"UN CRIPTOSISTEMA DE CLAVE PÚBLICA TIPO KNAPSACK BASADO EN CONJUNTOS B_h"

JAN ALEJANDRO MEDINA LÓPEZ

UNIVERSIDAD DEL CAUCA
FACULTAD DE CIENCIAS NATURALES, EXACTAS Y DE LA EDUCACIÓN
DEPARTAMENTO DE MATEMÁTICAS
POPAYÁN, CAUCA

"UN CRIPTOSISTEMA DE CLAVE PÚBLICA TIPO KNAPSACK BASADO EN CONJUNTOS B_h"

TRABAJO DE GRADO

En modalidad seminario presentado

como requisito parcial para optar al título de Matemático

JAN ALEJANDRO MEDINA LÓPEZ

Director:

Mg. DIEGO FERNANDO RUIZ

Codirector:

Dr. CARLOS ALBERTO TRUJILLO SOLARTE

UNIVERSIDAD DEL CAUCA

FACULTAD DE CIENCIAS NATURALES, EXACTAS Y DE LA EDUCACIÓN

DEPARTAMENTO DE MATEMÁTICAS

POPAYÁN, CAUCA

Agradecimientos

Agradezco en primer lugar a mis directores de tesis Diego Fernando Ruiz y Carlos Alberto Trujillo, a quienes admiro por sus cualidades humanas y profesionales. La ayuda, disposición, paciencia y constante motivación que me brindaron en el desarrollo de este trabajo son realmente invaluables.

A mi futura esposa Diana, por todo el amor y compañía que me ha ofrecido en estos dos último años.

A mis padres, mis hermanas y a mi hermano, quienes han creído en mi y que a pesar de todo, siempre están a mi lado.

Nota de aceptación _____

_____

_____

_____

Codirector:

_____

Dr. Carlos Alberto Trujillo Solarte

Jurados:

_____

Mg. Alfredo Gomez

_____

Esp. Siler Amador Donado

Fecha de sustentación:

Índice general

Lista de algoritmos

Capítulo 1

Introducción

Sea $\mathcal{A} = \{a_1, \ldots, a_n\}$ un conjunto de enteros positivos y sea s un entero positivo. Un problema sencillo de plantear pero difícil de resolver es el siguiente: ¿Es posible encontrar un subconjunto no vacío de $\mathcal{A}$ tal que la suma de sus elementos sea exactamente s? Este problema se conoce como *subset sum problem* (SSP) el cual es un problema que tiene gran importancia en teoría de la complejidad y criptografía.

Formalmente, el *SSP* se puede describir de la siguiente forma: Dado el conjunto $\mathcal{A}$ llamado mochila, determinar si existe un subconjunto de elementos $a_j \in \mathcal{A}$ tal que su suma sea s, o equivalentemente, determinar si existen $x_1, \ldots, x_n$ tales que

$$\sum_{j=1}^{n} x_j a_j = s, \;\; x_j \in \{0, 1\} \text{ para todo } j$$

El *SSP* es un problema de decisión NP-completo ([1], [2], [3]) que suele identificarse erróneamente con un problema más general llamado el problema *Knapsack (KP)* que en términos no matemáticos puede describirse como sigue: Dada una lista de n artículos (con tamaños distintos) donde cada uno aporta un beneficio y una mochila con capacidad para almacenar s kilos ¿es posible obtener como mínimo un beneficio determinado t de tal manera que la mochila no exceda su capacidad? Formalmente el problema *Knapsack* es el siguiente: Dados $\mathcal{A}$ y $\mathcal{B}$, determinar si

existe un conjunto $S \subseteq \{1, \ldots, n\}$ tal que

$$\begin{aligned} \sum_{j \in S} a_j &\leq s \\ \sum_{j \in S} b_j &\geq t \end{aligned}$$

El *SSP* es un caso particular del problema *knapsack* cuando $a_i = b_i$, $1 \leq i \leq n$ y $s = t$.

Un concepto muy importante relacionado con las mochilas es el concepto de densidad, el cual nos permite medir el tamaño de sus elementos. La densidad d de la mochila $\mathcal{A}$ se define como

$$d(\mathcal{A}) = \frac{n}{\log_2(\text{máx}\, \mathcal{A})}$$

Una de las aplicaciones más importantes del *SSP* se encuentra en criptografía[1], particularmente en el diseño de sistemas de cifrado de clave pública, de ahí que los sistemas cuya seguridad se basa en la solución del *SSP* se conocen como sistemas tipo mochila. En 1976, Ralph Merkle y Martin Hellman [4] idearon un sistema de cifrado muy sencillo de implementar, cuya seguridad se basa en la dificultad de resolver un caso particular del *SSP*, llamado *Problema de la Suma de un Subconjunto Supercreciente*. El sistema de cifrado tipo mochila de Merkle-Hellman es importante por razones históricas, ya que fue el primer sistema de cifrado de clave pública. Se han propuesto muchas variaciones de este sistema que usan mochilas de baja densidad, lo que lleva a demostrar que son inseguros [5]. Una excepción de estas variaciones es el sistema tipo mochila de Benny Chor y Ronald Rivest [6], el cual utiliza mochilas con densidad mayor que 1, haciendo ineficaces los ataques a sistemas tipo mochila de baja densidad.

El sistema de Chor-Rivest hace uso de la estructura y de la aritmética de los campos finitos para su implementación. Además de esto, utiliza un resultado clásico de la teoría aditiva de números, el Teorema de Bose-Chowla, el cual establece que para toda potencia prima q y un entero $h \geq 2$ existe un conjunto $B_h \pmod{q^h - 1}$ con q elementos (un conjunto $\mathcal{A}$ es un conjunto B_h en un grupo G si la suma de h elementos de $\mathcal{A}$ (incluyendo repeticiones) son diferentes).

La seguridad del sistema Chor-Rivest se basa en el *SSP* para mochilas de alta densidad, evitando así el ataque de Lagarias y Odlyzko [7] que hace uso del algoritmo L^3. El éxito de este ataque depende de dos factores: La densidad de la mochila y la facilidad para encontrar el vector más corto de un retículo $n + 1$-dimensional definido a partir del texto cifrado. Si la densidad d

[1] En criptografía el problema *knapsack y el SSP son considerados sinónimos*

de la mochila es baja, el vector más corto del retículo es la solución del texto cifrado y puede ser determinado de forma eficaz por el algoritmo L^3. Si la densidad de la mochila es alta entonces habrá muchos vectores cortos, por lo cual el más corto del retículo será difícil de encontrar, es mas, si $d < 0{,}645$ el ataque de Lagarias-Odlyzko es efectivo. En 1992 Coster et al. [8] mejoraron esta cota a 0,9408.

En Asiacrypt2005, Nguyen y Stern [9] introducen una nueva variante de densidad denominada "pseudo-densidad", la cual es efectiva para sistemas tipo mochila de bajo peso Hamming[2]. La pseudo-densidad se define como:

$$\kappa(\mathcal{A}) = \frac{r \log n}{\log_2(\text{máx}\, \mathcal{A})}$$

donde $r = \sum x_j^2$. Ellos mostraron que si $\kappa(\mathcal{A})$ es menor que 1, los sistemas tipo mochila de bajo peso Hamming se pueden resolver con el uso de SVP/CVP-oracle [9]. Es mas, mostraron que el sistema Chor-Rivest tiene pseudo densidad menor que 1.

En Africacrypt2008 Kunihiro [10] propone una nueva definición de densidad mucho más completa que las anteriores, la cual combina la definición de densidad usual, de pseudo-densidad y la función de entropía. Con base en esto se propone nuevas condiciones para tener en cuenta en el diseño de sistemas tipo mochila con bajo peso Hamming. Así, se define la densidad D de la mochila $\mathcal{A}$ de tamaño n y peso Hamming h como:

$$D(\mathcal{A}) = \frac{nH\left(\frac{h}{n}\right)}{\log_2(\text{máx}\, \mathcal{A})}$$

donde $H(x)$ es la función de entropía definida por $H(x) = x \log_2 x + (1-x) \log_2(1-x)$.

Este trabajo se enfoca en presentar en detalle el funcionamiento del sistema de clave pública presentado por Chor-Rivest en [6] basado en el *SSP* y la aritmética de campos finitos. Además, analizar el papel que juegan los conjuntos B_h obtenidos a través de la construcción Bose-Chowla en este sistema. Por último estudiar la propuesta de criptoanálisis desarrollado por Vaudenay dada en [11], el cual es hasta ahora el análisis más completo contra el sistema de Chor-Rivest.

El resto del trabajo se divide como sigue: en el Capítulo 2 se presenta el problema SSP, en el Capítulo 3 se estudia el sistema de Merkle-Hellman, en el Capítulo 4 se muestra la construcción de conjuntos B_h del tipo Bose-Chowla y se estudia de manera detallada el Sistema de Chor-Rivest,

[2]El peso Hamming hace referencia al total de bits 1 en un mensaje binario

en el Capítulo 5 se hace un estudio del criptoanálisis propuesto por Vaudenay, en el Apéndice A se presentan algunos cálculos basados en el nuevo concepto de densidad. Finalmente en el Apéndice B se muestran los algoritmos implementados en $SAGE$ utilizados en este trabajo.

1.1. Objetivos

1.1.1. General

Presentar la aplicación que tienen los conjuntos de Sidon tipo Bose-Chowla en el criptosistema de Chor-Rivest.

1.1.2. Específicos

1. Presentar un análisis de las propiedades matemáticas y problemas en los que se fundamenta el criptosistema de Chor-Rivest.

2. Mediante la teoría de campos finitos, explicar el criptoanalisis propuesto por Vaudenay para el sistema de Chor-Rivest.

3. Implementar en MuPAD los algoritmos básicos para la generación de las claves, el cifrado y descifrado en el criptosistema de Chor-Rivest.

Capítulo 2

El Problema de la Suma de un Subconjunto

Suponga que se tiene una mochila con capacidad para almacenar s kilos. Además, que se cuenta con una lista de n objetos con pesos $a_1, \ldots, a_n$ respectivamente y con valor también conocido $b_1, \ldots, b_n$. El problema Knapsack consiste en seleccionar una cierta cantidad de objetos de la lista de forma tal que la mochila quede completamente llena o los objetos en ella no pasen de s kilos y además que se obtenga como mínimo un valor determinado. El problema Knapsack forma parte de una lista histórica de problemas NP-Completos elaborada por Richard Karp en 1972. Éste problema se puede definir formalmente como sigue

Definición 2.1. Dados dos conjuntos $\mathcal{A} = \{a_1, \ldots, a_n\}$ y $\mathcal{B} = \{b_1, \ldots, b_n\}$ de enteros positivos y dados dos enteros positivos s y t, el problema *Knapsack* consiste en determinar si existe un subconjunto $S \subset \{1, 2, \ldots, n\}$ tal que

$$\begin{aligned} \sum_{i \in S} a_i &\leq s \\ \sum_{i \in S} b_i &\geq t \end{aligned}$$

El problema de la suma de un subconjunto (*SSP* por sus siglas del ingles) suele identificarse erróneamente con el problema knapsack; el *SSP* es un caso particular de éste problema cuando $a_i = b_i$, $1 \leq i \leq n$ y $s = t$. De manera formal, el *SSP* se define como sigue

Definición 2.2. Dado un conjunto de enteros positivos $\mathcal{A} = \{a_1, \ldots, a_n\}$ llamado mochila y un entero positivo s, el problema se trata de determinar si existe un subconjunto de $\mathcal{A}$ tal que la suma de sus elementos sea s. En otras palabras, determinar si existe un vector $x = (x_1, \ldots, x_n)$

con $x_i \in \{0,1\}$ para $1 \leq i \leq n$, tal que

$$\sum_{i=1}^{n} x_i a_i = s \tag{2.1}$$

Ejemplo 2.1. Sean $\mathcal{A} = \{2,3,4,9,14,23\}$ y $s = 27$. Por medio de ensayo y error se obtiene que la suma de los elementos del conjunto $\{4,9,14\}$ suman 27. Similarmente si $s = 29$, se encuentra que los conjuntos $\{2,4,23\}$ y $\{2,4,9,14\}$ son soluciones del problema.

Otra forma de plantear éste problema es el siguiente: Suponga que $\mathcal{A} = \{a_1, \ldots, a_n\}$ es un conjunto de enteros positivos. Considere un individuo E, el cual escoge de manera secreta un vector $x = (x_1, \ldots, x_n)$ donde $x_i \in \{0,1\}$. Con este vector se calcula el valor de s dado por la ecuación 2.1 y lo envía a un sujeto R. El *SSP* en este caso consiste en que el sujeto R determine cuál fue el vector utilizado por E para obtener el valor de s o que obtenga otro vector de 0's y 1's de manera que pueda obtener el mismo valor. Observe que el vector x de manera implícita determina un subconjunto de $\mathcal{A}$, ya que los x_i determinan que valores de la mochila pueden ser incluidos en la sumatoria (esto es cuando $x_i = 1$).

Claramente el sujeto R puede encontrar el vector x, si éste busca entre las 2^n posibilidades que existen, es mas, es posible reducir el exponente a la mitad si se utiliza un algoritmo simple de colisión [12] [1].

2.1. El Problema de la Suma de un Subconjunto Supercreciente

Suponga ahora que el sujeto R posee una información adicional sobre el conjunto $\mathcal{A}$, la cual es confidencial y que le permite encontrar de manera efectiva el vector x. Pero ¿qué tipo de información conoce R que le permita resolver el *SSP* fácilmente y que nadie más pueda hacerlo? Una opción es utilizar una variación del *SSP* que se puede resolver fácilmente, pero que de alguna manera disfrace esta solución fácil. Esta variación se conoce como el Problema de la Suma de un Subconjunto Supercreciente (*SSSP* por sus siglas en inglés).

Definición 2.3. Una sucesión supercreciente es un vector de enteros positivos $r = (r_1, r_2, \ldots, r_n)$ tal que $r_{i+1} \geq 2r_i$ para todo $1 \leq i \leq n-1$.

Lema 2.1. *Sea $r = (r_1, r_2, \ldots, r_n)$ una sucesión supercreciente. Entonces*

$$r_k > r_{k-1} + \cdots + r_2 + r_1 \quad \textit{para todo } 2 \leq k \leq n$$

Demostración. Por inducción sobre k.

(I) Para $k = 2$ se tiene $r_2 \geq 2r_1 > r_1$.

(II) (Hipótesis de inducción) Suponga ahora que la afirmación es válida para algún k con $2 \leq k \leq n$. Esto es $r_k > r_{k-1} + \cdots + r_2 + r_1$. Ahora, como r es una sucesión supercreciente se tiene $r_{k+1} \geq 2r_k$.

Luego, por hipótesis de inducción

$$r_{k+1} \geq 2r_k = r_k + r_k > r_k + (r_{k-1} + \cdots + r_2 + r_1)$$

Por lo anterior, se tiene que la afirmación es válida para todo k con $2 \leq k \leq n$. □

Definición 2.4. Sea $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ una mochila. Se dice que $\mathcal{A}$ es una mochila supercreciente si los elementos de $\mathcal{A}$ forman una sucesión supercreciente.

Note que el Problema de la Suma de un Subconjunto Supercreciente es un SSP que utiliza una mochila supercreciente. El algoritmo 1 resuelve éste problema eficientemente.

Algoritmo 1 Solución Problema de la Suma de un Subconjunto Supercreciente

Entrada: Una mochila supercreciente $\mathcal{A} = \{a_1, \ldots, a_n\}$ y un entero s (capacidad de la mochila)
Salida: $x = (x_1, \ldots, x_n)$ tal que $\sum_{i=1}^{n} x_i a_i = s$, $x_i \in \{0, 1\}$

```
1: i ← n
2: mientras i ≥ 1 hacer
3:    si s ≥ a_i entonces
4:       x_i ← 1
5:       s ← s − a_i
6:    si no
7:       x_i ← 0
8:    fin si
9:    i ← i − 1
10: fin mientras
11: retornar (x_1, ..., x_n)
```

Ejemplo 2.2. Sea $\mathcal{A} = \{3, 11, 24, 50, 115\}$ una mochila supercreciente y sea $s = 142$. Aplicando el Algoritmo 1 se obtiene que $x = (1, 0, 1, 0, 1)$ es la solución del problema. El funcionamiento del algoritmo se muestra en la Tabla 2.1.

s	i	x_5	x_4	x_3	x_2	x_1
142	5	1				
27	4	1	0			
27	3	1	0	1		
3	2	1	0	1	0	
3	1	1	0	1	0	1

Tabla 2.1: Ejecución del Algoritmo 1 con $\mathcal{A} = \{3, 11, 24, 50, 115\}$

Note que

$$1 \cdot 3 + 0 \cdot 11 + 1 \cdot 24 + 0 \cdot 50 + 1 \cdot 115 = 142$$

con lo que $\{3, 24, 115\} \subseteq \mathcal{A}$ es solución del problema planteado.

2.2. El concepto de densidad

Un concepto muy importante relacionado con las mochilas es el concepto de densidad, el cual de cierta forma permite medir el tamaño de sus elementos.

Definición 2.5. Sea $\mathcal{A} = \{a_1, \ldots, a_n\}$ una mochila. La densidad de $\mathcal{A}$ se define como

$$d(\mathcal{A}) = \frac{n}{\log_2(\text{máx}\,\mathcal{A})}$$

Se dice que la mochila es una mochila de baja densidad si $d(\mathcal{A}) < 0{,}9408$ (ver [8]).

Ejemplo 2.3. Sea $\mathcal{A} = \{1, 21, 27, 30\}$, la densidad de la mochila $\mathcal{A}$ es

$$d(\mathcal{A}) = \frac{4}{\log_2(30)} = 0{,}8151$$

Como $d(\mathcal{A}) < 0{,}9408$, se tiene que la mochila $\mathcal{A}$ es de baja densidad.

Ejemplo 2.4. Sea $\mathcal{B} = \{1, 21, 27, 30, 31, 42, 44\}$, la densidad de $\mathcal{B}$ es

$$d(\mathcal{B}) = \frac{4}{\log_2(44)} = 1{,}2821$$

2.3. Solución *SSP* para mochilas de baja densidad

El problema *SSP* que usa mochilas de baja densidad puede ser resuelto por medio del algoritmo de reducción de base L^3. Antes de mostrar cómo funciona este algoritmo, se introducirá una serie de definiciones que permitirá comprender su funcionamiento.

Definición 2.6. Sea $B = \{b_1 \ldots, b_m\}$ un conjunto de vectores linealmente independientes en $\mathbb{R}^n$ $(m \leq n)$. El conjunto L de todas las combinaciones lineales de $b_1, \ldots, b_m$ se denomina retículo de dimension m. Al conjunto B se le llama base del retículo L.

Un retículo puede tener diferentes bases. Una base que contiene vectores relativamente pequeños se llama base reducida. La siguiente definición proporciona una noción del concepto de base reducida, y es basada en el proceso de ortogonalización de Gram-Schmidt.

Definición 2.7. Sea $\mathbb{R}^n$ dotado de un producto interno $\langle \cdot, \cdot \rangle$ y sea $B = \{b_1, b_2, \ldots, b_n\}$ una base del retículo $L \subset \mathbb{R}^n$. Se definen los vectores b_i^* recursivamente por

- Para $i = 1, \quad b_1^* = b_1$
- Para $2 \leq i \leq n, \quad b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$ donde $\quad \mu_{i,j} = \dfrac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle}, \qquad 1 \leq j \leq i-1$

La base B es reducida si

$$|\mu_{i,j}| \leq \frac{1}{2}, \quad \text{para } 1 \leq j < i \leq n$$

y

$$\| b_i^* \|^2 \geq \left(\frac{3}{4} - \mu_{i,i-1}^* \right) \| b_{i,i-1}^* \|^2, \quad \text{para } 1 \leq i \leq n$$

La siguiente observación explica el hecho de que los vectores en una base reducida sean relativamente pequeños.

Observación 2.1. Sea $L \subset \mathbb{R}^n$ un retículo con base reducida $\{b_1, b_2, \ldots, b_n\}$.

1. Para todo vector no nulo $x \in L$, $\| b_1 \| \leq 2^{(n-1)/2} \| x \|$.
2. Para cualquier conjunto $\{c_1, c_2, \ldots, c_t\}$ de vectores linealmente independientes en L,

$$\| b_j \| \leq 2^{(n-1)/2} \, \text{máx}(\| c_1 \|, \| c_2 \|, \ldots, \| c_t \|) \quad \text{para } 1 \leq j \leq t.$$

Con el algoritmo de reducción de base L^3 (ver Algoritmo 2), que es de tiempo polinomial, es posible encontrar una base reducida a partir de una base de un retículo [1].

Algoritmo 2 Algoritmo L^3

Entrada: Una base $(b_1, b_2, \ldots, b_m)$ del retículo L en $\mathbb{R}^n$, $m \geq n$.
Salida: Una base reducida de L

1: $b_1^* \leftarrow b_1$, $B_1 \leftarrow \langle b_1^*, b_1^* \rangle$
2: **para** i desde 2 hasta n **hacer**
3: $\quad b_i^* \leftarrow b_i$
4: $\quad$ **para** j desde 1 hasta $i-1$ **hacer**
5: $\quad\quad \mu_{i,j} \leftarrow \langle b_i, b_j^* \rangle / B_j$, $b_i^* \leftarrow b_i^* - \mu_{i,j} b_j^*$, $B_i \leftarrow \langle b_i, b_j^* \rangle$
6: $\quad$ **fin para**
7: **fin para**
8: $k \leftarrow 2$
9: Ejecutar la subrutina $RED(k, k-1)$ para posiblemente actualizar algunos $\mu_{i,j}$
10: **si** $B_k < \left(\frac{3}{4} - \mu_{k,k-1}^2\right) B_{k-1}$ **entonces**
11: $\quad \mu \leftarrow \mu_{k,k-1}, B \leftarrow B_k + \mu^2 B_{k-1}$, $\mu k, k-1 \leftarrow \mu B_{k-1}/B$, $B_k \leftarrow B_{k-1} B_k / B, B_{k-1} \leftarrow B$
12: $\quad$ Intercambiar b_k y b_{k-1}
13: $\quad$ **si** $k > 2$ **entonces**
14: $\quad\quad$ Intercambiar $\mu_{k,j}$ y $\mu_{k-1,j}$ para $j = 1, 2, \ldots, k-2$
15: $\quad$ **fin si**
16: $\quad$ **para** $i = k+1, k+2, \ldots, n$ **hacer**
17: $\quad\quad t \leftarrow \mu_{i,k}$, $\mu_{i,k} \leftarrow \mu_{i,k-1} - \mu t$ y $\mu_{i,k-1} \leftarrow t + \mu_{k,k-1} \mu_{i,k}$
18: $\quad$ **fin para**
19: $\quad k \leftarrow \text{máx}(2, k-1)$
20: $\quad$ Ir al paso 9
21: **si no**
22: $\quad$ **para** $l = k-2, k-3, \ldots, 1$ **hacer**
23: $\quad\quad$ Ejecutar $RED(k, l)$
24: $\quad\quad k \leftarrow k+1$
25: $\quad$ **fin para**
26: **fin si**
27: **si** $k \leq n$ **entonces**
28: $\quad$ Ir al paso 9
29: **si no**
30: $\quad$ **retornar** $(b_1, b_2, \ldots, b_n)$
31: **fin si**

Algoritmo 3 Subrutina RED

1: **si** $|\mu_{k,l}| > \frac{1}{2}$ **entonces**
2: $\quad r \leftarrow \lfloor 0{,}5 + \mu_{k,l} \rfloor$, $b_k \leftarrow b_k - r b_l$
3: **fin si**
4: **para** j from 1 to $l-1$ **hacer**
5: $\quad \mu_{k,j} \leftarrow \mu_{k,j} - r \mu_{l,j}$
6: $\quad \mu_{k,l} \leftarrow \mu_{k,l} - r$
7: **fin para**

El Algoritmo 4 reduce el *SSP* al problema de encontrar un vector corto en un retículo. Por lo tanto, con el algoritmo L^3 puede encontrarse un vector que sea una solución al *SSP*, siempre y cuando este vector sea el más corto de todos los vectores distintos de cero en el retículo.

Algoritmo 4 Solución *SSP* de baja densidad usando el algoritmo L^3

Entrada: Un conjunto de enteros positivos $(a_1, a_2, \ldots, a_n)$ y un entero s.
Salida: $x_i \in \{0,1\}, 1 \leq i \leq n$, tal que $\sum_{i=1}^{n} a_i x_i = s$ siempre que los x_i existan

1: Sea $m = \lceil \frac{1}{2}\sqrt{n} \rceil$
2: Formar un retículo $(n+1)$-dimensional L cuya base $\bar{B}$ está conformada por los vectores

$$(1,0,0,\cdots,0,ma_1)\ (0,1,0,\cdots,0,ma_2)$$
$$\vdots$$
$$(0,0,0,\cdots,1,ma_n)\ (0,0,0,\cdots,0,-s)$$

3: $B \leftarrow L^3(\bar{B})$
4: Para cada vector $(b_1, b_2, \ldots, b_{n+1})$ en B, hacer:
5: **si** $b_{n+1} = 0$ y $b_i \in \{0,1\}$ para todo $i = 1, 2, \ldots, n$ **entonces**
6: **para** $i = 1, 2, \ldots, n$ **hacer**
7: $x_i \leftarrow b_i$
8: **fin para**
9: **si** $\sum_{i=1}^{n} a_i x_i = s$ **entonces**
10: **return** $(x_1, x_2, \ldots, x_n)$
11: **fin si**
12: **si no**
13: **return** "El algoritmo fallo"
14: **fin si**

Ejemplo 2.5. Considere la mochila $\mathcal{A} = \{295, 592, 301, 14, 28, 353, 120, 236\}$ cuya densidad es $d = 0{,}868$ y sea $s = 1129$. Aplicando el Algoritmo 4 se obtiene que el vector $(0,1,1,0,0,0,0,1)$ es una solución al problema.

Merkle y Hellman propusieron un sistema de clave pública basado en el *SSSP* que se oculta por medio de congruencias y una permutación. En el siguiente capítulo se mostrará detalladamente el funcionamiento de este sistema.

Capítulo 3

Criptosistema de Merkle-Hellman

El objetivo de la criptografía es la de asegurar la privacidad y la confidencialidad de las comunicaciones. Por otro lado, la meta del criptoanálisis es romper la seguridad y privacidad de este tipo de comunicaciones. En particular, en la criptografía de clave pública, cada usuario tiene dos claves: una que se conoce públicamente (clave pública) y que es utilizada por cualquier usuario para cifrar un mensaje y una privada, que se mantiene en secreto por el receptor y que es utilizado por él para descifrar los mensajes cifrados. En general, la criptografía de clave pública basa su seguridad en la insolubilidad computacional de algunos problemas de la teoría de números, como el problema de la factorización de números grandes, el problema del logaritmo discreto y el SSP.

3.1. Funcionamiento del Criptosistema

En 1978 Ralph Merkle y Martin Hellman [13] proponen un sistema de cifrado basado en el problema SSP conocido como criptosistema tipo mochila de Merkle-Hellman, el cual es importante por razones históricas, ya que fue el primer criptosistema de clave pública.

3.1.1. Generación de las claves

En el sistema de Merkle-Hellman, las dos claves están compuestas por mochilas. La clave privada es una mochila supercreciente, mientras que la clave pública no lo es. En este sistema la clave pública se genera a partir de la clave privada; para ello Merkle y Hellman haciendo uso de la aritmética modular, desarrollaron un procedimiento que permite convertir una mochila

supercreciente en una mochila normal.

Para la generación de la clave pública, el receptor debe escoger una mochila supercreciente $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ y dos enteros positivos w y p tales que $p > a_n + \cdots + a_2 + a_1$, $1 < w \leq p-1$ y $\text{mcd}(w, p) = 1$. Luego, debe seleccionar una permutación π del conjunto $\{1, 2, \ldots, n\}$ y calcular

$$b_i = w a_{\pi(i)} \text{ mód } p$$

Así, la clave pública que podrá utilizar el emisor será $\mathcal{B} = \{b_1, b_2, \ldots, b_n\}$ y la clave privada que mantendrá en secreto el receptor estará compuesta por π, p, w y $\{a_1, a_2, \ldots, a_n\}$.

3.1.2. Proceso de Cifrado

Para que el emisor pueda enviar un mensaje (el cual debe ser una sucesión binaria del mismo tamaño que la clave pública) por medio de un canal inseguro, éste debe cifrarse y para ello se eligen los elementos de la mochila $\mathcal{B}$ (clave pública) que correspondan al bit 1 en el mensaje, mientras que los elementos correspondientes al bit 0 son descartados. Luego, se suman los elementos que corresponden al bit 1, y el resultado será el texto cifrado. En otras palabras, para que el emisor pueda enviar un mensaje $\mathcal{M} = (m_1, m_2, \ldots, m_n)$, debe calcular

$$C = b_1 m_1 + b_2 m_2 + \cdots + b_n m_n = \sum_{i=1}^{n} b_i m_i; \qquad m_i \in \{0, 1\}$$

donde C será el mensaje cifrado. Por último, el emisor enviará el mensaje cifrado C a su destinatario.

Observación 3.1. En caso que el mensaje no sea de la misma longitud de la clave, se debe subdividir en sucesiones que tengan esta longitud y se aplica el mismo procedimiento a cada sucesión.

3.1.3. Proceso de Descifrado

Para que el receptor pueda descifrar el mensaje C debe encontrar enteros $x_1, x_2, \ldots, x_n$, con $x_i \in \{0, 1\}$ tales que $C = x_1 b_1 + x_2 b_2 + \cdots + x_n b_n$, esto es debe resolver un SSP, el cual es un problema NP-completo (ver [3] y [14]). Sin embargo, los enteros b_i fueron escogidos de tal manera que este problema se pueda solucionar fácilmente si se conocen $\mathcal{A}$, w, p y la permutación

π.

Para descifrar el mensaje el receptor encontrará un entero d tal que

$$d \equiv w^{-1}s \text{ mód } p \equiv w^{-1}\sum_{i=1}^{n} m_i b_i \text{ mód } p$$

como $1 \equiv ww^{-1}$ mód p y $b_i = wa_{\pi(i)}$ mód p se tiene

$$w^{-1}b_i \equiv w^{-1}wa_{\pi(i)} \text{ mód } p \equiv a_{\pi(i)} \text{ mód } p$$

Por lo tanto,

$$d \equiv \sum_{i=1}^{n} m_i a_i \text{ mód } p$$

Ahora, como $\sum_{i=1}^{n} a_i < p$ entonces $0 \leq d < p$. De esta manera, el receptor en vez de solucionar un problema tan difícil como el SSP debe solucionar un SSSP (el cual como se mostró anteriormente es fácil de resolver usando el Algoritmo 1). Haciendo uso de este algoritmo el receptor puede encontrar los enteros $x_1, x_2, \ldots, x_n$, con $x_i \in \{0, 1\}$ tales que $d = x_1a_1 + x_2a_2 + \cdots + x_na_n$. Así, el mensaje original estará dado por $m_i = r_{\pi(i)}$ para $i = 1, 2, \ldots, n$.

Observación 3.2. Note que la parte más importante del proceso de descifrado consiste en conocer $\mathcal{A}$, w, p y la permutación π, y encontrar w^{-1}.

3.1.4. Ejemplo de implementación

A continuación se presentará un ejemplo de implementación del criptosistema Merkle-Hellman con parámetros que permitan comprender el funcionamiento de cada uno de los procesos expuestos anteriormente.

Considere como parámetros del criptosistema la mochila supercreciente $\mathcal{A} = \{12, 17, 33, 74, 157, 316, 611, 1223\}$, $p = 2447$, $w = 1635$ y la permutación π del conjunto $\{1, 2, \ldots, 8\}$ dada por

$$\begin{aligned} \pi(1) &= 3, \qquad \pi(2) = 6 \\ \pi(3) &= 1, \qquad \pi(4) = 2 \\ \pi(5) &= 5, \qquad \pi(6) = 4 \\ \pi(7) &= 8, \qquad \pi(8) = 7 \end{aligned}$$

La clave privada del receptor en este caso estará dada por π, p, w y $\mathcal{A}$. A continuación se generará la clave pública, la cual se obtiene calculando $b_i = wa_{\pi(i)}$ mód p para $i = 1, 2, \ldots, 8$, esto es

$$\begin{aligned}
b_1 &= 1635 \cdot 33 \text{ mód } 2447 = 121 \\
b_2 &= 1635 \cdot 316 \text{ mód } 2447 = 343 \\
b_3 &= 1635 \cdot 12 \text{ mód } 2447 = 44 \\
b_4 &= 1635 \cdot 17 \text{ mód } 2447 = 878 \\
b_5 &= 1635 \cdot 157 \text{ mód } 2447 = 2207 \\
b_6 &= 1635 \cdot 74 \text{ mód } 2447 = 1087 \\
b_7 &= 1635 \cdot 1223 \text{ mód } 2447 = 406 \\
b_8 &= 1635 \cdot 611 \text{ mód } 2447 = 609
\end{aligned}$$

de donde se obtiene la clave pública $\mathcal{B} = \{121, 343, 44, 878, 2207, 1087, 406, 609\}$.
Ahora, suponga que el emisor quiere enviar el mensaje "Hola". Para que este mensaje pueda ser enviado de forma "segura" primero debe ser convertido en una sucesión binaria y luego ser cifrado, para ello se utilizó la codificación ASCII, de modo que

$$\begin{aligned}
m_1 &= 01001000 \qquad m_2 = 01101110 \\
m_3 &= 01101100 \qquad m_4 = 01100001
\end{aligned}$$

obteniendo el mensaje binario $\mathcal{M} = (m_1, m_2, m_3, m_4)$ donde m_1,m_2,m_3 y m_4 corresponde a la representación binaria de las letras H, o, l y a respectivamente, el mensaje $\mathcal{M}$ será cifrado por partes, esto es, en primer lugar se cifrará m_1, luego m_2 y m_3 y por último se cifrará m_4 de la siguiente manera

$$\begin{aligned}
s_1 &= 121 \cdot 0 + 343 \cdot 1 + 44 \cdot 0 + 878 \cdot 0 + 2207 \cdot 1 + 1087 \cdot 0 + 406 \cdot 0 + 609 \cdot 0 = 2550 \\
s_2 &= 121 \cdot 0 + 343 \cdot 1 + 44 \cdot 1 + 878 \cdot 0 + 2207 \cdot 1 + 1087 \cdot 1 + 406 \cdot 1 + 609 \cdot 0 = 4087 \\
s_3 &= 121 \cdot 0 + 343 \cdot 1 + 44 \cdot 1 + 878 \cdot 0 + 2207 \cdot 1 + 1087 \cdot 1 + 406 \cdot 0 + 609 \cdot 0 = 3681 \\
s_4 &= 121 \cdot 0 + 343 \cdot 1 + 44 \cdot 1 + 878 \cdot 0 + 2207 \cdot 0 + 1087 \cdot 0 + 406 \cdot 0 + 609 \cdot 1 = 996
\end{aligned}$$

Luego, el mensaje cifrado es $C = \{2250, 4087, 3681, 3203\}$ el cual será enviado al receptor. Una vez el receptor reciba el mensaje C, él procede a descifrarlo; para esto debe calcular $d_i = w^{-1}s_i$ mód p para $i = 1, 2, \ldots, 8$, esto es

$$\begin{aligned} d_1 = &\quad 1335 \cdot 2550 \text{ mód } 2447 &= 473 \\ d_2 = &\quad 1335 \cdot 4087 \text{ mód } 2447 &= 1782 \\ d_3 = &\quad 1335 \cdot 3681 \text{ mód } 2447 &= 559 \\ d_4 = &\quad 1335 \cdot 3203 \text{ mód } 2447 &= 1096 \end{aligned}$$

Luego, utilizando su clave privada $\mathcal{A}$, el Algoritmo 1 y la permutación π se obtiene

$$\begin{aligned} m_1 &= 01001000 \qquad & m_2 &= 01101110 \\ m_3 &= 01101100 \qquad & m_4 &= 01100001 \end{aligned}$$

el cual es el mensaje original.

Observación 3.3. En 1982, por desgracia para este histórico, elegante e importante criptosistema, Shamir y Zippel presenta un algoritmo en tiempo polinomial capaz de romper este sistema de cifrado (ver [5]). Actualmente el ataque más poderoso conocido para los criptosistema tipo mochila, es la técnica que reduce el problema SSP a un problema de encontrar un vector corto en un retículo $(n+1)$-dimensional conocido como SVP (por sus siglas en ingles) o el de encontrar el vector más cercano a otro dado, conocido como CVP, los cuales utilizan el algoritmo L^3 o alguna variación del mismo para su solución. Este ataque es un éxito si la densidad de la mochila definida en [8] es menor que $0,9408\ldots$ Esto es significante ya que la densidad de las mochilas en el sistema Merkle-Hellman es menor que 1, puesto que si sucede lo contrario habrán varios subconjuntos de la mochila que den la misma suma (es decir, el SSP no tendría solución única) lo cual implica que algunos textos cifrados no sean descifrados de forma única.

En 1988 Benny Chor y Ronald Rivest propusieron un nuevo sistema de cifrado tipo mochila conocido como el Criptosistema Chor-Rivest [6]. Hasta hace poco, este criptosistema fue el único sistema de cifrado tipo mochila que era seguro. Sin embargo, fue roto por S. Vaudenay en 2001 [11].

Capítulo 4

Conjuntos B_h tipo Bose-Chowla y el Criptosistema Chor-Rivest

4.1. Conjuntos B_h tipo Bose-Chowla

Sean $(G,+)$ un grupo conmutativo, h un entero mayor o igual que 2 y $\mathcal{A} = \{g_1, g_2, \ldots, g_k\} \subseteq G$. $\mathcal{A}$ es un conjunto B_h en G si todas las sumas de h elementos de $\mathcal{A}$ (no necesariamente distintos) son diferentes. Cuando $h = 2$, los conjuntos B_2 son llamados conjuntos de Sidon. Si $\mathcal{A}$ es un conjunto B_h en G se escribe $\mathcal{A} \in B_h(G)$ y si $\mathcal{A}$ es un conjunto B_h en $\mathbb{Z}_n$ se escribe $\mathcal{A} \in B_h(G)$ (mód n), el cual es llamado conjunto B_h módulo n.

Observación 4.1. Para demostrar que un conjunto $\mathcal{A}$ es $B_h(G)$, se supone que existen dos sumas de h elementos $a_1, a_2, \ldots, a_h, b_1, b_2, \ldots, b_h \in \mathcal{A}$ tal que

$$a_1 + a_2 + \cdots + a_h = b_1 + b_2 + \cdots + b_h$$

y se debe probar que

$$\{a_1, a_2, \ldots, a_h\} = \{b_1, b_2, \ldots, b_h\}$$

El siguiente lema es consecuencia directa de la definición de conjunto B_h y se utilizará para llevar conjuntos B_h de un grupo cíclico a los enteros modulares.

Lema 4.1. *Sean $(G_1,+)$ y $(G_2,*)$ grupos conmutativos y $\phi : G_1 \to G_2$ un homomorfismo inyectivo. Si $\mathcal{A} \in B_h(G_1)$ entonces $\phi(\mathcal{A}) \in B_h(G_2)$.*

Demostración. Sean $a_1, a_2, \ldots, a_h, b_1, b_2, \ldots, b_h \in \mathcal{A}$. Suponga que

$$\phi(a_1) * \phi(a_2) * \cdots * \phi(a_h) = \phi(b_1) * \phi(b_2) * \cdots * \phi(b_h)$$

Como ϕ es un homomorfismo inyectivo, entonces

$$\begin{aligned} \phi(a_1 + a_2 + \cdots + a_h) &= \phi(b_1 + b_2 + \cdots + b_h) \\ a_1 + a_2 + \cdots + a_h &= b_1 + b_2 + \cdots + b_h \end{aligned}$$

Ahora, como $\mathcal{A} \in B_h(G_1)$ se tiene

$$\{a_1, a_2, \ldots, a_h\} = \{b_1, b_2, \ldots, b_h\}$$

de donde

$$\{\phi(a_1), \phi(a_2), \ldots, \phi(a_h)\} = \{\phi(b_1), \phi(b_2), \ldots, \phi(b_h)\}$$

Por lo tanto, $\phi(\mathcal{A}) \in B_h(G_2)$. □

Teorema 4.2 (Construcción de Bose-Chowla)**.** *Para cada potencia prima q y todo entero $h \geq 2$, existe un conjunto $B_h(\text{mód} q^h - 1)$ con q elementos.*

Demostración. Sea $\theta \in \mathbb{F}_{q^h}$ un elemento primitivo, entonces el polinomio minimal de θ tiene grado h y además $\langle\theta\rangle = F^*_{q^h}$. Se probará que el conjunto

$$\theta + \mathbb{F}_q := \{\theta + a : a \in \mathbb{F}_q\}$$

es un conjunto $B_h(\mathbb{F}_{q^h})$ multiplicativo, donde $\mathbb{F}^*_{q^h}$ denota el grupo de unidades de $\mathbb{F}_{q^h}$.

Suponga lo contrario, esto es, que dos productos de h elementos de $\theta + \mathbb{F}_q$ son iguales

$$\prod_{k=1}^{h} (\theta + a_{i_k}) = \prod_{k=1}^{h} (\theta + a_{j_k}),$$

donde

$$\begin{aligned} 1 \leq i_1 \leq i_2 \leq \cdots \leq i_h \leq q, \qquad & 1 \leq j_1 \leq j_2 \leq \cdots \leq j_h \leq q, \\ (i_1, i_2, \ldots, i_h) \quad \neq \quad & (j_1, j_2, \ldots, j_h) \end{aligned} \tag{4.1}$$

Entonces por (4.1), el polinomio

$$p(x) = \prod_{k=1}^{h}(x + a_{ik}) - \prod_{k=1}^{h}(x + a_{jk}) \in \mathbb{F}_q[x]$$

es no nulo, además $grad(p) < h$ y se anula en θ, lo cual no puede ser posible.

Ahora, por el Lema 4.1, el conjunto

$$\log_\theta(\theta + \mathbb{F}_q) := \{\log_\theta(\theta + a) : a \in \mathbb{F}_q\},$$

es un conjunto $B_h(\text{mód} q^h - 1)$ bajo el isomorfismo

$$\begin{aligned} \log_\theta : \mathbb{F}^*_{q^h} &\rightarrow \mathbb{Z}_{q^h-1}, \\ \theta^k &\rightarrow k \end{aligned}$$

llamado logaritmo discreto en base θ. □

Observación 4.2. Note que la contradicción a la que se llegó en la demostración anterior consiste en que θ no puede ser raíz de un polinomio de grado menor que h, para lo cual no es necesario tomar un elemento primitivo para sumar a $\mathbb{F}_q$, bastará con que se considere $\beta \in \mathbb{F}_{q^h}$ algebraico de grado h en lugar de θ. Así, el conjunto $\beta + \mathbb{F}_q$ es un conjunto B_h multiplicativo en $\mathbb{F}^*_{q^h}$. Sin embargo para el logaritmo discreto es necesario tener como base un elemento primitivo.

Ejemplo 4.1 (Conjunto B_2)**.** Sean $q = 5$, $p(x) = x^2 + x + 2 \in \mathbb{F}_5[x]$. Ahora, como $p(x)$ es mónico e irreducible y si θ es una raíz de $p(x)$ entonces $\mathbb{F}^*_{25}$ es generado por θ. En efecto

k	θ^k
1	θ
2	$\theta^2 = 4\theta + 3$
3	$\theta^3 = 4\theta + 2$
4	$\theta^4 = 3\theta + 2$
5	$\theta^5 = 4\theta + 4$
6	$\theta^6 = 2$

k	θ^k
7	$\theta^7 = 2\theta$
8	$\theta^8 = 3\theta + 1$
9	$\theta^9 = 3\theta + 4$
10	$\theta^{10} = \theta + 4$
11	$\theta^{11} = 3\theta + 3$
12	$\theta^{12} = 4$

k	θ^k	k	θ^k
13	$\theta^{13} = 4\theta$	19	$\theta^{19} = 3\theta$
14	$\theta^{14} = \theta + 2$	20	$\theta^{20} = 2\theta + 4$
15	$\theta^{15} = \theta + 3$	21	$\theta^{21} = 2\theta + 1$
16	$\theta^{16} = 2\theta + 3$	22	$\theta^{22} = 4\theta + 1$
17	$\theta^{17} = \theta + 1$	23	$\theta^{23} = 2\theta + 2$
18	$\theta^{18} = 3$	24	$\theta^{24} = 1$

Luego, el conjunto

$$\theta + \mathbb{F}_5 = \{\theta, \theta + 1, \theta + 2, \theta + 3, \theta + 4\},$$

es un conjunto B_2 en el grupo multiplicativo $\mathbb{F}_{5^2}^*$. Ahora, por el teorema anterior, el conjunto

$$\log_\theta(\theta + \mathbb{F}_q) = \{1, 17, 14, 15, 10\}$$

es un conjunto B_2 módulo $5^2 - 1 = 24$ (como se puede verificar en la Tabla4.1)

Tabla 4.1: Sumas de $\{1, 17, 14, 15, 10\}$

1	10	14	15	17
2	11	15	16	18
	20	0	1	3
		4	5	7
			6	8
				10

Ejemplo 4.2. Sean $q = 5$, $p(x) = x^3 - 2x + 2 \in \mathbb{F}_5[x]$, α una raíz de $p(x)$. Como $p(x)$ es irreducible sobre $\mathbb{F}_5[x]$, entonces $\mathbb{F}_{5^3}$ es isomorfo a $\mathbb{F}_5[x]/\langle p(X)\rangle$ que a su vez es isomorfo a $\mathbb{F}_5(\alpha)$, es decir, $\mathbb{F}_{5^3} \cong \mathbb{F}_5[x]/\langle p(X)\rangle \cong \mathbb{F}_5(\alpha)$. Considere el elemento primitivo $\theta = 2\alpha^2 - 2\alpha - 2$. Entonces el conjunto

$$\alpha + \mathbb{F}_5 = \{2\alpha^2 - 2\alpha - 2, 2\alpha^2 - 2\alpha - 1, 2\alpha^2 - 2\alpha, 2\alpha^1 - 2\alpha + 1, 2\alpha^2 - 2\alpha + 2\},$$

es un conjunto B_3 en $(\mathbb{F}_{125}^*)$. Así,

$$\log_\theta(\alpha + \mathbb{F}_5) = \{1, 10, 12, 66, 110\}$$

es un conjunto B_3 módulo $5^3 - 1 = 124$.

4.2. Criptosistema Chor-Rivest

El criptosistema Chor-Rivest es un sistema de clave pública tipo mochila, basado en la estructura y aritmética de los campos finitos y en la construcción de conjuntos B_h tipo Bose-Chowla. Una de las ventajas de este criptosistema es que la densidad de la mochila considerada es mayor que 1, lo cual hace inefectivos los ataques a mochilas de baja densidad.

4.2.1. Generación de las claves

En primer lugar se determinarán los parámetros del criptosistema Chor-Rivest.

1. Considere $q = p^\lambda$ la potencia de un primo y sea $2 \leq h \leq q$ un entero tal que el cálculo del logaritmo discreto en el cuerpo $\mathbb{F}_{q^h}$ pueda hacerse de manera eficiente. [1]

2. Una vez determinado h, se procede a elegir de manera aleatoria un elemento algebraico α de grado h sobre $\mathbb{F}_q$, esto se logra encontrando un polinomio $p(x)$ mónico e irreducible de grado h sobre $\mathbb{F}_q[x]$ de manera que los elementos de $\mathbb{F}_{q^h}$ se puedan escribir como polinomios de grado menor o igual que $h-1$ en variable α y con coeficientes en $\mathbb{F}_q$.

3. Ahora, se elige un elemento primitivo θ de $\mathbb{F}_{q^h}$. Esto se puede realizar escogiendo de manera aleatoria un elemento $z \in \mathbb{F}_{q^h}$ que satisfaga $z^{(q^h-1)/s} \neq 1$ para todos los factores primos s de $q^h - 1$. Observe que es importante que $q^h - 1$ tenga factores primos pequeños y note también que el procedimiento anterior es eficiente puesto que la probabilidad de encontrar un generador de $\mathbb{F}^*_{q^h}$ es
$$\frac{\phi(q^h-1)}{q^h-1}$$
donde ϕ es la función phi de Euler.

[1] El cálculo del logaritmo discreto es un problema que es considerado computacionalmente difícil y por tanto es un problema que ha llamado la atención de los matemáticos en los últimos años. Pero, existen algunos algoritmos que permiten calcular el logaritmo discreto para casos especiales, estos son el de Coppersmith [15] y el de Pohlig-Hellman [16].
El algoritmo de Coppersmith funciona en campos con característica pequeña, y funciona de manera óptima en campos de característica 2. Para este último caso, el tiempo de ejecución es $e^{O(\sqrt[3]{n \ln^2 n})}$. Por otro lado, el algoritmo de Pohlig-Hellman funciona en cuerpos de cualquier característica pero siempre y cuando el orden del grupo multiplicativo del campo finito considerado tenga factores primos pequeños.

4. Luego, siguiendo la construcción de Bose-Chowla se calcula $a_i = \log_\theta(\alpha + t_i)$ para todo $t_i \in \mathbb{F}_q$. Seguidamente se reordenan los a_i obtenidos anteriormente, por medio de una permutación aleatoria π obteniendo los elementos $b_i = a_{\pi(i)}$. Se añade un ruido escogiendo aleatoriamente un entero tal que $0 \leq r \leq q^h - 2$ y se calcula $c_i = (b_i + r) \pmod{q^h - 1}$ para $i = 0, 1, \ldots, q-1$

5. De este modo, la clave pública del usuario está formada por $\{c_0, c_1, \ldots, c_{q-1}\}$, q y h. La clave privada será α, θ, π, r.

Observación 4.3. Cada usuario puede utilizar el mismo q y el mismo h puesto que la probabilidad de colisiones (es decir, que dos usuarios tengan la misma clave) es tan pequeña que se puede despreciar [6][17].

4.2.2. Proceso de Cifrado

Para cifrar un mensaje binario $\mathcal{M} = (m_0, m_1, \ldots, m_{q-1})$ de tamaño q y peso Hamming h (es decir, bits iguales a 1 en $\mathcal{M}$), se debe calcular la suma de los c_i's para los que $m_i = 1$, esto es,

$$C = \sum_{i=0}^{q-1} m_i c_i \pmod{q^h - 1}$$

4.2.3. Proceso de Descifrado

Para descifrar el mensaje $\mathcal{M}$ a partir de C, se debe realizar lo siguiente

1. Se calcula $s \equiv (C - hr) \pmod{q^h - 1}$.

2. Se obtiene el polinomio de grado menor o igual que $h - 1$ dado por

$$q(\alpha) \equiv \theta^s \pmod{p(\alpha)}$$

En efecto, sea I el conjunto de índices para los cuales a m_i le corresponde el bit 1, entonces se tiene

$$\theta^s = \theta^C \cdot \theta^{-hr} = \theta^{\sum_{i=0}^{q-1} m_i c_i} \cdot \theta^{-hr} = \prod_{i \in I} \theta^{c_i - r}$$

observe que

$$\sum_{i=0}^{q-1} m_i c_i = \sum_{i \in I} c_i$$

de donde

$$\begin{aligned}\theta^s &= \prod_{i\in I}\theta^{c_i-r} = \prod_{i\in I}\theta^{b_i} \\ &= \prod_{i\in I}\theta^{a_{\pi(i)}} = \prod_{i\in I}(\alpha + t_{\pi(i)}) \text{ con } t_{\pi(i)} \in \mathbb{F}_q\end{aligned}$$

3. Considere el polinomio $T(x) = p(x) + q(x)$ de grado h, observe que $T(x) \equiv \theta^s \pmod{p(x)}$, de ahí que T se puede descomponer en factores lineales en $\mathbb{F}_q$, de manera que

$$T(x) = \prod_{i\in I}(x + t_{\pi(i)})$$

Ahora, reemplazando los valores $t_0, t_1, \ldots, t_{q-1} \in \mathbb{F}_q$ en la expresión anterior, se obtienen las h raíces del polinomio T. Suponiendo que dichas raíces son $a_{j_1}, a_{j_2}, \ldots, a_{j_h}$ aplicando la permutación π^{-1} a los subíndices de estas raíces, se obtendrán los subíndices del mensaje en donde se encuentran los bit iguales a 1.

4.2.4. Transformando cadenas de bits

Cuando se estudió el proceso de cifrado, se trabajó con el supuesto de que el mensaje binario $\mathcal{M}$ era de tamaño q y tenía peso h, es decir un mensaje de q bits de los cuales h son iguales a 1. A continuación se presentará un procedimiento que permite transformar un mensaje binario en un conjunto de mensajes de tamaño q y peso h.

Sea $\mathcal{M}$ un mensaje binario cualquiera, primero el mensaje se dividirá en bloques de $\lfloor \log_2 \binom{q}{h} \rfloor$ bits. Cada uno de los bloques puede verse como la representación binaria de un numero n tal que $0 \leq n < \binom{q}{h}$. Para transformar estos números en vectores binarios de peso h se hará uso de la aplicación que preserva el orden inducido por el orden lexicográfico de los vectores y el orden natural de los números enteros.

Si n es más grande que $\binom{q-1}{h-1}$, el primer bit en el vector es 1, en caso contrario el bit será 0. Luego, se actualizan los valores de q y h y se itera q veces hasta que todos los q bits son determinados. El Algoritmo 5 transforma un número n en un vector binario v de tamaño q y peso h.

Algoritmo 5 Transformación entero a vector

Entrada: Un entero n y los parámetros q y h
Salida: Un vector binario v de tamaño q y peso h

1: **para** $i = 1, 2, \ldots, q$ **hacer**
2: **si** $n \geq \binom{q-1}{h}$ **entonces**
3: $v_i \leftarrow 1$
4: $n \leftarrow n - \binom{q-i}{h-1}$
5: $h \leftarrow h - 1$
6: **si no** $v_i \leftarrow 0$
7: **fin si**
8: **fin para**
9: **return** v

La transformación inversa, la cual es utilizada en la última etapa del proceso de descifrado, recupera un número n a partir de un vector binario v con las características mencionadas. El Algoritmo 6 muestra el procedimiento a seguir.

Algoritmo 6 Transformación vector a entero

Entrada: Un vector binario v y los parámetros q y h
Salida: Un entero n

1: $n \leftarrow 0$
2: **para** $i = 1, 2, \ldots, q$ **hacer**
3: **si** $v_i = 1$ **entonces**
4: $n \leftarrow n + \binom{q-i}{h}$
5: $h \leftarrow h - 1$
6: **fin si**
7: **fin para**
8: **return** n

Note que para una implementación eficiente, los $qh/4$ coeficientes binomiales anteriores, pueden ser calculados previamente y almacenados para ser después utilizados.

4.2.5. Ejemplo de implementación

A continuación se desarrollará un ejemplo donde se mostrará cómo llevar a cabo el proceso de cifrado y descifrado de un mensaje corto, tomando como parámetros números relativamente pequeños con el fin de hacer el proceso más interactivo con el lector.

Para esta implementación se generará en primer lugar la clave pública y la privada.

1. Como parámetros se utilizará $q = 17$ y $h = 3$

2. Sea $p(x) = x^3 + 4x^2 + 15x + 4 \in \mathbb{F}_{17}[x]$ un polinomio mónico e irreducible y sea $\alpha \in \mathbb{F}_{17^3}$ una raíz de $p(x)$.

3. Sea $\theta = \alpha + 14$ un elemento primitivo de $\mathbb{F}_{17^3}$. Si $n = 4912$, la elección se llevó a cabo verificando que $(\alpha + 14)^{n/s} \neq 1$ para todo factor primo s de n. Esto es,

$$\begin{aligned} (\alpha + 14)^{n/2} &= 16 \\ (\alpha + 14)^{n/307} &= 12\alpha^2 + \alpha + 14 \end{aligned}$$

4. Mediante la construcción de Bose-Chowla se procede a calcular $a_i = \log_\theta(\alpha + t_i)$ para todo $t_i \in \mathbb{F}_{17}$, obteniendo

$$\begin{array}{lll} a_0 = \log_\theta(\alpha) = 1548 & a_1 = \log_\theta(\alpha + 1) = 2142 & a_2 = \log_\theta(\alpha + 2) = 4400 \\ a_3 = \log_\theta(\alpha + 3) = 4226 & a_4 = \log_\theta(\alpha + 4) = 335 & a_5 = \log_\theta(\alpha + 5) = 1445 \\ a_6 = \log_\theta(\alpha + 6) = 3311 & a_7 = \log_\theta(\alpha + 7) = 3465 & a_8 = \log_\theta(\alpha + 8) = 1730 \\ a_9 = \log_\theta(\alpha + 9) = 1414 & a_{10} = \log_\theta(\alpha + 10) = 4482 & a_{11} = \log_\theta(\alpha + 11) = 959 \\ a_{12} = \log_\theta(\alpha + 12) = 3066 & a_{13} = \log_\theta(\alpha + 13) = 1799 & a_{14} = \log_\theta(\alpha + 14) = 1 \\ a_{15} = \log_\theta(\alpha + 15) = 361 & a_{16} = \log_\theta(\alpha + 16) = 3849 & \end{array}$$

   Ahora, considere la permutación π del conjunto $\{0, 1, \ldots, 16\}$ dada por

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 1 & 5 & 3 & 7 & 15 & 11 & 2 & 8 & 6 & 4 & 10 & 16 & 14 & 0 & 12 & 13 & 9 \end{pmatrix}$$

   de manera que

$$\begin{array}{llllll} b_0 = 2142 & b_1 = 1445 & b_2 = 4226 & b_3 = 3465 & b_4 = 351 & b_5 = 959 \\ b_6 = 4400 & b_7 = 1730 & b_8 = 3311 & b_9 = 335 & b_{10} = 4482 & b_{11} = 3849 \\ b_{12} = 1 & b_{13} = 1548 & b_{14} = 3066 & b_{15} = 1799 & b_{16} = 1414 & \end{array}$$

5. A continuación se añade el ruido $r = 2028$ y calculan los valores de la clave pública

$$\begin{array}{llllll} c_0 = 4170 & c_1 = 3473 & c_2 = 1342 & c_3 = 581 & c_4 = 2389 & c_5 = 2987 \\ c_6 = 1516 & c_7 = 3758 & c_8 = 427 & c_9 = 2363 & c_{10} = 1598 & c_{11} = 965 \\ c_{12} = 2029 & c_{13} = 3576 & c_{14} = 182 & c_{15} = 3827 & c_{16} = 3442 & \end{array}$$

Una vez obtenidas las claves, se procederá a cifrar el mensaje "Viva Colombia" de la siguiente forma, primero se escribirá el mensaje por medio del código ASCII, de manera que cada letra se escribirá con una longitud de $\lfloor \log_2 \binom{17}{3} \rfloor = 9$ bits.

$$\{001010110,\ 001101001,\ 001110110,\ 001100001,\ 001000000,\ 001000011,\ 001101111,$$
$$001101100,\ 001101111,\ 001101101,\ 001100010,\ 001101001,\ 001100001\}$$

Luego, se obtiene el número decimal que corresponde a cada bloque

$$212, 300, 220, 268, 4, 388, 492, 108, 492, 364, 140, 300, 268$$

A partir de los valores anteriores, se determinan sus correspondientes vectores binarios de tamaño $q = 17$ y peso $h = 3$ usando el Algoritmo 5

$m_0 = (0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0)$ $m_1 = (0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0,0)$
$m_2 = (0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1)$ $m_3 = (0,0,0,0,1,0,1,0,0,0,0,0,0,1,0,0,0)$
$m_4 = (0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1)$ $m_5 = (0,0,1,0,0,0,0,0,0,1,0,0,0,1,0,0,0)$
$m_6 = (0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0)$ $m_7 = (0,0,0,0,0,0,0,1,0,1,0,0,0,1,0,0,0)$
$m_8 = (0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0)$ $m_9 = (0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,1)$
$m_{10} = (0,0,0,0,0,0,1,0,0,0,1,1,0,0,0,0,0)$ $m_{11} = (0,0,0,1,0,0,0,0,0,0,0,1,1,0,0,0,0)$
$m_{12} = (0,0,0,0,1,0,1,0,0,0,0,0,0,1,0,0,0)$

Por último, se cifra el mensaje original, de donde

$$C = \{4685, 3575, 4746, 2569, 4386, 2369, 1234, 4785, 1234, 3699, 4079, 3575, 2569\}$$

Para el proceso de descifrado, se parte de C y se determinan los valores de s_i, $0 \leq i \leq n-1$ donde n es el número de elementos de C, esto es

$s_0 = 3513$ $s_1 = 2403$ $s_2 = 3574$ $s_3 = 1397$
$s_4 = 3214$ $s_5 = 1197$ $s_6 = 62$ $s_7 = 3613$
$s_8 = 62$ $s_9 = 2527$ $s_{10} = 2907$ $s_{11} = 2403$
$s_{12} = 1397$

Luego, se calculan los polinomios $q_i(\alpha)$

$$\begin{aligned}
q_0(\alpha) &= 4x^2 + 10x + 5 & q_1(\alpha) &= 16x^2 + 11x \\
q_2(\alpha) &= 16x^2 + 7x & q_3(\alpha) &= 13x^2 + 15x + 13 \\
q_4(\alpha) &= 15x^2 + 2x + 2 & q_5(\alpha) &= 3x^2 + 14x + 13 \\
q_6(\alpha) &= 5x^2 + 7x + 6 & q_7(\alpha) &= 8x^2 + 13 \\
q_8(\alpha) &= 5x^2 + 7x + 6 & q_9(\alpha) &= 4x^2 + 15x + 7 \\
q_{10}(\alpha) &= 7x^2 + 10x + 10 & q_{11}(\alpha) &= 16x^2 + 11x
\end{aligned}$$

y se factorizan las sumas de los polinomios $q_i(x) + p(x)$

$$\begin{aligned}
q_0(x) + p(x) &= (x+2)(x+11)(x+12) & q_1(x) + p(x) &= (x+7)(x+14)(x+16) \\
q_2(x) + p(x) &= (x+9)(x+13)(x+15) & q_3(x) + p(x) &= (x+9)(x+13)(x+14) \\
q_4(x) + p(x) &= x(x+3)(x+4) & q_5(x) + p(x) &= (x+5)(x+8)(x+13) \\
q_6(x) + p(x) &= x(x+4)(x+8) & q_7(x) + p(x) &= (x+5)(x+8)(x+13) \\
q_8(x) + p(x) &= (x+3)(x+9)(x+13) & q_9(x) + p(x) &= (x+2)(x+10)(x+16) \\
q_{10}(x) + p(x) &= (x+7)(x+14)(x+16) & q_{11}(x) + p(x) &= x(x+2)(x+15)
\end{aligned}$$

Una vez factorizados estos polinomios, se reordenan sus raíces según la permutación π, con lo que se obtienen los elementos del campo

$$\begin{array}{ccc}
\{5,6,14\} & \{3,11,12\} & \{4,15,16\} \\
\{4,6,13\} & \{12,15,16\} & \{2,9,13\} \\
\{1,7,15\} & \{7,9,13\} & \{2,15,16\} \\
\{6,10,11\} & \{3,11,12\} & \{4,6,13\}
\end{array}$$

Estos elementos indican la posición en la que se encuentra el bit 1 en cada mensaje parcial m_i. Por último, utilizando el Algoritmo 6, se transforma cada vector en un entero, obteniendo

$$212, 300, 220, 268, 4, 388, 492, 108, 492, 364, 140, 300, 268$$

los cuales al ser transformados a base binaria, se obtiene el mensaje binario original. De modo que es posible recuperar el mensaje inicial “Viva Colombia”.

4.3. Rendimiento del Criptosistema

Se analizarán tres parámetros básicos de un criptosistema

- Tiempo necesario para cifrar y descifrar un mensaje.
- El tamaño de las claves.
- Tasa de información en términos de la razón entre el número de bits del texto plano y el número de bits del texto cifrado.

4.3.1. Tiempo de cifrado y descifrado de un mensaje

Dado un mensaje binario de tamaño q y peso h, el proceso de cifrado consiste en sumar como máximo h enteros, cada uno más pequeño que q^h, de donde su tiempo de ejecución es del orden de $O(h^2 \log q)$. Por otro lado, el tiempo de ejecución para descifrar es mucho mayor puesto que este proceso es basado en la exponencial modular. Elevar un polinomio $q(x)$ de grado menor que h a una potencia en el rango $[1, q^h - 1]$ toma al menos $2h \log q$ multiplicaciones modulares. El modulo $p(x)$ es un polinomio de grado h con coeficientes en $\mathbb{F}_q$. Para realizar una multiplicación modular en $\mathbb{F}_q$ bastan $2h^2$ operaciones utilizando un algoritmo estándar de multiplicación de polinomios. De manera que en total son necesarias $4h^3 \log q$ operaciones en el campo $\mathbb{F}_q$, esto es $O(h^3 \log^2 q)$ operaciones en el cuerpo base. En particular cuando $q = 200$ y $h = 25$ (los parámetros originales propuesto por Chor y Rivest) este cálculo toma aproximadamente 500000 operaciones en $\mathbb{F}_q$, lo cual es muy favorable comparándolo con el tiempo de cifrado y descifrado del criptosistema RSA.

4.3.2. Tamaño de las claves

El tamaño de las claves, especialmente el tamaño de la clave pública es un factor importante en el diseño de cualquier criptosistema de clave pública. En este criptosistema el tamaño de la clave pública viene dado por q números en el intervalo $[1, q^h - 1]$. En términos de bits, estos son $q \log_2 q^h = qh \log_2 q$ bits. Para $p = 200$, $h = 25$ la clave pública ocupa menos de 40000 bits. A pesar que este número es mucho más grande que el propuesto por el RSA, aún se mantiene en los margenes aceptables para implementar el sistema.

4.3.3. Tasa de información

Como se mostró en el Capitulo 3, uno de los factores importantes en un criptosistema tipo mochila para evaluar su seguridad, es la densidad. Por tanto, es necesario conocer los valores aproximados para la densidad del sistema Chor-Rivest. Una cota inferior de la densidad de este criptosistema se puede obtener a partir de lo expuesto en 4.2.1. Como la permutación π no afecta al máx $\mathcal{A}$ y como $c_i \leq q^h - 1$, se tiene

$$\log_2 c_i \leq \log_2(q^h - 1) < \log_2 q^h = h \log_2 q$$

de donde

$$d = \frac{n}{\log_2(\text{máx}\,\mathcal{A})} \geq \frac{q}{h \log_2 q}$$

Para los valores propuesto por Chor y Rivest, $q = 197$ y $h = 24$, se obtiene $d \geq 1{,}0769$ la cual es mayor que 1. Por lo tanto, este criptosistema en principio no puede ser roto por los criptoanálisis basados en bajas densidades.

Por otro lado, la tasa de información de un criptosistema está definido como la razón entre el número de bits del texto plano y el número de bits del texto cifrado, de donde el criptosistema Chor-Rivest para los parámetros $q = 197$ y $h = 24$, la tasa de información es

$$R = \frac{\log \binom{q}{h}}{\log q^h} = 0{,}556$$

Para los parámetros restantes propuestos por Chor y Rivest, los valores de sus densidades y tasas de información son los siguientes

$$\begin{aligned} 211^{24} \rightarrow &\quad d = 1{,}139, \qquad R = 0{,}563 \\ 243^{24} \rightarrow &\quad d = 1{,}278, \qquad R = 0{,}576 \\ 256^{25} \rightarrow &\quad d = 1{,}280, \qquad R = 0{,}573 \end{aligned}$$

Capítulo 5

Criptoanálisis al sistema Chor-Rivest por Vaudenay

Chor y Rivest en [6] presentan como se genera y funciona el criptosistema, además muestran posibles ataques a este, entre estos se encuentran los ataques especializados (aquellos donde el criptonalista trata de reconstruir la clave privada a partir de la clave pública y de algún tipo de información parcial sobre la clave privada). Los ataques que se muestran son los siguientes

- Ataque conocido el generador del grupo multiplicativo g y el ruido r
- Ataque conocido la raíz α y el ruido r
- Ataque conocido α por Oded Goldreich
- Ataque conocido π por Odlyzko

También Chor y Rivest presentan un ataque donde no se necesita conocer información sobre la clave privada, este es el ataque de baja densidad propuesto por Lagarias y Odlyzko el cual es efectivo para mochilas de baja densidad (para más información ver [6]).

En este capítulo se presentará de forma detallada el ataque propuesto por S. Vaudenay, [11]. En primer lugar, se estudiarán algunas propiedades de las claves en este criptosistema que permiten reducir los ataques al hecho de buscar alguna de las claves que son equivalentes a la clave empleada en el criptosistema. Finalmente, se estudiarán dos ataques especializados presentados y desarrollados por Vaudenay y se presentará el criptoanálisis desarrollado por

Vaudenay donde se prueba que es posible romper el sistema para los parámetros propuestos por Chor y Rivest en [6].

5.1. Claves privadas simétricas

Recuerde que para la generación de las claves en el criptosistema Chor-Rivest se considera el campo finito $\mathbb{F}_{q^h}$ donde q es la potencia de un número primo y h un entero mayor o igual que 2. La clave privada estará conformada por los siguientes parámetros

- Una raíz $\alpha \in \mathbb{F}_{q^h}$ de un polinomio $f(x) \in \mathbb{F}_q[x]$ mónico e irreducible de grado h.
- Un generador θ del grupo multiplicativo $\mathbb{F}_{q^h}^*$.
- Un entero r tal que $0 \leq r \leq q^h - 2$.
- Una permutación π del conjunto $\{0, 1, \ldots, q-1\}$.

Observe que una cuenta sencilla permite mostrar que se pueden tener hasta $h \cdot \phi(q^h - 1) \cdot (q^h - 2) \cdot q!$ claves privada distintas.
Por otro lado, la clave pública será la mochila determinada por todos los c_i de la forma

$$c_i = \left(r + \log_\theta(\alpha + t_{\pi(i)})\right) \text{ mód } (q^h - 1),\ i = 0, 1, \ldots, q-1$$

Vaudenay muestra que existen diferentes claves privadas que dan lugar a la misma clave pública, de modo que dada una clave pública, cualquier clave privada que sea equivalente a la original permitirá romper el criptosistema.

En primer lugar, observe que si se reemplaza α y θ por su q^j-ésima potencia la clave pública no cambia puesto que

$$\log_{\theta^{q^j}}(\alpha^{q^j} + t_{\pi(i)}) = \frac{1}{q^j} \log_\theta(\alpha + t_{\pi(i)})^{q^j} = \log_\theta(\alpha + t_{\pi(i)})$$

En segundo lugar, si se reemplaza (α, t_π) por $(\alpha + z, t_\pi - z)$ para cualquier $z \in \mathbb{F}_q$ se obtiene la misma clave pública. Finalmente, si se reemplazan (α, r, t_π) por $(u\alpha, r - \log_\theta u, ut_\pi)$ para todo $u \in \mathbb{F}_q^*$ se obtiene nuevamente la misma clave pública. De lo anterior se puede concluir que por lo menos existen $q \cdot (q-1) \cdot h$ claves privadas equivalentes.

Las propiedades de equivalencia en las claves privadas mostradas anteriormente sugieren estudiar el polinomio $\prod_{i=1}^{h}\left(x-\alpha^{q^j}\right)$ del que los α's son raíces equivalentes.

5.2. Ataque conocida la norma del generador

En esta sección se estudiará el ataque al criptosistema Chor-Rivest dada la permutación π y la norma del generador θ del grupo $\mathbb{F}_{q^s}^*$ donde s es un divisor de h, dado que se hará uso de éste en el ataque en el que sólo se conoce la norma del generador $g_{q^s} \in \mathbb{F}_{q^s} \subset \mathbb{F}_{q^h}$. De aquí en adelante q será un número primo y no la potencia de un número primo a no ser que se diga lo contrario.

El siguiente resultado será de gran utilidad en lo que sigue de esta sección.

Teorema 5.1. *Para cualquier factor primo s de h, existe un generador $\bar{\theta}$ del grupo multiplicativo del subcampo $\mathbb{F}_{q^s}$ de $\mathbb{F}_{q^h}$ y un polinomio Q de grado h/s con coeficientes en $\mathbb{F}_{q^s}$, de modo que $-\alpha$ es una raíz, y tal que para cualquier i, se tiene*

$$Q(t_{\pi(i)}) = \bar{\theta}^{c_i}$$

Demostración. Sea $\bar{\theta}$ la norma de θ en $\mathbb{F}_{q^s} \subseteq \mathbb{F}_{q^h}$, esto es

$$\begin{aligned}\bar{\theta} &= \theta_{q^s} = N_{\mathbb{F}_{q^s}}^{\mathbb{F}_{q^h}}(\theta) = \theta^{1+q^s+q^{2s}+\cdots+q^{\left(\frac{h}{s}-1\right)s}} = \prod_{i=0}^{\frac{h}{s}-1}\theta^{q^{is}} \\ &= \theta^{\frac{q^h-sq^s-1}{q^s-1}} = \theta^{\frac{q^h-1}{q^s-1}}\end{aligned}$$

Sea

$$Q(x) = \theta_{q^s}^r \prod_{i=0}^{\frac{h}{s}-1}\left(x+\alpha^{q^{is}}\right) \tag{5.1}$$

Si $x \in \mathbb{F}_{q^s}$ entonces se tiene

$$\begin{aligned}x^{q^s} &= x \\ x^{q^{2s}} &= \left(x^{q^s}\right)^{q^s} = x^{q^s} = x \\ &\vdots \\ x^{q^{is}} &= x\end{aligned}$$

De donde

$$Q(x) = (\theta_{q^s})^r \prod_{i=0}^{\frac{h}{s}-1} (x+\alpha)^{q^{is}} = (\theta_{q^s})^r N_{\mathbb{F}_{q^s}}^{F_{q^h}}(x+\alpha)$$

Ahora, como la norma relativa de un elemento del cuerpo $\mathbb{F}_{q^s}$ cae en el mismo cuerpo, se tiene que $Q(x) \in \mathbb{F}_{q^s}$ para cualquier $x \in \mathbb{F}_{q^s}$. Por otro lado, como $q^s > h/s$ entonces todos los coeficientes de $Q(x)$ están en $\mathbb{F}_{q^s}$.

Por la Ecuación 5.1 y teniendo en cuenta que $C_i = r + \log_\theta(\alpha + t_{\pi(i)})$ se tiene

$$\alpha + t_{\pi(i)} = \theta^{c_i - r} = \frac{\theta^{c_i}}{\theta^r}$$

de donde

$$\begin{aligned} Q(t_{\pi(i)}) &= (\theta_{q^s})^r \prod_{i=0}^{\frac{h}{s}-1} \left(t_{\pi(i)} + \alpha^{q^{is}}\right) = (\theta_{q^s})^r \prod_{i=0}^{\frac{h}{s}-1} (\alpha + t_{\pi(i)})^{q^{is}} \\ &= (\theta_{q^s})^r \prod_{i=0}^{\frac{h}{s}-1} \left(\frac{\theta^{c_i}}{\theta^r}\right)^{q^{is}} = (\theta_{q^s})^r \frac{\prod_{i=0}^{\frac{h}{s}-1} \left(\theta^{q^{is}}\right)^{c_i}}{\prod_{i=0}^{\frac{h}{s}-1} \left(\theta^{q^{is}}\right)^r} \\ &= \prod_{i=0}^{\frac{h}{s}-1} \left(\theta^{q^{is}}\right)^{c_i} = (\theta_{q^s})^{c_i} \end{aligned}$$

□

5.2.1. Ataque conocida la norma del generador θ_{q^s} y la permutación π

Si se conoce θ_{q^s} y la permutación π se puede interpolar el polinomio $Q(x)$ (garantizado por el Teorema 5.1) con $h/s+1$ pares de la forma $\left(t_{\pi(i)}, \theta_{q^s}^{c_i}\right)$. Con esto se obtendrá un polinomio de grado h/s cuyas raíces son conjugadas de $-\alpha$. Resolviendo este último polinomio se obtiene la raíz α y se puede llevar a cabo cualquiera de los ataques en los que t es conocido (ver [6], [11]).

Suponga que sólo se conoce el valor de la norma θ_{q^s} en $\mathbb{F}_{q^s} \subset \mathbb{F}_{q^h}$. Sea $\{i_0, i_1, \ldots, i_{h/s}\}$ un conjunto de $h/s+1$ índices entre 0 y $q-1$, distintos dos a dos. Por la fórmula de interpolación de Lagrange para n puntos el polinomio $Q(x)$ puede interpolarse para todos los $\left(t_{\pi(i)}, \theta_{q^s}^{c_{i_j}}\right)$ obteniendo la relación

$$Q(t_{\pi(i)}) = \theta_{q^s}^{c_i} = \sum_{j=0}^{h/s} \theta_{q^s}^{c_{i_j}} \prod_{\substack{k=0 \\ k \neq j}} \frac{t_{\pi(i)} - t_{\pi(i_k)}}{t_{\pi(i_j)} - t_{\pi(i_k)}}, \qquad i = 0, 1, \ldots, q-1 \tag{5.2}$$

La expresión anterior se puede reescribir de la siguiente forma

$$\theta_{q^s}^{c_i} - \theta_{q^s}^{c_{i_0}} = \sum_{j=1}^{h/s} \left(\theta_{q^s}^{c_{i_j}} - \theta_{q^s}^{c_{i_0}}\right) \prod_{\substack{k=0 \\ k\neq j}} \frac{t_{\pi(i)} - t_{\pi(i_k)}}{t_{\pi(i_j)} - t_{\pi(i_k)}} \tag{5.3}$$

En efecto,

$$\begin{aligned}\sum_{j=1}^{h/s} \left(\theta_{q^s}^{c_{i_j}} - \theta_{q^s}^{c_{i_0}}\right) \prod_{\substack{k=0 \\ k\neq j}} \frac{t_{\pi(i)} - t_{\pi(i_k)}}{t_{\pi(i_j)} - t_{\pi(i_k)}} &= \sum_{j=1}^{h/s} \theta_{q^s}^{c_{i_j}} \prod_{\substack{k=0 \\ k\neq j}} \frac{t_{\pi(i)} - t_{\pi(i_k)}}{t_{\pi(i_j)} - t_{\pi(i_k)}} + \theta_{q^s}^{c_{i_0}} \prod_{k>0} \frac{t_{\pi(i)} - t_{\pi(i_k)}}{t_{\pi(i_j)} - t_{\pi(i_k)}} \\ &\quad -\theta_{q^s}^{c_{i_0}} \sum_{j=1}^{h/s} \prod_{\substack{k=0 \\ k\neq j}} \frac{t_{\pi(i)} - t_{\pi(i_k)}}{t_{\pi(i_j)} - t_{\pi(i_k)}} - \theta_{q^s}^{c_{i_0}} \prod_{k>0} \frac{t_{\pi(i)} - t_{\pi(i_k)}}{t_{\pi(i_j)} - t_{\pi(i_k)}}\end{aligned}$$

Ahora, como para cualquier $i = i_0, i_1, \ldots, i_{h/s}$ se tiene

$$\sum_{j=1}^{h/s} \prod_{\substack{k=0 \\ k\neq j}} \frac{t_{\pi(i)} - t_{\pi(i_k)}}{t_{\pi(i_j)} - t_{\pi(i_k)}} = 1$$

Entonces

$$\sum_{j=1}^{h/s} \left(\theta_{q^s}^{c_{i_j}} - \theta_{q^s}^{c_{i_0}}\right) \prod_{\substack{k=0 \\ k\neq j}} \frac{t_{\pi(i)} - t_{\pi(i_k)}}{t_{\pi(i_j)} - t_{\pi(i_k)}} = \theta_{q^s}^{c_i} - \theta_{q^s}^{c_{i_0}} \sum_{j=1}^{h/s} \prod_{\substack{k=0 \\ k\neq j}} \frac{t_{\pi(i)} - t_{\pi(i_k)}}{t_{\pi(i_j)} - t_{\pi(i_k)}} = \theta_{q^s}^{c_i} - \theta_{q^s}^{c_{i_0}}$$

En seguida se presentarán dos algoritmos que permiten encontrar la permutación π, de manera que si se conoce la norma θ_{q^s} y la permutación π entonces se podrá efectuar el ataque mencionado al inicio de esta subsección.

5.2.2. Algoritmos para encontrar la permutación π

Teniendo en cuenta la simetría de las claves los valores de πi_1 y $\pi(i_2)$ puede elegirse de manera aleatoria. Este algoritmo consiste en una búsqueda exhaustiva de los valores de π_{i_j} para $j = 0, 3, 4, \ldots, h/s$ hasta que la Ecuación 5.2 proporcione una permutación consistente, es decir, que sea compatible con la clave pública.

Algoritmo 7 Primer algoritmo de búsqueda de la permutación π

Entrada: Descriptores del cuerpo finito $\mathbb{F}_{q^h}$, un divisor s de h, la norma θ_{q^s} y la clave pública $\{c_0, c_1, \ldots, c_{q-1}\}$

Salida: Una clave privada cuya correspondiente clave pública es $\{c_0, c_1, \ldots, c_{q-1}\}$

1: Seleccione un conjunto de índices $\{i_0, i_1, \ldots, i_{h/s}\}$ en $\{0, 1, \ldots, q-1\}$ distintos dos a dos.
2: Elija de manera aleatoria los valores de $\pi(i_1)$ y $\pi(i_2)$ en $\{0, 1, \ldots, q-1\}$ de manera que sean distintos
3: Para todos los posibles valores de $\pi(i_0), \pi(i_3), \pi(i_4), \ldots, \pi(i_{h/s})$ defina $S = \{\pi(i_0), \ldots\}$ y luego hacer
4: Para todo $z \notin S$ se calcula el segundo término de la Ecuación 5.2 con $t_{\pi(i)} = t_z$.
5: Si el resultado es igual a $\theta_{q^s}^{c_i}$, de manera que $\pi(i)$ no sea definido, entonces se considera $\pi(i) = z$ en caso contrario se vuelve al paso 3
6: Ejecutar el ataque cuando se conoce θ_{q^s} y la permutación π

Si s es suficientemente grande, existe un segundo algoritmo, que es mucho más eficiente que el algoritmo anterior. Si $h/r \leq r$, esto es, si $r \geq \sqrt{h}$ los coeficientes en la Ecuación 5.2 son los únicos coeficientes en $\mathbb{F}_q$ que permiten escribir el valor de $\theta_{q^s}^{c_{i_j}} - \theta_{q^s}^{c_{i_0}}$ en la base dada por $\{\theta_{q^s}^{c_{i_1}} - \theta_{q^s}^{c_{i_0}}, \ldots, \theta_{q^s}^{c_{i_{h/s}}} - \theta_{q^s}^{c_{i_0}}\}$. Sea a_j^i el coeficiente de $\theta_{q^s}^{c_{i_j}} - \theta_{q^s}^{c_{i_0}}$ en la expresión de $\theta_{q^s}^{c_{i_j}} - \theta_{q^s}^{c_{i_0}}$ para $j = 1, 2, \ldots, h/s$. Entonces se tiene

$$\begin{aligned}
\frac{a_2^i}{a_1^i} &= \frac{\prod_{\substack{k=0 \\ k\neq 2}}^{h/s} \frac{t_{\pi(i)} - t_{\pi(i_k)}}{t_{\pi(i_2)} - t_{\pi(i_k)}}}{\prod_{\substack{k=0 \\ k\neq 1}}^{h/s} \frac{t_{\pi(i)} - t_{\pi(i_k)}}{t_{\pi(i_1)} - t_{\pi(i_k)}}} \\
&= \frac{\prod_{\substack{k=0 \\ k\neq 1}}^{h/s} (t_{\pi(i_1)} - t_{\pi(i_k)})}{\prod_{\substack{k=0 \\ k\neq 2}}^{h/s} (t_{\pi(i_2)} - t_{\pi(i_k)})} \frac{t_{\pi(i)} - t_{\pi(i_1)}}{t_{\pi(i)} - t_{\pi(i_2)}} \\
&= u \frac{t_{\pi(i)} - t_{\pi(i_1)}}{t_{\pi(i)} - t_{\pi(i_2)}} \qquad (5.4)
\end{aligned}$$

con $u \in \mathbb{F}_q$ tal que u no depende de i. Por lo tanto, si se escoge de manera aleatoria i_j, con $j = 0, 1, \ldots, h/s$, entonces se pueden escribir todos los $\theta_{q^s}^{c_{i_j}} - \theta_{q^s}^{c_{i_0}}$ en base $\{\theta_{q^s}^{c_{i_0}} - \theta_{q^s}^{c_{i_0}}, \ldots, \theta_{q^s}^{c_{i_{h/s}}} - \theta_{q^s}^{c_{i_0}}\}$. Ahora, si se elige el elemento $u \in \mathbb{F}_q$ los valores de $\pi(i)$ se pueden obtener por medio de la Ecuación 5.4.

Algoritmo 8 Segundo algoritmo de búsqueda de la permutación π

Entrada: Descriptores del cuerpo finito $\mathbb{F}_{q^h}$, un divisor s de h, la norma θ_{q^s} y la clave pública $\{c_0, c_1, \ldots, c_{q-1}\}$

Salida: Una clave privada cuya correspondiente clave pública es $\{c_0, c_1, \ldots, c_{q-1}\}$

1: Se elige un conjunto de índices $\{i_0, i_1, \ldots, i_{h/s}\}$ en $\{0, 1, \ldots, q-1\}$ distintos dos a dos

2: Se precalcula la matriz de cambio de base para la base

$$\theta_{q^s}^{c_{i_1}} - \theta_{q^s}^{c_{i_0}}, \ldots, \theta_{q^s}^{c_{i_{h/s}}} - \theta_{q^s}^{c_{i_0}}$$

3: Se elige aleatoriamente $\pi(i_1)$ y $\pi(i_2)$ de tal forma que sean diferentes en $\{0, 1, \ldots, q-1\}$

4: Para todo $v \in \mathbb{F}_q$ hacer lo siguiente

5: Para todo i escribir $\theta_{q^s}^{c_i} - \theta_{q^s}^{c_{i_0}}$ en la base dada y obtener a_0^i y a_1^i

6: De la expresión 5.4 obtener $\pi(i)$. Si este valor no es consistente con los $\pi(i')$ ya calculados, se vuelve al paso 4.

5.2.3. Encontrando la norma θ_{q^s}

La Ecuación 5.3 muestra que los valores de todos los $\theta_{q^s}^{c_i}$ se encuentran en el mismo subespacio afín $h/s-$dimensional de $\mathbb{F}_{q^s}$ sobre $\mathbb{F}_q$. Por lo tanto, si se asume que $h/s + 1 \leq s$, es decir. $s \geq \sqrt{h + \frac{1}{4}} + \frac{1}{2}$ entonces se puede elaborar un algoritmo que permita determinar θ_{q^s}.

Hecho 5.2. *Si existe un factor primo s de h tal que $s \geq \sqrt{h + \frac{1}{4}} + \frac{1}{2}$, entonces todos los valores de las normas $\theta_{q^s}^{c_i}$, se encuentran en el mismo espacio afín $h/s-$dimensional cuando se considera $\mathbb{F}_{q^s}$ como un espacio de dimension s sobre $\mathbb{F}_q$*

La existencia de este factor primo s de h puede verse como una condición inadecuada para llevar a cabo este ataque, pero dado que los parámetros del criptosistema Chor-Rivest se deben elegir de manera que sea fácil calcular los logaritmos discretos, se conoce con anterioridad que h tiene muchos factores primos. Si h no tiene tales factores entonces h es un número primo o el cuadrado de un número primo.

Teorema 5.3. *Si un número entero h no tiene factores primos s que verifiquen $s \geq \sqrt{h + \frac{1}{4}} + \frac{1}{2}$ entonces h es un número primo o es el cuadrado de un número primo.*

Demostración. Suponga que h no es primo y sea s un divisor primo de h tal que $s < \sqrt{h + \frac{1}{4}} + \frac{1}{2}$. Como s es divisor de h entonces $h = a \cdot s$ con $a \in \mathbb{Z}$. Si $a = s$ entonces $h = s^2$ lo que verificaría el teorema.

Suponga que $s > a$, como $s < \sqrt{h + \frac{1}{4}} + \frac{1}{2}$ entonces

$$\begin{aligned} s &< \sqrt{h + \frac{1}{4}} + \frac{1}{2} \\ \left(s - \frac{1}{2}\right)^2 &< h + \frac{1}{4} \\ s^2 - s + \frac{1}{4} &< a \cdot s + \frac{1}{4} \\ s^2 - s &< a \cdot s \\ s &< a - 1 \end{aligned}$$

lo cual es una contradicción ya que $s > a$. □

El siguiente algoritmo permite comprobar si un determinado candidato puede ser la norma de un generador θ_{q^s}. Este algoritmo comprueba si todos los valores $\theta_{q^s}^{c_i}$ son afínmente dependientes.

Algoritmo 9 Primer algoritmo de búsqueda de θ_{q^s}

Entrada: Descriptores del cuerpo finito $\mathbb{F}_{q^h}$, un divisor s de h y la clave pública $\{c_0, c_1, \ldots, c_{q-1}\}$
Salida: Posibles valores para la norma θ_{q^s}

1: Elegir un conjunto de índices $\{i_0, i_1, \ldots, i_{h/s}\} \in \{0, 1, \ldots, q-1\}$ distintos dos a dos.
2: Elegir un generador $\theta_{q^s}\mathbb{F}_{q^s}$
3: Determinar el espacio afín definido por $\{\theta_{q^s}^{c_{i_0}}, \ldots, \theta_{q^s}^{c_{i_{h/s}}}\}$, es decir

$$\{\theta_{q^s}^{c_{i_1}} - \theta_{q^s}^{c_{i_0}}, \ldots, \theta_{q^s}^{c_{i_{h/s}}} - \theta_{q^s}^{c_{i_0}}\}$$

4: Para los valores restantes de i, comprobar si los $\theta_{q^s}^{c_i}$ están en dicho espacio afín. En caso contrario ir al paso 2.
5: Ejecutar el ataque conocida la norma $\theta_{q^s}^{c_i}$

Se puede utilizar un algoritmo más eficiente si se emplean todos los factores de h en lugar de uno solo.

Hecho 5.4. *Si $Q(x)$ es un polinomio sobre $\mathbb{F}_{q^s}$ de grado d y si e es un entero tal que $1 \leq e \leq (q-1)/d$ entonces se tiene*

$$\sum_{t \in \mathbb{F}_q} Q(t)^e = 0$$

Con esta propiedad se puede buscar el valor de θ_{q^s} de forma más eficiente puesto que se puede trabajar en cualquier subcampo. El objetivo ahora, es calcular θ_{q^s} siendo s un divisor de h suficientemente grande. Para ello, se consideran los factores $s_1, \ldots, s_k$ de s y los valores de

$\theta_{q^{s_i}}$. Como

$$\theta_{q^{s_i}} = \theta_{q^s}^{1+q^{s_i}+q^{2s_i}+\cdots+q^{s-s_i}}$$

se tiene

$$\log_\sigma \theta_{q^s} = \frac{\log_\sigma \theta_{q^{s_i}}}{1+q^{s_i}+q^{2s_i}+\cdots+q^{s-s_i}} \pmod{q^{s_i}-1} \tag{5.5}$$

donde $\sigma \in \mathbb{F}_{q^s}$ es un elemento primitivo. El hecho de conocer todas las $\theta_{q^{s_i}}$ permite determinar el valor de $\log \theta_{q^s}$ módulo $l = \text{mcm}(q^{s_1}-1, \ldots, q^{s_k}-1)$, para lo que es necesario utilizar el Teorema Chino de los Restos cuando los módulos considerados no sean primos dos a dos.
Sea $y \equiv \log_\sigma \theta_{q^s} \pmod{l}$, entonces

$$\begin{aligned} \log_\sigma \theta_{q^s} &= y + lx \\ &= \log_\sigma \beta + \log_\sigma \sigma^{lx} \\ &= \log_\sigma(\beta\sigma^{lx}) \end{aligned}$$

es decir, se tiene que $\theta_{q^s} = \beta\sigma^{lx}$. Dado que σ es un elemento primitivo de $\mathbb{F}_{q^s}$ se tiene $0 \leq x \leq (q^s-1)/l$. De lo anterior, se deduce que sólo es necesario ejecutar el siguiente algoritmo $(q^s-1)/l$ veces, lo que supone $O(qs^2)$ operaciones en $\mathbb{F}_q$.

Algoritmo 10 Segundo algoritmo de búsqueda de θ_{q^s}

Entrada: Descriptores del cuerpo finito $\mathbb{F}_{q^h}$, factores s_i de s siendo s un divisor de h tal que $s \geq \sqrt{h+\frac{1}{4}}+\frac{1}{2}$, la clave pública $\{c_0, c_1, \ldots, c_{q-1}\}$ y los valores de $\theta_{q^{s_i}}$ para $i = 1, 2, \ldots, k$
Salida: Posibles valores para la norma θ_{q^s}

1: Resolver el sistema de ecuaciones dado en 5.5 para $i = 1, 2, \ldots, k$ y obtener el valor $\theta_{q^s} = \beta\sigma^{lx}$
2: Para $x = 0, \ldots, \frac{q-1}{l} - 1$ hacer
3: Calcular

$$\sum_{e=1}^{\frac{(q-1)s}{h}-1} \beta^{ec_i}\sigma^{ec_i lx}$$

si este valor es no nulo, volver a 2.
4: retornar $\theta_{q^s} = \beta\sigma^{lx}$

Nota 5.1. Como se vio en la sección anterior, para poder llevar a cabo el ataque de Vaudenay presentado es necesario conocer todos los valores de la clave pública. Por otro lado, Lenstra en [18] conjeturó que el hecho de conocer todos los valores de la clave pública puede suponer una debilidad para el criptosistema Chor-Rivest.

En [17], Queiruga Dios propone una nueva versión del criptosistema Chor-Rivest en la que el

emisor de un mensaje no conoce algunos valores de la clave pública.

El proceso de cifrado y descifrado de mensajes, es esencialmente el mismo presentado en el Capítulo 4. La única diferencia es que en la nueva versión solo se utilizan y se dan a conocer w elementos de la mochila pública ($w < q$).

Apéndice A

El nuevo concepto de densidad y el criptosistema de Chor-Rivest

Como se vio en la Sección 2.2, un concepto muy importante relacionado con las mochilas es el concepto de densidad, el cual de cierta forma permite medir el tamaño de sus elementos, este concepto fue introducido por Lagarias y Odlyzko. Ellos mostraron que un sistema tipo mochila puede ser roto si su densidad es lo suficientemente baja, es más, ellos propusieron una cota para determinar cuándo un sistema tipo mochila es de baja densidad. Coster et al. propusieron un algoritmo mejorado, que puede resolver este tipo de sistema con una densidad mayor a la propuesta por Lagarias y Odlyzko.

Los diseñadores de criptosistemas tipo mochila decidieron reducir el peso Hamming del mensaje con el fin de prevenir los ataques de baja densidad.

Omura-Tanaka e Izu et al. analizaron la seguridad de los criptosistemas tipo mochila con bajo peso Hamming y mostraron que este tipo de criptosistemas pueden ser rotos por medio del *Lattice attack*, incluso si la densidad es mayor que 1. En *Asiacrypt2005*, Nguyen y Stern [9] introducen una variante del concepto de densidad denominada "pseudo-densidad" y muestran que los sistemas tipo mochila de bajo peso Hamming pueden ser rotos con alta probabilidad por medio del *SVP/CVP-oracle* siempre que la pseudo-densidad fuera menor que 1. La pseudo-densidad de una mochila $\mathcal{A}$ se define como

Definición A.1. Sea $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ una mochila y sea k fijo el peso Hamming de la

solución del SSP. La pseudo-densidad se define como

$$\kappa(\mathcal{A}) = \frac{k \log_2 n}{\log_2(\text{máx}\, \mathcal{A})}$$

donde k es el peso Hamming.

Nguyen y Stern mostraron que la pseudo-densidad del criptosistema Chor-Rivest es menor que 1, con lo cual se muestra que el sistema puede ser resuelto por medio de *SVP/CVP-oracle*.

En *Africacrypt2008*, Kunihiro [10] introduce un nuevo concepto de densidad y además propone condiciones para determinar si un criptosistema tipo mochila de bajo peso Hamming es vulnerable al *Lattice attack*. El nuevo concepto de densidad integra el concepto usual mostrado en la Sección 2.2 y el concepto de pseudo-densidad.

Definición A.2. Sea $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ una mochila y sea k fijo el peso Hamming de la solución del SSP. La densidad D se define

$$D(\mathcal{A}) = \frac{nH(v)}{\log_2(\text{máx}\, \mathcal{A})}$$

donde $v = k/n$ y H es la función de entropía dada por $H(x) = x \log_2 x + (1-x)\log_2(1-x)$.

Kunihiro propone el siguiente algoritmo para determinar si un criptosistema tipo mochila de bajo peso Hamming es vulnerable al *Lattice attack*

Algoritmo 11 Vulnerabilidad al *Lattice attack*

Entrada: n, k y la mochila $\mathcal{A}$
Salida: Es vulnerable o no al *Lattice attack*

1: $p \leftarrow \frac{k}{n}$
2: $D \leftarrow \frac{nH(p)}{\log_2 \text{máx}\, \mathcal{A}}$
3: $g_{CJ} \leftarrow \frac{H(p)}{p-p^2+(1+p-p^2)H(1/(1+p-p^2))}$
4: Si $D < 0{,}8677$ entonces el sistema es vulnerable
5: Si $D < g_{CJ}$ entonces el sistema es vulnerable
6: $D < H(p)/f(p-p^2)$ entonces el sistema es vulnerable

La Tablas A.1 muestra la densidad usual d, la pseudo-densidad κ, la nuevo densidad D y la cota critica g_{CJ} para el criptosistema Chor-Rivest en los parámetros originales [6] y la Tabla A.2 muestra la misma información para algunos de los parámetros propuestos por Queiruga et al. en [19]. De los datos obtenidos y a partir del Algoritmo 11 se puede verificar que el criptosistema Chor-Rivest es vulnerable al *Lattice attack*.

Tabla A.1: Datos para los parámetros originales del criptosistema Chor-Rivest

q	h	d	κ	D	$g_{CJ}(p)$
197	24	1,08	1,005	0,58	0,87
211	24	1,14	1,002	0,58	0,87
243	24	1,28	1,001	0,59	0,87
256	25	1,28	1,00	0,59	0,87

Tabla A.2: Datos para los parámetros propuestos en [19] para el criptosistema Chor-Rivest

q	h	d	D	$g_{CJ}(p)$
409	17	2,773	0,69	0,88
601	13	5,008	0,75	0,89
631	13	5,218	0,76	0,89
839	13	6,644	0,77	0,89

Kunihiro concluye que el sistema Chor-Rivest es vulnerable al *Lattice attack* para cualquier parámetro, sin embargo se ha observado que la cota critica es superada para ciertos valores de h, actualmente la caracterización de dichos valores aún se está estudiando. La Tabla A.3 muestra algunos datos preliminares de la investigación

Tabla A.3: Posibles nuevos parámetros para el criptosistema Chor-Rivest

q	h	d	D	$g_{CJ}(p)$
211	2	13,664	1,056	0,897
211	3	9,109	0,980	0,891
211	4	6,831	0,926	0,888
211	5	5,466	0,883	0,886
409	2	23,570	1,051	0,904
409	3	15,714	0,983	0,899
409	4	11,785	0,935	0,896
409	5	9,428	0,898	0,894

A.1. Conclusión

Basándose en el nuevo concepto de densidad se puede verificar que los conjuntos de Sidon son óptimos para la generación de claves públicas en el sistema Chor-Rivest (ver Tabla A.4).

q	h	d	D	$g_{LO}(p)$	$g_{CJ}(p)$
409	2	23,570	1,050	0,900	0,904
601	2	32,552	1,047	0,905	0,909
631	2	33,919	1,047	0,906	0,909
839	2	43,191	1,045	0,910	0,911
409	3	15,713	0,982	0,894	0,899
601	3	21,701	0,984	0,900	0,904
631	3	22,612	0,984	0,900	0,904
839	3	28,794	0,985	0,904	0,907

Tabla A.4: Nueva densidad para conjuntos de Sidon

Apéndice B

Implementación en SAGE

B.1. Algoritmos

En este apéndice se presentan algunos de los algoritmos implementados en SAGE que fueron utilizados en este trabajo.

B.1.1. Algoritmo 1: Primitivo Aleatorio

Entrada: Un campo finito $\mathbb{F}_{q^h}$ y los enteros q y h.

Salida: Un elemento primitivo de $\mathbb{F}_{q^h}$

Descripción: El algoritmo toma un elemento aleatorio en el campo finito $\mathbb{F}_{q^h}$ y verifica que sea primitivo.

```
def random_primitive(p,h,F):
    s=p^h-1
    r=F.random_element()
    j=0
    if r!=0:
        for t in prime_factors(s):
            if r^(s/t)==1:
                j=j+1
        if j==0:
            return r
```

```
        else:
            return random_primitive(p,h)
    else:
        return random_primitive(p,h)
```

B.1.2. Algoritmo 2: Construcción Bose-Chowla

Entrada: Un campo finito $\mathbb{F}_{q^h}$, los enteros q y h, un elemento primitivo g y un entero r que será el ruido.

Salida: Un conjunto B_h del tipo Bose-Chowla que será la clave pública.

Descripción: El algoritmo genera un conjunto B_h del tipo Bose-Chowla, el cual será la clave publica del criptosistema Chor-Rivest.

```
def Boseh(p,h,g,F):
    if is_prime(p):
        a=[(log(g+i,g) + r) % (q^h-1) for i in F];
        return a
    else:
        print 'p debe ser primo'
```

B.1.3. Algoritmo 3: Vector a Entero

Entrada: Un vector binario V de tamaño q y peso h.

Salida: Un número entero positivo.

Descripción: El algoritmo transforma un vector binario de tamaño q y peso h en un numero entero.

```
def Vector_Entero(V,q,h):
    n=0
    for i in [1..q]:
        if V[i-1]==1:
            n=n+binomial(q-i,h)
            h=h-1
    return n
```

B.1.4. Algortimo 4: Entero a Vector

Entrada: Números enteros positivos n, q y h.

Salida: Un vector binario V de tamaño q y peso h.

Descripción: El algoritmo transforma un número entero a un vector binario de tamaño q y peso h.

```
def Entero_Vector(n,q,h):
    v=[]
    for i in [1..q]:
        if n>=binomial(q-i,h):
            v=v+[1];
            n=n-binomial(q-i,h);
            h=h-1;
        else:
            v=v+[0]
    return v
```

B.1.5. Algoritmo 5: Proceso de Cifrado

Entrada: La clave pública PK y un mensaje binario $\mathcal{M}$ de tamaño q y peso h.

Salida: El mensaje cifrado.

Descripción: El algoritmo permite cifrar un mensaje binario de tamaño q y peso h.

```
def Cifrar(PK,M,q,h):
    C=[(vector(m).dot_product(PK)) %(q^h-1)  for m in M]
    return C
```

B.1.6. Algoritmo 6: Proceso de Descifrado

Entrada: Un mensaje cifrado C, los enteros h y q descriptores del campo finito, el ruido r y el elemento primitivo g

Salida: Las posiciones donde está el bit 1 en el mensaje binario.

Descripción: El algoritmo permite determinar en que posicion se encuentra un 1 en un mensaje

binario de tamaño q y peso h, por medio de las raíces de los polinomios que se obtienen en el proceso de descifrado.

```
def Descifrar(C,h,q,r,g):
    S=[(i-3*r) % (17^3-1) for i in C];
    roots=[[0 for j in range(h)] for i in S]
    k=0
    for i in S:
        q=((g^i) % p).change_ring(F)
        q=(q+p).change_ring(L)
        T=q.factor()
        for j in range(h):
            roots[k][j]=T[j][0].constant_coefficient();
        k=k+1;
    return roots
```

B.1.7. Algoritmo 7: Raíz Vector

Entrada: Las posiciones del bit 1, los enteros h y q descriptores del campo finito.

Salida: El mensaje binario $\mathcal{M}$

Descripción: Conocidas las posiciones donde se encuentra el bit 1 en el mensaje, este algoritmo reconstruye el mensaje binario.

```
def raiz_vector(roots,q,h):
    Vector=[[0 for j in range(q)]for i in range(len(roots))]
    for i in range(len(roots)):
        for j in range(h):
            for k in range(q):
                if roots[i][j]==k:
                    Vector[i][k]=1
    return Vector
```

Bibliografía

[1] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*. CRC Press, Inc., 1st ed., 1996.

[2] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, Berlin, Germany, 2004.

[3] R. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations* (R. Miller and J. Thatcher, eds.), pp. 85–103, Plenum Press, 1972.

[4] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theor.*, vol. 22, no. 6, pp. 644–654, 2006.

[5] A. Shamir, "A polynomial-time algorithm for breaking the basic merkle - hellman cryptosystem," *IEEE Trans. Inf. Theor.*, vol. 30, no. 5, pp. 699–704, 2006.

[6] B. Chor and R. L. Rivest, "A knapsack-type public key cryptosystem based on arithmetic in finite fields," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 901–909, 1988.

[7] J. C. Lagarias and A. M. Odlyzko, "Solving low-density subset sum problems," *J. ACM*, vol. 32, no. 1, pp. 229–246, 1985.

[8] M. J. Coster, A. Joux, B. A. LaMacchia, A. M. Odlyzko, C.-P. Schnorr, and J. Stern, "Improved low-density subset sum algorithms.," *Computational Complexity*, vol. 2, pp. 111–128, 1992.

[9] P. Q. Nguyen and J. Stern, "Adapting density attacks to low-weight knapsacks," in *ASIACRYPT*, pp. 41–58, 2005.

[10] N. Kunihiro, "New definition of density on knapsack cryptosystems," in *AFRICACRYPT*, pp. 156–173, 2008.

[11] S. Vaudenay, "Cryptanalysis of the chor - rivest cryptosystem," *J. Cryptology*, vol. 14, no. 2, pp. 87–100, 2001.

[12] J. Hoffstein, J. Pipher, and J. Silverman, *An Introduction to Mathematical Cryptography*. Springer Publishing Company, Incorporated, 1 ed., 2008.

[13] R. C. Merkle and M. E. Hellman, "Hiding information and signatures in trapdoor knapsacks," *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 525–530, 1978.

[14] M. Garey and D. Johnson, *Computers and Intractability - A guide to the Theory of NP-Completeness*. San Fransisco: Freeman, 1979.

[15] D. Coppersmith, "Fast evaluation of logarithms in fields of characteristic two," *IEEE Transactions on Information Theory*, vol. 30, no. 4, pp. 587–593, 1984.

[16] S. C. Pohlig and M. E. Hellman, "An improved algorithm for computing logarithms over gf(p) and its cryptographic significance (corresp.)," *IEEE Transactions on Information Theory*, vol. 24, no. 1, pp. 106–110, 1978.

[17] A. Queiruga, *Avances recientes en el criptoanálisis del criptosistema de Chor-Rivest. Aplicaciones criptográficas*. PhD thesis, Universidad de Salamanca, Facultad de Ciencias, Departamento de Matemática Aplicada, 2006.

[18] H. W. L. Jr., "On the chor-rivest knapsack cryptosystem.," *J. Cryptology*, vol. 3, no. 3, pp. 149–155, 1991.

[19] L. H. Encinas, J. M. n. Masqué, and A. Q. Dios, "Analysis of the efficiency of the chor-rivest cryptosystem implementation in a safe-parameter range," *Inf. Sci.*, vol. 179, no. 24, pp. 4219–4226, 2009.