{f} \equiv \boldsymbol{1} \pmod p$ para calcular

$$\begin{aligned} \boldsymbol{c} &= \boldsymbol{F}_p \circledast \boldsymbol{b} \\ &= \boldsymbol{F}_p \circledast \boldsymbol{f} \circledast \boldsymbol{m} \\ &= \boldsymbol{m} \pmod p, \end{aligned}$$

lo que le permite recuperar el mensaje $\boldsymbol{m}$ de $\mathscr{E}$.

§5.4 Criterio de descifrado

Para que el proceso de descifrado funcione, es necesario que

$$\|p\ \boldsymbol{r} \circledast \boldsymbol{g} + \boldsymbol{f} \circledast \boldsymbol{m}\|_\infty < q.$$

Esto será cierto si se escogen parámetros tales que

$$\|\boldsymbol{f} \circledast \boldsymbol{m}\|_\infty \leq \frac{q}{4}, \quad \|p\ \boldsymbol{r} \circledast \boldsymbol{g}\|_\infty \leq \frac{q}{4},$$

y en vista de la proposición 15, esto sugiere tomar

$$\|\boldsymbol{f}\|\,\|\boldsymbol{m}\| \approx \frac{q}{4\gamma_2}, \quad \|\boldsymbol{r}\|\,\|\boldsymbol{g}\| \approx \frac{q}{4\gamma_2} \tag{5.2}$$

para una γ_2 correspondiente a valores pequeños de ε.

§5.5 Esquema de firma y autenticación polinomial (EFAP)

5.5.1 Parámetros

La tabla siguiente proporciona un conjunto de parámetros de EFAP.

	N	q	$d_{\boldsymbol{f}}$	$d_{\boldsymbol{g}}$	$d_{\boldsymbol{c}}$	$\mathbf{B}_{\boldsymbol{h}}$
Alta Seguridad	768	769	256	256	1	2.2
Ejemplo	16	17	5	5	1	2.2

También para el ejemplo tomaremos

$$\mathcal{S} = \{3,\ 5,\ 6,\ 7,\ 8,\ 10,\ 12,\ 15\} \pmod{17}.$$

Cada elemento de S tiene un inverso multiplicativo en S

$$3 \cdot 6 = 5 \cdot 7 = 8 \cdot 15 = 10 \cdot 12 = 1 \pmod{17}.$$

Aunque EFAP funciona aún si $\mathcal{S}$ no tiene esta propiedad, la seguridad del sistema se puede ver comprometida.

5.5.2 Creación de claves

$\mathscr{E}$ selecciona aleatoriamente 2 polinomios $\boldsymbol{f}_1,\ \boldsymbol{f}_2\ \in \mathcal{L}_{\boldsymbol{f}}$ y los mantiene en secreto, de lo contrario alguien podría pretender ser $\mathscr{E}$.

El siguiente paso de $\mathscr{E}$ es evaluar $\boldsymbol{f}_1$ y $\boldsymbol{f}_2$ en cada uno de los elementos de el conjunto $\mathcal{S}$

$$\begin{aligned} \boldsymbol{f}[\mathcal{S}] &= \{\boldsymbol{f}_1[\mathcal{S}], \boldsymbol{f}_2[\mathcal{S}]\} \\ &= \{\boldsymbol{f}_1(a_1), \boldsymbol{f}_1(a_2), \ldots, \boldsymbol{f}_1(a_{\frac{N}{2}}), \\ &\quad \boldsymbol{f}_2(a_1), \boldsymbol{f}_2(a_2), \ldots, \boldsymbol{f}_2(a_{\frac{N}{2}})\} \pmod{q}. \end{aligned}$$

El conjunto de $\boldsymbol{f}[\mathcal{S}]$ es la clave pública de autenticación de $\mathscr{E}$, y $\boldsymbol{f}_1,\ \boldsymbol{f}_2$ son su clave privada.

5.5.3 Autenticación

Si $\mathscr{E}$ quiere probar su autenticidad a $\mathscr{R}$ usando EFAP, tendrá que realizar los siguientes 4 pasos.

Convenio

$\mathscr{E}$ selecciona aleatoriamente 2 polinomios $\boldsymbol{g}_1,\ \boldsymbol{g}_2\ \in \mathcal{L}_{\boldsymbol{g}}$ y los evalúa en cada uno de los elementos del conjunto $\mathcal{S}$

$$\begin{aligned} \boldsymbol{g}[\mathcal{S}] &= \{\boldsymbol{g}_1[\mathcal{S}], \boldsymbol{g}_2[\mathcal{S}]\} \\ &= \{\boldsymbol{g}_1(a_1), \boldsymbol{g}_1(a_2), \ldots, \boldsymbol{g}_1(a_{\frac{N}{2}}), \\ &\quad \boldsymbol{g}_2(a_1), \boldsymbol{g}_2(a_2), \ldots, \boldsymbol{g}_2(a_{\frac{N}{2}})\} \pmod{q}. \end{aligned}$$

$\mathscr{E}$ mantiene en secreto $\boldsymbol{g}_1$ y $\boldsymbol{g}_2$, y envía el conjunto de valores $\boldsymbol{g}[\mathcal{S}]$ a $\mathscr{R}$, el cual es el convenio de $\mathscr{E}$.

Reto

$\mathscr{R}$ selecciona aleatoriamente 4 polinomios $\boldsymbol{c}_{11},\ \boldsymbol{c}_{12},\ \boldsymbol{c}_{21},\ \boldsymbol{c}_{22}\ \in \mathcal{L}_{\boldsymbol{c}}$ para luego enviarlos a $\mathscr{E}$. Estos 4 polinomios son el reto de $\mathscr{R}$.

Respuesta

$\mathscr{E}$ calcula y envía a $\mathscr{R}$ el polinomio

$$\begin{aligned}\boldsymbol{h} \quad = \quad & \boldsymbol{c}_{11} \circledast \boldsymbol{f}_1 \circledast \boldsymbol{g}_1 + \boldsymbol{c}_{12} \circledast \boldsymbol{f}_1 \circledast \boldsymbol{g}_2 + \\ & \boldsymbol{c}_{21} \circledast \boldsymbol{f}_2 \circledast \boldsymbol{g}_1 + \boldsymbol{c}_{22} \circledast \boldsymbol{f}_2 \circledast \boldsymbol{g}_2 \pmod q.\end{aligned}$$

El cual es la respuesta de $\mathscr{E}$.

Verificación

$\mathscr{R}$ checa que la respuesta de $\mathscr{E}$ satisfaga las siguientes 2 condiciones

1. $\mathscr{R}$ escribe $\boldsymbol{h}$ como

$$\boldsymbol{h} = h_0 + h_1 x + h_2 x^2 + \cdots + h_{N-1} x^{N-1}$$

con $h_i \in \left[-\frac{q}{2}, \frac{q}{2}\right]$, y entonces checa que

$$h_0^2 + h_1^2 + h_2^2 + \cdots + h_{N-1}^2 < (\mathbf{B}_{\boldsymbol{h}} q)^2.$$

2. Para cada $a \in \mathcal{S}$, $\mathscr{R}$ checa que

$$\begin{aligned}\boldsymbol{h}(a) \quad = \quad & \boldsymbol{c}_{11}(a) \circledast \boldsymbol{f}_1(a) \circledast \boldsymbol{g}_1(a) + \boldsymbol{c}_{12}(a) \circledast \boldsymbol{f}_1(a) \circledast \boldsymbol{g}_2(a) + \\ & \boldsymbol{c}_{21}(a) \circledast \boldsymbol{f}_2(a) \circledast \boldsymbol{g}_1(a) + \boldsymbol{c}_{22}(a) \circledast \boldsymbol{f}_2(a) \circledast \boldsymbol{g}_2(a) \pmod q.\end{aligned}$$

$\mathscr{R}$ puede checar la condición (2) porque conoce los conjuntos de valores $\boldsymbol{f}[\mathcal{S}]$ y $\boldsymbol{g}[\mathcal{S}]$, y claro conoce los $\boldsymbol{c}_{ij}$ polinomios, aunque no conozca los polinomios $\boldsymbol{f}$ y $\boldsymbol{g}$.

Si la respuesta $\boldsymbol{h}$ de $\mathscr{E}$ pasa las pruebas (1) y (2), entonces $\mathscr{R}$ acepta que $\mathscr{E}$ ha probado su autenticidad.

NOTA: Para prevenir que $\mathscr{R}$ obtenga información acerca de la clave de $\mathscr{E}$, es mejor crear el reto como sigue. $\mathscr{R}$ escoge una cadena aleatoria de bits $\mathbf{B}$ de una longitud específica, por ejemplo 80 bits. $\mathscr{R}$ envía $\mathbf{B}$ a $\mathscr{E}$. Entonces $\mathscr{R}$ y $\mathscr{E}$ "alimentan" esta cadena de bits a una función de digestión para crear los 4 polinomios $\boldsymbol{c}_{11},\ \boldsymbol{c}_{12},\ \boldsymbol{c}_{21},\ \boldsymbol{c}_{22}$. Esto previene a $\mathscr{R}$ de escoger los polinomios reto

con alguna propiedad especial, pues no tiene manera de predecir la salida de la función de digestión. Note la similitud en la manera en que un esquema de autenticación como lo es EFAP es convertido a un esquema de firmas digitales. Para hacer esto, se alimenta un mensaje $\boldsymbol{m}$ (y un convenio) en una función de digestión para producir un reto, es cual es a su vez usado para crear una respuesta.

5.5.4 Firmas digitales

Cualquier esquema de autenticación, del tipo convenio, reto, respuesta puede ser transformado, con la adición de una función de resumen, en un esquema de firma digital.

Para usar EFAP en la creación de una firma digital, se asume que se ha especificado una función de digestión $\mathbf{H}$ que toma una secuencia arbitraria de bits como entrada y produce como salida una cadena de 80 bits. También se usa una función de formateado $\mathbf{F}$ que toma la cadena de 80 bits y la transforma en un conjunto de válido de polinomios reto

$$\boldsymbol{c} = (\boldsymbol{c}_{11},\ \boldsymbol{c}_{12},\ \boldsymbol{c}_{21},\ \boldsymbol{c}_{22}).$$

Definición 38 *Aún más, si* $\mathbf{B}$ *y* $\mathbf{C}$ *son cadenas de bits, se escribe* $\mathbf{B}|\mathbf{C}$ *para denotar la* CONCATENACIÓN *de* $\mathbf{B}$ *y* $\mathbf{C}$.

Si $\mathscr{E}$ quiere firmar un mensaje $\boldsymbol{m}$ usando su clave de autenticación $\boldsymbol{f} = (\boldsymbol{f}_1, \boldsymbol{f}_2)$.

- $\mathscr{E}$ toma los polinomios de convenio $\boldsymbol{g} = (\boldsymbol{g}_1, \boldsymbol{g}_2)$ aleatoriamente y calcula su convenio $\boldsymbol{g}[\mathcal{S}]$.
- $\mathscr{E}$ construye los 4 polinomios reto $\boldsymbol{c} = (\boldsymbol{c}_{11},\ \boldsymbol{c}_{12},\ \boldsymbol{c}_{21},\ \boldsymbol{c}_{22})$ calculando $\boldsymbol{c} = \mathbf{F}(\mathbf{H}(\boldsymbol{g}[\mathbf{S}]|\boldsymbol{m}))$. Es decir, $\mathscr{E}$ simula un reto digeriendo juntos el convenio y el mensaje y usando el resultado para crear un polinomio reto.
- $\mathscr{E}$ calcula la respuesta $\boldsymbol{h}$ para el convenio $\boldsymbol{g}$ y para el reto $\boldsymbol{c}$, usando su clave privada $\boldsymbol{f}$, de la manera usual

$$\begin{aligned} \boldsymbol{h} \quad = \quad & \boldsymbol{c}_{11} \circledast \boldsymbol{f}_1 \circledast \boldsymbol{g}_1 + \boldsymbol{c}_{12} \circledast \boldsymbol{f}_1 \circledast \boldsymbol{g}_2 + \\ & \boldsymbol{c}_{21} \circledast \boldsymbol{f}_2 \circledast \boldsymbol{g}_1 + \boldsymbol{c}_{22} \circledast \boldsymbol{f}_2 \circledast \boldsymbol{g}_2 \pmod{q}. \end{aligned}$$

- El mensaje firmado de $\mathscr{E}$ es entonces el mensaje $\boldsymbol{m}$ seguido por la firma $(\boldsymbol{g}[\mathcal{S}], \boldsymbol{h})$.
- Para verificar que $\mathscr{E}$ firmó el mensaje $\boldsymbol{m}$, $\mathscr{R}$ calcula $\boldsymbol{c}$ a partir de $\boldsymbol{g}[\mathcal{S}]$ y $\boldsymbol{m}$ usando la función de resumen públicamente disponible $\mathbf{H}$ y la función de formateado $\mathbf{F}$. Entonces usa la clave pública de $\mathscr{E}$ $\boldsymbol{f}[\mathcal{S}]$ para verificar que la respuesta $\boldsymbol{h}$ fue generada por $\mathscr{E}$ *i. e.* por alguien con conocimiento de la clave privada $\boldsymbol{f}$. Esto verifica la firma de $\mathscr{E}$ sobre el mensaje $\boldsymbol{m}$.

5.5.5 Ejemplo

Creación de Claves

$\mathscr{E}$ crea su clave escogiendo aleatoriamente 2 polinomios $\boldsymbol{f}_1,\ \boldsymbol{f}_2$ con 5 coeficientes iguales 1 y a -1, y el resto a 0, digamos

$$\begin{aligned}\boldsymbol{f}_1 &= 1 - x + x^2 - x^3 + x^6 - x^7 + x^8 - x^{12} + x^{14} - x^{15},\\ \boldsymbol{f}_2 &= -x^2 - x^5 - x^6 - x^7 + x^9 + x^{10} + x^{11} + x^{13} + x^{14} - x^{15}.\end{aligned}$$

Calcula el valor de $\boldsymbol{f}_1$ y $\boldsymbol{f}_2$ en cada uno de los valores de $\mathcal{S}$

$$\begin{aligned}\boldsymbol{f}_1[\mathcal{S}] &= \{\boldsymbol{f}_1(3), \boldsymbol{f}_1(5), \boldsymbol{f}_1(6), \boldsymbol{f}_1(7), \boldsymbol{f}_1(8), \boldsymbol{f}_1(10), \boldsymbol{f}_1(12), \boldsymbol{f}_1(15)\}\\ &= \{9, 10, 5, 9, 1, 12, 15, 10\} \pmod{17},\\ \boldsymbol{f}_2[\mathcal{S}] &= \{\boldsymbol{f}_2(3), \boldsymbol{f}_2(5), \boldsymbol{f}_2(6), \boldsymbol{f}_2(7), \boldsymbol{f}_2(8), \boldsymbol{f}_2(10), \boldsymbol{f}_2(12), \boldsymbol{f}_2(15)\}\\ &= \{14, 9, 16, 5, 14, 1, 2, 8\} \pmod{17}.\end{aligned}$$

El conjunto

$$\boldsymbol{f}[\mathcal{S}] = (\boldsymbol{f}_1[\mathcal{S}], \boldsymbol{f}_2[\mathcal{S}])$$

es clave pública de autenticación de $\mathscr{E}$.

Convenio

Si $\mathscr{E}$ quiere probar su identidad a $\mathscr{R}$. Para ello selecciona aleatoriamente 2 polinomios $\boldsymbol{g}_1,\ \boldsymbol{g}_2 \in \mathcal{L}_5$, con 5 coeficientes iguales a 1, 5 coeficientes iguales a -1 y el resto a 0, digamos

$$\begin{aligned}\boldsymbol{g}_1 &= x^2 + x^3 - x^4 + x^5 - x^7 + x^8 - x^9 - x^{10} - x^{12} + x^{13},\\ \boldsymbol{g}_2 &= x + x^4 + x^5 - x^6 + x^7 - x^8 - x^9 - x^{11} - x^{13} + x^{15}.\end{aligned}$$

$\mathscr{E}$ mantiene en secreto $\boldsymbol{g}_1$ y $\boldsymbol{g}_2$, pero evalúa $\boldsymbol{g}_1$ y $\boldsymbol{g}_2$ en los elementos de $\mathcal{S}$ y envía esos valores a $\mathscr{R}$

$$\begin{aligned}\boldsymbol{g}_1[\mathcal{S}] &= \{\boldsymbol{g}_1(3), \boldsymbol{g}_1(5), \boldsymbol{g}_1(6), \boldsymbol{g}_1(7), \boldsymbol{g}_1(8), \boldsymbol{g}_1(10), \boldsymbol{g}_1(12), \boldsymbol{g}_1(15)\}\\ &= \{2, 16, 7, 10, 0, 14, 14, 10\} \pmod{17},\\ \boldsymbol{g}_2[\mathcal{S}] &= \{\boldsymbol{g}_2(3), \boldsymbol{g}_2(5), \boldsymbol{g}_2(6), \boldsymbol{g}_2(7), \boldsymbol{g}_2(8), \boldsymbol{g}_2(10), \boldsymbol{g}_2(12), \boldsymbol{g}_2(15)\}\\ &= \{8, 5, 1, 1, 5, 8, 2, 9\} \pmod{17}.\end{aligned}$$

Reto

$\mathscr{R}$ reta entonces a $\mathscr{E}$ enviándole sus 4 polinomios $\boldsymbol{c}_{11},\ \boldsymbol{c}_{12},\ \boldsymbol{c}_{21},\ \boldsymbol{c}_{22} \in \mathcal{L}$ seleccionados aleatoriamente con 1 coeficiente igual a 1 y otro igual a -1, y resto de los coeficientes iguales a 0, digamos

$$\boldsymbol{c}_{11} = 1 - x^{13},\ \boldsymbol{c}_{12} = -x^2 + x^4,\ \boldsymbol{c}_{21} = -x^2 + x^{15},\ \boldsymbol{c}_{22} = -x^7 + x^{13}.$$

Respuesta

$\mathscr{E}$ usa los polinomios reto $\boldsymbol{c}_{ij}$ de $\mathscr{R}$ y los polinomios $\boldsymbol{f}_i$ y $\boldsymbol{g}_j$ que solo él conoce para calcular la cantidad

$$\begin{aligned}\boldsymbol{h} &= \boldsymbol{c}_{11} \circledast \boldsymbol{f}_1 \circledast \boldsymbol{g}_1 + \boldsymbol{c}_{12} \circledast \boldsymbol{f}_1 \circledast \boldsymbol{g}_2 + \boldsymbol{c}_{21} \circledast \boldsymbol{f}_2 \circledast \boldsymbol{g}_1 + \boldsymbol{c}_{22} \circledast \boldsymbol{f}_2 \circledast \boldsymbol{g}_2 \\ &= -2 - 4x - 10x^2 - 9x^3 - 15x^4 - x^5 - 2x^6 - 5x^7 + x^8 + x^9 + \\ &\quad 9x^{10} + 15x^{11} + 2x^{12} + 16x^{13} + x^{14} + 3x^{15} \pmod{17}.\end{aligned}$$

Note que para calcular esta cantidad, $\mathscr{E}$ siempre pone $x^{16} = 1$, $x^{17} = x$, y así sucesivamente.

Verificación

$\mathscr{R}$ suma los cuadrados de los coeficientes de $\boldsymbol{h}$ y encuentra que

$$h_0^2 + h_1^2 + h_2^2 + \cdots + h_{N-1}^2 = 1034.$$

Como $(\mathbf{B}_{\boldsymbol{h}} q)^2 = 1398.6$ se tiene que

$$\sum_{i=0}^{N-1} h_i^2 = 1034 < 1398.6 = (\mathbf{B}_{\boldsymbol{h}} q)^2,$$

es decir, la respuesta de $\mathscr{E}$ pasa la prueba (1). En seguida $\mathscr{R}$ calcula los valores

$$\begin{aligned}&\boldsymbol{h}(a) \pmod{q}, \\ &\boldsymbol{c}_{11}(a) \circledast \boldsymbol{f}_1(a) \circledast \boldsymbol{g}_1(a) + \boldsymbol{c}_{12}(a) \circledast \boldsymbol{f}_1(a) \circledast \boldsymbol{g}_2(a) + \\ &\boldsymbol{c}_{21}(a) \circledast \boldsymbol{f}_2(a) \circledast \boldsymbol{g}_1(a) + \boldsymbol{c}_{22}(a) \circledast \boldsymbol{f}_2(a) \circledast \boldsymbol{g}_2(a) \pmod{q}\end{aligned}$$

para cada $a \in \mathcal{S}$ y checa si son iguales. Por ejemplo, si $a = 7$

$$\begin{aligned}\boldsymbol{h}(7) &= 9 \pmod{17}, \\ &\quad \boldsymbol{c}_{11}(7) \circledast \boldsymbol{f}_1(7) \circledast \boldsymbol{g}_1(7) + \boldsymbol{c}_{12}(7) \circledast \boldsymbol{f}_1(7) \circledast \boldsymbol{g}_2(7) + \\ &\quad \boldsymbol{c}_{21}(7) \circledast \boldsymbol{f}_2(7) \circledast \boldsymbol{g}_1(7) + \boldsymbol{c}_{22}(7) \circledast \boldsymbol{f}_2(7) \circledast \boldsymbol{g}_2(7) \\ &= 9 \pmod{17}.\end{aligned}$$

Al calcular esta última cantidad, no fue necesario los $\boldsymbol{f}_i$ y los $\boldsymbol{g}_j$ como polinomios; sólo fue necesario conocer sus valores en $a = 7$. Un cálculo similar para los otros a's en $\mathcal{S}$ revelan que la respuesta de $\mathscr{E}$ pasa la prueba (2), así que $\mathscr{R}$ acepta que $\mathscr{E}$ ha probado su autenticidad.

5.5.6 ¿Por qué funciona?

La respuesta $\boldsymbol{h}$ de $\mathscr{E}$ pasará la prueba (1) porque lo polinomios $\boldsymbol{f}$, $\boldsymbol{g}$ y $\boldsymbol{c}$ todos tienen coeficientes muy pequeños. Así el polinomio

$$\begin{aligned}\boldsymbol{h} &= \boldsymbol{c}_{11} \circledast \boldsymbol{f}_1 \circledast \boldsymbol{g}_1 + \boldsymbol{c}_{12} \circledast \boldsymbol{f}_1 \circledast \boldsymbol{g}_2 + \\ &\quad \boldsymbol{c}_{21} \circledast \boldsymbol{f}_2 \circledast \boldsymbol{g}_1 + \boldsymbol{c}_{22} \circledast \boldsymbol{f}_2 \circledast \boldsymbol{g}_2 \pmod{q}.\end{aligned}$$

tendrá también coeficientes pequeños, por lo que con una apropiada elección de parámetros (tal como la cota de la norma $\mathbf{B}_{\boldsymbol{h}}$), $\boldsymbol{h}$ seguramente pasará la prueba (1). Por otra parte, dado que $\boldsymbol{h}$ es de hecho igual a

$$\begin{aligned}&\boldsymbol{c}_{11} \circledast \boldsymbol{f}_1 \circledast \boldsymbol{g}_1 + \boldsymbol{c}_{12} \circledast \boldsymbol{f}_1 \circledast \boldsymbol{g}_2 + \\ &\boldsymbol{c}_{21} \circledast \boldsymbol{f}_2 \circledast \boldsymbol{g}_1 + \boldsymbol{c}_{22} \circledast \boldsymbol{f}_2 \circledast \boldsymbol{g}_2 \pmod q.\end{aligned}$$

la igualdad

$$\begin{aligned}\boldsymbol{h}(a) \;=\; &\boldsymbol{c}_{11}(a) \circledast \boldsymbol{f}_1(a) \circledast \boldsymbol{g}_1(a) + \boldsymbol{c}_{12}(a) \circledast \boldsymbol{f}_1(a) \circledast \boldsymbol{g}_2(a) + \\ &\boldsymbol{c}_{21}(a) \circledast \boldsymbol{f}_2(a) \circledast \boldsymbol{g}_1(a) + \boldsymbol{c}_{22}(a) \circledast \boldsymbol{f}_2(a) \circledast \boldsymbol{g}_2(a) \pmod q.\end{aligned}$$

será verdadera para todos los valores de $a \pmod q$. En particular, será verdadera $\forall\ a \in \mathcal{S}$, así que la respuesta de $\mathscr{E}$ también pasará la prueba (2).

§5.6 Esquema de firma NTRU (NSS)

La idea central de NSS es como sigue.

La clave privada de quien firma es un vector corto generador de cierto retículo y su clave pública es un vector mucho más largo generador del mismo retículo.

La firma en un documento digital es un vector en el retículo con 2 propiedades importantes

i) El documento digital es codificado dentro del vector firma.

ii) El vector firma demuestra conocimiento de un vector corto en el retículo.

NSS usa 2 parámetros públicos: $(N, q) = (251, 128)$ o $(N, q) = (503, 256)$ y las operaciones básicas se efectuan en $\boldsymbol{\mathcal{R}}$.

Definición 39 *El* Retículo de Convolución Modular $\Lambda_{\boldsymbol{h}}$ *asociado al polinomio* $\boldsymbol{h}(x) = c_0 + c_1 x + \cdots + c_{N-1}x^{N-1} \in \boldsymbol{\mathcal{R}}$ *es*

$$\Lambda_{\boldsymbol{h}} = \left\{ (\boldsymbol{u}, \boldsymbol{v}) \in \boldsymbol{\mathcal{R}} \times \boldsymbol{\mathcal{R}} \cong \mathbb{Z}^{2N} \,\middle|\, \boldsymbol{v} = \boldsymbol{h} \circledast \boldsymbol{u} \bmod (q) \right\}.$$

Proposición 16 $\Lambda_{\boldsymbol{h}}$ *es generado por las filas de la matriz*

$$\left(\begin{array}{cccc|cccc} \alpha & 0 & \cdots & 0 & c_0 & c_1 & \cdots & c_{N-1} \\ 0 & \alpha & \cdots & 0 & c_{N-1} & c_0 & \cdots & c_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha & c_1 & c_2 & \cdots & c_0 \\ \hline 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{array}\right)$$

Observación 7 *$\Lambda_{\boldsymbol{h}}$ tiene una propiedad de invariancia rotacional; si $(\boldsymbol{u},\boldsymbol{v}) \in \Lambda_{\boldsymbol{h}} \Longrightarrow \quad \left(x^i \circledast \boldsymbol{u}, x^i \circledast \boldsymbol{v}\right) \in \Lambda_{\boldsymbol{h}} \quad \forall \quad 0 \leq i < N$.*

Observación 8 *Cada una de las rotaciones $\left(x^i \circledast \boldsymbol{u}, x^i \circledast \boldsymbol{v}\right)$ tiene la misma norma centrada que $(\boldsymbol{u},\boldsymbol{v})$.*

$$\boldsymbol{\mathcal{R}} \circledast (\boldsymbol{u},\boldsymbol{v}) = \Lambda\{(\boldsymbol{u},\boldsymbol{v})\} \cup \{\text{rotaciones de } (\boldsymbol{u},\boldsymbol{v})\}$$

es un subretículo de $\Lambda_{\boldsymbol{h}}$.

Definición 40 *Si $\boldsymbol{h} \in \boldsymbol{\mathcal{R}}$ tiene una descomposición de la forma*

$$\boldsymbol{h} \equiv \boldsymbol{f}^{-1} \circledast \boldsymbol{g} \bmod (q)$$

donde $\boldsymbol{f}$ y $\boldsymbol{g}$ tienen coeficientes pequeños, entonces se dice que $\Lambda_{\boldsymbol{h}}$ es un RETÍCULO NTRU*, denotado por $\Lambda_{\boldsymbol{h}}^{NT}$.*

N y q se relacionan por $q = O(N)$. Generalmente se toma $\dfrac{N}{3} \leq q \leq \dfrac{2N}{3}$.

Un polinomio pequeño es un polinomio cuyos coeficientes son $O(1)$, *i. e.* un poliomio cuyos coeficientes estan acotados independientemente de N. En la práctica los polinomios pequeños pueden tener la forma $\boldsymbol{f} + p\boldsymbol{g}$ donde $\boldsymbol{f}$, $\boldsymbol{g}$ polinomios binarios o terciarios y $p = 3$. La norma centrada de un polinomio pequeño $\boldsymbol{f}$ satisface

$$||\boldsymbol{f}|| = O\left(\sqrt{N}\right).$$

La heurística Gaussiana proporciona un método para predecir propiedades de un retículo aleatorio Λ (de dimensión grande), que el vector más corto tiene un tamaño aproximado de

$$\lambda_{\text{Gauss}}(\Lambda) = \sqrt{\frac{\dim(\Lambda)}{2\pi e}} \ \text{Disc}(\Lambda)^{\frac{1}{\dim(\Lambda)}}.$$

Similarmente la mayoría de problemas del vector más cercano para Λ tienen soluciones cuyo tamaño es $\approx \lambda_{\text{Gauss}}(\Lambda)$.

$\Lambda_{\boldsymbol{h}}$ tiene dimensión $2N$ y discriminante q^N, así que su vector más corto probable y vector más cercano tienen un tamaño aproximado

$$\lambda_{\text{Gauss}}(\Lambda_{\boldsymbol{h}}) = \sqrt{\frac{Nq}{\pi e}} = O(N).$$

$\Lambda_{\boldsymbol{h}}$ contiene N vectores l. i. de longitud $q = O(N)$, que son de hecho las N filas interiores de su matriz. Combinaciones lineales pequeñas de estos q-vectores son los vectores obvios de longitud $O(N)$ en $\Lambda_{\boldsymbol{h}}$.

Si $(\boldsymbol{u},\boldsymbol{v}) \in \Lambda_{\boldsymbol{h}}$, entonces el vector obtenido al reducir las coordenadas de $\boldsymbol{u}$ & $\boldsymbol{v}$ (mod q) está en $\Lambda_{\boldsymbol{h}}$. Así $\Lambda_{\boldsymbol{h}}$ contiene muchos vectores de longitud

$O\left(q\sqrt{N}\right) = O\left(N^{\frac{3}{2}}\right)$, y es, claramente, trivial también encontrar vectores en $\Lambda_{\boldsymbol{h}}$ cuya distancia a un vector dado es a lo más $O\left(N^{\frac{3}{2}}\right)$.

En $\Lambda_{\boldsymbol{h}}^{NT}$ el polinomio $\boldsymbol{h}$ tiene la forma $\boldsymbol{h} = \boldsymbol{f}^{-1} \circledast \boldsymbol{g} \bmod (q)$ para polinomios pequeños $\boldsymbol{f}$ & $\boldsymbol{g}$, así que $\Lambda_{\boldsymbol{h}}^{NT}$ contiene el vector corto $(\boldsymbol{f}, \boldsymbol{g})$. Más generalmente, todas las rotaciones de $(x^i \circledast \boldsymbol{f}, x^i \circledast \boldsymbol{g})$ son vectores cortos en $\Lambda_{\boldsymbol{h}}^{NT}$ con longitud $O(N)$. Basado en la heurística Gaussiana, estos vectores cortos secretos son probablemente $O(\sqrt{N})$ más pequeños que cualquier vector que no esté en el subespacio $\boldsymbol{\mathcal{R}} \circledast (\boldsymbol{f}, \boldsymbol{g})$ que generan.

Definición 41 *Sea $\Lambda_{\boldsymbol{h}}^{NT}$ un retículo NTRU. El* Problema de la Clave del Retículo NTRU *es encontrar un vector de longitud $O(\sqrt{N})$ en $\Lambda_{\boldsymbol{h}}^{NT}$.*

El retículo NTRU $\Lambda_{\boldsymbol{h}}^{NT}$ contiene al subespacio $\boldsymbol{\mathcal{R}} \circledast (\boldsymbol{f}, \boldsymbol{g})$ generado por los vectores de longitud $O(\sqrt{N})$, por lo que el problema de la clave del retículo NTRU siempre tiene soluciones. La heurística Gaussiana predice que el vector más pequeño en $\Lambda_{\boldsymbol{h}}^{NT}$ que es l. i. al subespacio $\boldsymbol{\mathcal{R}} \circledast (\boldsymbol{f}, \boldsymbol{g})$ tiene longitud $O(N)$. Luego es altamente probable que todas las soluciones al problema de la clave del retículo NTRU estén dadas por múltiplos pequeños $(\boldsymbol{u} \circledast \boldsymbol{f}, \boldsymbol{u} \circledast \boldsymbol{g})$, *i. e.*, con $\boldsymbol{u} \in \boldsymbol{\mathcal{R}}$ un polinomio de tamaño $||\boldsymbol{u}|| = O(1)$.

5.6.1 Verificación y PVMCe

En esta sección se da un argumento heurístico para mostrar que una firma NSS válida proporciona una solución para un problema del vector más cercano aproximado en un retículo NTRU, donde el vector objetivo depende del mensaje que se va a firmar. De esta forma se prueba que firmar es al menos tan difícil como resolver el problema indicado del vector más cercano aproximado.

Si $\boldsymbol{s}$ es una firma NSS válida para el mensaje $\boldsymbol{m}$ con clave pública $\boldsymbol{h}$ y sean

$$\boldsymbol{t} \equiv \boldsymbol{h} \circledast \boldsymbol{s} \bmod (q) \tag{5.3}$$

$$\boldsymbol{s}' = \boldsymbol{p}^{-1} \circledast (\boldsymbol{s} - \boldsymbol{m}) \bmod (q) \tag{5.4}$$

$$\boldsymbol{t}' = \boldsymbol{p}^{-1} \circledast (\boldsymbol{t} - \boldsymbol{m}) \bmod (q) \tag{5.5}$$

Sustituimos las fórmulas para $\boldsymbol{s}'$ y $\boldsymbol{t}'$ en

$$\boldsymbol{t} \equiv \boldsymbol{h} \circledast \boldsymbol{s} \pmod{q}$$

y con algo de álgebra

$$\boldsymbol{t}' \equiv \boldsymbol{h} \circledast \boldsymbol{s}' - \boldsymbol{p}^{-1} \circledast (\boldsymbol{m} - \boldsymbol{h} \circledast \boldsymbol{m}) \bmod (q).$$

El polinomio $\boldsymbol{A}_m = \boldsymbol{p}^{-1} \circledast (\boldsymbol{m} - \boldsymbol{h} \circledast \boldsymbol{m})$ es una cantidad conocida que depende de $\boldsymbol{m}$ y de $\boldsymbol{h}$.

El retículo NTRU asociado a $\boldsymbol{h}$ es el conjunto de vectores

$$\Lambda_{\boldsymbol{h}}^{NT} = \{(\boldsymbol{u}, \boldsymbol{h} \circledast \boldsymbol{u}) \mid \boldsymbol{u} \in \mathbb{Z}^N[x]\}$$

En particular, $\Lambda_{\boldsymbol{h}}^{NT}$ contiene al vector

$$\begin{aligned}(\boldsymbol{s}', \boldsymbol{h} \circledast \boldsymbol{s}') &= (\boldsymbol{s}', \boldsymbol{t}' + \boldsymbol{A}_m) \\ &= (\boldsymbol{s}', \boldsymbol{t}') + (\boldsymbol{0}, \boldsymbol{A}_m).\end{aligned}$$

La prueba de la norma $\|\cdot\|$ para una firma válida incluye la condición que el vector $(\boldsymbol{s}', \boldsymbol{t}')$ tiene norma no mayor que $\|\cdot\| NormBound$. Ahora consideramos la cuestión de cuan pequeña se puede poner la norma $\|\cdot\| NormBound$ y que siga permitiendo que las firmas válidas sean eficientemente generadas. Un formato típico para $\boldsymbol{s}$ es

$$\begin{aligned}\boldsymbol{s} &= \boldsymbol{f} \circledast \boldsymbol{w} \\ &= (\boldsymbol{u} + p \cdot \boldsymbol{F}) \circledast (\boldsymbol{w}_0 + p \cdot \boldsymbol{w}),\end{aligned}$$

donde

- $\boldsymbol{u}$, $\boldsymbol{F}$, $\boldsymbol{w}$ son polinomios pequeños
- $p \in \mathbb{Z}$ pequeño independiente de N y q, $(p, q) = 1$, por ejemplo $(p, q) = (3, 2^k)$, así $p = O(1)$
- el polinomio $\boldsymbol{w}_0 = \boldsymbol{u}^{-1} \circledast (\boldsymbol{m} + \boldsymbol{w}_1) \bmod (p)$
- $\boldsymbol{u}^{-1}$ inverso de $\boldsymbol{u} \bmod (p)$.

$\boldsymbol{w}_0$ es pequeño, pues sus coeficientes estan acotados por p y $p = O(1)$.

$\boldsymbol{w}_1$ tiene un número muy pequeño de coeficientes no 0. Más precisamente $||\boldsymbol{w}_1|| = O(1)$; por lo que $\boldsymbol{w}_1$ es aún más pequeño que un polinomio pequeño.

Se define $\boldsymbol{b} \in \boldsymbol{\mathcal{R}}$ por

$$\boldsymbol{u} \circledast \boldsymbol{w}_0 = \boldsymbol{m} + \boldsymbol{w}_1 + p\,\boldsymbol{b}.$$

Usando esta fórmula y la fórmula para $\boldsymbol{s}$, calculamos

$$\begin{aligned}\boldsymbol{s}' &\equiv p^{-1}\,(\boldsymbol{s} - \boldsymbol{m}) \bmod (q) \\ &\equiv p^{-1}\,(\boldsymbol{u} \circledast \boldsymbol{w}_0 + p\,\boldsymbol{u} \circledast \boldsymbol{w} + p\,\boldsymbol{F} \circledast \boldsymbol{w} - \boldsymbol{m}) \bmod (q) \\ &\equiv p^{-1}\,(\boldsymbol{m} + \boldsymbol{w}_1 + p\,\boldsymbol{b} + p\,\boldsymbol{u} \circledast \boldsymbol{w} + p\,\boldsymbol{F} \circledast \boldsymbol{w} - \boldsymbol{m}) \bmod (q) \\ &\equiv p^{-1}\,\boldsymbol{w}_1 + \boldsymbol{b} + \boldsymbol{u} \circledast \boldsymbol{w} + \boldsymbol{F} \circledast \boldsymbol{w} \bmod (q).\end{aligned}$$

Acotemos la norma de $\boldsymbol{s}$ usando la siguiente estimación

$$p^{-1} \bmod (q) = O(q) = O(N),$$

$$\begin{aligned} ||\boldsymbol{b}|| = ||\boldsymbol{u} \circledast \boldsymbol{w}_0 + \boldsymbol{m} + \boldsymbol{w}_1|| &<< ||\boldsymbol{u}|| \circledast ||\boldsymbol{w}_0|| + ||\boldsymbol{m}|| + ||\boldsymbol{w}_1|| = O(N) \\ ||\boldsymbol{m} + \boldsymbol{w}|| &<< ||\boldsymbol{u}|| \circledast ||\boldsymbol{w}|| = O(N) \\ ||\boldsymbol{F} \circledast \boldsymbol{w}|| = ||\boldsymbol{F} \circledast (\boldsymbol{w}_0 + p\boldsymbol{w})|| &<< ||\boldsymbol{F}||||\boldsymbol{w}_0|| + p||\boldsymbol{F}||||\boldsymbol{w}|| = O(N). \end{aligned}$$

De aquí

$$||\boldsymbol{s}'|| = O(N),$$

y un cálculo similar proporciona la misma cota para $||\boldsymbol{t}'||$. Esto prueba que usando la clave privada $(\boldsymbol{f}, \boldsymbol{g})$ es posible encontrar una firma válida $\boldsymbol{s}$ en el documento $\boldsymbol{m}$ que satisface la cota

$$||\boldsymbol{s}', \boldsymbol{t}'|| \leq O(N)$$

Por lo que podemos poner $\|\cdot\|NormBound = O(N)$.

Luego una firma creada usando la clave privada da un vector en $\Lambda_{\boldsymbol{h}}^{NT}$ cuya distancia a el vector $\boldsymbol{A}_m$ es $O(N)$. Esto puede ser comparado con la heurística Gaussiana, la cual predice que la la mayoría de los problemas del vector más cercano tienen una solución cuya longitud es $O(\sqrt{qN}) = O(N)$. En otras palabras, el conocimiento de la clave privada permite a una persona encontrar un vector en $\Lambda_{\boldsymbol{h}}^{NT}$ cuya distancia al vector objetivo $\boldsymbol{A}_m$ es múltiplo constante de la distancia más corta probable (como predijo la heurística Gaussiana).

§5.7 Análisis de seguridad

5.7.1 Ataques de fuerza bruta

Un atacante $\mathscr{I}$ puede recuperar la clave privada probando todos los posibles $\boldsymbol{f} \in \mathcal{L}_{\boldsymbol{f}}$ y probando si $\boldsymbol{f} \circledast \boldsymbol{h} \pmod q$ tiene entradas pequeñas probando todos los $\boldsymbol{g} \in \mathcal{L}_{\boldsymbol{g}}$ y checando si $\boldsymbol{g} \circledast \boldsymbol{h}^{-1} \pmod q$ tiene entradas pequeñas. Similarmente $\mathscr{I}$ puede recuperar un mensaje probando todos los posibles $\boldsymbol{r} \in \mathcal{L}_{\boldsymbol{r}}$ y checando si $\boldsymbol{e} - \boldsymbol{r} \circledast \boldsymbol{h} \pmod q$ tiene entradas pequeñas. En la práctica $\mathcal{L}_{\boldsymbol{g}}$ será más pequeño que $\mathcal{L}_{\boldsymbol{f}}$, así que la seguridad de la clave está determinada por $\sharp\mathcal{L}_{\boldsymbol{g}}$ y la seguridad de un mensaje individual está determinada por $\sharp\mathcal{L}_{\boldsymbol{r}}$. De cualquier manera, como se describe en la siguiente sección, hay un ataque por en medio el cual (suponiendo almacenamiento suficiente) corta el tiempo de búsqueda a la raíz cuadrada. De aquí que el nivel de seguridad esté dado por

$$\left(\begin{array}{c}\text{Seguridad}\\ \text{de la clave}\end{array}\right) = \sqrt{\sharp\mathcal{L}_{\boldsymbol{g}}} = \frac{1}{d_{\boldsymbol{g}}!}\sqrt{\frac{N!}{(N-2d_{\boldsymbol{g}})!}}$$

$$\left(\begin{array}{c}\text{Seguridad de}\\ \text{el mensaje}\end{array}\right) = \sqrt{\sharp\mathcal{L}_{\boldsymbol{r}}} = \frac{1}{d_{\boldsymbol{r}}!}\sqrt{\frac{N!}{(N-2d_{\boldsymbol{r}})!}}$$

5.7.2 Ataques por en medio

Un mensaje cifrado luce como $\boldsymbol{e} \equiv \boldsymbol{r} \circledast \boldsymbol{h} + \boldsymbol{m} \pmod{q}$. Existe un ataque por en medio el cual puede ser usado en contra de $\boldsymbol{r}$, y se observa que ataques similares se aplican también a la clave privada $\boldsymbol{f}$. Brevemente, se separa $\boldsymbol{f}$ a la mitad, digamos $\boldsymbol{f} = \boldsymbol{f}_1 + \boldsymbol{f}_2$ y entonces se compara $\boldsymbol{f}_1 \circledast \boldsymbol{e}$ contra $-\boldsymbol{f}_2 \circledast \boldsymbol{e}$, buscando $(\boldsymbol{f}_1, \boldsymbol{f}_2)$ tal que los coeficientes correspondientes tengan aproximadamente el mismo valor. Así que para obtener un nivel de seguridad de digamos 2^{80}, se debe escoger $\boldsymbol{f}$, $\boldsymbol{g}$ y $\boldsymbol{r}$ de conjuntos que contengan aproximadamente 2^{160} elementos.

5.7.3 Ataques de transmisión múltiple

Si $\mathscr{E}$ envía un único mensaje $\boldsymbol{m}$ varias veces usando la misma clave pública pero diferente $\boldsymbol{r}$ aleatorio, entonces $\mathscr{I}$ será capaz de recuperar la mayor parte del mensaje. Brevemente, suponga que $\mathscr{E}$ transmite $\boldsymbol{e}_i \equiv \boldsymbol{r}_i \circledast \boldsymbol{h} + \boldsymbol{m} \pmod{q}$ para $i = 1,\ 2, \ldots, t$. $\mathscr{I}$ puede entonces capturar $(\boldsymbol{e}_i - \boldsymbol{e}_1) \circledast \boldsymbol{h}^{-1} \pmod{q}$, y por tanto recuperar $\boldsymbol{r}_i - \boldsymbol{r}_1 \pmod{q}$. De cualquier manera, los coeficientes de $\boldsymbol{r}$ son tan pequeños que $\mathscr{I}$ recupera exactamente $\boldsymbol{r}_i - \boldsymbol{r}_1$, y de esto recuperará muchos de los coeficiente de $\boldsymbol{r}_1$. Si t es par de tamaño moderado (digamos 4 o 5), $\mathscr{I}$ recuperará suficiente de $\boldsymbol{r}_1$ como para poder probar todas las posibilidades para los coeficientes restantes por fuerza bruta, y de aquí recuperar $\boldsymbol{m}$. Por lo tanto la transmisión múltiple no es recomendable sin alguna transformación adicional del mensaje a transmitir. Note que si $\mathscr{I}$ descifra un uńico mensaje de esta manera, esta información no le servirá para descifrar cualquier mensaje subsecuente.

5.7.4 Ataques basados en retículos

El objeto de esta sección es dar un breve análisis del conocido ataque de retículos sobre la clave pública $\boldsymbol{h}$ y el mensaje $\boldsymbol{m}$. El objetivo de la reducción de retículos es encontrar uno o más vectores "pequeños" en un retículo dado. En teoría, el vector más pequeño puede se encontrado por busqueda exhaustiva, pero en la práctica esto no es posible si la dimensión del retículo es grande. El algoritmo LLL mejorado encontrará vectores relativamente pequeños en tiempo polinomial, pero aún a LLL le tomará un tiempo largo encontrar el vector más pequeño dado que el vector más pequeño no es mucho más pequeño que la longitud esperada del vector más pequeño.

Ataque de retículos sobre una clave privada de NTRU

Considere la matriz $2N \times 2N$ compuesta de los 4 bloques $N \times N$

$$\left(\begin{array}{cccc|cccc} \alpha & 0 & \cdots & 0 & h_0 & h_1 & \cdots & h_{N-1} \\ 0 & \alpha & \cdots & 0 & h_{N-1} & h_0 & \cdots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha & h_1 & h_2 & \cdots & h_0 \\ \hline 0 & 0 & \cdots & 0 & q & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & q & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & q \end{array}\right)$$

donde α es un parámetro escogido previamente. Sea Λ el retículo generado por las filas de esta matriz. El determinante de Λ es $q^N \alpha^N$.

Dado que la clave pública es $\boldsymbol{h} = \boldsymbol{g} \circledast \boldsymbol{f}^{-1}$, el retículo Λ contendrá el vector $\tau = (\alpha \boldsymbol{f}, \boldsymbol{g})$, lo cual es el vector de $2N$ entradas que consiste de los N coeficientes de $\boldsymbol{f}$ multiplicados por α, seguidos de los N coeficientes de $\boldsymbol{g}$. Por la heurística Gaussiana, el tamaño esperado del vector más pequeño en un retículo Λ aleatorio de dimensión $2N$ está entre

$$\mathrm{d}(\Lambda)^{\frac{1}{2N}} \sqrt{\frac{N}{\pi e}} \quad \text{y} \quad \mathrm{d}(\Lambda)^{\frac{1}{2N}} \sqrt{\frac{2N}{\pi e}}.$$

En este caso $\mathrm{d}(\Lambda) = q^N \alpha^N$, así que la longitud más pequeña esperada es mayor (pero no mucho mayor) que

$$s = \sqrt{\frac{N \alpha q}{\pi e}}.$$

Una implementación del algoritmo de reducción de retículos tendrá la mejor oportunidad de localizar τ, u otro vector cuya longitud sea cercana a la de τ, si $\mathscr{I}$ escoje α para maximizar la razón $\dfrac{s}{\|\tau\|}$. Elevando al cuadrado esta razón se observa que el atacande debe de escojer α de tal manera que maximize

$$\frac{\alpha}{\alpha^2 \|\boldsymbol{f}\|^2 + \|\boldsymbol{g}\|^2} = \left(\alpha \|\boldsymbol{f}\|^2 + \alpha^{-1} \|\boldsymbol{g}\|^2\right)^{-1}$$

Esto se logra escogiendo $\alpha = \dfrac{\|\boldsymbol{g}\|}{\|\boldsymbol{f}\|}$, $\|\boldsymbol{g}\|$ y $\|\boldsymbol{f}\|$ son cantidades públicas.

Cuando α es escogido de esta manera, definimos una constante $c_{\boldsymbol{h}}$ poniendo $\|\tau\| = c_{\boldsymbol{h}} s$. Así que $c_{\boldsymbol{h}}$ es la razón de la longitud del vector objetivo y la longitud del vector mas pequeño esperado. El valor más pequeño de $c_{\boldsymbol{h}}$, será aquel para el cual sea más fácil encontrar el vector objetivo. Sustituyendo en lo anterior, obtenemos

$$c_{\boldsymbol{h}} = \sqrt{\frac{2\pi e \|\boldsymbol{f}\| \, \|\boldsymbol{g}\|}{Nq}}.$$

Para un par dado $(\boldsymbol{f}, \boldsymbol{g})$ usado para el criptosistema, $c_{\boldsymbol{h}}$ puede ser visto como una medida de cuan lejos está el retículo asociado de uno aleatorio. Si $c_{\boldsymbol{h}}$ es cercano a $\mathbf{1}$, entonces Λ se parecerá a un retículo aleatorio y los métodos de reducción de retículos tendrán un tiempo difícil para encontrar un vector corto en general, y encontrar τ en particular. Tanto como $c_{\boldsymbol{h}}$ decrece los a algoritmos de reducción de retículos les será más fácil encontrar τ. Basados en la evidencia limitada que hemos obtenido, parece que el tiempo requerido es, por lo menos, exponencial en N, con una constante en el exponente proporcional a $c_{\boldsymbol{h}}$.

Ataque de retículos sobre un mensaje de NTRU

Un ataque de retículos puede ser dirigido en contra de un mensaje individual $\boldsymbol{m}$. Aquí el problema de retículos asociado es muy similar a aquel para $\boldsymbol{h}$, y el vector objetivo tendrá la forma $(\alpha \boldsymbol{m}, \boldsymbol{r})$. Como antes, $\mathscr{I}$ debe balancear el retículo usando $\alpha = \dfrac{\|\boldsymbol{r}\|}{\|\boldsymbol{m}\|}$, el cual lleva al valor

$$c_{\boldsymbol{m}} = \sqrt{\frac{2\pi e \|\boldsymbol{m}\| \, \|\phi\|}{Nq}}.$$

Esta constante $c_{\boldsymbol{m}}$ da una medida de la vulnerabilidad de un mensaje individual a un ataque de retículos, similar a la forma en que $c_{\boldsymbol{h}}$ lo hace para un ataque sobre $\boldsymbol{h}$. Un mensaje cifrado es más vulnerable si $c_{\boldsymbol{m}}$ es pequeño, y es menos tanto como $c_{\boldsymbol{m}}$ se acerca a $\mathbf{1}$.

A fin de hacer los ataque sobre $\boldsymbol{h}$ y $\boldsymbol{m}$ igual de difíciles, se debe tomar $c_{\boldsymbol{m}} \approx c_{\boldsymbol{h}}$, o equivalentemente, $\|\boldsymbol{f}\| \, \|\boldsymbol{g}\| \approx \|\boldsymbol{m}\| \, |\boldsymbol{r}\|$. En concreto, se restringe al caso $p = 3$; otros valores pueden ser analizados de manera similar. Para $p = 3$, un mensaje $\boldsymbol{m}$ en promedio consistirá de $\dfrac{N}{3}$ 1's, $\dfrac{N}{3}$ 0's, $\dfrac{N}{3}$ -1's, así que $\|\boldsymbol{m}\| \approx \sqrt{\dfrac{2N}{3}}$. Similarmente, $\boldsymbol{r}$ consiste de d 1's, d -1's y el resto 0's, así que $\|\boldsymbol{r}\| = \sqrt{2d_{\boldsymbol{r}}}$. Por lo que queremos poner

$$\|\boldsymbol{f}\| \, \|\boldsymbol{g}\| \approx \sqrt{\frac{4Nd_{\boldsymbol{r}}}{3}}.$$

Esto puede ser combinado con el criterio de descifrado (5.2) como ayuda para escoger los parámetros.

Ataque de retículos sobre una clave espuria

En vez de tratar de encontrar la clave privada $\boldsymbol{f}$, $\mathscr{I}$ puede usar el retículo descrito anteriormente, sección 5.7.4, y tratar de encontrar algún otro vector corto en el retículo, digamos de la forma $\tau' = (\alpha \boldsymbol{f}', \boldsymbol{g}')$. Si este vector es lo suficientemente corto, entonces $\boldsymbol{f}'$ actuará como una clave de descifrado. Más precisamente, si

$$\boldsymbol{f}' \circledast \boldsymbol{e} \equiv p\boldsymbol{r} \circledast \boldsymbol{g}' + \boldsymbol{m} \circledast \boldsymbol{f}' \pmod{q}$$

satisface $|p\boldsymbol{r}\circledast\boldsymbol{g}'+\boldsymbol{m}\circledast\boldsymbol{f}'|_\infty < q$ con alta probabilidad, el descifrado tendrá éxito; y aún si este ancho es $2q$ o $3q$, es posible que el mensaje pueda ser recuperado por técnicas correctoras de error, especialmente si varios τ''s pueden ser encontrados. De cualquier manera la evidencia experimental sugiere que la existencia de una clave espuria no plantea una amenaza de seguridad.

§5.8 Implementaciones prácticas de NTRU

Las normas de $\boldsymbol{f}$ y $\boldsymbol{g}$ han sido escogidas de tal manera que la falla de descifrado ocurra con una probabilidad menor que 5×10^{-5}.

5.8.1 Seguridad baja

Los parámetros de seguridad moderada son convenientes para situaciones en las cuales el valor intrínseco de un mensaje individual es pequeño, y en las cuales las claves se cambiarán con frecuencia razonable (cifrado de televisión, transmisiones de teléfono celular).

$$\begin{aligned} (N,p,q) &= (107,3,64)\\ \mathcal{L}_{\boldsymbol{f}} &= \mathcal{L}(15,14)\\ \mathcal{L}_{\boldsymbol{g}} &= \mathcal{L}(12,12)\\ \mathcal{L}_{\boldsymbol{r}} &= \mathcal{L}(5,5)\\ i.\ e.\quad d_{\boldsymbol{r}} &= 5. \end{aligned}$$

En otras palabras, $\boldsymbol{f}$ es escogida con 15 1's y 14 -1's; $\boldsymbol{g}$ es escogida con 12 1's y 12 -1's; y $\boldsymbol{r}$ es escogida con 5 1's y 5 -1's. Esto da los tamaños de las claves

$$\text{Clave Privada} = 340 \text{ bits} \quad \text{y} \quad \text{Clave Pública} = 642 \text{ bits}$$

y niveles de seguridad para un ataque por en medio.

$$\text{Seguridad de la clave} = 2^{50} \quad \text{y} \quad \text{Seguridad del mensaje} = 2^{26.5}.$$

Notemos otra vez que un ataque por en medio requiere una gran cantidad de almacenamiento de cómputo; para búsqueda directa para ataques por fuerza bruta los niveles de seguridad deben de ser elevados al cuadrado. Sustituyendo los valores anteriores en las fórmulas apropiadas obtenemos los valores del retículo

$$c_{\boldsymbol{h}} = 0.257, \quad c_{\boldsymbol{m}} = 0.258, \quad s = 0.422q.$$

5.8.2 Seguridad media

(N, p, q)	=	$(167, 3, 128)$
$\mathcal{L}_{\boldsymbol{f}}$	=	$\mathcal{L}(61, 60)$
$\mathcal{L}_{\boldsymbol{g}}$	=	$\mathcal{L}(20, 20)$
$\mathcal{L}_{\boldsymbol{r}}$	=	$\mathcal{L}(18, 18)$
i. e. $d_{\boldsymbol{r}}$	=	18
Clave Privada	=	530 bits
Clave Pública	=	1169 bits
Seguridad de la clave	=	$2^{82.9}$
Seguridad del mensaje	=	$2^{77.5}$
$c_{\boldsymbol{h}}$	=	0.236
$c_{\boldsymbol{m}}$	=	0.225
s	=	$0.296q$

5.8.3 Seguridad alta

(N, p, q)	=	$(503, 3, 256)$
$\mathcal{L}_{\boldsymbol{f}}$	=	$\mathcal{L}(216, 215)$
$\mathcal{L}_{\boldsymbol{g}}$	=	$\mathcal{L}(72, 72)$
$\mathcal{L}_{\boldsymbol{r}}$	=	$\mathcal{L}(55, 55)$
i. e. $d_{\boldsymbol{r}}$	=	55
Clave Privada	=	1595 bits
Clave Pública	=	2024 bits
Seguridad de la clave	=	2^{285}
Seguridad del mensaje	=	2^{170}
$c_{\boldsymbol{h}}$	=	0.182
$c_{\boldsymbol{m}}$	=	0.160
s	=	$0.0.365q$

6

NTRU y cómputo cuántico

§6.1 Introducción

En este capítulo se describe, a grandes rasgos, un nuevo método de reducción reticular. Este algoritmo aproxima vectores más cortos en un retículo salvo un factor de a lo más $\left(\frac{k}{6}\right)^{\frac{N}{2k}}$, donde k es el tamaño de bloque y N es la dimensión del retículo. Este método tiene un tiempo esperado de corrida de a lo más

$$O\left(N^3\left(\frac{k}{6}\right)^{\frac{k}{8}}A + N^4A\right).$$

Esto es aproximadamente la raíz cuadrada de

$$O\left(N^3\left(\frac{k}{6}\right)^{\frac{k}{4}} + N^4A\right),$$

que es el tiempo esperado de corrida del algoritmo de reducción por muestreo aleatorio de Schnorr. Aquí A es el número de operaciones bit para la aritmética sobre enteros de $O(N^2)$ bits.

Con este resultado se demuestra que la disponibilidad de computadoras cuánticas afectará no solo la seguridad de criptosistemas basados en factorización entera (RSA) 2.5, o logaritmos discretos (CE) 2.6, si no también a los basados en retículos.

§6.2 Algunos antecedentes

Se cree que las computadoras cuánticas no serán capaces de resolver problemas $\mathcal{NP}-$difíciles en tiempo polinomial, como lo son el PVMC y el PVMCe en un retículo [Ajt98, Mic01b].

A la fecha, no hay evidencia de que la seguridad de criptosistemas del tipo GGH [GGH97, Mic01a], que se basan en PVMC y PVMCe en retículos arbitrarios, se afecte por la disponibilidad, en un futuro, de computadoras cuánticas.

Tampoco existe tal resultado para el NTRU, el cual se basa en la dificultad del PVMC en una clase especial de retículos.

Los algoritmos clásicos de reducción en un retículo son:

i) el de Kannan [Kan], que calcula un vector más corto en tiempo exponencial.

ii) LLL renovado [LLL82], y sus variantes que calculan en tiempo polinomial un vector de a lo más $\left(\frac{4}{3}+\varepsilon\right)^{\frac{N-1}{2}}$ veces el vector más corto.

iii) el algoritmo (2k) de Schnorr [Sch87] que aplica el de Kannan a bloques de longitud $2k$ en la base del retículo. Con esto se mejora el factor de aproximación de LLL a $\left(\frac{k}{3}\right)^{\frac{N}{k}}$ para un k suficientemente grande, a costa de un tiempo adicional de corrida de $O\left(N^3 k^{k+o(k)} A\right)$.

iv) el método Dual Primario de Koy [Sch03a], que reduce el tiempo adicional de corrida a $O\left(N^3 k^{\frac{k}{2}+o(k)} A\right)$ con un factor de aproximación de a lo más $\left(\frac{k}{6}\right)^{\frac{k}{N}}$.

v) una variante del algoritmo $2k$, llamado BKZ (Block Korkine-Zolotarev) [SE91a], del cual no se ha probado que corra en tiempo polinomial en N.

§6.3 Algunos resultados con cómputo cuántico

Siempre que el método $2k$ el primer vector base, toma en cuenta sólo a los primeros $2k$ vectores. Schnorr [Sch03a], propuso un algoritmo que busca un reemplazo en el generado por la base completa para en el primer vector base, pero solo la contribución de los últimos vectores base puede ser variada. Si una cantidad suficiente de estos vectores se toma como muestra, entonces se puede encontrar, con alta probabilidad, un vector lo suficientemente corto. El tiempo adicional de corrida de esta reducción por muestreo aleatorio (RSR) es $O\left(N^3\left(\frac{k}{6}\right)^{\frac{k}{4}} A\right)$ y garantiza un factor de aproximación de $\left(\frac{k}{6}\right)^{\frac{N}{2k}}$. Si el tiempo de corrida asignado se fija, RSR reduce el factor de aproximación a aproximadamente su raíz cuadrada comparado con el método dual primario.

EL tiempo de corrida de RSR está dominado por el hecho de que el tamaño de la muestra de vectores que se toman es exponencial (en k) para luego ser descartados hasta que se encuentra un vector lo suficientemente pequeño. El

conjunto de vectores del cual se toma esta muestra aleatoria no tiene una estructura inherente que permita acelerar el tiempo de búsqueda. Por esto, se propone usar el algoritmo cuántico de búsqueda de Grover [Gro96a]. Se demuestra que una computadora cuántica encuentra un vector suficientemente corto con $O\left(\left(\frac{k}{6}\right)^{\frac{k}{8}}\right)$ evaluaciones de una función que es tan cara de evaluar como una sola llamada al algoritmo de muestreo de Schnorr. Esto lleva al algoritmo de reducción por búsqueda cuántica (QSR) que desarrolla $O\left(N^3\left(k^6\right)^{\frac{k}{8}}A\right)$ operaciones adicionales y obtiene un factor de aproximación de a lo más $\left(\frac{k}{6}\right)^{\frac{N}{2k}}$. De aquí que QSR mejora en tiempo fijo el factor de aproximación a aproximadamente la 8^a raíz comparado con el método dual primario.

Si

$$\max\{\|\vec{\boldsymbol{b}}_j\| \mid 1 \leq j \leq N\} = 2^{O(N)}$$

entonces el algoritmo LLL opera sobre enteros de longitud binaria $O(N^2)$.

Sea $\boldsymbol{B} = \hat{\boldsymbol{B}}\boldsymbol{R}$ la descompocisión de Gram-Schmidt de $\boldsymbol{B}$, es decir, las columnas $\hat{\vec{\boldsymbol{b}}}_j$ de $\hat{\boldsymbol{B}} \in \mathbb{Q}^{n\times n}$ son perpendiculares a pares y $\boldsymbol{R} = (\mu_{ij}) \in \mathbb{Q}^{n\times n}$ es triangular superior unitaria.

Se tiene que

$$\|\vec{\boldsymbol{b}}_1\| \leq \left(\delta - \frac{1}{4}\right)^{-\frac{n-1}{2}} \lambda_1$$

SA selecciona aleatoriamente vectores reticulares con coeficientes que satisfagan

$$\begin{array}{rcll} \nu_j & \in & \left(-\frac{1}{2}, \frac{1}{2}\right] & \text{para } 1 \leq j \leq n-k' \\ \nu_j & \in & (-1,1] & \text{para } n-k' < j < n \\ \nu_n & \in & \{1,2\} & \end{array} \tag{6.1}$$

para algún entero k'. Se denota

$$D_{n,k'} = \left(-\frac{1}{2}, \frac{1}{2}\right]^{n-k'} \times (-1,1]^{k'-1} \times \{1,2\}.$$

$$\|\vec{\boldsymbol{b}}\|^2 \leq 0.99\|\vec{\boldsymbol{b}}_1\|^2. \tag{6.2}$$

$$3(k+1) + \frac{k-1}{4}\log_2\left(\frac{k}{6}\right) \leq n. \tag{6.3}$$

La idea principal de reducción de búsqueda cuántica es reemplazar el algoritmo SHORT por una búsqueda cuántica de un vector que satisfaga 6.2. Más precisamente, se busca un vector suficientemente corto en

$$V_{\boldsymbol{B},k} = \left\{ \vec{\boldsymbol{v}} \in \Lambda(\boldsymbol{B}) \;\middle|\; \vec{\boldsymbol{v}} = \hat{\boldsymbol{B}}\vec{\boldsymbol{u}},\ \vec{\boldsymbol{u}} \in D_{n,k'} \text{ con } k' = 1 + \left\lceil \frac{k-1}{4} \log_2 \left(\frac{k}{6} \right) \right\rceil \right\}.$$

Sea $N = 2^{k'} = \min \left\{ 2^t \mid 2^t \geq 2 \left(\frac{k}{6} \right)^{\frac{k-1}{4}} \right\}$. Existe un algoritmo de tiempo $O(n^2 A)$ que enumera $V_{\boldsymbol{B},k}$. En particular, $\sharp(V_{\boldsymbol{B},k}) = N$. Este algoritmo es esencialmente el de Schnorr (SA), excepto que los bits aleatorios son reemplazados por el índice de entrada.

El algoritmo de enumeración de $V_{\boldsymbol{B},k}$, (ENUM), básicamente calcula en $O(n^2)$ pasos aritméticos el vector $\vec{\boldsymbol{v}}_i$ en alguna enumeración de $V_{\boldsymbol{B},k}$. Aquí $\boldsymbol{B}$ es una base $\delta - LLL$ reducida con descomposición de Gram-Schmidt $\hat{\boldsymbol{B}}\boldsymbol{R}$, $\boldsymbol{R} = [\mu_1, \mu_2, \ldots, \mu_n]$ y $k \geq 24$ un entero sujeto a 6.3, y la entrada es $\boldsymbol{B}$, k y un índice $0 \leq i < N$.

El vector regresado por ENUM satisface 6.1, pues $\boldsymbol{R}$ es triangular superior. Como los coeficientes en 6.1 están restringidos a intervalos semi-abiertos, la enumeración de $V_{\boldsymbol{B},k}$ es exhaustiva.

La caja negra oráculo de la búsqueda cuántica se basa en

$$f_{\boldsymbol{B},k} \colon \{0, 1, \ldots, N-1\} \longrightarrow \{0, 1\}$$

con

$$f_{\boldsymbol{B},k}(i) = 1 \iff \|ENUM(\boldsymbol{B}, k, i)\|^2 \leq .99\|\vec{\boldsymbol{b}}_1\|^2.$$

La valuación de $f_{\boldsymbol{B},k}$ requiere $O(n^2)$ pasos aritméticos sobre enteros de longitud $O(n^2)$, mientras que $f_{\boldsymbol{B},k}$ requiere $O(n^2 A)$ operaciones cuánticas y $O(n^2 A)$ cubits.

Ahora se aplica el algoritmo de BÚSQUEDA CUÁNTICA DE VECTORES CORTOS (QSHORT) para encontrar un vector $\vec{\boldsymbol{b}} \in \Lambda(\boldsymbol{B})$ que satisfaga 6.2. La entrada de este algoritmo es $\boldsymbol{B}$ una base $\delta - LLL$ reducida y un entero $k \geq 24$ sujeto a 6.3, usando RA y GSA sobre $O\left(n^2 \left(\frac{k}{6} \right)^{\frac{k-1}{8}} A \right)$ operaciones esperadas sobre $O(n^2 A)$ cubits.

Reemplazando SHORT por QSHORT, se obtiene un algoritmo que obtiene el mismo factor de aproximación que RSR en muchas menos operaciones elementales. Este es el algoritmo de Búsqueda Cuántica por Reducción (QSR). Si $\boldsymbol{B} = [\vec{\boldsymbol{b}}_1, \vec{\boldsymbol{b}}_2, \ldots, \vec{\boldsymbol{b}}_n]$ es una base $\delta - LLL$ reducida y $k \geq 24$ un entero sujeto a

6.3, usando RA y GSA sobre $\boldsymbol{B}$ y k, QSR calcula una base $\delta - LLL$ reducida $\boldsymbol{B}' = [\vec{\boldsymbol{b}}_1', \vec{\boldsymbol{b}}_2', \ldots, \vec{\boldsymbol{b}}_n']$ que satisface

$$\|\vec{\boldsymbol{b}}_1'\| \leq \left(\frac{k}{6}\right)^{\frac{n}{2k}} \lambda_1.$$

Para esto usa en promedio $O\left(n^2 \left(\frac{k}{6}\right)^{\frac{k-1}{8}} A + n^3 A\right)$ operaciones.

Este tiempo es la raíz cuadrada del tiempo de corrida del algoritmo clásico conocido más rápido (RSR). Un ataque con QSR a NTRU-503 requerirá un estimado de 5.8×10^3 MIPS-años, por lo que éste sería no feasible sólo si el parámetro de seguridad de NTRU se incrementara a al menos 1277.

N	**RSR**	**QSR**	**Tiempo RSR**	**Tiempo QSR**
503	3.3×10^8	5.8×10^3	55 días	1 min. 26 seg.
709	2.8×10^{11}	5.3×10^5	131 años	2 hrs. 10 min.
809	2.3×10^{13}	4.8×10^6	1.07×10^4 años	20 hrs.
1277	1.9×10^{22}	1.4×10^{11}	8.85×10^{12} años	66 años
1511	5.6×10^{26}	2.4×10^{13}	2.61×10^{17} años	11.18×10^3 años

Tiempo estimado de corrida (en MIPS[1]) para recuperar una llave privada NTRU en una computadora con un procesador a 2 GHz.

[1] Million Instructions per Second

7 Conclusiones

En el primer capítulo de este trabajo he abordado temas básicos matemáticos para el entendimiento de esquemas criptográficos basados en aritmética modular, curvas elípticas y retículos. También se abordaron temas básicos relacionados con la complejidad algorítmica, lo cual nos permite precisar la noción de seguridad de los esquemas al poder decir que un problema general es "difícil". Cabe mencionar que el hecho de que un tipo de problemas sea "difícil" no impide que alguna de sus instancias sea muy "fácil" de resolver. Además de incluir el pseudocódigo de algunos algoritmos.

En el segundo capítulo he introducido algunos conceptos básicos sobre aspectos criptográficos y nos enfocamos a los esquemas de firma digital. Los esquemas de firma digital tienen una gran importancia en el mundo electrónico y se han hecho grandes esfuerzos en varias partes del mudo (United Nations Commission on International Trade Law[1]) y México (NOM-151-SCFI-2002, prácticas comerciales[2]) para legislar su uso en aplicaciones comerciales. Aunque estos aspectos son meramente legales, resalta la gran importancia que tienen los esquemas de firmas digitales en cuanto a su seguridad, es decir, entre más difícil sea falsificar una instancia de firma digital, más seguro es el esquema de firma del cual proviene tal instancia. Por lo anterior es de suma importancia proveer de esquemas de firmas que sean seguros, y por ello mi enfoque en este tema.

En el tercer capítulo he abarcado el tema del cómputo cuántico. Aún no se sabe sobre la posibilidad o imposibilidad de la existencia de computadoras cuánticas capaces de manejar miles de qubits, por lo que los esfuerzos hasta ahora realizados son inofensivos a los esquemas de firma existentes y ampliamente usados como los basados en RSA y curvas elípticas. En caso de que se llegue a desarrollar computadoras cuánticas "grandes", es necesario contar con esquemas de firmas que puedan resistir ataques con éstas. Shor presentó un algoritmo cuántico teórico en el cual se calcula "fácilmente" la factorización de un entero y el logaritmo discreto, por lo que los esquemas de firma ampliamente usados como los basados en RSA y curvas elípticas quedan a merced de las computadoras cuánticas. De aquí se ve la importancia de contar con esquemas de firma digital que sean prácticos en cuanto a su eficiencia e implementación

[1] UNCITAL: `http://www.uncitral.org`

[2] `http://www.economia.gob.mx/work/normas/noms/2002/151scfi.pdf`

en computadoras "clásicas" y que sean resistentes al cómputo cuántico.

En los capítulos cuarto y quinto he abarcado el tema de retículos y el de esquemas de firma basado en ellos. La importancia de estas estructuras yace en la dificultad en hallar un vector de un retículo dado, por un lado, no nulo de longitud mínima y, por otro, más cercano a un vector dado en el espacio vectorial generado por el retículo. Schnorr presentó un algoritmo que encuentra un vector paralelo al más corto de manera subexponencial. Este algortimo permite atacar a los esquemas de firma basados en NTRU, pero de manera todavía ineficiente. Ludwig [Lud03] mejoró ese algoritmo utilizando el algoritmo cuántico de Grover (el cual busca un elemento dentro de una lista desordenada de forma eficiente: $O(\sqrt{N})$, con N el número de elementos en la lista), pero aún así estos esquema de firma basados en NTRU prevalecen seguros ante este ataque cuántico al incrementar sus parámetros en un factor tal que siguen siendo eficientes en su implementación y uso.

Bibliografía

[AB06] Simon Anders and Hans J. Briegel. *Fast simulation of stabilizer circuits using a graph-state representation. Phys. Rev. A*, 73(2):022334, Feb 2006.

[Ajt96] Miklós Ajtai. *Generating Hard Instances of Lattice Problems.* pages 99–108, May 1996.

[Ajt98] Miklós Ajtai. *The shortest vector problem in L2 is NP-hard for randomized reductions (extended abstract).* In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 10–19, New York, NY, USA, 1998. ACM Press.

[AKN98] Dorit Aharonov, Alexei Kitaev, and Noam Nisan. *Quantum circuits with mixed states.* pages 20–30, 1998.

[AR03] Dorit Aharonov and Oded Regev. *A lattice problem in quantum NP*, 2003.

[AR05] Dorit Aharonov and Oded Regev. *Lattice problems in* NP∩coNP. *J. ACM*, 52(5):749–765, 2005.

[ASM98] Kiyomichi Araki, Takakazu Satoh, and Shinji Miura. *Overview of elliptic curve cryptography.* In *PKC '98: Proceedings of the First International Workshop on Practice and Theory in Public Key Cryptography*, pages 29–49, London, UK, 1998. Springer-Verlag.

[Bea03] Stéphane Beauregard. *Circuit for Shor's algorithm using* $2n + 3$ *qubits.* 2003.

[Ben80] P. Benioff. *The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. J. Stat. Phys.*, 22:563–591, 1980.

[Ben82a] P. Benioff. *Quantum mechanical Hamiltonian models of Turing machines. J. Stat. Phys.*, 29:515, 1982.

[Ben82b] P. Benioff. *Quantum mechanical models of Turing machines that dissipate no energy. Phys. Rev. Lett.*, 48:1581–1585, 1982.

[Ber] André Berthiaume. *Quantum Computation.*

[Ber68] Elwyn R. Berlekamp. *Algebraic coding theory.* McGraw-Hill, New York, 1968.

[Beu94] Albrecht Beutelspacher. *Cryptology.* Mathematical Association of America, Washington, DC, USA, 1994.

[Blu82] M. Blum, M. & Shub. *Comparison of two pseudo-random number generators.* In *Conf. Proc. Crypto 82*, pages 61–78, 1982.

[Bra88] Gilles Brassard. *Modern cryptology - A tutorial*, volume 325 of *Lecture Notes in Computer Science.* Springer-Verlag, 1988.

[Bra95] Samuel L. Braunstein. *Quantum computation: a tutorial.* 1995.

[BS96] Eric Bach and Jeffrey Shallit. *Algorithmic number theory.* MIT Press, Cambridge, MA, USA, 1996.

[BV] Dan Boneh and Ramarathnam Venkatesan. *Breaking RSA may be easier than factoring.*

[Can02] Richard Cannings. *A quantum algorithm to break the key exchange system over imaginary quadratic fields.* December 2002.

[CLO+92] Matthijs J. Coster, Brian A. LaMacchia, Andrew M. Odlyzko, Claus-Peter Schnorr, Jacques Stern, and Antoine Joux. *Improved low-density subset sum algorithms. Comput. Complex.*, 2(2):111–128, 1992.

[CLOS91] Matthijs J. Coster, Brian A. LaMacchia, Andrew M. Odlyzko, and Claus-Peter Schnorr. *An Improved low-density subset sum algorithm. Lecture Notes in Computer Science*, 547:54–??, 1991.

[COS86] Don Coppersmith, Andrew M. Odlyzko, and Richard Schroeppel. *Discrete logarithms in* GF(p). *Algorithmica*, 1(1):1–15, 1986.

[CR85] Benny Chor and Ronald L. Rivest. *A knapsack type public key cryptosystem based on arithmetic in finite fields.* In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 54–65, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[CS00] Ronald Cramer and Victor Shoup. *Signature schemes based on the strong RSA assumption. ACM Transactions on Information and System Security*, May 2000.

[CS06] An Commeine and Igor Semaev. *An algorithm to solve the discrete logarithm problem with the number field sieve.* In *Public Key Cryptography*, pages 174–190, 2006.

[Deu85] David Deutsch. *Quantum theory, the Church-Turing principle and the universal quantum computer. Proceedings of the Royal Society of London Ser. A*, A400:97–117, 1985.

[DH76] Whitfield Diffie and Martin E. Hellman. *New directions in cryptography. IEEE Transactions on Information Theory*, IT-22(6):644–65, November 1976.

[DO86] Yvo Desmedt and Andrew M. Odlyzko. *A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes.* In *CRYPTO '85: Advances in Cryptology*, pages 516–522, London, UK, 1986. Springer-Verlag.

[Dwo03] Cynthia Dwork. *Lattices and their aplication to cryptography.* Lecture Notes, June 2003.

[EHI01] Artur Ekert, Patrick Hayden, and Hitoshi Inamori. *Basic concepts in quantum computation.* January 2001.

[Eis99] D. Eisenbud. *Commutative algebra. With a view toward algebraic geometry.* Springer-Verlag, 0 edition, 2 1999.

[Fei73] H. Feistel. *Cryptography and computer privacy. Scientific American*, 228(5):15–23, May 1973.

[Fey82] R. P. Feynman. *Simulating physics with computers. Int. J. of Theor. Phys.*, 21:467, 1982.

[Fey84] R. P. Feynman. *Quantum-Mechanical computers. J. Opt. Soc. Am. B*, 1:464, 1984.

[Fey86] R. P. Feynman. *Quantum-Mechanical computers. Found. Phys.*, 16:507–531, 1986.

[Gar94] Simson Garfinkel. *PGP: Pretty Good Privacy.* O'Reilly Media, November 1994.

[GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. *Public-key cryptosystems from lattice reduction problems. Lecture Notes in Computer Science*, 1294:112+, 1997.

[GJSS01] Craig Gentry, Jakob Jonsson, Jacques Stern, and Michael Szydlo. *Cryptoanalysis of the NTRU Signature Scheme (NSS) from Eurocrypt 2001.* In *Asiacrypt 2001, LNCS*, volume 2248, pages 1–20. Springer-Verlag, 2001.

[GMSS] Oded Goldreich, Daniele Micciancio, Shmuel Safra, and Jean-Pierre Seifert. *Approximating shortest lattice vectors is not harder than approximating closest lattice vectors.*

[Gol98] Oded Goldreich. *Modern cryptography, probabilistic proofs and pseudorandomness.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[Gol03] Oded Goldreich. *Cryptography and cryptographic protocols. Distrib. Comput.*, 16(2-3):177–199, 2003.

[Gor98] Daniel M. Gordon. *A survey of fast exponentiation methods.* In *Journal of Algorithms*, volume 27, pages 129–146, 1998.

[Gro96a] Lov K. Grover. *A fast quantum mechanical algorithm for database search.* In *STOC 1996*, pages 212–219, 1996.

[Gro96b] Lov K. Grover. *A fast quantum mechanical algorithm for database search.* In *STOC 1996*, pages 212–219, 1996.

[GS] Craig Gentry and Mike Szydlo. *Cryptoanalysis of the revised NTRU Signature Scheme.*

[Hug97] Richard J. Hughes. *Criyptography, quantum computation and trapped ions.* 1997.

[Hun74] Thomas W. Hungerford. *Algebra.* Number 73 in Graduate Texts in Mathematics. Springer-Verlag, New York, 1974.

[JHS98] Jill Pipher Jeffrey Hoffstein and Joseph H. Silverman. *NTRU: A ring based public key cryptosystem*, 1998.

[JHW02] Jill Pipher Joseph H. Silverman Jeffrey Hoffstein, Nick Howgrave-Graham and William Whyte. *NTRUSign: Digital signatures using the NTRU lattice.* Preprint, April 2002.

[Joz96] Richard Jozsa. *Quantum factoring, discrete logarithms, and the hidden subgroup problem. IEEE MultiMedia*, 3(2):34–43, 1996.

[Kak02] Subhash Kak. *General qubit errors cannot be corrected.* June 2002.

[Kan] Ravi Kannan. *Minkowski's convex body theorem and integer programming.*

[Kan87] Ravi Kannan. *Algorithmic geometry of numbers. Annual Reviews in Computer Science*, 2:231–267, 1987.

[KM] Neal Koblitz and Alfred J. Menezes. *A survey of public-key cryptosystems.*

[KMWZ98] Neal Koblitz, Alfred J. Menezes, Yi-Hong Wu, and Robert J. Zuccherato. *Algebraic aspects of cryptography.* Springer-Verlag New York, Inc., New York, NY, USA, 1998.

[Kob87] Neal Koblitz. *A course in number theory and cryptography.* Springer-Verlag New York, Inc., New York, NY, USA, 1987.

[Kon81] Alan G. Konheim. *Cryptography, a Primer*. John Wiley & Sons, Inc., New York, NY, USA, 1981.

[KS95] Erich Kaltofen and Victor Shoup. *Subquadratic-time factoring of polynomials over finite fields*. In *STOC '95: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 398–406, New York, NY, USA, 1995. ACM Press.

[KS97] Erich Kaltofen and Victor Shoup. *Fast polynomial factorization over high algebraic extensions of finite fields*. In *ISSAC '97: Proceedings of the 1997 international symposium on Symbolic and algebraic computation*, pages 184–188, New York, NY, USA, 1997. ACM Press.

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, and László Lovász. *Factoring polynomials with rational coefficients. Mathematische Annalen*, 261:515–534, 1982. URL: `http://cr.yp.to/bib/entries.html#1982/lenstra-lll`.

[LO91] Brian A. LaMacchia and Andrew M. Odlyzko. *Computation of discrete logarithms in prime fields (Extended Abstract)*. In *CRYPTO '90: Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, pages 616–618, London, UK, 1991. Springer-Verlag.

[Lud03] C. Ludwig. *A faster lattice reduction method using quantum search*, 2003.

[LV01] Arjen K. Lenstra and Eric R. Verheul. *Selecting cryptographic key sizes. Journal of Cryptology*, 2001. URL: `http://www.cryptosavvy.com/Joc.ps`. Note: to appear.

[Mat94] Mitsuru Matsui. *Linear cryptanalysis method for DES cipher*. In *EUROCRYPT '93: Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, pages 386–397, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[McL03] McLaughlin, Philip B. *New Frameworks for Montgomery's Modular Multiplication Method. Mathematics of Computation*, 2003.

[McL06] Laurianne McLaughlin. Philip zimmermann on what's next after pgp. *IEEE Security and Privacy*, 4(1):10, 2006.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of lattice problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, Mar 2002.

[Mic98] Daniele Micciancio. *On the hardness of the shortest vector problem*. PhD thesis, Sep 1998. Supervisor Shafi Goldwasser.

[Mic01a] Daniele Micciancio. *Improving lattice based cryptosystems using the Hermite normal form.* In *CaLC '01: Revised Papers from the International Conference on Cryptography and Lattices*, pages 126–145, London, UK, 2001. Springer-Verlag.

[Mic01b] Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. *SIAM J. Comput.*, 30(6):2008–2035, 2001.

[Mir01] Ilya Mironov. *A note on cryptoanalysis of the preliminary version of the NTRU Signature Scheme.* 2001.

[Mon85] Peter L. Montgomery. *Modular Multiplication Without Trial Division. Mathematics of Computation*, 44(170):519–521, April 1985.

[MVO91] Alfred Menezes, Scott Vanstone, and Tatsuaki Okamoto. *Reducing elliptic curve logarithms to logarithms in a finite field.* In *STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 80–89, New York, NY, USA, 1991. ACM Press.

[MZ03] Michele Mosca and Christof Zalka. *Exact quantum Fourier transform and discrete logarithm algorithms.* January 2003.

[NC97] M. A. Nielsen and Isaac L. Chuang. *Programmable quantum gate arrays. Phys. Rev. Lett.*, 79(2):321–324, Jul 1997.

[NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information.* Cambridge University Press, New York, NY, USA, October 2000.

[NHGW] Joseph H. Silverman Nick Howgrave-Graham and William Whyte. *Choosing parameter sets for NTRU-Encrypt with NAEP and SVES-3.*

[NS99] Phong Nguyen and Jacques Stern. *The Hardness of the Hidden Subset Sum Problem and its Cryptographic Implications.* In *Advances in Cryptology–Procedings of CRYPTO'99 Lecture Notes in Computer Sciens.* Springer-Verlag, August 1999.

[Odl00a] Andrew M. Odlyzko. *Discrete logarithms: The past and the future. Des. Codes Cryptography*, 19(2-3):129–145, 2000.

[Odl00b] Andrew M. Odlyzko. *Integer factorization and discrete logarithms (Abstract).* In *LATIN '00: Proceedings of the 4th Latin American Symposium on Theoretical Informatics*, page 258, London, UK, 2000. Springer-Verlag.

[Pol71] J. M. Pollard. *The fast Fourier transform in a finite field. Matematics of computation*, 25(114):365–374, April 1971.

[Pol75] J. M. Pollard. *Monte Carlo method for factorization. bit*, 15:331–334, 1975.

[Pom94] Carl Pomerance. *The number field sieve.* In Walter Gautschi, editor, *Mathematics of Computation 1943–1993: a half-century of computational mathematics*, volume 48, pages 465–480, Providence, RI, 1994. American Mathematical Society.

[Reg02a] O. Regev. *Quantum computation and lattice problems*, 2002.

[Reg02b] Oded Regev. *Quantum computation and lattice problems.* April 2002.

[Reg03] Oded Regev. *New lattice based cryptographic constructions.* September 2003.

[SA98] T. Satoh and K. Araki. *Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. Comment. Math. Univ. Sancti Pauli*, 47(1):81–92, 1998.

[Sch] Ed. Schaefer. *An introduction to cryptography.* Lecture Notes.

[Sch87] Claus-Peter Schnorr. *A hierarchy of polynomial time lattice basis reduction algorithms. Theoretical Computer Science*, 53(2-3):201–224, 1987.

[Sch93] Bruce Schneier. *Applied cryptography: protocols, algorithms and cource code in C.* John Wiley & Sons, Inc., New York, NY, USA, 1993.

[Sch03a] Claus-Peter Schnorr. *Lattice reduction by random sampling and birthday methods*, 2003.

[Sch03b] Claus-Peter Schnorr. *Lattice reduction by random sampling and birthday methods*, 2003.

[SE91a] Claus-Peter Schnorr and M. Euchner. *Lattice basis reduction: improved practical algorithms and solving subset sum problems.* In *FCT '91: Proceedings of the 8th International Symposium on Fundamentals of Computation Theory*, pages 68–85, London, UK, 1991. Springer-Verlag.

[SE91b] Claus-Peter Schnorr and M. Euchner. *Lattice basis reduction: improved practical algorithms and solving subset sum problems.* In *Fundamentals of computation theory*, pages 68–85, 1991.

[SE93] Claus-Peter Schnorr and M. Euchner. *Lattice basis reduction: improved practical algorithms and solving subset sum problems.* July 1993.

[Sem98a] Igor A. Semaev. *An algorithm for evaluation of discrete logarithms in some nonprime finite fields. Math. Comput.*, 67(224):1679–1689, 1998.

[Sem98b] Igor A. Semaev. *Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. Math. Comput.*, 67(221):353–356, 1998.

[Sem01] Igor A. Semaev. *A 3-dimensional lattice reduction algorithm.* In *CaLC*, pages 181–193, 2001.

[SH94] Claus-Peter Schnorr and H. H. Hörner. *Attacking the Chor-Rivest cryptosystem by improved lattice reduction*, 1994.

[SH95] Claus-Peter Schnorr and H. H. Hörner. *Attacking the Chor-Rivest cryptosystem by improved lattice reduction. Lecture Notes in Computer Science*, 921:1–12, 1995.

[Sha79] Adi Shamir. *On the cryptocomplexity of knapsack systems.* pages 118–129, 1979.

[Sho93] Victor Shoup. *Fast construction of irreducible polynomials over finite fields.* In *SODA '93: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 484–492, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[Sho94] Peter W. Shor. *Algorithms for quantum computation: discrete logarithms and factoring.* In *IEEE Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[Sho97] Peter W. Shor. *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J.SCI.STATIST.COMPUT.*, 26:1484, 1997.

[Sho98] Peter W. Shor. *Quantum computing.* In *ICM: Proceedings of the International Congress of Mathematicians*, 1998.

[Sho99] Victor Shoup. *Efficient computation of minimal polynomials in algebraic extensions of finite fields.* In *ISSAC '99: Proceedings of the 1999 international symposium on Symbolic and algebraic computation*, pages 53–58, New York, NY, USA, 1999. ACM Press.

[Sma99a] N. P. Smart. *Elliptic curve cryptosystems over small fields of odd characteristic. Journal of Cryptology*, 12(2):141–151, 1999.

[Sma99b] N. P. Smart. *The discrete logarithm problem on elliptic curves of trace one. J. Cryptology*, 12(3):193–196, October 1999.

[SOOS95a] Richard Schroeppel, Hilarie Orman, Sean O'Malley, and Oliver Spatscheck. *Fast key exchange with elliptic curve systems. Lecture Notes in Computer Science*, 963:43–56, 1995.

[SOOS95b] Richard Schroeppel, Hilarie Orman, Sean W. O'Malley, and Oliver Spatscheck. Fast key exchange with elliptic curve systems. In *CRYPTO '95: Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology*, pages 43–56, London, UK, 1995. Springer-Verlag.

[SS71] A. Schönhage and V. Strassen. *Schnelle multiplikation grosser zahlen. Springer Verlag*, pages 281–292, 1971.

[Szy03] Michael Szydlo. *Hypercubic lattice reduction and analysis of GGH and NTRU signatures.* In *Eurocrypt 2003, LNCS*, volume 2656, pages 433–448, 2003.

[Ver26] G. S. Vernam. Cipher printing telegraph systems for secret wire and radio telegraphic communications. *Journal American Institute of Electrical Engineers*, XLV(??):109–115, 1926.

[Vol01] I. V. Volovich. *Quantum computing and Shor's factorin algorithm.* 2001.

[VP98] Vlatko Vedral and Martin B. Plenio. *Basics of quantum computation.* February 1998.

[WZ82] W. K. Wootters and W. H. Zurek. *A single quantum cannot be cloned. Nature*, 299(5886):802–803, October 1982.

[Zal98] C. Zalka. *Fast versions of Shor's quantum factoring algorithm*, 1998.