Sección de Matemáticas
Universidad de La Laguna

Itciar Fernández Elízaga

Ataque algebraico por inyección de fallos a la familia de cifrados en flujo Snow

Algebraic Fault Injection Attack on the Snow Family of Stream Ciphers

Trabajo Fin de Grado
Grado en Matemáticas
La Laguna, Mayo de 2023

Dirigido por
Irene Márquez Corbella

Irene Márquez Corbella
Matemáticas, Estadística e
Investigación Operativa
Universidad de La Laguna
38200 La Laguna, Tenerife

Agradecimientos

Quiero agradecer a mi familia, amigos y pareja por el apoyo incondicional a lo largo de estos 4 duros y bonitos años. También quiero agradecer a mi tutora Irene, por su guía y enseñanza en este nuevo campo.

Itciar Fernández Elízaga
La Laguna, 22 de mayo de 2023

Resumen · Abstract

Resumen

El cifrado de flujo Snow 3G es crucial para garantizar la seguridad de las comunicaciones en redes 4G. Se basa en su predecesor, el Snow 2.0, pero ha sido diseñado para ser más resistente a ataques algebraicos. En este trabajo se estudiarán las componentes de los cifrados de flujo, en particular los registros de desplazamiento de retroalimentación lineal (LFSR), analizando las secuencias que generan y sus propiedades. Además, se analizará la estructura del Snow 3G y se planteará un ataque basado en el método de Armknecht y Meier, el cual permite recuperar la clave secreta mediante la inducción de fallos en los LFSR para obtener sistemas de ecuaciones no lineales. Para resolver estos sistemas se utilizará la teoría de bases de Gröbner.

Palabras clave: *LFSR – Cuerpos finitos – Cifrado en flujo – Snow 3G – Bases de Gröbner*

Abstract

The Snow 3G stream cipher is crucial for ensuring the security of communications in 4G networks. It is based on its predecessor, Snow 2.0, but has been designed to be more resistant to algebraic attacks. This work will study the components of stream ciphers, particularly the linear feedback shift registers (LFSRs), analyzing the sequences they generate and their properties. Additionally, the structure of Snow 3G will be analyzed, and an attack based on the Armknecht and Meier method will be proposed, which allows to recover the secret key by inducing faults in the LFSR to obtain systems of nonlinear equations. To solve these systems, the theory of Gröbner bases will be used.

Keywords: *LFSR – Finite fields – Stream cipher – Snow 3G – Gröbner basis.*

Contenido

Introducción

La criptografía se puede definir como el estudio de técnicas de comunicaciones seguras (criptosistemas) que permiten que dos personas puedan intercambiar mensajes de forma privada. En paralelo a la criptografía se ha desarrollado el criptoanálisis, que busca debilidades de los criptosistemas que nos permitan romper su seguridad y acceder a conversaciones privadas. El triunfo de unos supone el fracaso de otros. Esta lucha la define muy bien el filósofo Edgar Allan Poe:

"Es dudoso que el género humano logre crear un enigma que el mismo ingenio humano no resuelva"

En la historia hemos tenido multitud de ejemplos de criptosistemas. Los primeros ejemplos datan de más de 2500 años (la escítala espartana, siglo V a. C.), basada en un cilindro cuyo grosor servía como clave, este cilindro enrrollaba el mensaje y luego al desenrollarlo las letras salían traspuestas (en otro orden). Para recuperar el mensaje original se requería de una escítala del mismo tamaño. Otros ejemplos son: el cifrado de Polybios, el primer cifrado por sustitución de caracteres; o el cifrado del César (de los romanos, siglo I a. C.) que consistía en sustituir cada letra por la que ocupaba tres posiciones hacia la derecha en el abecedario. Y si hacemos un resumen de criptografía no podemos olvidarnos de la máquina de cifrado más famosa de la historia: la máquina Enigma. El matemático Alan Turing, padre de los ordenadores modernos, fue clave para conseguir "criptoanalizar"la máquina Enigma, lo que se dice que adelantó el final de la Segunda Guerra Mundial.
Hasta mediados del siglo XX, cuando el uso del internet no se había extendido, la criptografía utilizada era criptografía de clave secreta o simétrica. En este tipo de criptografía se requiere de un acuerdo previo (o una clave) para cada par de personas que quieran comunicarse. En este trabajo se hará uso de la criptografía de clave secreta y, en particular, de cifrados en flujo. Un cifrado de flujo es un cifrado de clave simétrica donde los dígitos del texto plano (texto sin cifrar) se combinan con un flujo de dígitos de cifrado pseudoaleatorios (flujo de salida).

Hoy en día, la criptografía es clave para garantizar la seguridad de la transmisión de la información entre dispositivos informáticos y, sobre todo, en teléfonos móviles. Actualmente, el sistema de comunicación más usado es el conocido como tecnología 5G, que surge para mejorar los problemas de seguridad del predecesor, el 4G, cuyo criptosistema está basado en el generador Snow 3G, ambos pertenecientes a la familia de cifrados en flujo Snow.
La primera versión del cifrado Snow, ahora conocido como Snow 1.0 fue presentado al proyecto europeo Nessie (New European Schemes for Signatures, Integrity and Encryption) en el año 2000. Durante su evaluación en este proyecto, ya se encontraron debilidades que permitían recuperar la clave secreta solo observando la salida del algoritmo. Es por ello que, en el mismo proyecto, los autores crearon el Snow 2.0, que evitaba las debilidades encontradas. Esta nueva versión, utiliza un LFSR (generador de secuencias pseudoaleatorias de bits) y una máquina de estados finitos (FSM) como su predecesor, pero el polinomio de actualización del LFSR ha sido modificado. Este algoritmo sigue trabajando con palabras de 32 bits y con claves de 256 bits. Trabajaremos en detalle con este algoritmo en la Sección 2.2.2. Este cifrado en flujo fue elegido como estándar en las normas ISO/IEC 18033-4.
Sin embargo, el diseño del Snow 2.0 no es resistente a ataques algebraicos (como se verá en la Sección 2.2.2). De ahí que surja un nuevo diseño conocido como Snow 3G, que es el objeto de estudio del Capítulo 2. Este cifrado fue adoptado como cifrado estándar para proteger la privacidad y seguridad de los datos en las comunicación móviles de tercera generación (3G) y cuarta generación (4G), conocida como 3GPP. En el Capítulo 2 estudiaremos un ataque algebraico contra este criptosistema. Este y otros ataques ponen al descubierto las debilidades de este algoritmo.
Evitando los ataques anteriores han aparecido nuevas versiones como son el Snow-V y Snow-Vi que son candidatos para su inclusión en el estándar 5G.
El objetivo de este trabajo es analizar y realizar un ataque al generador Snow 3G con métodos algebraicos, haciendo uso de bases de Gröbner. En el primer capítulo, se introducirá uno de los elementos principales del Snow 3G, los LFSR, estudiando sus propiedades y usos. Para posteriormente, en el segundo capítulo, describir el Snow 3G y plantear un modelo de ataque basado en el método de Armknecht y Meier, y realizar una simulación del mismo. En el apéndice, se definirán y comentarán propiedades de cuerpos finitos, ya que estarán presente durante todo el desarrollo, así como las bases de Gröbner, fundamentales para poder realizar el ataque planteado.

1

Cifrados en flujo basados en LFSR

En el campo de la criptografía, un registro de desplazamiento con retroalimentación lineal (en inglés: Linear Feedback Shift Register que abreviaremos por LFSR), es un componente esencial de cifrados en flujo[1]. Un LFSR es un dispositivo que genera secuencias de bits pseudoaleatorias, es decir, son generadas por un algoritmo, aunque parecen ser producidas al azar. Dentro de los cifrados en flujo, la elección del LFSR se debe a las ventajas que este ofrece, como su rendimiento, las buenas propiedades que tienen las secuencias que produce y su coste de implementación. En este capítulo se explicará en detalle el funcionamiento del LFSR, así como sus propiedades.

1.1. Definiciones básicas de recurrencias lineales

Definición 1.1 (LFSR). *Un LFSR de longitud L sobre el cuerpo finito $\mathbb{F}_q$ es un automáta finito o una máquina de estado finito, es decir, un modelo que en cada instante tiene un único estado y tiene un número finito de estados posibles. El LFSR produce una secuencia de elementos de $\mathbb{F}_q$. A la secuencia producida la denotaremos por $s = (s_t)_{t\geq 0}$ siendo s_0 el estado en el instante $t = 0$, el* estado inicial. *Esta secuencia satisface una relación de recurrencia lineal de grado L sobre $\mathbb{F}_q$ como sigue:*

$$s_{t+L} = \sum_{i=1}^{L} c_i s_{t+L-i}, \forall t \geq 0 \tag{1.1}$$

donde $c_1, c_2, \ldots, c_L$ son elementos del cuerpo $\mathbb{F}_q$, que llamaremos ***coeficientes de retroalimentación*** *del LFSR.*

El registro consta de L celdas, llamadas también *etapas*, cada una de ellas contiene un elemento de $\mathbb{F}_q$. El contenido de las L etapas $s_t, s_{t+1}, \ldots, s_{t+L-1}$ forma el estado del LFSR en el instante t. El instante $t = 0$, forma el estado inicial y

[1] Un cifrado de flujo es un cifrado de clave simétrica donde los dígitos del texto plano se combinan con un flujo de dígitos pseudoaleatorios.

los elementos en ese instante suelen ser elegidos de forma arbitraria en $\mathbb{F}_q$. La secuencia de salida de un LFSR está únicamente determinada por sus coeficientes de retroalimentación y su estado inicial $(s_0, s_1, \ldots, s_{L-1})$.

1.2. Representación de Fibonacci

En la Figura 1.1 se muestra la *representación de Fibonacci* de un LFSR de longitud L sobre $\mathbb{F}_q$. Esta es la representación estándar de un LFSR, en la que cada nuevo término es generado a partir de los anteriores. El registro de desplazamiento está controlado por un reloj externo. En cada unidad de tiempo, cada dígito es desplazado a la derecha, siendo s_t la salida en el instante t. El nuevo contenido s_{t+L} se conoce como *bit de retroalimentación* y se obtiene a partir de una combinación lineal del contenido de las etapas anteriores del registro, siguiendo la relación de recurrencia definida en la Ecuación (1.1).

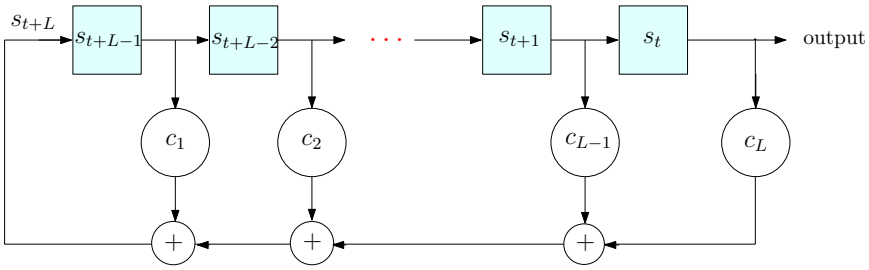

Figura 1.1: Representación de Fibonnaci de un LFSR de longitud L.

Ejemplo 1.2. Vamos a considerar un LFSR binario (definido en $\mathbb{F}_2$) de longitud 4. Fijamos como coeficientes de retroalimentación $c_1 = c_2 = 0$, $c_3 = c_4 = 1$ y su estado inicial como $(s_0, s_1, s_2, s_3) = (1, 0, 1, 1)$. Por lo tanto, utilizando la relación de recurrencia

$$s_{t+4} = \sum_{i=1}^{4} c_i s_{t+4-i} = s_{t+1} + s_t \bmod 2$$

su secuencia de salida generada es $(s_t)_{t\geq 0} = 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, \ldots$
La Figura 1.2 presenta la representación de Fibonacci de este ejemplo.
Los estados sucesivos del LFSR quedan descritos en la Tabla 1.1.

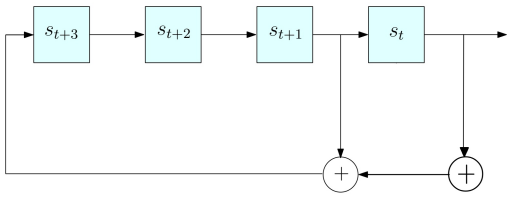

Figura 1.2: Representación de Fibonacci del Ejemplo 1.2.

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
s_t	1	0	1	1	1	1	0	0	0	1	0	0	1	1	0
s_{t+1}	0	1	1	1	1	0	0	0	1	0	0	1	1	0	1
s_{t+2}	1	1	1	1	0	0	0	1	0	0	1	1	0	1	0
s_{t+3}	1	1	1	0	0	0	1	0	0	1	1	0	1	0	1

Tabla 1.1: Estados del LFSR definidos en el Ejemplo 1.2.

1.3. Polinomio característico

Definición 1.3. *Los coeficientes de retroalimentación de un LFSR de longitud L se pueden representar por el* ***polinomio de retroalimentación*** *o* ***polinomio de conexión*** *del LFSR, definido por:*

$$P(X) = 1 - \sum_{i=1}^{L} c_i X^i$$

También se puede usar como alternativa el ***polinomio característico****, definido como el recíproco del polinomio de retroalimentación. Su expresión es la siguiente:*

$$P^*(X) = X^L P(\frac{1}{X}) = X^L - \sum_{i=1}^{L} c_i X^{L-i}$$

Ejemplo 1.4. Si volvemos al ejemplo anterior, es claro que

$$P(X) = 1 + X^3 + X^4 \quad \text{y} \quad P^*(X) = 1 + X + X^4.$$

representan el polinomio de retroalimentación y el polinomio característico del LFSR de la Figura 1.2.

Definición 1.5. *Se dice que un LFSR es* ***no singular*** *si el grado de su polinomio de retroalimentación es igual a la longitud del LFSR, es decir, $c_L \neq 0$.*

Proposición 1.6. *La secuencia generada por un LFSR no singular de longitud L es periódica y su periodo es menor que $q^L - 1$.*
Es más, un LFSR de longitud L tiene a lo sumo q^L estados distintos.

Demostración. Sabemos que cada estado del LFSR está completamente determinado por su estado anterior. Luego, si en un cierto instante $t+k$ ocurre que un estado es el mismo que el estado t es decir, $s_t = s_{t+k}$, entonces los estados entre t y $t+k$ se volverán a repetir, obteniendo así la periodicidad del registro. Sabemos que en un LFSR de longitud L cada una de las L etapas tiene q posibles valores diferentes en $\mathbb{F}_q$. Luego, hay q^L estados posibles distintos. Del razonamiento anterior, se deduce que el registro tiene periodicidad con periodo $s \leq q^L$.
Además, el estado con todos sus valores igual a cero no es posible si queremos tener un periodo largo pues tras el estado $(0, \ldots, 0)$ todos los valores serán 0 y, en este caso, su periodo será $s = 1$. Por tanto, $s \leq q^L - 1$. □

Hemos obtenido entonces que, dado un LFSR de longitud L definido en $\mathbb{F}_q$ se pueden generar a lo sumo q^L estados diferentes. Es decir, podemos considerar el conjunto de los estados diferentes que produce el LFSR como el espacio $\mathbb{F}_q^L$, este espacio tiene estructura de $\mathbb{F}_q$-espacio vectorial.

Definición 1.7. *Consideremos un LFSR de longitud L definido en $\mathbb{F}_q$, denotamos como* ***función de generación*** *a la representación de la secuencia producida por el LFSR entendida como una serie en $\mathbb{F}_q[X]$. Es decir, si $s = (s_t)_{t\geq 0}$ es la secuencia, entonces la función tiene la forma siguiente:*

$$\sum_{t\geq 0} s_t X^t \in \mathbb{F}_q[X]$$

Teorema 1.8. *Una secuencia $(s_t)_{t\geq 0}$ está generada por un LFSR de longitud L definido en $\mathbb{F}_q$ con polinomio de retroalimentación P si, y solo si, existe un polinomio $Q \in \mathbb{F}_q[X]$ con $\deg(Q) < L$ tal que la función de generación de $(s_t)_{t\geq 0}$, satisface que:*

$$\sum_{t\geq 0} s_t X^t = \frac{Q(X)}{P(X)}$$

Además, el polinomio Q está completamente determinado por los coeficientes de P y por el estado inicial del LFSR. Es más,

$$Q(X) = -\sum_{j=0}^{L-1} X^j \Big(\sum_{k=0}^{j} s_k c_{j-k}\Big) \quad \textit{donde} \quad P(X) = -\sum_{i=0}^{L} c_i X^i$$

Demostración. Sea $(s_t)_{t\geq 0}$ la secuencia generada por un LFSR con polinomio de retroalimentación P. Hacemos el producto de $P(X)$ con la función de generación.

$$P(X)\left(\sum_{t\geq 0} s_t X^t\right) = \left(-\sum_{i=0}^{L} c_i X^i\right)\left(\sum_{t=0}^{\infty} s_t X^t\right) =$$

$$= \sum_{j=0}^{\infty} X^j \left(- \sum_{k=\text{máx }\{0,j-L\}}^{j} s_k c_{j-k} \right) =$$

$$= \underbrace{- \sum_{j=0}^{L-1} X^j \left(\sum_{k=0}^{j} s_k c_{j-k} \right)}_{Q(X)} + \underbrace{\sum_{j=L}^{\infty} X^j \left(\sum_{k=j-L}^{j} s_k c_{j-k} \right)}_{A(X)}$$

Luego, si demostramos que $A(X) = 0$, habremos encontrado un polinomio $Q(X) \in \mathbb{F}_q[X]$ con $\deg(Q) < L$ que verifica el enunciado. Veamos que, efectivamente $A(X) = 0$:

$$A(X) = \sum_{j=L}^{\infty} X^j (\sum_{k=j-L}^{j} s_k c_{j-k}) = 0 \Longleftrightarrow (\sum_{k=j-L}^{j} s_k c_{j-k}) = 0$$

Como los c_i son los coeficientes asociados al polinomio de retroalimentación $P(X)$ y este polinomio tiene grado a lo sumo L, tenemos que $c_m = 0, \forall m > L$. Luego, $A(X) = 0$.
Recíprocamente, supongamos que existe $Q(X) \in \mathbb{F}_q[X]$ con $\deg(Q) < L$ tal que

$$\left(\sum_{t \geq 0} s_t X^t \right) = \frac{Q(X)}{P(X)} \quad \text{con} \quad Q(X) = - \sum_{j=0}^{L-1} X^j (\sum_{k=0}^{j} s_k c_{j-k})$$

Veamos que la secuencia $(s_t)_{t \geq 0}$ está generada por un LFSR con polinomio de retroalimentación P.
Observamos que,

$$P(X) \cdot \left(\sum_{t \geq 0} s_t X^t \right) = - \sum_{j=0}^{L-1} X^j \left(\sum_{k=0}^{j} s_k c_{j-k} \right) \Longrightarrow P(X) = - \sum_{i=0}^{L} c_i X^i$$

Por lo tanto, $P(X)$ define un polinomio de retroalimentación del LFSR con secuencia $(s_t)_{t \geq 0}$. □

Este resultado implica que existe una correspondencia biyectiva entre las secuencias generadas por un LFSR de longitud L con polinomio de retroalimentación P y las fracciones $\frac{Q(X)}{P(X)}$ con $\deg(Q) < L$.

Observación 1.9. Supongamos que tenemos un LFSR generado por un polinomio $P(X)$ y con función de generación $(s_t)_{t \geq 0}$ tal que

$$\sum s_t X^t = \frac{Q(X)}{P(X)} \quad \text{pero} \quad mcd(P, Q) \neq 1$$

Entonces, se puede encontrar P', Q' tal que

$$\sum s_t X^t = \frac{Q'(X)}{P'(X)} \quad \text{con} \quad \deg(P') < \deg(P)$$

y donde P y P' son polinomios que generan el mismo LFSR.

Observación 1.10. Veamos cómo se relacionan los polinomios que generan una cierta secuencia. Sean P_1 y P_2 los polinomios de retroalimentación asociados a un cierto LFSR de longitud L. Por el teorema anterior sabemos que existirán polinomios Q_1 y Q_2 verificando que:

$$\sum_{t\geq 0} s_t X^t = \frac{Q_1(X)}{P_1(X)} \quad \text{y} \quad \sum_{t\geq 0} s_t X^t = \frac{Q_2(X)}{P_2(X)}$$

Por tanto, tenemos que

$$\frac{Q_1(X)}{P_1(X)} = \frac{Q_2(X)}{P_2(X)} \Rightarrow P_1(X) = \frac{Q_1(X)\cdot P_2(X)}{Q_2(X)}$$

Ejemplo 1.11. Sea $(s_t)_{t\geq 0}$ una secuencia binaria que satisface:

$$s_{t+6} = s_{t+4} + s_{t+3} + s_{t+1} + s_t, \forall t \geq 0 \tag{1.2}$$

Para hallar el polinomio de retroalimentación tenemos que calcular los coeficientes de retroalimentación. Sabemos que la relación entre la secuencia y los coeficientes de retroalimentación viene dada por la Ecuación (1.1). Por tanto, se deduce que

$$c_1 = 0, c_2 = 1, c_3 = 1, c_4 = 0, c_5 = 1, c_6 = 1$$

Luego, como $P(X) = 1 - \sum_{i=1}^{L} c_i X^i$, se tiene que $P(X) = 1 + X^2 + X^3 + X^5 + X^6$ es el polinomio de retroalimentación de esta secuencia. Por otra parte,

$$\begin{aligned} s_{t+7} &= s_{t+5} + s_{t+4} + s_{t+2} + s_{t+1} \\ s_{t+8} &= s_{t+6} + s_{t+5} + s_{t+3} + s_{t+2} \\ &= s_{t+4} + s_{t+3} + s_{t+1} + s_t + s_{t+5} + s_{t+3} + s_{t+2} \\ &= s_{t+5} + s_{t+4} + s_{t+2} + s_{t+1} + s_t \\ &= s_{t+7} + s_t \end{aligned}$$

De la relación anterior podemos deducir otro polinomio de retroalimentación de nuestro LFSR (que no es irreducible)

$$P'(X) = 1 + X + X^8 = (1 + X^2 + X^3 + X^5 + X^6)\cdot(1 + X + X^2)$$

Vamos a ejemplificar la Observación 1.9, para ello supongamos que tenemos un LFSR con el siguiente estado inicial $(s_0, s_1, \ldots, s_5) = (1, 0, 0, 1, 0, 0)$ y hallemos el polinomio $Q(X)$ que se deduce del Teorema 1.8 utilizando $P(X)$

$$Q(X) = \sum_{j=0}^{L-1} X^j (\sum_{k=0}^{j} s_k c_{j-k}) = 1 + 1 \cdot X^2 + (1+1) \cdot X^3 + (1+1+0) \cdot X^5 = 1 + X^2$$

Luego,

$$\sum_{t \geq 0} s_t X^t = \frac{Q(X)}{P(X)} = \frac{1 + X^2}{1 + X^2 + X^3 + X^5 + X^6} \quad \text{con } \operatorname{mcd}(Q(X), P(X)) = 1$$

Si ahora hallamos el polinomio $Q'(X)$ que se deduce del Teorema 1.8 pero utilizando $P'(X)$, tenemos que $Q'(X) = 1 + X + X^3 + X^4$. Luego,

$$\sum_{t \geq 0} s_t X^t = \frac{Q'(X)}{P'(X)} = \frac{1 + X + X^3 + X^4}{1 + X + X^8} =$$

$$= \frac{(1 + X + X^2) \cdot (1 + X^2)}{(1 + X^2 + X^3 + X^5 + X^6) \cdot (1 + X + X^2)} =$$

$$= \frac{1 + X^2}{1 + X^2 + X^3 + X^5 + X^6} = \frac{Q(X)}{P(X)}$$

En el primer caso tenemos un LFSR definido por $P(X)$ de longitud 6 y en el segundo otro LFSR definido por $P'(X)$ de longitud 8. Ambos LFSR generan la misma secuencia. Por lo tanto por la Proposición 1.6 su periodo será $\leq q^6 - 1$.

1.3.1. Polinomio mínimo

Definición 1.12. *El **polinomio mínimo** de una secuencia de recurrencia lineal $s = (s_t)_{t \geq 0}$ de elementos de $\mathbb{F}_q$ es el polinomio $P(X)$ en $\mathbb{F}_q[X]$ de menor grado mónico, tal que $(s_t)_{t \geq 0}$ es la secuencia generada por el LFSR con polinomio característico $P(X)$, es decir, $P(X) = X^L - \sum_{i=1}^{L} p_i X^{L-i}$ es el polinomio de menor grado que satisface la secuencia:*

$$s_{t+L} + \sum_{i=0}^{L-1} p_i s_{t-i} = 0, \ t \geq 0$$

En la sección siguiente se verá la relación del polinomio mínimo con la longitud de un LFSR.

Lema 1.13. *El polinomio mínimo de una secuencia de recurrencia lineal es único.*

Demostración. Supongamos que existen P_1, P_2 dos polinomios mínimos que generan la misma secuencia. Por la Observación 1.10 y el Teorema 1.8 se tiene que

$$P_1(X) = \frac{Q_1(X) \cdot P_2(X)}{Q_2(X)} \quad \text{con} \deg(Q_1),\ \deg(Q_2) < L$$

$$\text{Como } \deg(P_1) = \deg(P_2) \Rightarrow \frac{Q_1(X)}{Q_2(X)} = \lambda. \text{ Luego, } P_1(X) = \lambda P_2(X),\ \lambda \in \mathbb{F}_q$$

Además, $P_1(X)$ es mónico, entonces $\lambda = 1$, por tanto, $P_1(X) = P_2(X)$, obteniendo así la unicidad. □

Lema 1.14. *Si P es el polinomio mínimo de una secuencia de recurrencia lineal $s = (s_t)_{t\geq 0}$ y Q es el polinomio definido en el Teorema 1.8. Entonces $mcd(P, Q) = 1$.*

Demostración. Se deduce de la Observación 1.9.

El polinomio mínimo no tiene que ser irreducible, como se puede comprobar en el ejemplo siguiente.

Ejemplo 1.15. Consideremos un LFSR binario de longitud 10 con polinomio de retroalimentación $P(X) = 1 + X + X^3 + X^4 + X^7 + X^{10}$ y estado inicial $(s_0, s_1, \ldots, s_9) = (1, 0, 0, 1, 0, 0, 1, 1, 0, 1)$. Sabemos entonces que $c_1 = c_3 = c_4 = c_7 = c_{10} = 1$ y $c_2 = c_5 = c_6 = c_8 = c_9 = 0$. Luego, la relación de recurrencia es la siguiente

$$s_{t+10} = s_{t+9} + s_{t+7} + s_{t+6} + s_{t+3} + s_t \bmod 2$$

Y su representación de Fibonacci es la mostrada en la Figura 1.3.

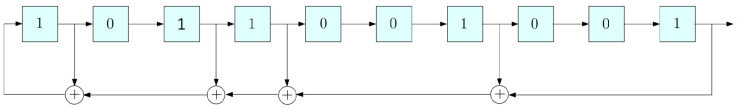

Figura 1.3: Representación de Fibonacci del Ejemplo 1.15

Hallemos ahora el polinomio $Q(X)$ definido en el Teorema 1.8.

$$Q(X) = -\sum_{j=0}^{9} X^j \Big(\sum_{k=0}^{j} s_k c_{j-k}\Big),\ \text{con } s_1 = s_2 = s_4 = s_5 = s_7 = s_8 = 0$$

Sabemos que este polinomio verifica que

$$\sum_{t\geq 0} s_t X^t = \frac{Q(X)}{P(X)} = \frac{1 + X + X^7}{1 + X + X^3 + X^4 + X^7 + X^{10}} = \frac{1}{1 + X^3}$$

Ya que $(1 + X + X^3 + X^4 + X^7 + X^{10}) = (1 + X + X^7) \cdot (1 + X^3)$
Esto nos dice que la secuencia $(s_t)_{t\geq 0}$ está también generada por el polinomio de retroalimentación $P_0(X) = 1 + X^3$ representado en la Figura 1.4.

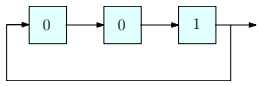

Figura 1.4: Otra representación de Fibonacci del Ejemplo 1.15.

El polinomio mínimo de la secuencia es P_0 y, sin embargo, este no es irreducible en $\mathbb{F}_2$[X], pues 1 es raíz de $P_0(X)$.

1.3.2. Periodo de una secuencia de recurrencia lineal

Otro aspecto importante del polinomio mínimo es que determina el periodo de una secuencia de recurrencia lineal, para probarlo veamos un resultado previo.

Proposición 1.16. *Sea $s = (s_t)_{t\geq 0}$ una sucesión recursiva lineal en $\mathbb{F}_q$. El conjunto de polinomios en $\mathbb{F}_q[X]$ que generan a s forma un ideal principal $I = (m)$ en $\mathbb{F}_q[X]$. Si tomamos m mónico, m es el polinomio mínimo de la sucesión s.*

Demostración. Definimos la aplicación ϕ_n entre $\mathbb{F}_q[X]$ y $\mathbb{F}_q$ como:

$$\phi_n(a_k X^k + \cdots a_1 X + a_0) := a_k s_{k+n} + \cdots + a_1 s_{1+n} + a_0 s_n$$

Es sencillo comprobar que:

- ϕ_n es homomorfismo entre $\mathbb{F}_q-$espacios vectoriales.
- $\phi_{n+1}(f) = \phi_n(X \cdot f)$, $\forall f \in \mathbb{F}_q[X]$.

Además, se tiene la propiedad de que una sucesión $s = (s_t)_{t\geq 0}$ está generada por el polinomio $f \in \mathbb{F}_q[X]$ si, y solo si, para todo $n \geq 0$, $\phi_n(f) = 0$. En particular,

1. si $f_1, f_2 \in \mathbb{F}_q[X]$ generan la misma sucesión $s = (s_t)_{t\geq 0}$. Entonces, como ϕ_n es homomorfismo, se tiene que $\phi_n(\alpha_1 \cdot f_1 + \alpha_2 \cdot f_2) = \alpha_1\phi_n(f_1) + \alpha_2\phi_n(f_2) = 0$ $\forall \alpha_1, \alpha_2 \in \mathbb{F}_q$.
2. Además $\phi_n(X^r \cdot f) = \phi_{n+r}(f)$ por lo tanto $\forall f \in \mathbb{F}_q[X]$, $\forall r \in \mathbb{N}$, si f genera $s = (s_t)_{t\geq 0}$ entonces $X^r \cdot f$ también genera $s = (s_t)_{t\geq 0}$.

De las propiedades anteriores, si I es el conjunto de polinomios en $\mathbb{F}_q[X]$ que genera $s = (s_t)_{t\geq 0}$, entonces I tiene estructura de ideal en $\mathbb{F}_q[X]$. Como $\mathbb{F}_q$ es cuerpo se tiene que $\mathbb{F}_q[X]$ es dominio de ideales principales. Luego, $I = (m)$ para cierto $m \in \mathbb{F}_q[X]$ que podemos considerar mónico. □

Proposición 1.17. *Una secuencia generada por un LFSR con polinomio mínimo P tiene periodo m si, y solo si, $P(X)$ divide a $1 - X^m$.*

Demostración. Lo veremos por doble implicación.
Si suponemos que la secuencia $s = (s_t)_{t\geq 0}$ tiene periodo m, entonces el estado en el instante t se repite en $t+m$. Es decir, $s_j = s_{j+m},\ \forall j \geq 0$. Por tanto, la función de generación admite la siguiente escritura:

$$\begin{aligned}\sum_{t\geq 0} s_t X^t = (s_0 + s_1 X + \cdots + s_{m-1}X^{m-1}) + \\ + X^m(s_0 + s_1 X + \cdots + s_{m-1}X^{m-1}) + \\ + X^{2m}(s_0 + s_1 X + \cdots + s_{m-1}X^{m-1}) + \cdots = \\ = (s_0 + s_1 X + \cdots + s_{m-1}X^{m-1}) \cdot (1 + X^m + X^{2m} + \dots) = \\ = (s_0 + s_1 X + \cdots + s_{m-1}X^{m-1}) \cdot \frac{1}{1 - X^m}\end{aligned} \tag{1.3}$$

Donde la tercera igualdad es consecuencia de que el polinomio $(1 + X^m + X^{2m} + \dots)$ sea una suma geométrica infinita de razón X^m. Ahora, sea $P(X)$ el polinomio mínimo asociado a la secuencia $s = (s_t)_{t\geq 0}$ y utilizando el Teorema 1.8 sabemos que existe un polinomio $Q(X)$ con $\deg(Q) < m$ tal que

$$\sum_{t\geq 0} s_t X^t = \frac{Q(X)}{P(X)} \tag{1.4}$$

Además por el Lema 1.14 se tiene que $mcd(P, Q) = 1$. De la expresión (1.4) y de lo obtenido en (1.3) se deduce que:

$$P(X) \cdot (s_0 + s_1 X + \cdots + s_{m-1}X^{m-1}) = (1 - X^m) \cdot Q(X)$$

Como $mcd(P, Q) = 1$ se tiene que $P(X)$ divide a $1 - X^m$.
Supongamos ahora que $P(X)$ divide a $1 - X^m$. Primero sabemos que $1 - X^m$ genera una secuencia de periodo m pues $s_{t+m} = s_t$. Además, como $P(X)$ divide a $1 - X^m$ se tiene que $(1 - X^m) = P(X) \cdot R(X)$ con $R(X) \in \mathbb{F}_q[X]$. Luego, es claro que $(1 - X^m) \in (P(X))$, por lo que se concluye que $P(X)$ y $1 - X^m$ generan la misma secuencia por lo probado en la Proposición 1.16. □

Corolario 1.18. *Una secuencia generada por un LFSR con polinomio mínimo P tiene periodo m si, y solo si, m es el menor entero positivo tal que $P(X)$ divide a $1 - X^m$.*

Corolario 1.19. *Si una secuencia tiene un polinomio característico irreducible de grado L, el periodo de la secuencia es un factor de $q^L - 1$.*

Definición 1.20. *Diremos que la secuencia generada por un LFSR no singular de longitud L es máxima si su periodo es $q^L - 1$.*

Teorema 1.21. *Sea $P(X) \in \mathbb{F}_q[X]$ el polinomio mínimo que genera la secuencia $s = (s_t)_{t\geq 0}$. La secuencia s es máxima si $P(X)$ es irreducible.*

Demostración. Por lo visto en la Proposición 1.17, el periodo de $s = (s_t)_{t\geq 0}$ es el grado de $P(X)$. Supongamos por reducción al absurdo que $P(X)$ es reducible. Luego, existen dos polinomios $u(X), v(X) \in \mathbb{F}_q[X]$ con $\deg(u) = L_1$ y $\deg(v) = L_2$ tales que $P(X) = u(X) \cdot v(X)$. Sean $(s'_t)_{t\geq 0}$ y $(s''_t)_{t\geq 0}$ las secuencia generadas por $u(X)$ y $v(X)$, respectivamente. Entonces, por el Teorema 1.8 existen $Q'(X)$ y $Q''(X)$ tales que $\frac{Q'(X)}{u(X)}$, $\frac{Q''(X)}{v(X)}$ son series de potencias periódicas con periodos a lo sumo $q^{L_1} - 1$ y $q^{L_2} - 1$. Por lo tanto

$$\frac{Q'(X)}{u(X)} + \frac{Q''(X)}{v(X)} = \frac{v(X) \cdot Q'(x) + u(X) \cdot Q''(X)}{u(X) \cdot v(X)} = \frac{R(X)}{P(X)}$$

Tenemos que, $\frac{R(X)}{P(X)}$ genera una secuencia con periodo a lo sumo $mcm(q^{L_1} - 1, q^{L_2} - 1)$ Además,

$$L_1 + L_2 = L \text{ y } mcm(q^{L_1} - 1, q^{L_2} - 1) \leq (q^{L_1} - 1) \cdot (q^{L_2} - 1)$$

Por tanto,

$$\begin{aligned} q^L - 1 \leq (q^{L_1} - 1) \cdot (q^{L_2} - 1) &= q^{L_1+L_2} - q^{L_1} - q^{L_2} + 1 \\ &\leq q^L - q - q + 1 = q^L - 2q + 1 < q^L - 1 \end{aligned}$$

Llegamos entonces a un absurdo, que proviene de suponer que $P(X)$ es reducible. □

Sin embargo, el recíproco no es cierto en general. Existen polinomios irreducibles que se corresponden con secuencias que no son de longitud máxima.

Ejemplo 1.22. Consideremos $P(X) = X^4 + X^3 + X^2 + X + 1 \in \mathbb{F}_2[X]$. $P(X)$ es irreducible en $\mathbb{F}_2[X]$ pues no tiene raíces en $\mathbb{F}_2[X]$ y no se puede poner como producto de dos polinomios de grado menor que 4. Además, como $1 + X + X^2 + X^3 + X^4 = \frac{1-X^5}{1-X}$ por ser una suma geométrica finita, entonces $P(X)$ divide a $1 - X^5$. La Proposición 1.17 nos dice que este polinomio genera a una secuencia de periodo 5, mucho menor que el periodo máximo que es $2^4 - 1 = 15$.

En el siguiente resultado utilizaremos el concepto de polinomio primitivo que hemos trabajado en el Anexo A.1.

Lema 1.23. *Si $P(X)$ es primitivo de grado L entonces el menor entero positivo m tal que $P(X)$ divide a $1 - X^m$ es $m = q^L$.*

Demostración. Si $P(X)$ divide a $1 - X^m$ con $m < q^L$ y α es raíz de $P(X)$ entonces, $\alpha^m = 1$ por tanto existe α raíz de $P(X)$ con orden $m < q^L$ y por la Definición A.10 de polinomio primitivo todas las raíces de $P(X)$ tienen que generar la extensión $\mathbb{F}_{q^L}$, por tanto, llegamos a un absurdo que viene de suponer que $m < q^L$. □

Teorema 1.24. *La secuencia $s = (s_t)_{t\geq 0}$ generada por un LFSR de longitud L no singular es máxima si, y solo si, su polinomio característico $P(X)$ es primitivo.*

Demostración. Si la secuencia es de longitud máxima, por resultados anteriores se tiene que su polinomio característico es irreducible y el periodo de la secuencia es $q^L - 1$ que es el menor entero positivo tal que $P(X)$ divide a $1 - X^{q^L-1}$. Por tanto, $P(X)$ es primitivo. Recíprocamente, si $P(X)$ es primitivo, por el Lema 1.23 y por el Corolario 1.18 la secuencia $s = (s_t)_{t\geq 0}$ tiene longitud máxima. □

Corolario 1.25. *Si $q^L - 1$ es primo, cada polinomio irreducible de grado L se corresponde con una secuencia de desplazamiento de longitud máxima.*

1.3.3. Representación de Galois

Consideramos un LFSR con polinomio característico $P^*(X)$. Supongamos que $P^*(X)$ es irreducible en $\mathbb{F}_q[X]$ de grado L. Entonces, el cuerpo $\mathbb{F}_{q^L} = \mathbb{F}_q[\alpha]$ es una extensión de $\mathbb{F}_q$ siendo α una raíz de $P^*(X)$. Además, $A = \{1, \alpha, \ldots, \alpha^{L-1}\}$ es una base de $\mathbb{F}_{q^L}$ como $\mathbb{F}_q$-espacio vectorial (que se conoce como base polinómica de $\mathbb{F}_{q^L}$). Para más detalles sobre cuerpos finitos véase Anexo A.1.
Veamos en esta sección que la salida del LFSR se corresponde con una multiplicación en $\mathbb{F}_{q^L}$. En efecto, si identificamos para cada instante t, el estado del LFSR con la L-tupla: $(s_t, \ldots, s_{t+L-1})$ y consideramos $B = \{\beta_0, \ldots, \beta_{L-1}\}$ una base dual de A entonces se tiene el siguiente resultado:

Teorema 1.26. *La salida del LFSR con polinomio característico $P^*(X)$ en el instante $t+1$ se corresponde con la multiplicación de $s = s_{t+L-1}\beta_{L-1} + \cdots + s_t\beta_0$ por α.*

Demostración. Por definición de base dual (véase en A.18) se tiene que:

$$tr(\alpha^i s) = \sum_{j=0}^{L-1} tr(s_j \alpha^i \beta_j) = \sum_{j=0}^{L-1} s_j tr(\alpha^i \beta_j) = s_i,\ 0 \leq i \leq L$$

Por lo tanto, la i-ésima coordenada de s respecto de la base $\{\beta_0, \ldots, \beta_{L-1}\}$ es $tr(\alpha^i x)$. Sea $y = \alpha \cdot s$. Entonces $tr(\alpha^i y) = tr(\alpha^{i+1} s)$.

- Si $i < L - 1$. Entonces la i-ésima coordenada de y se corresponde con la $(i+1)$-ésima coordenada de s.
- Si $i = L - 1$. La última coordenada de y se define como

$$tr(\alpha^{L-1} y) = tr(\alpha^L s) = \sum_{j=0}^{L-1} s_j tr(\alpha^L \beta_j) = \sum_{j=0}^{L-1} s_j tr(\sum_{i=0}^{L-1} c_i \alpha^i \beta_j) = \sum_{j=0}^{L-1} c_j s_j$$

Puesto que $P^*(X) = X^L - \sum_{i=1}^{L-1} c_i X^i$. □

Una representación de Galois es un autómata que implementa esta multiplicación.

Ejemplo 1.27 (Continuación Ejemplo 1.2). Vamos a hacer uso del Lema A.20 para calcular la salida del LFSR definido en el Ejemplo 1.2 pero utilizando la multiplicación en un cuerpo. Así como de la herramienta de software algebraico Sagemath. Consideramos el polinomio $P^*(X) = 1 + X + X^4$ que es irreducible en $\mathbb{F}_2[X]$. Por lo tanto

$$\mathbb{F}_{2^4} = \frac{\mathbb{F}_2[X]}{P^*(X)} = \mathbb{F}_2[\alpha], \text{ siendo } \alpha \text{ raíz de } P^*(X)$$

```
> sage: K.<a>=GF(2^4)
> sage: R.<x> = PolynomialRing(K)
## El polinomio que define el cuerpo se puede cambiar.
> sage: f=x^4+x+1; derivative(f, x)
> f /(x-a) x^3 + a*x^2 + a^2*x + a^3 + 1
> A=[1,a,a^2,a^3] ## base polinómica
> B=[1+a^3,a^2,a,1] ## base dual de A
```

Hemos comprobado que realmente son bases duales, comprobando que se verifica la definición con el siguiente comando $(\texttt{A[i]} * \texttt{B[j]}).\texttt{trace}()$ con $i, j \in \{0, 1, 2\}$ y $i \neq j$. Vamos a calcular la salida del LFSR utilizando la multiplicación por a:

```
> sage: S=[1,0,1,1]
> sage: out=S[0]*B[0]+S[1]*B[1]+S[2]*B[2]+S[3]*B[3]
a^3 + a
> sage: for i in range(10): print (out*a^i).trace()
Salida: 1 0 1 1 1 1 0 0 0 1
```

Los 4 primeros coinciden con S y el resto con la salida del Ejemplo 1.2.

1.3.4. Representación matricial

En esta sección hemos querido dar unas breves pinceladas de otras propiedades del LFSR utilizando su representación matricial.

Definición 1.28. *Sea $\{s_n\}_{n=0}^{\infty}$ una secuencia generada por un LFSR de longitud L en $\mathbb{F}_q$ y definida por la relación de recurrencia lineal: $s_{t+L} = \sum_{i=i}^{L} c_i s_{t+L-i}$, $\forall t \geq 0$. Podemos asociar a esta secuencia la matriz*

$$M = \begin{pmatrix} c_1 & 1\,0 \cdots 0\,0 \\ c_2 & 0\,1 \cdots 0\,0 \\ c_3 & 0\,0 \cdots 0\,0 \\ \vdots & \vdots\,\vdots\,\ddots\,\vdots\,\vdots \\ c_{L-1} & 0\,0 \cdots 0\,1 \\ c_L & 0\,0 \cdots 0\,0 \end{pmatrix} \tag{1.5}$$

De forma que,

$$(s_{t+L-1}, s_{t+L-2}, \ldots, s_t) \cdot M = (\underbrace{s_{t+L-1}c_1 + s_{t+L-2}c_2 + \cdots + s_t c_L}_{s_{t+L}}, s_{t+L-1}, \ldots, s_{t+1})$$

Luego, una multiplicación del estado t por la matriz M nos proporciona el nuevo estado s_{t+L}. De esta forma se puede trabajar indistintamente con la matriz asociada al LFSR o con la relación de recurrencia. Esto abre un nuevo campo: poder aplicar resultados conocidos de matrices para deducir propiedades del LFSR. En esta sección, por falta de espacio, solo veremos algunas pinceladas en esta dirección pero queda como trabajo futuro explorar más este nexo con las matrices.
Observamos que, como $det(M) = c_L$, M es invertible si, y solo si, $c_L \neq 0$. Es decir, si el LFSR es no singular (Definición 1.5). Además, si estudiamos el polinomio característico[2] de M se tiene que:

$$p(X) = det(M - XI) = \begin{vmatrix} c_1 - X & 1 & 0 & \cdots & 0 & 0 \\ c_2 & -X & 1 & \cdots & 0 & 0 \\ c_3 & 0 & -X & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ c_L & 0 & 0 & \cdots & 0 & -X \end{vmatrix} = \tag{1.6}$$

$$= (-X)^{L-1}(c_1 - X) - c_2(-X)^{L-2} + c_3(-X)^{L-3} - \cdots + (-1)^L c_L =$$

$$= (-X)^L \left[1 - \frac{c_1}{X} - \frac{c_2}{X^2} - \cdots - \frac{c_L}{X^L}\right] = (-1)^L \left(X^L - \sum_{i=1}^{L} c_i X^{L-i}\right) = \pm P^*(X)$$

coincide con el polinomio característico del LFSR.

Periodicidad y ecuación característica

El Teorema de Cayley-Hamilton afirma que toda matriz anula su polinomio característico. Es decir, $P(M) = 0$. De aquí se obtiene la relación de recurrencia que define el LFSR. Si el polinomio característico divide a $1 - X^m$, entonces, $M^m = I$. Por lo tanto, por el Corolario 1.18 encontrar el periodo de una secuencia es equivalente a encontrar el menor m tal que $M^m = I$.

[2] El polinomio característico de una matriz M se define como $P_M(X) = det(M - XI)$, donde I de la nota la matriz identidad de tamaño el mismo que el de M.

2

Snow 3G

En este capítulo se realizará un análisis del diseño del cifrado Snow 3G, que constituye el núcleo en el que se basa la integridad y la confidencialidad de las comunicaciones 4G. Se hará además, un estudio teórico de un ataque por inducción de fallos contra la seguridad del Snow 3G. En este capítulo se han utilizado varias referencias, en particular; para más detalles referimos al lector a [8], [12] y [5].

2.1. Descripción del snow 3G

El algoritmo de cifrado Snow 3G es un sistema criptográfico que ha sido adoptado como el algoritmo de cifrado estándar para proteger la privacidad y seguridad de los datos en las comunicaciones móviles de tercera generación (3G) y cuarta generación (4G), conocida como 3GPP. Es un algoritmo de cifrado simétrico, lo que significa que utiliza una sola clave secreta para cifrar y descifrar los datos.
El algoritmo Snow 3G utiliza una función de generación de secuencias para cifrar los datos que se transmiten entre un dispositivo móvil y una red móvil. Produce una secuencia de palabras de 32 bits utilizando una clave de 128 bits y un estado de inicialización de 128 bits.
La fortaleza del Snow 3G radica en su capacidad para generar secuencias de bits pseudoaleatorias de alta calidad y su resistencia a diferentes tipos de ataques criptográficos como, por ejemplo, a ataques basados en métodos de enmascaramiento lineal[1], que eran efectivos con el generador Snow 2.0. El diseño del Snow 3G, se basa en su predecesor pero incluyendo cambios para evitar los ataques existentes. La Figura 2.1 muestra el esquema del generador Snow 3G. Como se puede observar, este cifrado consta de dos partes principales: un LFSR y una máquina de estados finitos (FSM).

[1] Los ataques basados en métodos de enmascaramiento lineal son ataques que se aprovechan de la linealidad del "enmascaramiento"de los datos para inferir información sobre los datos originales y la clave secreta usada. La clave está en analizar los valores de salida y encontrar patrones que revelen información sobre la clave secreta.

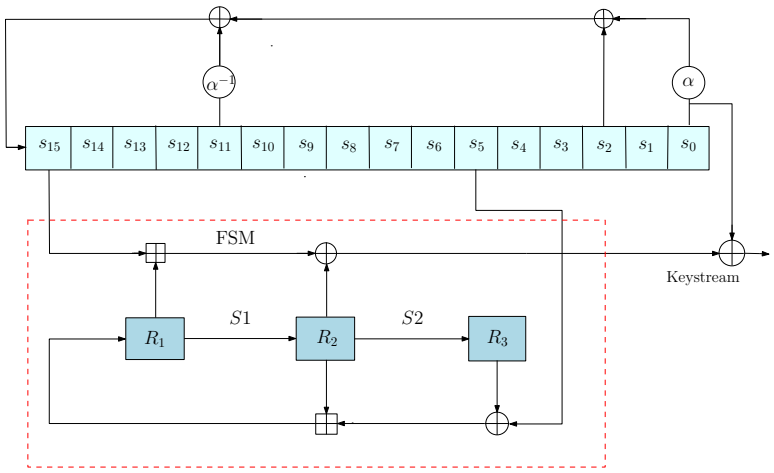

Figura 2.1: Representación del Snow 3G.

El LFSR puede usarse en modos de operación diferentes: de inicialización y de generación de la secuencia cifrante. En el modo de inicialización, este se inicia con una clave de 128 bits que consiste en 4 palabras de 32 bits, escogidas arbitrariamente k_0, k_1, k_2, k_3 y una variable de inicialización (IV) que consiste en 4 palabras de 32 bits IV_0, IV_1, IV_2, IV_3 como sigue: (Supongamos que $\mathbf{1}$ denota la palabra de todo unos)

$$s_{15} = k_3 \oplus IV_0;\ \ s_{14} = k_2;\ \ s_{13} = k_1;\ \ s_{12} = k_0 \oplus IV_1;$$

$$s_{11} = k_3 \oplus \mathbf{1};\ \ s_{10} = k_2 \oplus \mathbf{1} \oplus IV_2;\ \ s_9 = k_1 \oplus \mathbf{1} \oplus IV_3;\ \ s_8 = k_0 \oplus \mathbf{1}$$

$$s_7 = k_3;\ \ s_6 = k_2;\ \ s_5 = k_1;\ \ s_4 = k_0$$

$$s_3 = k_3 \oplus \mathbf{1};\ \ s_2 = k_2 \oplus \mathbf{1}; s_1 = k_1 \oplus \mathbf{1};\ \ s_0 = k_0 \oplus \mathbf{1}$$

Se inicializan además los registros de la FSM con $R_1 = R_2 = R_3 = 0$. Luego, el cifrado se ejecuta en un modo especial sin producir ninguna salida, tal como se muestra en la Figura 2.2. Y se repite el algoritmo siguiente 32 veces:

1. Se actualiza el estado de la FSM.
2. Se obtiene un valor de salida de la FSM, de 32 bits, que se usa para obtener el estado inicial del LFSR.

Para la generación del flujo de salida, con cada pulso de reloj $t > 0$ la salida f^t del FSM y la palabra de salida z^t vienen dadas por las siguientes ecuaciones:

$$f^t = (s_{15}^t \boxplus R_1^t) \oplus R_2^t \tag{2.1}$$

$$z^t = s_0^t \oplus f^t \tag{2.2}$$

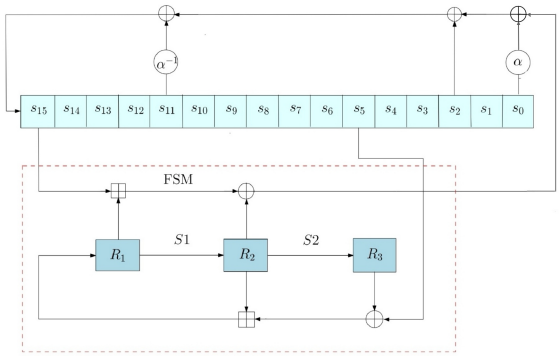

Figura 2.2: Representación del Snow 3G en modo de inicialización.

donde $\boxplus$ es la suma módulo 2^{32}, definida en la Sección 2.1 y $\oplus$ es la operación XOR, que consiste en operar bit a bit haciendo uso de la suma módulo 2 sin acarreo. Es decir, la salida de $a \oplus b$ es uno si $a \neq b$ y cero si $a = b$.
El LFSR consta de 16 etapas $s_0, \ldots, s_{15}$ cada una de ellas contiene 32 bits. La función de actualización se define por su polinomio de retroalimentación:

$$P(x) = \alpha x^{16} + x^{14} + \alpha^{-1}x^5 + 1 \ \in \mathbb{F}_{2^{32}}[x] \tag{2.3}$$

con $c_5 = \alpha^{-1}, c_{14} = 1, c_{16} = \alpha$, donde α es una raíz del polinomio

$$f(x) = x^4 + \beta^{23}x^3 + \beta^{245}x^2 + \beta^{48}x + \beta^{239} \in \mathbb{F}_{2^8}[x] \tag{2.4}$$

y β es una raíz de

$$g(x) = x^8 + x^7 + x^5 + x^3 + 1 \in \mathbb{F}_2[x] \tag{2.5}$$

Es decir, hemos construido la extensión $\mathbb{F}_{(2^{32})^{16}} = \mathbb{F}_2[\beta, \alpha, \gamma]$ con β raíz de $g(x)$, α raíz de $f(x)$ y γ raíz de $P(x)$. Con sage se puede comprobar que:

- $g(x)$ es irreducible en $\mathbb{F}_2[x]$ y además es primitivo.
- $f(x)$ es irreducible en $\mathbb{F}_{2^8}[x]$ y además es primitivo.

Queda entonces la siguiente relación de recurrencia del LFSR:

$$s_{t+16} = \alpha s_t \oplus s_{t+2} \oplus \alpha^{-1}s_{t+11} = \alpha s_t + s_{t+2} + \alpha^{-1}s_{t+11} \bmod 2, \ \forall t > 0$$

Además, las dos multiplicaciones implicadas en el LFSR pueden ser implementadas como desplazamientos de bytes y una aplicación XOR con alguno de los 2^8 patrones posibles.

- Dado que β es raíz del polinomio primitivo $g(x)$, es decir, es irreducible y β genera a $\mathbb{F}_{2^8}$. Por tanto, el conjunto $\{0, 1, \beta, \beta^2, \ldots, \beta^{2^8-2}\}$ representa todo

el cuerpo $\mathbb{F}_{2^8}$, por lo que, cualquier elemento del cuerpo puede representarse como la evaluación en β de un polinomio en $\mathbb{F}_2[x]$ de grado menor que 8. O bien, con un byte cuyos bits se corresponden con los coeficientes de dicho polinomio. Así, la multiplicación de dos elementos en $\mathbb{F}_{2^8}$ resulta de la multiplicación de los dos polinomios correspondientes módulo $g(x)$.

- Como α es una raíz del polinomio primitivo $f(x)$ en $\mathbb{F}_{2^8}[x]$, se puede generar el cuerpo extendido $\mathbb{F}_{2^{32}}$ como sucesivas potencias de α, de forma que el conjunto $\{0, 1, \alpha, \alpha^2, \ldots, \alpha^{2^{32}-2}\}$ define todo el cuerpo $\mathbb{F}_{2^{32}}$. Por tanto, cualquier elemento del cuerpo puede ser representado mediante un polinomio en $\mathbb{F}_{2^8}[x]$ de grado menor que 4, o bien con una palabra de 4 bytes correspondientes a los 4 coeficientes de dicho polinomio. Así, las operaciones en $\mathbb{F}_{2^{32}}$ se corresponden con operaciones de polinomios módulo $f(x)$. Es decir, sea $(c_3, c_2, c_1, c_0) \in \mathbb{F}_{2^8}^4$ una palabra de 4 bytes que se corresponde con el polinomio $c_3\alpha^3 + c_2\alpha^2 + c_1\alpha + c_0 \in \mathbb{F}_{2^{32}}$. La operación $\alpha \cdot (c_3, c_2, c_1 c_0) \mod f(\alpha) = (c_2 + c_3\beta^{23}, c_1 + c_3\beta^{245}, c_0 + c_3\beta^{48}, c_3\beta^{239}) \in \mathbb{F}_{2^8}^4$ se puede implementar de forma rápida mediante tablas precalculadas. Análogamente, el resultado de multiplicar α^{-1} por cualquier palabra de 4 bytes es $(c_0\beta^{16}, c_3 + c_0\beta^{239}, c_2 + c_0\beta^6, c_1 + c_0\beta^{64})$. En conclusión, una rápida implementación binaria de esta operación puede estar basada en tablas precalculadas de los valores $(c\beta^{16}, c\beta^{39}, c\beta^6, c\beta^{64}), \forall c \in \mathbb{F}_{2^8}$.

La máquina de estados finitos FSM constituye la parte no lineal del generador y se alimenta de dos valores de entrada, los estados s_0 y s_{15}, procedentes del LFSR. La FSM está formada por tres registros de 32 bits, R_1, R_2 y R_3 y dos cajas de sustitución o S-boxes[2], S_1 y S_2, que se utilizan para actualizar los registros R_2 y R_3. La actualización de la FSM viene dada por las siguientes ecuaciones:

$$R_1^{t+1} = (s_5^t \oplus R_3^t) \boxplus R_2^t \tag{2.6}$$

$$R_2^{t+1} = S_1(R_1^t) \tag{2.7}$$

$$R_3^{t+1} = S_2(R_2^t) \tag{2.8}$$

Cada caja S_i de sustitución está implementada con 4 cajas de sustitución $(S_i - T_0, \ldots, S_i - T_3)$ que toman 8 bits de entrada y los transforman en 32 bits de salida. De esta forma los 32 bits de entrada se dividen en 4 entradas de 8 bits, luego cada una de esas entradas es utilizada en una caja $S_i - T_j$ y, finalmente, las 4 salidas se combinan con una operación XOR. En la Figura 2.3 se muestra un esquema de las S-boxes. Las tablas de búsqueda de S_1 y S_2 son distintas, haciendo de este modo más seguro el algoritmo.

[2] Las cajas S-boxes son tablas bidimensionales que contienen una lista de valores de entrada y una lista de valores de salida. Cada valor de entrada se sustituye por el valor correspondiente de salida en la tabla, produciendo una transformación no lineal de los datos de entrada, intentando evitar los ataques que eran efectivos en las versiones anteriores del Snow.

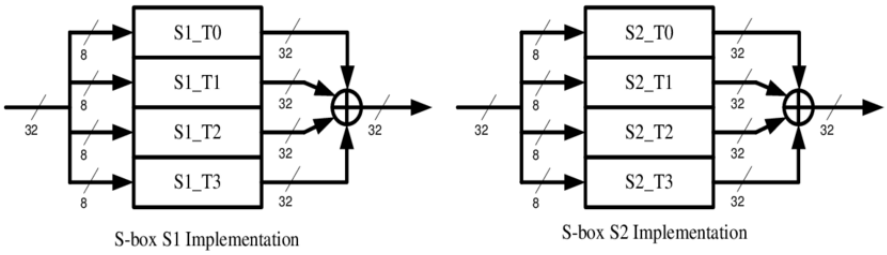

Figura 2.3: S-boxes

- La caja S_1 se basa en la función del algoritmo de Rindjael [7], donde tanto la entrada como la salida se expresa en forma hexadecimal, sustituyendo el valor de entrada por el correspondiente en la Figura 2.4. Por ejemplo, $S_1(42) = S_1(0 \times 2A) = 0 \times E5 = 229$. Véase más información de esta implementación en [20].
- La caja S_2 funciona análogamente, con la diferencia que su tabla de búsqueda es la que se muestra en la Figura 2.5.

x_1 / x_0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Figura 2.4: Tabla de búsqueda S_1.

x_1 / x_0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	25	24	73	67	D7	AE	5C	30	A4	EE	6E	CB	7D	B5	82	DB
1	E4	8E	48	49	4F	5D	6A	78	70	88	E8	5F	5E	84	65	E2
2	D8	E9	CC	ED	40	2F	11	28	57	D2	AC	E3	4A	15	1B	B9
3	B2	80	85	A6	2E	02	47	29	07	4B	0E	C1	51	AA	89	D4
4	CA	01	46	B3	EF	DD	44	7B	C2	7F	BE	C3	9F	20	4C	64
5	83	A2	68	42	13	B4	41	CD	BA	C6	BB	6D	4D	71	21	F4
6	8D	B0	E5	93	FE	8F	E6	CF	43	45	31	22	37	36	96	FA
7	BC	0F	08	52	1D	55	1A	C5	4E	23	69	7A	92	FF	5B	5A
8	EB	9A	1C	A9	D1	7E	0D	FC	50	8A	B6	62	F5	0A	F8	DC
9	03	3C	0C	39	F1	B8	F3	3D	F2	D5	97	66	81	32	A0	00
A	06	CE	F6	EA	B7	17	F7	8C	79	D6	A7	BF	8B	3F	1F	53
B	63	75	35	2C	60	FD	27	D3	94	A5	7C	A1	05	58	2D	BD
C	D9	C7	AF	6B	54	0B	E0	38	04	C8	9D	E7	14	B1	87	9C
D	DF	6F	F9	DA	2A	C4	59	16	74	91	AB	26	61	76	34	2B
E	AD	99	FB	72	EC	33	12	DE	98	3B	C0	9B	3E	18	10	3A
F	56	E1	77	C9	1E	9E	95	A3	90	19	A8	6C	09	D0	F0	86

Figura 2.5: Tabla de búsqueda S_2.

Los microprocesadores en los teléfonos móviles trabajan con palabras de 32 bits, así que este algoritmo descrito en $\mathbb{F}_{2^{32}}$ se supone eficiente para esta arquitectura.

2.2. Ataque por inducción de fallos

Un ataque por inducción de fallos es una técnica de ataque en criptografía que consiste en introducir de manera intencional errores en el sistema que se utiliza con el fin de obtener información sensible del mismo, como la clave secreta utilizada para cifrar los datos, y así comprometer su seguridad. Estos errores

pueden ser producidos de diversas formas como, por ejemplo, aplicando choques eléctricos, manipulando el reloj del sistema o sobrecargando la memoria.
En esta sección se describirá el ataque por inducción de fallos propuesto contra el cifrado Snow 3G sin especificar cómo producir estos fallos.

2.2.1. Representación algebraica de $\boxplus$

El primer paso es obtener una representación algebraica de la operación $\boxplus$. La expresión de cualquier número natural en base q es única por el *Teorema fundamental de la Representación en Base q* [16][páginas 194-213]. Es decir, en particular para $q = 2$, existen unos únicos $x_i \in \{0, 1\}$ $i = 0, \ldots, n-1$ tales que:

$$x = \sum_{i=0}^{n-1} 2^i x_i = (x_{n-1}, \ldots, x_0)_2, \ x_i \in \{0, 1\}$$

Definición 2.1 (Aritmética módulo 2^n). *Sean x, y números enteros positivos menores que $2^n + 1$, se define su suma en aritmética módulo 2^n como la suma de sus representaciones binarias donde se tiene en cuenta las unidades que se llevan al siguiente nivel (se trata de sumas de cadenas de n bits). De esta forma, si las representaciones binarias de x e y son $x = \sum_{i=0}^{n-1} 2^i x_i$ e $y = \sum_{i=0}^{n-1} 2^i y_i$;*

$$x + y \ mod \ 2^n \ se \ define \ como \ \sum_{i=0}^{n-1} 2^i (x_i + y_i + c_i \ mod \ 2)$$

donde c_i es el acarreo de la suma de los coeficientes del nivel anterior.

En las siguientes líneas se verá cómo describir la suma en aritmética módulo 2^n. Para ello, primero necesitamos estudiar algunas propiedades en $\mathbb{F}_2$.

- Para todo $a \in \mathbb{F}_2$ se tiene que $a + a = 0$ y $a^2 = a$
- El operador lógico "and" ($\wedge$) se puede escribir de forma aritmética como

$$a \wedge b = a \cdot b. \tag{2.9}$$

- El operador lógico "or"($\vee$) se puede escribir de forma aritmética como

$$a \vee b = a + b + a \cdot b. \tag{2.10}$$

Recordemos cómo funcionan los operadores "and" ($\wedge$) y "or"($\vee$) con la Tabla 2.1. Con estas propiedades sigamos con nuestro objetivo de describir la suma en aritmética módulo 2^n de dos secuencias de n bits utilizando operaciones algebraicas.
Vamos a realizarla para $x, y \in \mathbb{N}$ con $x, y \leq 2^n$. Es decir, que podemos calcular $x + y = z \bmod 2^n$ donde

a	b	$\wedge$	$\vee$
1	1	1	1
1	0	0	1
0	1	0	1
0	0	1	0

Tabla 2.1: Operaciones "and" y "or"

$$x = \sum_{i=0}^{n-1} 2^i x_i = (x_{n-1}, \ldots, x_0)_2, \; y = \sum_{i=0}^{n-1} 2^i y_i = (y_{n-1}, \ldots, y_0)_2,$$

$$z = \sum_{i=0}^{n-1} 2^i z_i = (z_{n-1}, \ldots, z_0)_2$$

siendo x_0, y_0, z_0 los bits menos significativos.

- En el bit z_0 no tenemos acarreo, luego $z_0 = x_0 + y_0$.
- Pero, para $i \geq 1$ se tiene que $z_i = x_i + y_i + c_i$ donde c_i es el acarreo de la posición anterior.
- Además observamos que $z_i = x_i + y_i + c_i$ con $c_{i+1} = 1$ si al menos dos de los bits x_i, y_i, c_i son unos. En particular, $c_1 = x_0 \cdot y_0$.

Es decir: $c_{i+1} = (x_i \wedge y_i) \vee (x_i \wedge c_i) \vee (y_i \wedge c_i)$.
Ahora, utilizando las expresiones (2.9) y (2.10) se tiene que:

$$c_{i+1} = (x_i y_i) \vee (x_i c_i) \vee (y_i c_i) = (x_i y_i + x_i c_i + x_i y_i c_i) \vee (y_i c_i) =$$

$$= x_i y_i + x_i c_i + x_i y_i c_i + y_i c_i + (x_i y_i + x_i c_i + x_i y_i c_i)(y_i c_i) =$$

$$= x_i y_i + x_i c_i + x_i y_i c_i + y_i c_i + x_i y_i^2 c_i + x_i y_i c_i^2 + x_i y_i^2 ci^2 =$$

$$= x_i y_i + x_i c_i + y_i c_i + 4 x_i y_i c_i = x_i y_i + x_i c_i + y_i c_i =$$

$$c_{i+1} = x_i y_i + (x_i + y_i) c_i$$

En la segunda igualdad hemos aplicado la propiedad asociativa. En la quinta igualdad hemos aplicado propiedades de $\mathbb{F}_2$ y en la última igualdad la propiedad distributiva.
Por tanto, la suma $x + y = z \bmod 2^n$ se puede escribir como un sistema de n ecuaciones de grado a lo sumo 2 como sigue:

$$\begin{cases} z_0 = x_0 + y_0 \\ z_1 = x_1 + y_1 + c_1 \\ \quad \vdots \\ z_{n-1} = x_{n-1} + y_{n-1} + c_{n-1} \end{cases} \quad \text{con} \quad \begin{cases} c_1 = x_0 y_0 \\ c_2 = x_1 y_1 + (x_1 + y_1) c_1 \\ \quad \vdots \\ c_{n-1} = x_{n-2} + y_{n-2} + (x_{n-2} + y_{n-2}) c_{n-2} \end{cases}$$

Sustituyendo las ecuaciones de c_i en el primer sistema, se llega al siguiente sistema de ecuaciones algebraicas:

$$\begin{cases} z_0 = x_0 + y_0 \\ z_1 = x_1 + y_1 + x_0 y_0 \\ z_2 = x_2 + y_2 + x_1 y_1 + (x_1 + y_1) x_0 y_0 = x_2 + y_2 + (x_1 + y_1)(x_1 + y_1 + z_1) \\ \quad \vdots \\ z_i = x_i + y_i + x_i y_i + (x_{i-1} + y_{i-1})(x_{i-1} + y_{i-1} + z_{i-1}) \\ \quad \vdots \\ z_{n-1} = x_{n-1} + y_{n-1} + x_{n-2} y_{n-2} + (x_{n-2} + y_{n-2})(x_{n-2} + y_{n-2} + z_{n-2}) \end{cases}$$

2.2.2. Método de Armknecht y Meier

Para presentar el ataque que se realizará se explicará primero el método de Armcknecht y Meier que muestra un ataque por inducción de fallos para el caso genérico de un generador combinado con memoria. Esta sección servirá como base para comprender el ataque al Snow 3G propuesto en [8]. Para este método de ataque podemos asumir que:

- El atacante tiene acceso al dispositivo físico.
- Puede causar un fallo desconocido en alguno de los registros. Conoce el registro afectado pero no controla ni el instante del fallo, ni el error inducido.
- Puede reiniciar el dispositivo con los mismos valores iniciales todas las veces que él quiera.

La idea de este ataque es encontrar el conjunto de todos los estados iniciales del LFSR posibles en relación a un flujo de salida observado e inducir un fallo en los registros de memoria para analizar los efectos producidos, reduciendo de tamaño el conjunto de los estados iniciales posibles. Con las diferencias observadas el atacante recupera ecuaciones que tienen por variables los registros de memoria (entre otras variables). Si el atacante obtiene un numero suficiente de ecuaciones aplicando técnicas adecuadas puede recuperar el estado inicial del LFSR.
Muchos sistemas de cifrado por flujo combinan uno o varios LFSR y bits de memoria actualizados de manera no lineal. A estos sistemas se les denomina de manera más formal como un generador combinado con memoria.

Definición 2.2. *Un **generador combinado con memoria** es un par C=(f,ϕ) que toma k elementos como entrada y l bits de memoria, y se define como una máquina de estados finitos compuesta por:*

1. *Una función de salida, que produce z_t:*

$$f : \{0,1\}^l \times \{0,1\}^k \longrightarrow \{0,1\}$$

2. *Una función de actualización de la memoria:*

$$\phi : \{0,1\}^l \times \{0,1\}^k \longrightarrow \{0,1\}^l$$

En cada pulso de reloj t, $X_t \in \{0,1\}^k$ *es la palabra de salida producida por el LFSR y* $Q_t \in \{0,1\}^l$ *es el conjunto de los bits de memoria. La salida correspondiente se define según las siguientes ecuaciones:*

$$z_t = f(Q_t, X_t) \text{ y } Q_{t+1} = \phi(Q_t, X_t) \ \forall t \geq 0 \tag{2.11}$$

Como Q_{t+1} depende solo de Q_0 y $X_0, \ldots, X_t$, por la Ecuación (2.11). Entonces, para un τ fijo, una secuencia producida por bits de claves de flujo consecutivas $z_0, \ldots, z_\tau$ depende únicamente de $Q_0, X_0, \ldots, X_\tau$. La idea es ignorar Q_0 y buscar el subconjunto de todos los $X_0, \ldots, X_t$ que son compatibles con la secuencia de salida. Luego, mediante la inyección de un fallo el atacante puede reducir el tamaño de este conjunto, ya que obtiene una nueva ecuación con las mismas variables.
Describimos el método de manera resumida en el siguiente algoritmo. Para obtener más detalles consultar [2].

INPUT: Un generador combinado con memoria $C(k,l)$, $N \geq 1$ y $\tau \geq 0$.
OUTPUT: El estado inicial del LFSR.

- Deducir para cada una de las salidas $(z_0, \ldots, z_\tau)$, el conjunto de entradas posibles $\{(X_0, \ldots, X_\tau)\}$.
- Ejecutar el generador una vez y observar el flujo de salida $z_0, \ldots, z_R$.
- Para $t = 1, \ldots, R - \tau$ estudiar el conjunto de entradas posibles $\{(X_t, \ldots, X_{t+\tau})\}$.
- Mientras que el número de ecuaciones de grado d sea más pequeño que N realizar:
  1. Inducir un fallo en uno de los registros de memoria.
  2. Buscar el primer instante t' en el que existe una diferencia $z_{t'} \neq z'_{t'}$.
  3. Determinar dicha diferencia : $\delta(t) := X_t \oplus X'_t \ \forall t \geq t'$.
  4. Para $t \geq t'$ reducir el número de entradas posibles $\{(X_t, \ldots, X_{t+\tau})\}$.
  5. Contar el número de ecuaciones sobre las variables $\{(X_t, \ldots, X_{t+\tau})\}$ de grado $\leq d$ para todo $t = 0, \ldots, T - \tau$.
- Buscar la solución al sistema recuperando así el estado inicial del LFSR.

Podemos observar del algoritmo anterior, que el objetivo del atacante es encontrar un sistema de ecuaciones S que relacione únicamente los bits desconocidos del LFSR. Como todos los bits pueden ser expresados como combinación lineal de los bits del estado inicial del LFSR, sustituyendo las variables del LFSR por sus correspondientes expresiones lineales, el atacante puede obtener un sistema S' de ecuaciones en los bits desconocidos del estado inicial del LFSR, del mismo grado máximo que el sistema S. Si ha obtenido suficientes ecuaciones y si el sistema se puede resolver en la práctica, podrá recuperar los bits del estado inicial. La mejor solución para el atacante es encontrar ecuaciones lineales, que

pueden ser resueltas por eliminación Gaussiana[3] en tiempo polinomial. Una parte difícil de esta tarea es encontrar el sistema inicial de ecuaciones S, así como, determinar la diferencia $\delta(t) := X_t \oplus X'_t \ \forall t \geq t'$ cuando la función de salida es altamente no lineal.

Ataque por inducción de fallos en el Snow 2.0

El cifrado en flujo Snow 2.0 (predecesor del Snow 3G) tiene la estructura que se muestra en la Figura 2.6. Se encuentran dos diferencias principales entre Snow 2.0 y Snow 3G: El Snow 2.0 únicamente tiene dos registros R_1 y R_2 y su caja S es diferente a la caja S_1 del Snow 3G, aunque su diseño es muy similar. Las

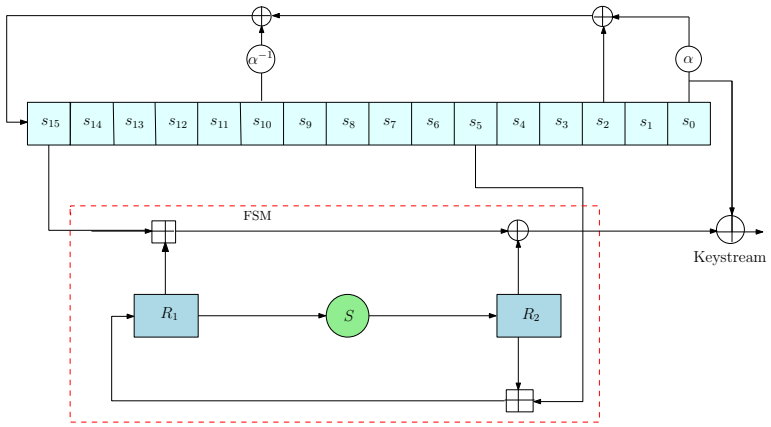

Figura 2.6: Representación del Snow 2.0

ecuaciones que describen el funcionamiento del Snow 2.0 son las siguientes:

- Ecuación de la flujo de salida:

$$z^t = (s_{15} \boxplus R_1^t) \oplus R_2^t \oplus s_0 \tag{2.12}$$

- Actualización de la FSM:

$$R_2^{t+1} = S(R_1^t) \tag{2.13}$$

$$R_1^{t+1} = R_2^t \boxplus s_5^t \tag{2.14}$$

El principio del ataque de Armknecht y Meier en Snow 2.0 es inyectar un fallo que modifique el valor del registro R_2 antes de la obtención de la palabra de salida. Denotemos por $R_1'^t$, $R_2'^t$ los registros de memoria con fallos. Una característica del Snow 2.0 es que, si en un instante t del algoritmo, se induce un fallo en R_1^t

[3] Eliminación Gaussiana: El número de operaciones en $\mathbb{F}$ requeridas para aplicar el método de Gauss-Jordan, es decir, obtener la forma escalonada reducida en una matriz cuadrada $M \in M_n(\mathbb{F})$ es del orden n^3.

o en R_2^t, entonces en el siguiente instante $t+1$ el registro que se ha modificado tiene su valor normal, mientras que, el otro registro ahora está perturbado, es decir no pueden tener un valor erróneo ambos al mismo tiempo (siempre que solo se induzca un fallo).
Se supone además que la modificación inducida debe cambiar el bit de menor peso del registro R_2 y no debe cambiar el segundo bit de menor peso. Es decir, el fallo inducido tiene la forma: $R_2^t \oplus R_2'^t = (*, *, \ldots, 0, 1)$.
Usando esta propiedad y la Ecuaciones (2.12) y (2.14) obtenemos:

$$z_{t+1} = \underbrace{(s_{15}^{t+1} \boxplus R_2^t \boxplus s_5^t)}_{P_1} \oplus R_2^{t+1} \oplus s_0^{t+1} = P_1 \oplus R_2^{t+1} \oplus s_0^{t+1}$$

$$z'_{t+1} = \underbrace{(s_{15}^{t+1} \boxplus R_2'^t \boxplus s_5^t)}_{P_2} \oplus R_2^{t+1} \oplus s_0^{t+1} = P_2 \oplus R_2^{t+1} \oplus s_0^{t+1}$$

Así:

$$z_{t+1} \oplus z'_{t+1} = P_1 \oplus P_2$$

El segundo bit menos significativo de $z_{t+1} \oplus z'_{t+1}$ es 1 si, y solo si, el bit menos significativo de $s_{15}^t \boxplus s_5^t$ es 1. Por lo tanto, el atacante puede utilizar la diferencia de salida para obtener una ecuación lineal en los bits del LFSR.
Si el atacante realiza este paso al menos en $512 = 32 \times 16$ momentos diferentes del algoritmo, obtendrá suficientes ecuaciones lineales para recuperar el estado inicial del LFSR. Es posible que se necesiten más ecuaciones si no son linealmente independientes. En promedio, tras estudios analizados en [2] cada cuatro fallos inducidos en R_2 uno tiene la forma requerida y el atacante solo puede explotar el fallo en la segunda actualización del algoritmo, por lo que el atacante necesita al menos $4 \times 2 \times 512 = 2^{12}$ palabras de la secuencia cifrada producida por la ejecución normal del algoritmo y 2^{12} palabras de la secuencia cifrada producida por el algoritmo con fallo inducido. Los detalles de este ataque están en [2]. Este ataque sigue el modelo descrito en el algoritmo ya expuesto, sin embargo, resulta fácil observar que este ataque no se puede aplicar al algoritmo Snow 3G, ya que si se induce un fallo en el registro R_2 después de 3 actualizaciones del FSM, los valores de los 3 registros R_1, R_2 y R_3 se ven afectados.

2.2.3. Modelo del ataque al Snow 3G

Para este ataque vamos a suponer que el atacante:

- Está en posesión del dispositivo físico.
- Puede reinicializar el dispositivo con el estado original e inducir un nuevo fallo.
- Conoce el registro afectado pero no puede elegir ni el momento exacto de la perturbación, ni el error inducido.

Se supone una arquitectura de software de 32 bits. En el instante t, el estado del LFSR se denota como $(s_{15}^t, \ldots, s_0^t)$. Este modelo supone que el atacante es capaz de modificar el valor de 32 bits s_i^t donde i es elegido por el atacante, pero no necesariamente el instante t. Para simplificar el modelo, se supone que el atacante induce el fallo en el espacio de RAM donde se registra la palabra s_i^t. El primer paso del ataque debe ser obtener la localización o el instante adecuado. Se supone que no solo el atacante puede inyectar diferentes fallos en los mismos instantes de ejecución, sino también en instantes diferentes.

2.2.4. Paso preliminar: localizar el fallo

El primer paso que vamos a estudiar en este ataque es deducir dónde es más adecuado inyectar un fallo, es decir, en qué localización el ataque es más efectivo, o de forma equivalente, dónde obtenemos más ecuaciones. El atacante tendrá que determinar además dónde está produciendo un error.
Vamos a suponer que el atacante ha encontrado la localización del estado del Snow 3G en el chip y podrá, por tanto, analizar las diferencias obtenidas en el flujo de salida, inducidas tras algún fallo.
Veamos por ejemplo, que ocurre si se inyecta un fallo en la etapa del LFSR s_0^t:

1. En el primer toque de reloj este fallo se transmite en la salida produciendo la primera diferencia con el cifrado original. Además, en este primer paso, por el polinomio de retroalimentación, este fallo también se trasmite a la etapa s_{15}^{t+1}.
2. En el segundo toque de reloj, como tenemos una variación en la etapa s_{15}^{t+1} tendremos de nuevo una diferencia en la salida (segunda diferencia frente al cifrado sin fallos). Además, el fallo se desplaza a la etapa s_{14}^{t+2}.
3. En el tercer toque de reloj, el fallo se desplaza a la etapa s_{13}^{t+3}. En este toque no tendremos ninguna diferencia en la salida frente al cifrado original.
4. En el cuarto toque de reloj el fallo se desplaza a la etapa s_{12}^{t+4}. Nuevamente aquí no tendremos ninguna diferencia en la salida frente al cifrado original.
5. En el quinto toque de reloj el fallo se desplaza a la etapa s_{11}^{t+5}, e igual que antes, no observamos diferencias en la salida.
6. En el sexto toque de reloj el fallo se desplaza a la etapa s_{15}^{t+6} por el polinomio de retroalimentación, así como a la etapa s_{10}^{t+6}, sin observar diferencias en la salida.
7. En el séptimo toque de reloj, como tenemos una variación en s_{15}^{t+6}, tendremos una diferencia en la salida. Además, el fallo se desplaza tanto a la etapa s_9^{t+7} como a s_{14}^{t+7}.
8. En el octavo toque de reloj el fallo se desplaza a las etapas s_8^{t+8} y s_{13}^{t+8}, nuevamente no observamos ninguna diferencia en la salida.
9. En el noveno toque de reloj el fallo se desplaza a las etapas s_7^{t+9} y s_{12}^{t+9}, sin producir diferencias en la salida.

10. En el décimo toque de reloj el fallo se desplaza a las etapas s_6^{t+10} y s_{11}^{t+10}, sin producir de nuevo ninguna diferencia en la salida.

De ahí que, si representamos con un uno la diferencia en la salida y cero si no hay diferencia, el resultado de inyectar un fallo en s_0^t sería: 1100001000. En la Figura 2.7 se muestra este ejemplo de forma visual.
En la Tabla 2.2 se muestra el patrón de fallos que se obtiene al inducir un error en la localización que se indica en la columna de la izquierda. Todos los patrones parten de la primera diferencia que se observa. Es decir, si inyectamos un fallo en s_1^t, en el primer toque de reloj no se obtiene ninguna diferencia en la salida, pero en el siguiente toque el patrón de fallos será el mismo que hemos explicado para s_0^t.

Localización del fallo	Diferencia en la salida
R_1, R_2, R_3	1 1 1 1 1 1 1 1 1 1
s_0, s_1	1 1 0 0 0 0 1 0 0 0
s_2, s_3, s_4	1 1 1 0 0 1 0 1 0 0
$s_5, \ldots, s_{10}$	1 1 1 1 1 1 1 1 1 1
$s_{11}, \ldots, s_{14}$	1 0 0 0 0 1 1 1 1 1
$s_{11}, \ldots, s_{14}$	1 0 0 0 0 1 0 0 0 0

Tabla 2.2: Diferencias en el flujo de salida inducidas por un fallo.

Solo se considerarán fallos que modifiquen el valor antes de cualquier computación que lo involucre.
El atacante sabrá que ha encontrado la localización correcta cuando obtenga el patrón dado por la tabla anterior.

Observación 2.3. Cabe destacar que un patrón esperado podría no aparecer después de inducir un fallo, es decir, que tras el fallo inducido el valor quede el mismo, pero vamos a probar que esta situación es muy poco probable, por lo que, este caso podemos considerarlo despreciable. Consideramos los valores de las palabras de salida como aleatorios y uniformemente distribuidos.

- La probabilidad de que el flujo de salida no coincida con el valor regular es equivalente a que haya al menos un bit, de los 32, que sea distinto al esperado. O análogamente, será 1 menos la probabilidad de que todos los bits sean iguales a los esperados en el cifrado original, es decir $1 - \frac{1}{2} \cdot \ldots \cdot \frac{1}{2} = 1 - \frac{1}{2^{32}}$.
- De hecho, la probabilidad de que el patrón de salida no tenga la forma dada en la tabla es equivalente a que haya al menos 1 salida distinta, o de forma análoga a lo anterior, 1 menos la probabilidad de que hayan k salidas iguales siendo k el número de unos. Por tanto la probabilidad máxima en nuestro caso (estamos estudiando el patrón de 10 salidas) es

$$1 - (1 - \frac{1}{2^{32}})^{10} \approx 2.33 \cdot 10^{-9}$$

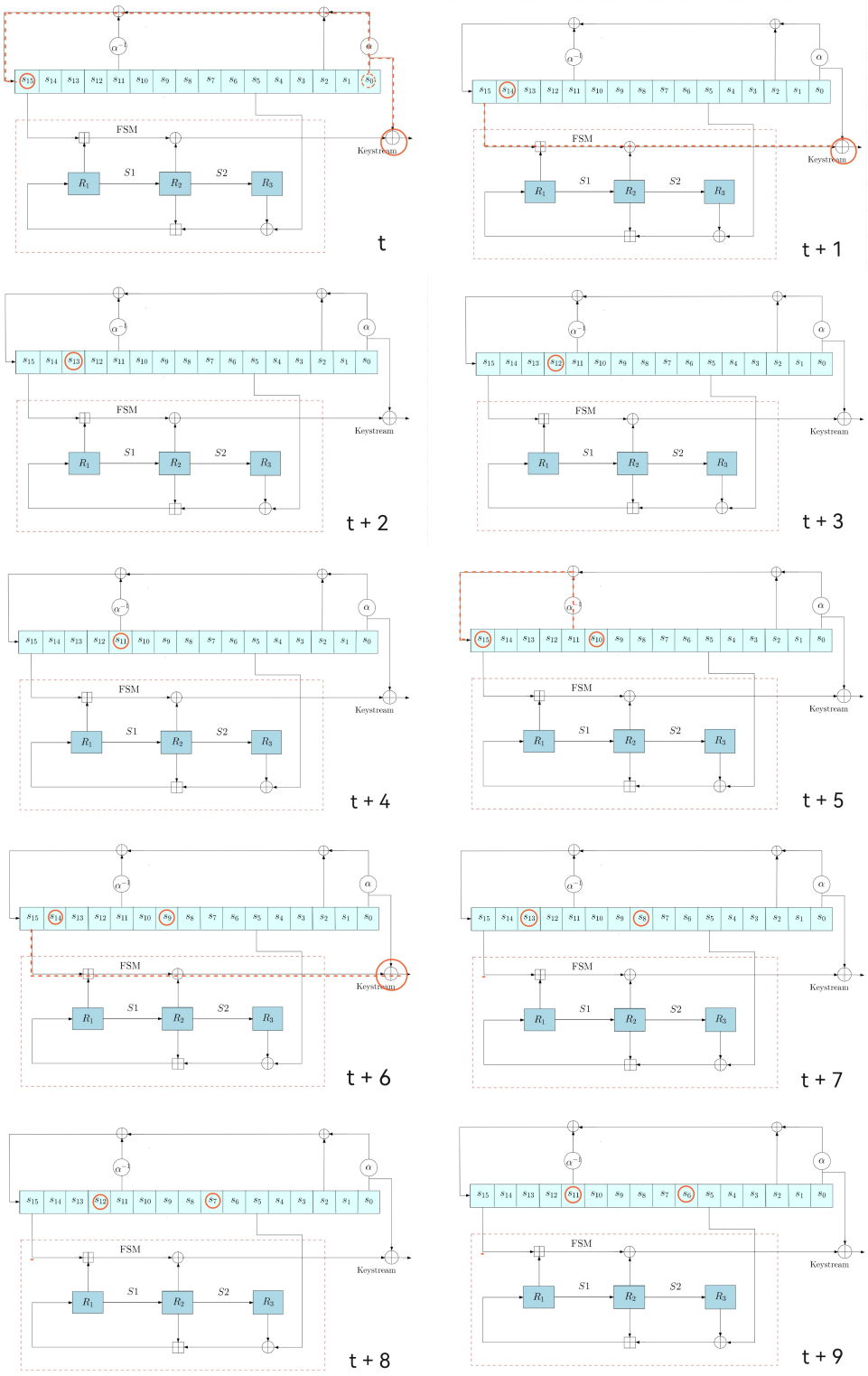

Figura 2.7: Representación de 10 pulsos de reloj al inyectar un fallo en s_0^t. Señalamos con un círculo rojo cómo se transmiten los fallos inducidos.

2.2.5. Visión general del ataque en el Snow 3G

El segundo paso del ataque que se realizará consiste en recuperar un estado completo del LFSR. En primer lugar, el atacante debe inducir un fallo en una de estas tres palabras del LFSR, s_0, s_3, s_4 (se justificará durante esta sección por qué se han elegido estas posiciones). El método para localizar donde está induciendo un fallo el atacante fue explicado en la sección 2.2.4. Sabemos dónde hemos inducido un fallo pero no controlamos qué fallo hemos introducido, esto se observa un toque de reloj después de ver la primera diferencia en la salida. La nueva diferencia es exactamente la que induce el fallo en la palabra del LFSR. En efecto, si el fallo modifica $s_3^{t_0}$, en el instante $t_0 + 3$, tendremos, usando las Ecuaciones (2.1) y (2.2), la ecuación siguiente:

$$z^{t_0+3} \oplus z'^{t_0+3} = (s_0^{t_0+3} \oplus f^{t_0+3}) \oplus (s_0'^{t_0+3} \oplus f'^{t_0+3}) = (s_3^{t_0} \oplus f^{t_0+3}) \oplus (s_3'^{t_0} \oplus f'^{t_0+3})$$

$$= (s_3^{t_0} \oplus [(s_{15}^{t_0+3} \boxplus R_1^{t_0+3}) \oplus R_2^{t_0+3}]) \oplus (s_3'^{t_0} \oplus [(s_{15}'^{t_0+3} \boxplus R_1'^{t_0+3}) \oplus R_2'^{t_0+3}])$$

Sabemos que $R_1'^{t_0+3} = R_1^{t_0+3}$ y $R_2'^{t_0+3} = R_2^{t_0+3}$, pues el error todavía no ha llegado a modificar la máquina FSM. Además $s_{15}^{t_0+3} = s_4^{t_0} = s_{15}'^{t_0+3}$. Luego, se tiene que:

$$z^{t_0+3} \oplus z'^{t_0+3} = (s_3^{t_0} \oplus s_3'^{t_0}) \oplus 2[(s_{15}^{t_0+3} \boxplus R_1^{t_0+3}) \oplus R_2^{t_0+3}]$$

Luego,

$$z^{t_0+3} \oplus z'^{t_0+3} = s_3^{t_0} \oplus s_3'^{t_0} \tag{2.15}$$

De este modo conocemos la diferencia entre $s_3^{t_0}$ original y el nuevo $s_3'^{t_0}$ en el que hemos inducido un fallo. Cabe aclarar que esto solo ocurre si se inyecta el fallo antes de s_5^t.

Observación 2.4. Además, por la linealidad del LFSR vamos a conocer

$$\triangle s_i^{t_0+j} \ \forall i \in \{0, \ldots, 15\}, \ \forall j \geq 0$$

donde $\triangle s_i^{t_0+j} = s_i^{t_0+j} \oplus s_i'^{t_0+j}$. En particular $\forall t \geq t_0$ conocemos $\triangle s_{15}^t$.

De la ecuación del flujo de salida: $z^t = s_0^t \oplus f^t = s_0^t \oplus (s_{15}^t \boxplus R_1^t) \oplus R_2^t$ y por propiedades del operador XOR, llegamos a $z^t \oplus R_2^t \oplus s_0^t = (s_{15}^t \boxplus R_1^t)$. Por lo tanto, para cada pulso de reloj, tenemos la siguiente ecuación:

$$s_{15}^t \boxplus R_1^t = w^t \tag{2.16}$$

donde $w^t = R_2^t \oplus s_0^t \oplus z^t$. Esta ecuación siguiendo la sección 2.2.1 puede ser descrita como un sistema de 32 ecuaciones en el cuerpo $\mathbb{F}_2$, de grado a lo sumo 2, en 32×3 variables, que son los bits desconocidos de s_{15}^t, R_1^t y w^t (recordemos que cada uno de estos elementos está formado por 32 bits).
Los elementos s_{15}^t, R_1^t y w^t pueden escribirse como:

$$s_{15}^t = (a_{31}^{15t}, \ldots, a_0^{15t}),\ R_1^t = (r_{31}^{1t}, \ldots, r_0^{1t}),\ w^t = (w_{31}^t, \ldots, w_0^t)$$

con $a_i^{15t}, r_i^{1t}, w_i^t \in \mathbb{F}_2\ \forall i \in \{0, \ldots, 31\}$, $\forall t \geq t_0$.
Ahora, utilizando la Sección 2.2.1 se buscan soluciones en $\mathbb{F}_2$ del siguiente sistema:[4]

$$\begin{cases} w_0^t = a_0^{15t} + r_0^{1t} \\ w_1^t = a_1^{15t} + r_1^{1t} + a_0^{15t} r_0^{1t} \\ w_2^t = a_2^{15t} + r_2^{1t} + (a_1^{15t} + r_1^{1t})(a_1^{15t} + r_1^{1t} + w_1^t) \\ \quad \vdots \\ w_i = a_i^{15t} + r_i^{1t} + a_i^{15t} r_i^{1t} + (a_{i-1}^{15t} + r_{i-1}^{1t})(a_{i-1}^{15t} + r_{i-1}^{1t} + w_{i-1}^t) \\ \quad \vdots \\ w_{31}^t = a_{31}^{15t} + r_{31}^{1t} + a_{30}^{15t} r_{30}^{1t} + (a_{30}^{15t} + r_{30}^{1t})(a_{30}^{15t} + r_{30}^{1t} + w_{30}^t) \end{cases}$$

Luego, la ecuación tras la ejecución con el fallo es: $(s_{15}^t \oplus \triangle s_{15}^t) \boxplus (R_1^t \oplus \triangle R_1^t) = w^t \oplus \triangle w^t$. Si este fallo no modifica la máquina FSM, se tiene un nuevo sistema asociado a la ecuación:

$$\underbrace{(s_{15}^t \oplus \triangle s_{15}^t)}_{s_{15}'^t} \boxplus R_1^t = \underbrace{w^t \oplus \triangle w^t}_{w_t'} \tag{2.17}$$

Por la Nota 2.4 conocemos $\triangle s_{15}^t$. Por otra parte: $\triangle w^t = \triangle R_2^t \oplus \triangle s_0^t \oplus \triangle z^t$, donde $\triangle R_2^t = 0$ pues el error no ha llegado a la máquina FSM. Además $\triangle s_0^t$ es conocido por la Nota 2.4 y $\triangle z^t$ también es conocido por el atacante pues se trata de la diferencia que se observa en la salida. Luego, $\triangle w^t$ también es conocido por el atacante (es una constante, no aporta nuevas variables).
Por tanto, la Ecuación (2.17) representa un sistema de 32 ecuaciones en las mismas 32×3 variables que la ecuación (2.16), pero en general con coeficientes distintos. En este modelo, $\triangle s_{15}^t$ es un valor aleatorio de 32 bits. Si el atacante reinicia el dispositivo con los mismos valores iniciales e induce un fallo en la misma localización, obtendrá entonces un nuevo sistema de ecuaciones en las mismas variables: $(s_{15}^t \oplus \triangle s_{15}^t) \boxplus R_1^t = w^t \oplus \triangle w^t$. Si se inducen n fallos siguiendo este mismo proceso en las localizaciones mencionadas, el atacante obtendrá un sistema de $(n+1) \times 32$ ecuaciones en 32×3 variables.
A este sistema queremos aplicar un algoritmo de bases de Gröbner, explicado en A.2, con un orden monomial lexicográfico para extraer ecuaciones lineales que dependan únicamente de los bits de la palabra s_{15}^t. El hecho de que sean lineales se debe a que las ecuaciones inducidas por el cuerpo $\mathbb{F}_2$ no permite exponentes

[4] Como buscamos soluciones de este sistema en $\mathbb{F}_2$ debemos añadir las ecuaciones:

$$(w_i^t)^2 - w_1^t = 0;\ (r_i^{1t})^2 - r_i^{1t} = 0;\ (a_i^{15t})^2 - a_i^{15t} = 0;\ \text{con } i \in \{0, \ldots, 31\}$$

mayores que 1. Al realizar este modelo de ataque estamos siguiendo la estrategia dada por Armknecht-Meier vista en la Sección 2.2.2.
Si el atacante recupera 32 ecuaciones lineales libres en los bits de s_{15}^t ya tendría toda esta etapa. Para recuperar más ecuaciones en otras etapas del LFSR tendrá que inyectar fallos en otros instantes. En particular, podrá aplicarlo para los $t > t_0$ que cumplan que:

- Las diferencias $\triangle s_{15}^t, \triangle s_{15}^{'t}, \ldots$ son distintas de 0 (de no serlo no obtendríamos un sistema de ecuaciones diferente al que proporciona el algoritmo sin fallos).
- Las diferencias $\triangle R_1^t$ y $\triangle R_2^t$ son 0 (si no lo fueran, el atacante obtendría más variables ya que no conoce las diferencias en la memoria de la FSM, debido a su carácter no lineal).

Es decir, debe contar la cantidad de instantes de tiempo, que tras inducir el fallo, tiene un impacto distinto de 0 en s_{15}^t y antes de tener un impacto en el estado de la FSM.

Elección de la localización del fallo

Si se induce un fallo en s_2, s_3 o s_4, el atacante obtendrá 5 sistemas que serán diferentes con alta probabilidad, puesto que s_{15} tendrá un valor no nulo en 5 pulsos de reloj, antes de que el fallo llegue a la FSM, tal y como se muestra a continuación en la Figura 2.8. De hecho, si estudiamos las diferencias obtenidas en s_{15} induciendo el fallo en cada uno de los estados del LFSR, siguiendo este mismo proceso y teniendo en cuenta que, una vez el error inducido llegue al estado s_5, en el siguiente pulso pasará a la FSM y que lo obtenido en la Ecuación (2.15) solo se tiene si se inyecta antes de s_5, obtenemos los resultados mostrados en la Tabla 2.3.
Teniendo en cuenta que el atacante necesita el mayor número de ecuaciones para poder resolver el sistema, con la tabla se justifica la elección de s_2, s_3 y s_4 como estados escogidos donde inducir el fallo.

Palabra donde se induce el fallo	Número de sistemas obtenidos
Palabra s_0	2
Palabra s_1	3
Palabra s_2	5
Palabra s_3	5
Palabra s_4	5
Palabras s_i $i \geq 5$	0

Tabla 2.3: Número de sistemas obtenidos según la localización donde se inyecta el fallo.

Siguiendo con el modelo de ataque que se está estudiando, como se está induciendo el error en s_2, s_3 o s_4 implica la obtención de 5 sistemas, lo que significa

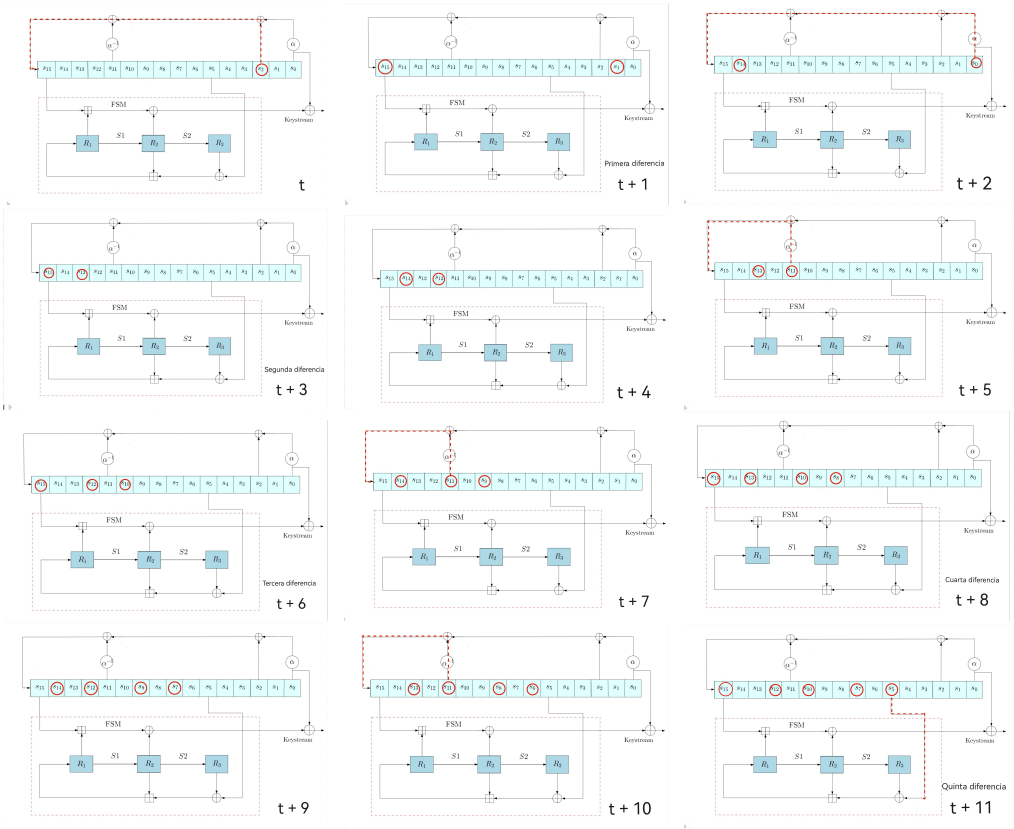

Figura 2.8: Representación de las diferencias obtenidas en s_{15} al inducir un fallo en s_2, s_3 o s_4.

que el atacante puede conseguir a lo sumo 32×5 sistemas de ecuaciones sobre los bits del LFSR. El número de ecuaciones lineales a priori no se conoce pero de la simulación que tenemos, tras más de cien simulaciones del ataque se deduce que de media por ejemplo en un fallo se obtienen unas 5 ecuaciones lineales. Necesita al menos $512 = (16 \times 32)$ ecuaciones linealmente independientes para obtener el estado inicial completo del LFSR (una por cada bit del estado inicial), por lo que el número de ecuaciones no es suficiente. Por lo tanto, el atacante debe inducir no solamente fallos en el mismo instante, sino también fallos a instantes diferentes.
Además debe buscar el equilibrio entre el número de fallos inducidos en la misma localización e instante de tiempo, y el número de instantes diferentes de tiempo $t_0, \ldots, t_k$ que escoge para inyectarlos, con el objetivo de minimizar el número total de fallos, ya que en la vida real, inducir un fallo es un proceso costoso.

Número de fallos	Número medio de ecuaciones lineales
5	18,66
4	17,08
3	13,95
2	9,64
1	4,31

Tabla 2.4: Número medio de ecuaciones lineales obtenidas.

Recuperando la clave secreta

Una vez un estado completo del LFSR ha sido recuperado, el atacante debe usarlo para recuperar la clave secreta del cifrado de flujo. Para hacerlo, debe seguir los pasos siguientes:

1. Recuperar un estado completo del cifrado en el instante t.
2. Usando un algoritmo de bases de Gröbner con el orden monomial apropiado al sistema de ecuaciones dado por $s_{15}^t \boxplus R_1^t = w^t$ y $(s_{15}^t \oplus \triangle s_{15}^t) \boxplus R_1^t = w^t \oplus \triangle w^t$, se obtiene información sobre algunos bits de R_1 y se realiza una suposición para el resto.
3. Usando las ecuaciones: $f^t = (s_{15}^t \boxplus R_1^t) \oplus R_2^t$ y $z^t = s_0^t \boxplus f^t$ el atacante obtiene R_2^t en el instante t.
4. Con las ecuaciones (2.6), (2.7) y (2.8) el atacante obtiene R_1^{t+1}, R_2^{t+1} y R_3^{t+1} de esta forma puede ver si las suposiciones realizadas sobre R_1^t son correctas. En caso de no serlo, realiza otras suposiciones.
5. Una vez que el atacante obtiene el estado completo puede retroceder en el algoritmo y encontrar los valores que tenía el LFSR al principio, que están en función de la clave secreta.

Un pequeño estudio que hemos hecho del Snow V, nos hace indicar que el ataque que explicamos en este capítulo no se puede llevar a cabo en esta versión. Queda como trabajo futuro realizar un estudio exhaustivo. La principal dificultad radica en que en este nuevo diseño las diferencias que se inyectan en el LFSR no se pueden recuperar en la salida. Por lo tanto no es posible conseguir el número de ecuaciones suficientes para recuperar la clave inicial. Cada vez que inyectamos un nuevo fallo, obtenemos nuevas ecuaciones pero se incluyen nuevas variables.

Simulación del ataque

Primero nos crearemos el anillo de polinomios con 32×3 variables en $\mathbb{F}_2$ y con orden monomial lexicográfico donde las variables x_i que representan las variables de la etapa s_{15} del LFSR son las más pequeñas.

```
> sage: R.<z0, ...,z31,y0,...,y31, ...,x0,...,x31>=
PolynomialRing(GF(2),32*3,'z0,...,z31,y0,...,y31,x1,...,x31',
```

```
order='lex')
```

Luego nos creamos el ideal generado por los polinomios:

- Las ecuaciones relacionadas con el cuerpo $\mathbb{F}_2$, es decir:
  $z_i^2 + z_i, y_i^2 + y_i$ y $x_i^2 + x_i$ con $i \in \{0, \ldots, 31\}$
- Las ecuaciones aritméticas de la suma módulo 2^n del algoritmo sin fallos (véase Sección 2.2.1).
- Las ecuaciones aritméticas de la suma módulo 2^n del algoritmo pero induciendo un fallo. La diferencia en la salida:
  $(1,0,1,1,1,0,1,0,1,0,0,1,0,1,1,0,1,0,1,1,1,1,0,0,1,0,0,1,0,1,1,1)$.
  Diferencia en el LFSR:
  $(1,0,1,1,0,1,0,1,1,0,0,1,1,1,0,1,1,0,1,0,1,0,1,0,1,0,0,1,0,1,0,1)$.

De esta forma tenemos un ideal generado por 160 polinomios. Creamos el ideal y calculamos una base de Gröbner de dicho ideal.

```
> sage: I=R.ideal(f0,..,f159)
> sage: B=I.groebner_basis('libsingular:slimgb')
```

Tras el cálculo de Gröbner respecto al orden lexicográfico obtenemos 5 ecuaciones lineales en las variables x_i.

```
(1): x3 + x7 + 1
(2): x5 + x7 + 1
(3): x11 + x12 + 1
(4): x13 + x15 + 1
(5): x20 + x22 + 1
```

Hay que repetir este paso con distintos fallos y distintos instantes hasta recuperar un estado completo del LFSR. Al hacerlo en distintos instantes, no tenemos ecuaciones sobre s_{15}^t, sino sobre s_{15}^{ti} que se corresponderá con otra etapa s_i^t de forma lineal y de este modo recuperamos ecuaciones sobre todos los bits s_i^t del LFSR.

A

Apéndice

A.1. Cuerpos finitos

En el primer anexo vamos a recordar propiedades de cuerpos finitos. No hemos incluido demostraciones en este anexo por falta de espacio pero a todos los resultados se les incluye referencias para el lector.
A partir de ahora p denota un número primo.

Definición A.1 (Cuerpo). *Diremos que $\mathbb{F}$ es cuerpo si $\mathbb{F}$ es un anillo conmutativo y unitario en el que todo elemento distinto de cero tiene inverso respecto del producto.*

Un cuerpo $\mathbb{F}$ es finito si posee un número finito de elementos. No es un resultado trivial, pero todos los cuerpos finitos con el mismo número de elementos son isomorfos entre sí, como evidenciaremos en las siguientes líneas. Por lo tanto, vamos a denotar por $\mathbb{F}_q$ al cuerpo finito con q elementos.
La característica[1] de un cuerpo finito $\mathbb{F}_q$ es un número primo.
El conjunto de los enteros $\mathbb{Z}$ módulo p forma un cuerpo, denotado por $\mathbb{Z}_p$. Esto es cierto si p es primo. Sin embargo, $\mathbb{Z}_4$ no es cuerpo. $\mathbb{Z}_p$ con p primo son los ejemplos más simples de cuerpos finitos. Es más, todo cuerpo finito contiene a algún $\mathbb{Z}_p$ como subcuerpo.

Definición A.3. *Dado un cuerpo $\mathbb{F}$, se define su subcuerpo primo $\mathbb{P}_\mathbb{F}$ como el menor subcuerpo de $\mathbb{F}$.*

[1] Sea $\mathbb{A}$ un anillo unitario, se define la característica de $\mathbb{A}$ como:

$$\text{car}(\mathbb{A}) := \min\{m \in \mathbb{Z}^+ \ / \ \overbrace{1_A + \cdots + 1_A}^{m} = 0_A\}$$

donde 1_A denota el neutro del producto en $\mathbb{A}$ y 0_A el elemento neutro de la suma en A. Si este entero no existe, decimos que la característica de $\mathbb{A}$ es 0. Además, se tiene el siguiente resultado:

Lema A.2. *Sea $(\mathbb{A}, +, \cdot)$ un anillo unitario. Si $\mathbb{A}$ es finito, entonces car$(\mathbb{A}) \neq 0$. Si $\mathbb{A} = \mathbb{F}$ cuerpo, entonces car$(\mathbb{F}) = 0$ o bien car$(\mathbb{F}) = p$.*

Dado un cuerpo $\mathbb{F}$, su cuerpo primo es copia de $\mathbb{Q}$ o de algún $\mathbb{Z}_p$. En particular, si $\mathbb{F} = \mathbb{F}_q$ existe p primo tal que $\mathbb{P}_{\mathbb{F}} \cong \mathbb{Z}_p$. Además, si p es primo, $\mathbb{F}_p \cong \mathbb{Z}_p$ es un cuerpo finito con p elementos.

Proposición A.4. *Consideremos $\mathbb{F}_q$, entonces existe p primo tal que:*

a) $car(\mathbb{F}_q) = p$.
b) $\mathbb{F}_p \subseteq \mathbb{F}_q$.
c) $\mathbb{F}_q$ *es un* $\mathbb{F}_p$*-espacio vectorial de dimensión finita.*
d) $(\alpha + \beta)^p = \alpha^p + \beta^p$, $\forall \alpha, \beta \in \mathbb{F}_q$.
e) $\alpha^q = \alpha$, $\forall \alpha \in \mathbb{F}_q$.
f) $(\alpha + \beta)^q = \alpha^q + \beta^q$, $\forall \alpha, \beta \in \mathbb{F}_q$.

Demostración. Véase en [17][Páginas 16 y 44].

Estudiemos ahora para qué otros valores de q existe un cuerpo finito con q elementos.

A.1.1. Construcción explícita de $\mathbb{F}_q$

Si $f(x) \in \mathbb{F}_q[x]$ es un polinomio irreducible en $\mathbb{F}_q[x]$ de grado r, entonces el anillo cociente

$$\mathbb{F}_q \cong \frac{\mathbb{F}_p[x]}{\langle f(x) \rangle}$$

es un cuerpo con $q = p^r$ elementos, donde $\langle f(x) \rangle$ representa el ideal generado por $f(x)$. Por abuso de notación, podemos representar la clase de x módulo $f(x)$ por x, de esta forma los monomios $\{1, x, \ldots, x^{r-1}\}$ forman una base de $\mathbb{F}_q$ como $\mathbb{F}_p$- espacio vectorial. Por lo tanto, cualquier elemento en $\mathbb{F}_q$ se representa de forma única como un polinomio $g(x) \in \mathbb{F}_p[x]$ de grado menor que r. Esta representación de los elementos de $\mathbb{F}_q$ no es única.

$$\mathbb{F}_q \cong \frac{\mathbb{F}_p[x]}{\langle f(x) \rangle} = \{g(x) \in \mathbb{F}_p[x] / \deg(g) < r\}$$

De forma equivalente, tenemos:

$$\mathbb{F}_q \cong \frac{\mathbb{F}_p}{\langle f(x) \rangle} = \mathbb{F}_p[\alpha] = \{g(\alpha) \ / g(x) \in \mathbb{F}_p[x] \text{ y } \deg(g) < r\}$$

donde α es una raíz de $f(x)$.

Ejemplo A.5.

$$\mathbb{F}_8 = \frac{\mathbb{F}_2[x]}{\langle x^3 + x^2 + x + 1 \rangle} = \{g(x) \in \mathbb{F}_2[x] / \deg(g) < 3\} =$$

$$= \{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$$

Otra representación de $\mathbb{F}_8$ se puede obtener utilizando el polinomio irreducible $f(x) = x^3 + x + 1 \in \mathbb{F}_2[x]$.

¿Es posible siempre encontrar un polinomio irreducible en $\mathbb{F}_p[x]$ de grado r, con $r \in \mathbb{N}$? La respuesta es sí. Es más, el número de polinomios mónicos irreducibles que existen sobre un cuerpo finito $\mathbb{F}_q$ viene dado por la **Fórmula de Gauss**.

Teorema A.6 (Fórmula de Gauss). *El número de polinomios mónicos irreducibles de grado n que existen en $\mathbb{F}_p$ viene dado por:*

$$\frac{1}{n}\sum_{d|n}\mu\left(\frac{n}{d}\right)q^d$$

donde d son todos los divisores positivos de n y $\mu(r)$ es la función de Möbius.[2]

Demostración. Véase en [17][Teorema 3.25].

El siguiente resultado nos indica que no existen otros valores de q con los que podamos construir un cuerpo finito.

Teorema A.7. *Para cualquier primo p y para todo $r \in \mathbb{N}$, existe un cuerpo finito con $q = p^r$ elementos. Además, es único salvo isomorfismos.*

Demostración. Véase en [17][Teorema 2.5].

Por lo tanto, podemos hablar de *el cuerpo finito de q elementos* como ya habíamos adelantado, que denotaremos por $\mathbb{F}_q$.

A.1.2. Más propiedades de cuerpos finitos

Proposición A.8. *$(\mathbb{F}_q \setminus \{0\}, \cdot)$ es un grupo abeliano cíclico, es decir, existe un elemento $\alpha \in \mathbb{F}_q \setminus \{0\}$ tal que $\mathbb{F}_q \setminus \{0\} = \langle\alpha\rangle$. A estos elementos se les conoce como **elementos primitivos** de $\mathbb{F}_q$.*

Demostración. Véase en [17][Teorema 2.8].

Proposición A.9. *El número de elementos primitivos de $\mathbb{F}_q$ es $\phi(q-1)$, siendo ϕ la función de Euler.*

Demostración. Véase en [17][Teorema 1.15].

[2] La función de Möbius se define como:

$$\mu(n) = \begin{cases} 1 & \text{si n es libre de cuadrados y tiene un número par de factores primos distintos} \\ 0 & \text{si n es libre de cuadrados y tiene un número impar de factores primos distintos} \\ (-1)^k & \text{si n es divisible por algún cuadrado} \end{cases}$$

Definición A.10 (Polinomio primitivo). *Un polinomio $P(x) \in \mathbb{F}_q[x]$ de grado n es primitivo si es irreducible y todas las raíces de $P(x)$ generan la extensión $\mathbb{F}_{q^n}$.*

Observación A.11. Si $P(x)$ es primitivo, entonces toda raíz de $P(x)$ $\alpha \in \mathbb{F}_{q^n}[x]$ tiene orden[3] p^n.

Una pregunta que nos surge: si consideramos $\mathbb{F}_q$ con $q = p^r$. **¿Siempre existe un polinomio primitivo $\mathbf{f(x)} \in \mathbb{F}_\mathbf{p}[\mathbf{x}]$ de grado r tal que $\mathbb{F}_\mathbf{q} \cong \frac{\mathbb{F}_\mathbf{p}[\mathbf{x}]}{\langle \mathbf{f(x)} \rangle}$?** La respuesta es sí. Es más, el número de polinomios primitivos de grado r en $\mathbb{F}_p[x]$ es conocido y se describe en el siguiente Teorema.

Teorema A.12. *El número de polinomios primitivos en $\mathbb{F}_p[x]$ de grado m es:*

$$\frac{\phi(p^m - 1)}{m}$$

Demostración. Véase en [17][Teorema 3.5].

Ejemplo A.13. El polinomio $P(x) = x^4 + x^3 + x^2 + x + 1 \in \mathbb{F}_2[x]$ no es primitivo pues si α es raíz de $P(x)$ entonces:

$$\alpha^5 = \alpha^4 \cdot \alpha = (\alpha^3 + \alpha^2 + \alpha + 1) \cdot \alpha = \alpha^4 + \alpha^3 + \alpha^2 + \alpha =$$

$$= (\alpha^3 + \alpha^2 + \alpha + 1) + \alpha^3 + \alpha^2 + \alpha = 1$$

Por lo tanto α no genera a $\mathbb{F}_{2^4} \setminus \{0\}$. Sin embargo, el polinomio $R(x) = x^4 + x + 1$ sí es primitivo pues $\mathbb{F}_{2^4} \setminus \{0\} = \langle \beta \rangle$, con β raíz de $R(x)$ ya que $\beta^{15} = 1$ y no existe una potencia menor que lo verifique[4].

Observación A.14. Todo polinomio primitivo es irreducible, pero no todo polinomio irreducible es primitivo como se comprueba en el ejemplo anterior.

Observación A.15. Tenemos que si $q = p^r$ entonces:

$$\mathbb{F}_q^* = \mathbb{F}_p(\alpha) = \langle \beta \rangle$$

donde α es raíz de un polinomio irreducible de grado r. Sin embargo no siempre α es un elemento primitivo.

[3] Sea $(\mathbb{G}, \cdot)$ un grupo, se define el orden de un elemento $g \in G$ como el entero positivo m más pequeño tal que $g^m = 1_\mathbb{G}$, donde $1_\mathbb{G}$ denota el neutro de $\mathbb{G}$.

[4] $\beta^5 = \beta^2 + \beta$;$\beta^6 = \beta^3 + \beta^2$; $\beta^7 = \beta^3 + \beta + 1$; $\beta^8 = \beta^2 + 1$; $\beta^9 = \beta^3 + \beta$; $\beta^{10} = \beta^2 + \beta + 1$; $\beta^{11} = \beta^3 + \beta^2 + \beta$; $\beta^{12} = \beta^3 + \beta^2 + \beta + 1$; $\beta^{13} = \beta^3 + \beta^2 + 1$; $\beta^{14} = \beta^3 + 1$; $\beta^{15} = 1$

Base dual y traza

Definición A.16 (Función traza). *La **traza** es una aplicación $tr : \mathbb{F}_q \longrightarrow \mathbb{F}_q$ con $q = p^r$, definida por:*

$$tr(u) := \sum_{i=0}^{r-1} u^{p^i} = u + u^p + \cdots + u^{p^{r-1}}.$$

Observación A.17. Es fácil comprobar que la traza es una aplicación lineal. Además, $tr(u) = u \ \forall u \in \mathbb{F}_p$ y $tr(u^p) = u \ \forall u \in \mathbb{F}_q$.

Definición A.18 (Base dual). *Sea $A = \{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$ una base de $\mathbb{F}_{q^n}$. Diremos que $B = \{\beta_0, \beta_1, \ldots, \beta_{n-1}\}$ es una **base dual** de A si:*

$$tr(\alpha_i \cdot \beta_i) = \delta_{ij}, \quad \text{donde } \delta_{ij} = \begin{cases} 1 & \text{si } i = j, \\ 0 & \text{si } i \neq j \end{cases} \quad 0 \leq i, j \leq n$$

siendo δ_{ij} la delta de Kronecker.

Lema A.19. *Para cada base de $\mathbb{F}_{q^n}$ existe una única base dual.*

Demostración. Véase en [17][página 54].

Como hemos visto, si $q = p^r$, $\mathbb{F}_q = \frac{\mathbb{F}_p[x]}{\langle f(x) \rangle} = \mathbb{F}_p[\alpha]$ con $f(x) \in \mathbb{F}_p[x]$ polinomio irreducible de grado r y α raíz de $f(x)$. De esta forma, $A = \{1, \alpha, \ldots, \alpha^{r-1}\}$ es una base de $\mathbb{F}_q$ como $\mathbb{F}_p$-espacio vectorial. A esta base se le conoce como base polinómica de $\mathbb{F}_q$. El siguiente resultado nos permite definir de forma explícita una base dual de A en $\mathbb{F}_q$.

Lema A.20. *Sea $A = \{1, \alpha, \ldots, \alpha^{r-1}\}$ una base polinómica de $\mathbb{F}_q$ con $q = p^r$, $f(x) \in \mathbb{F}_p[x]$ polinomio irreducible de grado r y α raíz de $f(x)$. Consideremos $\gamma = f'(\alpha)$ y $\frac{f(x)}{(x-\alpha)} = \beta_0 + \beta_1 x + \cdots + \beta_{r-1} x^{r-1} \in \mathbb{F}_q[x]$.*
Entonces, $B = \{\beta_0 \gamma^{-1}, \beta_1 \gamma^{-1}, \ldots, \beta_{r-1} \gamma^{r-1}\}$ es una base dual de A en $\mathbb{F}_q$.

Demostración. Véase en [13][página 59].

A.2. Bases de Gröbner

Las bases de Gröbner fueron introducidas por Bruno Buchberger en su tesis doctoral en 1965 [3], realizada bajo la dirección de Wolfgang Gröbner. En su tesis, Buchberger presenta un algoritmo que permite calcular una base de Gröbner de un ideal. Este algoritmo está actualmente implementado en muchos sistemas de álgebra computacional (Maple, Mathematica, Macaulay, Cocoa, Singular, Sage,...). Buchberger decía los siguiente respecto a las bases de Gröbner:

"Nuestro algoritmo permite calcular una nueva base de un ideal que permite resolver el problema de decidir si un polinomio pertenece a un ideal de forma fácil."

Sea $\mathbb{F}$ un cuerpo, en esta sección vamos a trabajar con el anillo de polinomios en varias variables $\mathbb{F}[x_1, \ldots, x_n]$ y coeficientes en $\mathbb{F}$. Estudiaremos los conceptos necesarios para definir una base de Gröbner y una de sus aplicaciones que nos permitirá resolver el sistema planteado en el ataque estudiado en el Capítulo 2.

A.2.1. Orden monomial

Definición A.21. *Un* ***monomio*** *en* $\mathbb{F}[x_1, \ldots, x_n]$ *es un elemento de la forma* $x^\alpha := x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, *siendo* $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}_{\geq 0}^n$. *Donde* $|\alpha| := \alpha_1 + \cdots + \alpha_n$ *es el grado del monomio.*

A continuación se define un orden total en $\mathbb{F}[x_1, \ldots, x_n]$, concepto que generaliza el orden propio de los polinomios en una variable, dado por su grado.

Definición A.22 (Orden monomial). *Un* ***orden monomial*** *en* $\mathbb{F}[x_1, \ldots, x_n]$ *es una relación binaria* $\geq$ *en el conjunto de los monomios* $M = \{x^\alpha / \alpha \in \mathbb{Z}_{\geq 0}^n\}$ *que verifica:*

- $\geq$ *es un orden total en* M.
- *Si* $x^\alpha \geq x^\beta$ *entonces* $x^\alpha x^\gamma \geq x^\beta x^\gamma$ *con* $x^\alpha, x^\beta, x^\gamma \in M$.
- $x^\alpha \geq x^0$ *para todo* $x^\alpha \in M$, *con* $0 = (0, \ldots, 0)$.

Equivalentemente, se puede estudiar como relación binaria $\geq$ *en* $\mathbb{Z}_{\geq 0}^n$ *que verifica:*

- $\geq$ *es orden total en* $\mathbb{Z}_{\geq 0}^n$.
- *Si* $\alpha \geq \beta$ *entonces* $\alpha + \gamma \geq \beta + \gamma$ *con* $\alpha, \beta, \gamma \in \mathbb{Z}_{\geq 0}^n$.
- $\alpha \geq 0$ *para todo* $\alpha \in \mathbb{Z}_{\geq 0}^n$.

Existen diversos tipos de órdenes monomiales, en esta sección trabajaremos con uno en particular, el orden lexicográfico.

Definición A.23 (Orden lexicográfico). *Sean* $\alpha, \beta \in \mathbb{Z}_{\geq 0}^n$ *con* $\alpha = (\alpha_1, \ldots, \alpha_n)$ *y* $\beta = (\beta_1, \ldots, \beta_n)$. *Se dice que* $\alpha >_{lex} \beta$ *si el primer valor de la tupla* $\alpha - \beta$ *es positivo. Luego, si* $\alpha >_{lex} \beta$, *entonces se dice que* $x^\alpha >_{lex} x^\beta$.

Ejemplo A.24. 1. $\alpha = (1, 2, 0) >_{lex} (0, 3, 4) = \beta$ ya que $\alpha - \beta = (1, -1, -4)$.
2. Las variables $x_1, \ldots, x_n$ están ordenadas en la forma usual por el orden lexicográfico:

$$(1, 0, \ldots, 0) >_{lex} (0, 1, \ldots, 0) >_{lex} \cdots >_{lex} (0, 0, \ldots, 1)$$

Luego, $x_1 >_{lex} x_2 >_{lex} \cdots >_{lex} x_n$.

Proposición A.25. *El orden lexicográfico es un orden monomial.*

Demostración. La primera y tercera condición se verifican por definición de orden lexicográfico y porque $\mathbb{Z}^n_{\geq 0}$ está totalmente ordenado. Veamos que cumple la segunda como relación binaria en $\mathbb{Z}^n_{\geq 0}$. Sean $\alpha, \beta, \gamma \in \mathbb{Z}^n_{\geq 0}$. Es claro que, $(\alpha + \gamma) - (\beta + \gamma) = \alpha - \beta$. Luego, si $\alpha >_{lex} \beta$, entonces $\alpha + \gamma >_{lex} \beta + \gamma$. Por tanto, efectivamente se trata de un orden monomial. □

Definición A.26. *Sea $f = \sum_\alpha a_\alpha x^\alpha$ un polinomio no nulo en $\mathbb{F}[x_1, \ldots, x_n]$ y sea $\geq$ un orden monomial. Se define entonces:*

- ***Multigrado de f**: $mdeg := máx\{\alpha \in \mathbb{Z}^n_{\geq 0} / a_\alpha \neq 0\}$ (tomando el máximo respecto al orden monomial).*
- ***Coeficiente principal de f**: $LC(f) = a_{mdeg(f)} \in \mathbb{F}$.*
- ***Monomio principal de f**: $LM(f) = x^{mdeg(f)}$.*
- ***Término principal de f**: $LT(f) = LC(f) \cdot LM(f) = a_{mdeg(f)} \cdot x^{mdeg(f)}$.*

A.2.2. Algoritmo de la división en $\mathbb{F}[x_1, \ldots, x_n]$

Vamos a generalizar el algoritmo de división de polinomios en una variable a $\mathbb{F}[x_1, \ldots, x_n]$. El objetivo es dividir $f \in \mathbb{F}[x_1, \ldots, x_n]$ por $f_1, \ldots, f_s \in \mathbb{F}[x_1, \ldots, x_n]$. Es decir, expresar f como:

$$f = q_1 f_1 + \cdots + q_s f_s + r$$

donde $q_i, r \in \mathbb{F}[x_1, \ldots, x_n]$ para $i \in \{1, \ldots, s\}$.

Teorema A.27 (Algoritmo de la división). *Fijado un orden monomial $\geq$ en $\mathbb{Z}^n_{\geq 0}$ y siendo $F = (f_1, \ldots, f_s)$ una s-tupla ordenada de polinomios en $\mathbb{F}[x_1, \ldots, x_n]$. Entonces, todo $f \in \mathbb{F}[x_1, \ldots, x_n]$ se puede expresar como:*

$$f = q_1 f_1 + \cdots + q_s f_s + r$$

*con $q_i, r \in \mathbb{F}[x_1, \ldots, x_n]$ para $i \in \{1, \ldots, s\}$. Además, se verifica que $r = 0$ o bien r es combinación lineal de monomios de $\mathbb{F}[x_1, \ldots, x_n]$, donde ninguno de los monomios es divisible por ningún término principal, $LT(f_i)$, con $i \in \{1, \ldots, s\}$. Llamaremos r al **resto de la división entre f y F**. Además, si $q_i, f_i \neq 0$, entonces $mdeg(f) \geq mdeg(q_i f_i)$.*

Algorithm 1 Algoritmo de la división en $\mathbb{K}[x_1, \ldots, x_n]$

Entrada: $f, f_1, \ldots, f_s \in \mathbb{K}[x_1, \ldots, x_n]$.
Salida: $q_1, \ldots, q_s, r \in \mathbb{K}[x_1, \ldots, x_n]$ tal que $f = q_1 f_1 + \ldots + q_s f_s + r$ donde $r = 0$ o ningún monomio de r es divisible por $\mathrm{LT}(f_i)$ con $i \in \{1, \ldots, s\}$.
Inicialización $f^{(0)} := f$, $r^{(0)} := 0$, $q_1 = \ldots = q_s = 0$ $i := 0$
while $f^{(i)} = 0$ **do**
  if *existe* $j = \text{mín}\left\{k \mid \mathrm{LT}(f_k) \text{ divide } \mathrm{LT}(f^{(i)})\right\}$ **then**
    $f^{(i+1)} = f^i - \frac{\mathrm{LT}(f^{(i)})}{\mathrm{LT}(f_j)} f_j$ $r^{(i+1)} = r^{(i)}$ $q_j^{(i+1)} = q_j^{(i)} + \frac{\mathrm{LT}(f^{(i)})}{\mathrm{LT}(f_j)}$ $q_k^{(i+1)} = q_k^{(i)}$, $\forall k \neq j$
  else
    $f^{(i+1)} = f^{(i)} - \mathrm{LT}(f^{(i)})$ $r^{(i+1)} = r^{(i)} + \mathrm{LT}(f^{(i)})$ $q_k^{(i+1)} = q_k^{(i)}$
  end
  $i = i + 1$
end
return $q_1^{(i)}, \ldots, q_s^{(i)}, r^{(i)}$

Demostración. Lo que tenemos que demostrar es que el Algoritmo termina en un número finito de pasos. El Algoritmo empieza planteando dos casos: cuando existe j tal que $LT(f^{(i)})$ es divisible por $LT(f_j)$ y cuando no. En ambos casos se define un nuevo $f^{(i+1)}$ que verifica que $LT(f^{(i)}) > LT(f^{(i+1)})$. Además, estamos en un anillo noetheriano $\mathbb{F}[x_1, \ldots, x_n]$ y estamos considerando un orden monomial (un buen orden en $\mathbb{F}[x_1, \ldots, x_n]$). Luego la cadena:

$$LT(f^{(i)}) > LT(f^{(i+1)}) > \cdots$$

tiene mínimo (se obtiene como consecuencia de [6][Teorema 7]) lo que nos asegura que el Algoritmo termina. □

Veamos un ejemplo usando el algoritmo mostrado.

Ejemplo A.28. Sea $f = xy^2 + 1$, lo dividiremos por $f_1 = xy + 1$ y $f_2 = y + 1$, usando el orden lexicográfico y haciendo uso del Algoritmo 1. Tenemos que $LT(f) = xy^2$, $LT(f_1) = xy$ y $LT(f_2) = y$. Luego como $LT(f_1)|LT(f)$, entonces:

$$f^{(1)} = -y + 1,\ r^{(1)} = 0,\ q_1^{(1)} = y,\ q_2^{(1)} = 0$$

Como $LT(f_2)|LT(f^{(1)})$ nos queda:

$$f^{(2)} = 2,\ r^{(2)} = 0,\ q_1^{(2)} = y,\ q_2^{(2)} = -1$$

Ahora, como los términos principales de f_1 y f_2 no dividen a $f^{(2)}$ se obtiene:

$$f^{(3)} = 0,\ r^{(3)} = 2,\ q_1^{(3)} = y,\ q_2^{(3)} = -1$$

Finalizando así el algoritmo. Por tanto, podemos escribir f como:

$$f = xy^2 + 1 = y \cdot (xy + 1) - 1 \cdot (y + 1) + 2 = y \cdot f_1 - f_2 + 2$$

A.2.3. Bases de Gröbner

Definición A.29. *Un ideal J de $\mathbb{F}[x_1, \ldots, x_n]$ se dice* ***ideal monomial*** *si existe un subconjunto $A \subseteq \mathbb{Z}^n_{\geq 0}$, que puede ser infinito, tal que $J = \langle x^\alpha / \alpha \in A \rangle$.*

Lema A.30. *Sea $J = \langle x^\alpha / \alpha \in A \rangle$ un ideal monomial. Entonces el monomio $x^\beta \in J$ si, y solo si, x^β es divisible por x^α para algún $\alpha \in A$.*

Demostración. Sea $x^\beta \in J$. Entonces, $x^\beta = \sum_{i=1}^{s} h_i x^{\alpha(i)}$, donde $\alpha(i) \in A$ y $h_i \in \mathbb{F}[x_1, \ldots, x_n]$ para $1 \leq i \leq s$. Si expandimos cada h_i como una combinación lineal de sus monomios, obtenemos:

$$x^\beta = x^{\alpha(1)}(c_{1,1}x^{\beta_{1,1}} + \cdots + c_{1,s}x^{\beta_{1,s}}) + \cdots + x^{\alpha(s)}(c_{s,1}x^{\beta_{s,1}} + \cdots + c_{s,s}x^{\beta_{s,s}})$$

Luego, cada término de la expresión de la derecha es divisible por un cierto $x^{\alpha(i)}$. En consecuencia, x^β es divisible por x^α para un cierto $\alpha \in A$.
Recíprocamente, si x^β es un múltiplo de x^α para algún $\alpha \in A$, entonces $x^\beta \in J$ por definición de ideal. □

Teorema A.31 (Teorema de la Base de Hilbert). *En $\mathbb{F}[x_1, \ldots, x_n]$ todo I ideal tiene un conjunto de generadores finito, es decir, $I = \langle g_1, \ldots, g_t \rangle$ para ciertos $g_1, \ldots, g_t \in I$.*

Demostración. En la demostración de este Teorema se requiere el uso del algoritmo de la división 1. Veáse en [6][Teorema 4].

Definición A.32. *Sea J un ideal no nulo de $\mathbb{F}[x_1, \ldots, x_n]$ y fijemos un orden monomial en el anillo de polinomios. Denotamos por $\langle LM(J) \rangle$ al ideal de $\mathbb{F}[x_1, \ldots, x_n]$ definido como:*

$$\langle LM(J) \rangle = \langle LM(f) / f \in J - \{0\} \rangle$$

Proposición A.33. *Sea J un ideal no nulo de $\mathbb{F}[x_1, \ldots, x_n]$. Entonces:*

- *$\langle LM(J) \rangle$ es un ideal monomial.*
- *Existen $g_1, g_2, \ldots, g_t \in J$ tales que $\langle LM(J) \rangle = \langle LM(g_1), \ldots, LM(g_s) \rangle$.*

Demostración. La primera parte es directa por la definición de $\langle LM(J) \rangle$. La segunda condición se cumple como consecuencia del Teorema de la base de Hilbert. □

En $\mathbb{F}[x_1, \ldots, x_n]$ fijamos un orden monomial $\geq$.

Definición A.34 (Base de Gröbner). *Un subconjunto finito $G = \{g_1, \ldots, g_t\}$ de un ideal $J \subseteq \mathbb{F}[x_1, \ldots, x_n]$, con $J \neq \{0\}$ se denomina* ***base de Gröbner*** *de J respecto del orden si*

$$\langle LM(J) \rangle = \langle LM(g_1), \ldots, LM(g_t) \rangle$$

Por convenio, $\langle \emptyset \rangle = \{0\}$ y $\emptyset$ es la base de Gröbner del ideal $\{0\}$.

Proposición A.35 (Existencia). *Fijado un orden monomial, todo ideal J de $\mathbb{F}[x_1, \ldots, x_n]$ admite una base de Gröbner. Es más, toda base de Gröbner de J es un sistema generador del ideal $\langle LM(J) \rangle$.*

Demostración. Si $J = \{0\}$, por convenio su base de Gröbner es $\emptyset$. Supongamos que $J \neq \{0\}$. Por lo visto en la Proposición A.33 sabemos que existen $g_1, \ldots, g_t \in J$ tal que $\langle LM(J) \rangle = \langle LM(g_1), \ldots, LM(g_t) \rangle$. Así, tenemos que $G = \{g_1, \ldots, g_t\}$ es una base de Gröbner de J. Falta probar que $J = \langle g_1, \ldots, g_t \rangle$.
Es claro que, $\langle g_1, \ldots, g_t \rangle \subseteq J$ ya que $\{g_1, \ldots, g_t\} \subseteq J$.
Veamos que se verifica la otra inclusión. Sea $f \in J$. Como se ha fijado un orden monomial podemos realizar la división de f entre $F = (g_1, \ldots, g_t)$. Luego,

$$f = q_1 g_1 + \cdots + q_t g_t + r$$

y ninguno de los términos de r es divisible por $LT(g_i)$, con $1 \leq i \leq s$. Vamos a probar que $r = 0$. Por la expresión anterior, obtenemos que $r = f - q_1 g_1 - \cdots - q_t g_t \in J$. Supongamos por reducción al absurdo que $r \neq 0$. Entonces, se tiene que $LT(r) \in \langle LT(J) \rangle = \langle LT(g_1), \ldots, LT(g_t) \rangle$. Como $\langle LT(J) \rangle$ es monomial, por lo visto en el Lema A.30 existe $g_i \in G$ tal que $LT(r)$ es divisible por $LT(g_i)$, que no es posible por cómo se define r en el Algoritmo de la división 1. Por tanto, $f = q_1 g_1 + \cdots + q_t g_t$ y, en consecuencia, $f \in \langle g_1, \ldots, g_t \rangle$. □

En la Tesis doctoral de Buchberger, se presenta un algoritmo que permite calcular una base de Gröbner de un ideal. Véase los detalles en [6].

Definición A.36 (S-polinomio). *Sean $f, g \in \mathbb{F}[x_1, \ldots, x_n] - \{0\}$. Se define el S-polinomio de f y g, denotado por $S(f, g)$ como:*

$$S(f, g) := \frac{x^\gamma}{LT(f)} f - \frac{x^\gamma}{LT(g)} g$$

donde $x^\gamma = mcm(LM(f), LM(g))$.

Ejemplo A.37. Sean $f = x + xy$, $g = z + zy$ polinomios en $\mathbb{F}[x, y, z]$. El S-polinomio de f y g es entonces:

$$S(f, g) = \frac{xyz}{xy} \cdot (x + xy) - \frac{xyz}{yz} \cdot (z + zy) = z \cdot (x + yx) - x \cdot (z + zy)$$

Denotaremos por $R(S(f, g), \tilde{\mathrm{G}})$ al resto de la división del polinomio $S(f, g)$ entre $\tilde{\mathrm{G}}$.

Teorema A.38 (Algoritmo de Buchberger). *Sea $J = \langle f_1, \ldots, f_s \rangle$ ideal de $\mathbb{F}[x_1, \ldots, x_n]$. El algoritmo de Buchberger construye una base de Gröbner de J en un número finito de pasos.*

Algorithm 2 Algoritmo de Buchberger

Entrada: $F = (f_1, \ldots, f_s)$ sistema generador de J.
Salida: $G = (g_1, \ldots, g_t)$ base de Gröbner de J, con $F \subseteq G$.
Inicialización $G := F$;
while $\tilde{G}$:=G. *Para cada par* $\{p, q\}$ *con* $p \neq q$ *en* $\tilde{G}$ **do**
  $r = R(S(f, g), \tilde{\mathrm{G}})$
  if $r \neq 0$ **then**
    $G := G \cup \{r\}$
  end
end
return G

Demostración. Veáse en [6][Teorema 2, página 91].

Ejemplo A.39. Consideremos $F = \{f_1 = x^2y + x + 1, f_2 = xy + y^2 + 1\}$. Vamos a utilizar el orden lexicográfico con $x >_{lex} y$. Inicialiamos $\tilde{\mathrm{G}} = \{f_1, f_2\}$.
Calculamos $R(S(f_1, f_2), \tilde{\mathrm{G}}) = y^3 + y + 1 = f_3$. Ponemos $\tilde{\mathrm{G}} = G \cup \{f_3\}$.
Calculamos:

- $R(S(f_1, f_2), G) = 0$ pues $f_3 \in G$
- $R(\underbrace{S(f_1, f_3)}_{-x^2y+xy^2-x^2+y^2}, G) = -x^2 + x + y^2 + 2 = f_4$
- $R(\underbrace{S(f_2, f_3)}_{-xy-x+y^4+y^2}, G) = -x + y^2 - y + 1 = f_5$

Luego, $\tilde{\mathrm{G}} = \{f_1, f_2, f_3, f_4, f_5\}$.

A.2.4. Ideales y soluciones de un sistema de polinomios

¿Cómo se relacionan bases de Gröbner y el conjunto de soluciones de un sistema polinomial? Esta es la idea clave de porqué elegimos utilizar bases de Gröbner en este trabajo.
Se define *variedad afín* como el conjunto de soluciones de un sistema polinomial. Es decir,

$$V(f_1, \ldots, f_s) = \{(a_1, \ldots a_n) \in \mathbb{F}^n / f_i(a_1, \ldots, a_n) = 0 \text{ con } i \in \{1, \ldots, s\}\}$$

Consideremos ahora el ideal $F = \langle f_1, \ldots, f_s \rangle$ en $\mathbb{F}[x_1, \ldots, x_n]$. Podemos definir el concepto de variedad afín asociada a un ideal:

$$V(F) = \{(a_1, \ldots, a_n) \in \mathbb{F}^n / f(a_1, \ldots, a_n) = 0 \ \forall f \in I\}$$

Un hecho crucial es que las variedades solo dependen del ideal, no de la base elegida.

Proposición A.40. *Sean $F = \{f_1, \ldots, f_s\}$, $G = \{g_1, \ldots, g_s\}$ bases del mismo ideal I. Entonces, $V(I) = V(f_1, \ldots, f_s) = V(g_1, \ldots, g_s)$.*

Demostración. Veamos primero que $V(f_1, \ldots, f_s) = V(g_1, \ldots, g_s)$.
Para todo elemento $(a_1, \ldots, a_n) \in V(f_1, \ldots, f_s)$ se tiene que $f_i(a_1, \ldots, a_n) = 0$ con $i \in \{1, \ldots, s\}$. Además, como $g_j \in I$ y F es base de I, se tiene que: $g_j = h_1 f_1 + \cdots h_s f_s$ para ciertos $h_i \in \mathbb{F}[x_1, \ldots, x_n]$. De donde se deduce que $(a_1, \ldots, a_n) \in V(g_1, \ldots, g_s)$. El recíproco se obtiene de forma similar. Veamos ahora que $V(I) = V(f_1, \ldots, f_s)$. Es claro que $V(f_1, \ldots, f_s) \subseteq V(I)$ pues $f_i \in I$ $\forall i \in \{1, \ldots, s\}$. De forma similar al caso anterior se obtiene la otra inclusión. □

Esta proposición nos relaciona variedades de polinomios con variedades de ideales y viceversa.

Lema A.41. *El resto del algoritmo de la división entre f y $F = (f_1, \ldots, f_s)$ es único.*

Demostración. Denotamos $g = q_1 f_1 + \cdots + q_s f_s$. Supongamos por reducción al absurdo que existe r' tal que $r \neq r'$, luego existe g' de forma que $f = g + r = g' + r'$. Por tanto, $LT(r - r') \in \langle LT(J) \rangle = \langle LT(f_1), \ldots, LT(f_t) \rangle$ y por el Lema A.30, $LT(r - r')$ es divisible por $LT(f_i)$ para cierto f_i con $i \in \{1, \ldots, t\}$. Sin embargo, esto no es posible por lo visto en el Teorema A.27, luego $r = r'$. □

Proposición A.42. *Sea $J = (f_1, \ldots, f_t)$ un ideal de $\mathbb{F}[x_1, \ldots, x_n]$, $G = \{g_1, \ldots, g_t\}$ una base de Gröbner de J y $f \in \mathbb{F}[x_1, \ldots, x_n]$. Entonces:*

$$f \in J \iff \textit{el resto de la división de } f \textit{ entre } G \textit{ es cero.}$$

Es decir: $f = f_1 g_1 + \cdots + f_t g_t$.

Demostración. Sea $f \in J$. Si efectuamos la división entre f y G obtenemos que $f = \underbrace{f_1 g_1 + \cdots f_t g_t}_{g} + r$ donde el Algoritmo de la división 1 nos indica que $r = 0$ o bien r es combinación lineal de monomios de $\mathbb{F}[x_1, \ldots, x_n]$ pero ninguno de ellos es divisible por $LT(f_i)$ con $1 \leq i \leq t$. Luego, $f = g + r$. Por tanto, como $f \in J$, f se puede escribir como $f = f + 0$ y por el Lema A.2.4 tenemos que $r = 0$.
Recíprocamente, si el resto es 0, tenemos que $f = f_1 g_1 + \cdots f_t g_t$, por lo que $f \in J$. □

Corolario A.43. *Sea J un ideal de $\mathbb{F}[x_1, \ldots, x_n]$, $G = \{g_1, \ldots, g_t\}$ una base de Gröbner de J. Entonces,*

$$V(J) = V(G)$$

Por lo tanto, encontrar las soluciones de un sistema de polinomios $F = \{f_1, \ldots, f_s\}$ es equivalente a encontrar las soluciones del sistema asociado a una base de Gröbner del ideal $I = \langle F \rangle$.

A.2.5. Eliminación de variables

Definición A.44. *Sea* $J = \langle f_1, \ldots, f_s \rangle \subseteq \mathbb{F}[x_1, \ldots, x_n]$. *Se define el* ***l-ésimo ideal de eliminación*** *como el ideal de* $\mathbb{F}[x_{l+1}, \ldots, x_n]$*:*

$$J_l = J \cap \mathbb{F}[x_{l+1}, \ldots, x_n]$$

Es decir, en general, dado un sistema de s ecuaciones $f_1 = 0, \ldots, f_s = 0$ donde $f_i \in \mathbb{F}[x_1, \ldots, x_n]$ para $1 \leq i \leq s$ podemos considerar el ideal $I = \langle f_1, \ldots, f_s \rangle$ y las intersecciones:

- $I \cap \mathbb{F}[x_2, \ldots, x_n]$ que elimina la variable x_1,
- $I \cap \mathbb{F}[x_3, \ldots, x_n]$ que elimina las variables x_1, x_2

  $\vdots$
- $I \cap \mathbb{F}[x_n]$ que elimina las variables $x_1, \ldots, x_{n-1}$

El siguiente resultado es fundamental en el Capítulo 2. Este resultado nos permite definir una base de Gröbner respecto del orden lexicográfico para el ideal $J_l = J \cap \mathbb{F}[x_{l+1}, \ldots, x_n]$, ideal en el que hemos eliminado las variables $x_1, \ldots, x_l$.

Teorema A.45 (Eliminación). *Sea* $J \subseteq \mathbb{F}[x_1, \ldots, x_n]$ *un ideal y* G *una base de Gröbner de* J *respecto de* $>_{lex}$ *con* $x_1 >_{lex} \cdots >_{lex} x_n$*. Entonces, para todo* $0 \leq l \leq n$, $G_l = G \cap \mathbb{F}[x_{l+1}, \ldots, x_n]$ *es una base de Gröbner de* J_l*.*

Demostración. Sea $0 \leq l \leq n$. Como $G \subseteq J$, es claro que $G_l \subseteq J_l$. Luego, basta ver que $\langle LT(J_l) \rangle = \langle LT(G_l) \rangle$.
Veamos primero que $\langle LT(J_l) \rangle \subseteq \langle LT(G_l) \rangle$. Sea $f \in J$ y $LT(f)$, su término principal. Basta probar que $LT(f)$ es divisible por $LT(g)$ para algún $g \in G_l$. En efecto, como $f \in J$ y G es base de Gröbner de J, existe $g \in G$ tal que $LT(f)$ es divisible por $LT(g)$. Por otra parte, $f \in J_l$ y como $LT(g) | LT(f)$, se tiene que $LT(g) \in \mathbb{F}[x_{l+1}, \ldots, x_n]$. Además, por hipótesis todo monomio en el que aparezca alguna de las variables $x_1, \ldots, x_l$ es mayor que los monomios de $\mathbb{F}[x_{l+1}, \ldots, x_n]$. Por tanto, $g \in \mathbb{F}[x_{l+1}, \ldots, x_n]$ ya que su término principal pertenece a este anillo y el resto de términos son menores. Entonces, se tiene que $g \in G_l = G \cap \mathbb{F}[x_{l+1}, \ldots, x_n]$, luego $LT(f)$ es divisible por $LT(g)$ para cierto $g \in G_l$ obteniendo así esta inclusión.
La otra inclusión es evidente, puesto que $G_l \subseteq J_l$, luego $LT(G_l) \subseteq LT(J_l)$ y, en consecuencia, $\langle LT(G_l) \rangle \subseteq \langle LT(J_l) \rangle$. □

La aplicación de este sistema es obtener un sistema con menos variables del que podemos obtener una solución particular y con esta solución resolver el sistema inicial.

Ejemplo A.46. El siguiente ejemplo lo resolveremos haciendo uso del software algebraico sage.

```
> sage: R = PolynomialRing(QQ, 2, 'xy', order = 'lex')
> sage: x,y = R.gens()
> sage: I = (y^2-x^2-3, y-2*x)*R
> sage: B = I.groebner_basis(); B
  [y-2*x, x^2-1]
```

Por lo tanto $x = \pm 1$ y $y = 2x$ son las soluciones del sistema.

Ejemplo A.47. Sea $I = \langle \underbrace{x^2+y^2+z^2-1}_{f_1}, \underbrace{x^2+z^2+y}_{f_2}, \underbrace{x-z}_{f_3} \rangle$. Calculamos una base de Gröbner de este ideal: $G = \{g_1, g_2, g_3\}$ con

1. $g_1 = x - z$
2. $g_2 = y - 2z^2$
3. $g_3 = z^4 + \frac{1}{2}z^2 - \frac{1}{4}$

Observamos que g_3 solo utiliza la variable z. Resolviendo la ecuación obtenemos las siguientes soluciones del sistema:

$$z = \pm\frac{1}{2}\sqrt{\pm\sqrt{5}-1}$$

Luego, sustituyendo en $g_2 = 0$ y $g_1 = 0$, obtenemos las soluciones para x y para y. Por tanto, las posibles soluciones del sistema inicial son:

$$\left\{\left(\pm\frac{1}{2}\sqrt{\pm\sqrt{5}-1}, \frac{\pm\sqrt{5}}{2} - \frac{1}{2}, \pm\frac{1}{2}\sqrt{\pm\sqrt{5}-1}\right)\right\}$$

Bibliografía

[1] Alexandrov Nikolov, P. (2019). Analysis and design of a stream cipher. Universidad de Alicante.

[2] Armknecht, F., & Meier, W. (2005). Fault Attacks on Combiners with Memory. *Lecture Notes in Computer Science* (pp. 36-50).

[3] Buchberger B. (1965). An algorithm for Finding a Basis for the Residue Class Ring of a Zerodimensional Ideal. *Ph. D. Thesis*. University of Innsbruck, Math. Inst.

[4] Canteaut, A. (2011). *Linear Feedback Shift Register. In: van Tilborg, H.C.A., Jajodia, S. (eds) Encyclopedia of Cryptography and Security.* Springer, Boston, MA.

[5] Courtois, Nicolas T. and Debraize, Blandine. (2008). Algebraic Description and Simultaneous Linear Approximations of Addition in Snow 2.0.*Lecture Notes in Computer Science.*

[6] Cox, D. G., Little, J. B., & O'Shea, D. F. (1993). *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra.*

[7] J. Daemen, V. Rijmen. (2002). *The design of Rijndael.* Springer Verlag Series on Information Security and Cryptography, Springer Verlag.

[8] Debraize, B., & Corbella, I. M. (2009). Fault Analysis of the Stream Cipher Snow 3G. *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC).* Lusanne, Switzerland (pp. 103-110).

[9] Ekdahl, P., Johansson, T., Maximov, A., & Yang, J. (2019). A new SNOW stream cipher called SNOW-V. *IACR transaction on symmetric cryptology, 1-42.*

[10] Ekdahl, P., Maximov, A., Johansson, T., & Yang, J. (2021). SNOW-Vi.

[11] Ghizlane, O., Said, E. H.,& Youssef, B. (2010). SNOW 3G stream cipher operation and complexity study. *Contemporary engineering sciences, 3(3)* (pp. 97-111).

[12] Gil, J. M. M., Gil, P. C., & Sabater, A. F. (2014). Análisis e implementación del generador SNOW 3G utilizado en las comunicaciones 4G. *RECSI XIII: Actas de la XIII Reunión Española sobre Criptología y Seguridad de la Información* Alicante (pp. 51-56).

[13] Gollmann, D. (1999). Dual bases and bit-serial multiplication in Fqn. *Theoretical Computer Science, 226(1-2)* (pp. 45-59).

[14] Hulle, Nagnath and B., Prathiba and Khope, Sarika and Anuradha, K. and Borole, Yogini and Kotambkar, D. (2021). Optimized architecture for SNOW 3G.

[15] Klimov, A., & Shamir, A. (2002). A New Class of Invertible Mappings. *Lecture Notes in Computer Science.*

[16] Knuth, D. E. (1998). *The Art of Computer Programming: Volume 3: Sorting and Searching. Addison-Wesley Professional.*

[17] Lidl, R., & Niederreiter, H. (1994). *Introduction to Finite Fields and Their Applications. Cambridge University Press.*

[18] Moro, E. M., Gómez, C. M., & Benito, D. R. (2007). Bases de Gröbner: Aplicaciones a la codificación algebraica. Instituto Venezolano de Investigaciones Científicas.

[19] Sadurní, J. M. (2021, 19 febrero). Alan Turing, el arma secreta de los aliados.

[20] (2006). *Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2.*

Algebraic Fault Injection Attack on the Snow Family of Stream Ciphers

Itciar Fernández Elízaga
Facultad de Ciencias • Sección de Matemáticas
Universidad de La Laguna
alu01013233848@ull.edu.es

Abstract

The Snow 3G stream cipher is crucial for ensuring the security of communications in 4G networks. It is based on its predecessor, Snow 2.0, but has been designed to be more resistant to algebraic attacks. In this work we will study the main components of stream ciphers, in particular the Linear Feedback Shift Registers (LFSRs). Specifically, we will study in detail the algorithm Snow 3G and a fault attack against this algorithm based on the Armknecht and Meier method. This method allows us to recover the secret key by inducing faults to obtain a system of nonlinear equations. By using the theory of Gröbner bases and a suitable monomial order we will be able to recover a complete state of the LFSR used.

1. LFSR

An LFSR of lenght L over $\mathbb{F}_q$ is a shift register which produces a sequence of elements of $\mathbb{F}_q$, $s = (s_t)_{t\geq 0}$, satisfying a linear recurrence relation of degree L over $\mathbb{F}_q$

$$s_{t+L} = \sum_{i=1}^{L} c_i s_{t+L-i},\ \forall t \geq 0$$

The L coefficients $c_1, \ldots, c_L$ are elements of $\mathbb{F}_q$ called the *feedback coefficients* of the LFSR.

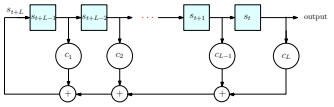

Figure 1: *Fibonacci representation of an LFSR.*

Definition. The feedback coefficients can be represented by *characteristic polynomial*, which is define by:

$$P^*(X) = X^L - \sum_{i=1}^{L} c_i X^{L-i}$$

The characteristic polynomial of $s = (s_t)_{t\geq 0}$ of minimum degree and monic is called the *minimal polynomial* of the sequence.

The minimum polynomial determines the period of a linear recurrence sequence.

Theorem. The sequence $s = (s_t)_{t\geq 0}$ generated by a nonsingular LFSR of length L is maximal if and only if its characteristic polynomial is primitive.

2. Snow 3G

Figure 2 shows the snow 3G generator scheme. Snow 3G is made up of an LFSR and an FSM (Finite State Machine). The LFSR has two modes of operation: initialization, in which it does not produce any output, and generation.

Keystream generation
$f^t = (s_{15}^t \boxplus R_1^t) \oplus R_2^t$
$z^t = s_0^t \oplus f^t$

Update of the FSM
$R_1^{t+1} = (s_5^t \oplus R_3^t) \boxplus R_2^t$
$R_2^{t+1} = S_1(R_1^t)$
$R_3^{t+1} = S_2(R_2^t)$

Where $\boxplus$ is the addition modulo 2^{32}, $\oplus$ is a XOR and S_1, S_2 represent two different 32-bit S-boxes which take 8 input bits and transform them into 32 output bits.

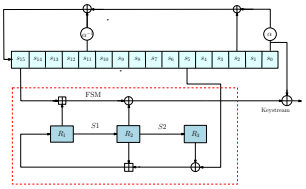

Figure 2: *Representation of snow 3G.*

3. Algebraic representation of the modular addition

Let us consider 3 n-bit words: $x = (x_{n-1}, \ldots, x_0)$, $y = (y_{n-1}, \ldots, y_0)$, $z = (z_{n-1}, \ldots, z_0)$. An algebraic representation of the addition $z = x \boxplus y \ mod\ 2^n$ is given by:

$$\begin{cases} z_0 = x_0 + y_0 \\ z_1 = x_1 + y_1 + x_0 y_0 \\ z_2 = x_2 + y_2 + x_1 y_1 + (x_1 + y_1) x_0 y_0 = x_2 + y_2 + (x_1 + y_1)(x_1 + y_1 + z_1) \\ \quad \vdots \\ z_i = x_i + y_i + x_i y_i + (x_{i-1} + y_{i-1})(x_{i-1} + y_{i-1} + z_{i-1}) \\ \quad \vdots \\ z_{n-1} = x_{n-1} + y_{n-1} + x_{n-2} y_{n-2} + (x_{n-2} + y_{n-2})(x_{n-2} + y_{n-2} + z_{n-2}) \end{cases}$$

4. Scheme of the attack

1. The attacker induces a fault on one of the three LFSR words s_2, s_3, s_4.
2. If the fault modifies $s_3^{t_0}$ at instant $t_0 + 3$, we have the equation: $z^{t_0+3} \oplus z'^{t_0+3} = s_3^{t_0} \oplus s_0'^{t_0}$, where $s_3'^{t_0}$ is the fault word.
3. By linearity of the LFSR, we will obtain each differences: $\triangle s_i^{t_0+j} = s_i^{t_0+j} \oplus s_i'^{t_0+j}\ \forall t \geq t_0$. In particular, we get $\triangle s_{15}^t\ \forall t \geq 0$.
4. We have the equation: $s_{15}^t \boxplus R_1^t = \underbrace{R_2^t \oplus s_0^t \oplus z^t}_{w^t}$. The system can also be described as a system of 32 equations on the field $\mathbb{F}_2$ of degree at most 2, with 32×3 variables which are the unknown bits of s_{15}^t, R_1^t and w^t.
5. The attacker produces a new fault on the system and obtain a new system of equations on the same 32×3 variables.
6. The attacker applies Gröbner bases with respect to lexicographic order to obtain a system of linear equations depending only on the bits of s_{15}^t.
7. The attacker must induce faults at different execution times to obtain enough equations to recover the initial state of the LFSR.

References

[1] Canteaut, A. (2011). *Linear Feedback Shift Register. In: van Tilborg, H.C.A., Jajodia, S. (eds) Encyclopedia of Cryptography and Security.* Springer, Boston, MA.

[2] Debraize, B., & Corbella, I. M. (2009). Fault Analysis of the Stream Cipher Snow 3G. *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. Lusanne, Switzerland (pp. 103-110).