COMPARTICIÓN DE SECRETOS: UNA APLICACIÓN A IMÁGENES

TESIS QUE PRESENTA

ANAHÍ RENDÓN ESPINOSA

PARA OBTENER EL TÍTULO DE
LICENCIADO EN MATEMÁTICAS APLICADAS

ASESOR: DR. CARLOS ALBERTO LÓPEZ ANDRADE

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA
Facultad de Ciencias Físico Matemáticas
http://www.fcfm.buap.mx/
Enero 2018

WEBSITE:
http://www.fcfm.buap.mx/clopez

E-MAIL:
anahi.rendon@gmail.com

AGRADECIMIENTOS

Agradezco a mis padres por estar siempre, por su apoyo incondicional. A ustedes, Javier y Rocío, dedico este esfuerzo.

Agradezco al Dr. Carlos Alberto López Andrade por presentarme al álgebra y compartirme sus implacables ganas de comprenderla. De usted he aprendido que la paciencia y el trabajo son aliados de esta aventura.

Agradezco a la Mtra. Edith Salgado Cuevas por las tardes de té, risa e inspiración. Por guiarme en la escritura de este trabajo, cuidando mi esencia, dejándome ser.

Al M.C. Edgar Santiago Móyotl Hernández y a todos aquellos que me extendieron la mano.

A las dulces y amargas coincidencias de la vida, que me han hecho ser quien soy y estar acompañada de tanto cariño. Gracias a todos los que no me dejan nunca ir para atrás.

Danke, Schatz.

Los autores agradecen al Laboratorio Nacional de Supercómputo del Sureste de México, perteneciente a la red de laboratorios nacionales CONACYT, por los recursos computacionales, el apoyo y la asistencia técnica.

Esta investigación fue realizada gracias al Consejo de Ciencia y Tecnología del Estado de Puebla.

INTRODUCCIÓN

La comunicación ha sido un elemento fundamental para el desarrollo del hombre. Su importancia y alcance van desde expresar necesidades y sentimientos hasta intereses políticos y económicos, con mensajes que no siempre se quiere, sean de conocimiento público. Distribuir información secreta fue, indiscutiblemente, una inspiración que consolidaría más tarde a una nueva ciencia aplicada llamada Criptografía, cuyo objetivo principal quedaría definido como la protección de la información.

Los primeros registros del uso de la escritura secreta datan del Siglo V a. C. durante la guerra de Atenas y Esparta. Desde entonces, el peligro de que un mensaje transmitido sea interceptado por un enemigo o llegue a manos de alguien no deseado ha sido latente. Secretos descubiertos han provocado incomodidades diplomáticas leves hasta conflictos militares. Un ejemplo es el conocido telegrama Zimmermann, cuyo mensaje fue descifrado por parte de los británicos; en este, el ministro alemán intentaba convencer a Japón y México para que invadieran EE. UU. El descubrimiento de este mensaje fue la pieza clave para que EE. UU. decidiera participar en la Primera Guerra Mundial y, posiblemente, una de las más grandes motivaciones para el desarrollo de la Criptografía.

En México, los primeros registros de correspondencia codificada fueron los que mantuvieron la Corona Española y los gobernantes en la Nueva España. Algunos historiadores suponen que tras la independencia de México se utilizó algún tipo de critografía, pero no se han encontrado evidencias. Lo que sí se sabe es que a partir del telégrafo, el uso de la criptografía creció considerablemente. Tal es el caso de Benito Juárez, quien mantuvo una constante comunicación cifrada con Don Ignacio L. Vallarta, gobernador de Jalisco; ellos usaron un sistema de sustitución simple. Otro caso de uso de textos cifrados es el que utilizó Porfirio Díaz para comunicarse con gobernadores y jefes militares importantes.

Las raíces etimológicas de la palabra criptografía son *kriptos*, que significa oculto y *graphos*, que se traduce como escribir, a partir de lo cual una definición clásica de criptografía es el arte de escribir mensajes en clave secreta o enigmáticamente. Fue en 1949, cuando Shannon publicó la Teoría de las comunicaciones secretas [CE49]; a partir de ello, la criptografía pasó de ser considerada un arte a una ciencia aplicada, debido a su estrecha relación con otras ciencias como la estadística, la teoría de números, la teoría de la información y la teoría de la complejidad computacional.

El alcance de las aplicaciones criptográficas ha aumentado con el desarrollo de la cultura informática y la internet, ya que ha surgido la necesidad de proteger datos durante su transmisión y su almacenamiento. Además, se ha producido un cambio radical en el concepto de seguridad de sistemas criptográficos, pues aquellos que eran seguros frente a cálculos manuales se volvieron vulnerables ante la eficacia de las computadoras actuales. De esta forma, la seguridad de los sistemas clásicos ha pasado de ser un ejercicio de aficionados a un desarrollo científico en la constante búsqueda de una seguridad matemática y computacionalmente demostrable. De igual forma, las necesidades y lineamientos de la protección de la información están en constante evolución, por lo que el desarrollo de esta ciencia ha tenido que ajustarse a dichos cambios y a plantearse nuevas técnicas de seguridad. Por ejemplo, en la criptografía clásica los protocolos criptográficos contaban con solo dos participantes: el poseedor de la información original y el receptor de dicha información. Sin embargo, existen situaciones en las que la información no es generada exclusivamente por personas físicas, sino por empresas o entidades o bien, en las que la seguridad de la misma afecta a un grupo mayor de personas. Un sencillo ejemplo son los populares grupos en WhatsApp. Este nuevo requerimiento, donde los involucrados pasaron a ser más de dos participantes, fue la inspiración para un nuevo tema de estudio: la compartición de secretos.

Los esquemas de compartición de secretos se introdujeron de forma independiente por Blackley y Shamir en 1979. La idea desde ese entonces ha sido repartir el valor de un secreto en fragmentos entre los participantes de un conjunto P, de tal manera que solo los subconjuntos autorizados puedan reconstruir el secreto a partir de sus fragmentos.

Más adelante, nace la Criptografía Visual, que es una técnica de codificación con la que se desea compartir a una imagen secreta. Fue propuesta por primera vez por Naor y Shamir en la *Eurocrypt 1994 Conference*. Uno de los principales intereses de esta propuesta recae en que la implementación de estos esquemas no necesita ningún tipo de conocimiento o cómputo criptográfico, para su decodificación, dejando a la misma en las manos del sistema visual humano.

Los esquemas de compartición de secretos, cuando el secreto es un número o una imagen binaria, son un tema cada vez más popular en muchos libros de criptografía; sin embargo, pocas veces las soluciones propuestas son llevadas a la práctica y los ejemplos, en la mayoría de los textos que desarrollan dicho tema, se reducen a trabajar el tema para un conjunto de dos participantes. El propósito de esta tesis fue proponer soluciones a problemas de compartición de secretos, cuando el secreto es una imagen digital binaria y que dichas soluciones puedan ser llevadas a la práctica.

Esta tesis consta de 2 capítulos. El primer capítulo consiste en el estudio de los esquemas de compartición de secretos, cuando el secreto es un número y en la teoría matemática que hay detrás de dichos esquemas. Para esta parte, se toma como referencia al capítulo 11 del libro *Cryptography Theory And Practice*. Se partió del estudio de un esquema de compartición de secretos en el que, si se reúne un número mínimo de participantes, se tiene acceso al secreto. Posteriormente, se trabajó con esquemas más complejos en los que los conjuntos de participantes que pueden tener acceso al secreto son establecidos y su cardinalidad no es una característica principal. Para ello se recurrió al uso de definiciones y teoremas que reforzaron dicha teoría; también, se utilizó el algoritmo de Quine-McCluskey que se sugiere como herramienta complementaria para encontrar soluciones mejoradas. El capítulo concluye con el desarrollo de un problema de aplicación que abarca a la mayoría de la teoría trabajada hasta el momento.

El segundo capítulo trata sobre la compartición de secretos visuales. Se partió del concepto de imagen digital. Posteriormente, se definió a una solución para un problema de compartición de secretos para una imagen binaria y se continuó con el estudio de una serie de soluciones. Algunas de ellas fueron tomadas de la bibliografía y algunas otras fueron propuestas. Este capítulo termina con la presentación de los programas en MATLAB® que permiten compartir una imagen digital binaria con base a un (2,2), (2,3), (2,4), (2,n) con técnica de replicación, (3,3), (3,4), (3,n) con técnica de replicación y (4,4) esquema visual umbral.

ÍNDICE GENERAL

1 ESQUEMAS DE COMPARTICIÓN DE SECRETOS

Un esquema de compartición de secretos es un método que permite dividir un secreto entre un conjunto de n participantes de modo que solo determinados subconjuntos calificados, bajo criterios de acuerdo con los intereses del grupo, puedan recuperar el secreto original. La idea básica de estos esquemas consiste en permitir que un distribuidor (o tercera parte de confianza), en el que confían todos los participantes, divida el secreto a compartir en tantas piezas como participantes haya. Dichas piezas son distribuidas entre los n participantes, de modo que para recuperar el secreto original se deben unir todas las piezas de algún subconjunto calificado.

Existen diferentes tipos de esquemas de compartición de secretos. En este capítulo, se presenta el esquema umbral de Shamir, para después abordar un problema más general, donde los subconjuntos calificados para recuperar el secreto no requieren tener un número de participantes mínimo requerido.

1.1 EL ESQUEMA UMBRAL DE SHAMIR

En un banco, hay una bóveda que debe ser abierta cada día. El banco emplea tres cajeros de alto nivel, pero no confía la totalidad de la combinación a ninguno de ellos. Sin embargo, el banco desea un sistema de seguridad, donde dos de los tres cajeros de alto nivel puedan tener acceso a la bóveda, sin que ninguno pueda hacerlo de forma individual. Este tipo de problema se puede resolver por medio de un *esquema de compartición de secretos*.

La revista *Time*, en 1992, presentó un ejemplo de un esquema de compartición de secretos sobre una situación real. Este caso fue el control de armas nucleares en Rusia, cuyas partes involucradas eran el Presidente, el Ministro de Defensa y el Ministerio de Defensa. El mecanismo era similar al expuesto en el párrafo anterior: *two out of three*.

Se puede observar que en ambos ejemplos, ya sea para tener acceso a la bóveda o para tener el control de las armas nucleares de Rusia, es necesaria la participación de dos de los tres miembros involucrados. La solución a la generalidad de este tipo de problemas se ilustra con la siguiente definición.

Definición 1.1. *Sean* t, w *enteros positivos tales que* $t \leqslant w$. *Un* (t, w)*—esquema umbral es un método para compartir una llave* K *entre un grupo de* w *participantes (denotado por* $\mathcal{P}$*), de tal manera que, cualesquiera* t *participantes puedan calcular el valor de* K*, pero ningún grupo de* $t-1$ *participantes pueda hacerlo.*

Los dos ejemplos descritos anteriormente corresponden a la definición de un $(2, 3)$—esquema umbral.

Ahora, dado que $\mathcal{P}$ es el conjunto de los w participantes entre los cuales se compartirá la llave, es decir,

$$\mathcal{P} = \{P_i \mid 1 \leqslant i \leqslant w\},$$

decimos que el valor de K es elegido por un participante especial llamado *distribuidor*. El distribuidor es denotado por D y asumimos que $D \notin \mathcal{P}$. Cuando D quiere compartir la llave K entre los participantes en $\mathcal{P}$, él da a cada participante una pieza de la información necesaria para determinar el valor de K. A cada una de estas piezas se le llamará *parte*. Las partes deben distribuirse secretamente, de manera que ningún participante conozca la parte dada a otro participante.

En un momento posterior, un subconjunto de los participantes $B \subseteq \mathcal{P}$ juntará sus partes en un intento por determinar el valor de K (alternativamente, podrían dar sus partes a una autoridad de confianza que llevara a cabo el cálculo para ellos). Si $|B| \geqslant t$, entonces, ellos deberían ser capaces de determinar el valor de K en función de las partes que conjuntamente poseen; si $|B| < t$, entonces, no deberían ser capaces de determinar dicho valor. Se llama $\mathcal{K}$ al *conjunto de llaves* (el conjunto de todas las posibles llaves); y S al *conjunto de partes* (el conjunto de todos las posibles partes).

A continuación se presenta el algoritmo 1, el cual es un método para construir un (t, w)-esquema umbral. Este método se conoce como **esquema umbral de Shamir** y fue inventado por Shamir en 1979. Sea $\mathcal{K} = \mathbb{Z}_p$ con p primo y $p \geqslant w + 1$. Además, sea $S = \mathbb{Z}_p$. Esto significa que tanto la llave

como las partes dadas a cada participante serán elementos de $\mathbb{Z}_p$. En este esquema, el distribuidor construye un polinomio aleatorio $a(x) \in \mathbb{Z}_p[x]$ de grado a lo más $t-1$, donde el término constante de dicho polinomio es, precisamente, la llave K. Cada participante P_i obtiene un punto (x_i, y_i) en este polinomio.

Algoritmo 1 (t,w)-esquema umbral de Shamir

1: D elige w elementos de $\mathbb{Z}_p$ distintos entre sí y diferentes de cero, denotados por x_i, $1 \leqslant i \leqslant w$ (aquí es donde se requiere que $p \geqslant w+1$). Para $1 \leqslant i \leqslant w$, D da el valor de x_i a P_i. Los valores x_i son públicos.
2: Supongamos que D quiere compartir una llave $K \in \mathcal{K} = \mathbb{Z}_p$. D elige secretamente $t-1$ elementos de $\mathbb{Z}_p$ (independientes y aleatorios), los cuales son denotados por $a_1, \ldots, a_{t-1}$.
3: Para $1 \leqslant i \leqslant w$, D calcula $y_i = a(x_i)$, donde

$$a(x) = K + \sum_{j=1}^{t-1} a_j x^j \ mod\ p.$$

4: Para $1 \leqslant i \leqslant w$, D da la parte y_i a P_i.

Ahora, se verá cómo un subconjunto B de t participantes pueden reconstruir al polinomio $a(x)$ y encontrar el valor de K. Esto se logra básicamente por medio de un polinomio de interpolación. Para ello, se describirán un par de métodos.

PRIMER MÉTODO. Supongamos que los participantes $P_{i1}, \ldots, P_{it}$ quieren determinar el valor de K. Se sabe que

$$y_{ij} = a(x_{ij}),$$

$1 \leqslant j \leqslant t$, donde $a(x) \in \mathbb{Z}_p[x]$ es el polinomio secreto elegido por D. Como $a(x)$ es de grado a lo más $t-1$, $a(x)$ puede escribirse como

$$a(x) = a_0 + a_1 x + \ldots + a_{t-1} x^{t-1},$$

donde los coeficientes $a_0, \ldots, a_{t-1}$ son elementos desconocidos de $\mathbb{Z}_p$, y $a_0 = K$ es la llave. Como $y_{ij} = a(x_{ij})$ para $1 \leqslant j \leqslant t$, el conjunto B de t participantes puede obtener t ecuaciones lineales con t incógnitas $a_0, \ldots, a_{t-1}$, donde toda la aritmética se realiza en $\mathbb{Z}_p$. Si las filas de la matriz de coeficientes del sistema no homogéneo son l.i., se tendrá una única solución y a_0 será revelada como el valor de la llave K.

Ejemplo 1.1. *Sea* $p = 17$, $t = 3$ *y* $w = 5$, *y se tienen* x*—coordenadas públicas que son* $x_i = i$, $1 \leqslant i \leqslant 5$. *Supongamos que* $B = \{P_1, P_3, P_5\}$ *es el conjunto de participantes cuyas partes son* $8, 10$ *y* 11, *respectivamente. Se escribe el polinomio* $a(x)$ *como*

$$a(x) = a_0 + a_1 x + a_2 x^2,$$

se calcula $a(1)$, $a(3)$ *y* $a(5)$, *y se obtienen las siguientes tres ecuaciones lineales en* $\mathbb{Z}_{17}$*:*

$$\begin{aligned} a_0 + a_1 + a_2 &= 8 \\ a_0 + 3a_1 + 9a_2 &= 10 \\ a_0 + 5a_1 + 25a_2 &= 11. \end{aligned}$$

Este sistema tiene una solución única en $\mathbb{Z}_{17}$*:* $a_0 = 13$, $a_1 = 10$ *y* $a_2 = 2$. *La llave, por lo tanto, es* $K = a_0 = 13$.

Definición 1.2. *Sean* $\alpha_1, \ldots, \alpha_n \in F$, *un campo. La matriz de Vandermonde generada por los puntos* $\alpha_1, \ldots, \alpha_n$ *se define mediante la siguiente fórmula:*

$$V(\alpha_1, \alpha_2, ..., \alpha_n) := \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \ldots & \alpha_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \ldots & \alpha_n^{n-1} \end{bmatrix}$$

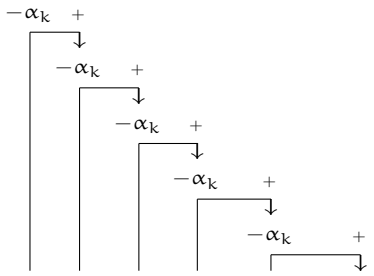
Teorema 1.1. *El determinante de una matriz de Vandermonde puede ser calculado como sigue:*

$$\det V(\alpha_1, \alpha_2, \ldots, \alpha_n) = \prod_{1 \leqslant i < j \leqslant n} (\alpha_j - \alpha_i).$$

Demostración. Por inducción.
Se usará a $n = 2$ como base de la inducción. Veamos que la fórmula es válida para $n = 2$:

$$\det V(\alpha_1, \alpha_2) = \begin{vmatrix} 1 & \alpha_1 \\ 1 & \alpha_2 \end{vmatrix} = \alpha_2 - \alpha_1 = \prod_{1 \leqslant i < j \leqslant 2} (\alpha_j - \alpha_i).$$

Ahora, supongamos que la fórmula es válida para $n = k-1$ y demostraremos que se cumple para $n = k$.
Para producir ceros en todas las entradas del último renglón excepto la primera entrada, se realizarán las siguientes operaciones elementales en las columnas de izquierda a derecha:



$$\det V(\alpha_1, \ldots, \alpha_k) = \begin{vmatrix} 1 & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^{k-2} & \alpha_1^{k-1} \\ 1 & \alpha_2 & \alpha_2^2 & \ldots & \alpha_2^{k-2} & \alpha_2^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \alpha_k & \alpha_k^2 & \ldots & \alpha_k^{k-2} & \alpha_k^{k-1} \end{vmatrix}$$

$$= \begin{vmatrix} 1 & \alpha_1 - \alpha_k & \alpha_1^2 - \alpha_1\alpha_k & \ldots & \alpha_1^{k-2} - \alpha_1^{k-3}\alpha_k & \alpha_1^{k-1} - \alpha_1^{k-2}\alpha_k \\ 1 & \alpha_2 - \alpha_k & \alpha_2^2 - \alpha_2\alpha_k & \ldots & \alpha_2^{k-2} - \alpha_2^{k-3}\alpha_k & \alpha_2^{k-1} - \alpha_2^{k-2}\alpha_k \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & 0 & \ldots & 0 & 0 \end{vmatrix}$$

$$= (1)(-1)^{k+1} \begin{vmatrix} \alpha_1 - \alpha_k & \alpha_1^2 - \alpha_1\alpha_k & \ldots & \alpha_1^{k-2} - \alpha_1^{k-3}\alpha_k & \alpha_1^{k-1} - \alpha_1^{k-2}\alpha_k \\ \alpha_2 - \alpha_k & \alpha_2^2 - \alpha_2\alpha_k & \ldots & \alpha_2^{k-2} - \alpha_2^{k-3}\alpha_k & \alpha_2^{k-1} - \alpha_2^{k-2}\alpha_k \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \alpha_{k-1} - \alpha_k & \alpha_{k-1}^2 - \alpha_{k-1}\alpha_k & \ldots & \alpha_{k-1}^{k-2} - \alpha_{k-1}^{k-3}\alpha_k & \alpha_{k-1}^{k-1} - \alpha_{k-1}^{k-2}\alpha_k \end{vmatrix}$$

Para cada $i \in 1, \ldots, k-1$ del i–renglón factoricemos $\alpha_i - \alpha_k$

$$\det V(\alpha_1, \ldots, \alpha_k) = \begin{vmatrix} \alpha_1 - \alpha_k & \alpha_1(\alpha_1 - \alpha_k) & \ldots & \alpha_1^{k-3}(\alpha_1 - \alpha_k) & \alpha_1^{k-2}(\alpha_1 - \alpha_k) \\ \alpha_2 - \alpha_k & \alpha_2(\alpha_2 - \alpha_k) & \ldots & \alpha_2^{k-3}(\alpha_2 - \alpha_k) & \alpha_2^{k-2}(\alpha_2 - \alpha_k) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \alpha_{k-1} - \alpha_k & \alpha_{k-1}(\alpha_{k-1} - \alpha_k) & \ldots & \alpha_{k-1}^{k-3}(\alpha_{k-1} - \alpha_k) & \alpha_{k-1}^{k-2}(\alpha_{k-1} - \alpha_k) \end{vmatrix}$$

Se sacan los factores comunes por renglón y se obtiene:

$$\det V(\alpha_1,\ldots,\alpha_k) = (-1)^2(-1)^{k-1}\prod_{1\leqslant i\leqslant k-1}(\alpha_i-\alpha_k)\begin{vmatrix} 1 & \alpha_1 & \ldots & \alpha_1^{k-3} & \alpha_1^{k-2} \\ 1 & \alpha_2 & \ldots & \alpha_2^{k-3} & \alpha_2^{k-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \alpha_{k-1} & \ldots & \alpha_{k-1}^{k-3} & \alpha_{k-1}^{k-2} \end{vmatrix}$$

Usando el factor $(-1)^{k-1}$ se cambia el signo de los factores $(\alpha_i-\alpha_k)$ para $i\in\{1,\ldots,k-1\}$:

$$\det V(\alpha_1,\ldots,\alpha_k) = (-1)^2\prod_{1\leqslant i\leqslant k-1}(\alpha_k-\alpha_i)\begin{vmatrix} 1 & \alpha_1 & \ldots & \alpha_1^{k-3} & \alpha_1^{k-2} \\ 1 & \alpha_2 & \ldots & \alpha_2^{k-3} & \alpha_2^{k-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \alpha_{k-1} & \ldots & \alpha_{k-1}^{k-3} & \alpha_{k-1}^{k-2} \end{vmatrix}$$

$$= \prod_{1\leqslant i\leqslant k-1}(\alpha_k-\alpha_i)\begin{vmatrix} 1 & \alpha_1 & \ldots & \alpha_1^{k-3} & \alpha_1^{k-2} \\ 1 & \alpha_2 & \ldots & \alpha_2^{k-3} & \alpha_2^{k-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & \alpha_{k-1} & \ldots & \alpha_{k-1}^{k-3} & \alpha_{k-1}^{k-2} \end{vmatrix}$$

Se puede observar que el último determinante es el $\det V(\alpha_1,\ldots,\alpha_{k-1})$. Usando la hipótesis inductiva se tiene que:

$$\det V(\alpha_1,\ldots,\alpha_k) = \prod_{1\leqslant i\leqslant k-1}(\alpha_k-\alpha_i)\prod_{1\leqslant i<j\leqslant k-1}(\alpha_j-\alpha_i) = \prod_{1\leqslant i<j\leqslant k}(\alpha_j-\alpha_i).$$

□

Es fundamental que el sistema de t ecuaciones lineales, que se obtienen con el método de contrucción de un (t,w)—esquema umbral, tenga una única solución como en el ejemplo 1.1. Se mostrará ahora que este es siempre el caso. En general, se tiene que

$$y_{i_j} = a(x_{i_j}),$$

$1\leqslant j\leqslant t$, donde

$$a(x) = a_0 + a_1x + \ldots + a_{t-1}x^{t-1},$$

y

$$a_0 = K.$$

El sistema de ecuaciones lineales (en $\mathbb{Z}_p$) es el siguiente:

$$\begin{aligned} a_0 + a_1x_{i_1} + a_2x_{i_1}^2 + \ldots + a_{t-1}x_{i_1}^{t-1} &= y_{i_1} \\ a_0 + a_1x_{i_2} + a_2x_{i_2}^2 + \ldots + a_{t-1}x_{i_2}^{t-1} &= y_{i_2} \\ &\vdots \\ a_0 + a_1x_{i_t} + a_2x_{i_t}^2 + \ldots + a_{t-1}x_{i_t}^{t-1} &= y_{i_t} \end{aligned}$$

Esto puede ser escrito en forma matricial como:

$$\begin{pmatrix} 1 & x_{i_1} & x_{i_1}^2 & \ldots & x_{i_1}^{t-1} \\ 1 & x_{i_2} & x_{i_2}^2 & \ldots & x_{i_2}^{t-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{i_t} & x_{i_t}^2 & \ldots & x_{i_t}^{t-1} \end{pmatrix}\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{pmatrix} = \begin{pmatrix} y_{i_1} \\ y_{i_2} \\ \vdots \\ y_{i_t} \end{pmatrix}$$

La matriz de coeficientes A es una matriz de Vandermonde. Haciendo uso de la fórmula para calcular el determinante de una matriz de Vandermonde, se tiene que

$$\det A = \prod_{1 \leqslant j < k \leqslant t} (x_{i_k} - x_{i_j}) \bmod p.$$

Recordemos que las $x_i's$ son todas distintas, así que ningún término $x_{i_j} - x_{i_k}$ en este producto es igual a 0. Además, el producto es calculado en $\mathbb{Z}_p$, donde p es primo, así que $\mathbb{Z}_p$ es un campo. Como el producto de términos distintos de cero en un campo es siempre distinto de cero, se tiene que $\det A \neq 0$. Dado que el determinante de la matriz de coeficientes es distinto de cero, entonces, se concluye que tiene una única solución en el campo $\mathbb{Z}_p$. Esto establece también que cualquier grupo de t participantes podría acceder a recuperar la llave en este esquema umbral.

¿Qué ocurre si un grupo de $t-1$ participantes intenta calcular a K?

La respuesta es la siguiente: ellos obtendrían un sistema de $t-1$ ecuaciones con t incógnitas. Supongamos que ellos tienen un valor hipotético para el valor de y_0 (la llave). Ya que la llave es $a_0 = a(0)$, esto da paso a una t−ésima ecuación y la matriz de coeficientes del sistema resultante de t ecuaciones es una matriz de Vandermonde, nuevamente. Como antes, esto tiene una solución única. Por consiguiente, para cada valor hipotético y_0 de la llave en $\mathbb{Z}_p$ existe un único polinomio $a_{y_0}(x)$ tal que

$$y_{i_j} = a_{y_0}(x_{i_j})$$

$1 \leqslant j \leqslant t-1$, y tal que

$$y_0 = a_{y_0}(0).$$

Por lo tanto, ningún valor de la llave puede ser descartado y un grupo de $t-1$ participantes no puede obtener información alguna de la llave.

SEGUNDO MÉTODO. (Método basado en el polinomio de interpolación de Lagrange)
El polinomio de interpolación de Lagrange es una fórmula explícita para el único polinomio $a(x)$ de grado a lo más $t-1$ que se exhibe a continuación:

$$a(x) = \sum_{j=1}^{t} y_{i_j} \prod_{1 \leqslant k \leqslant t, k \neq j} \frac{x - x_{i_k}}{x_{i_j} - x_{i_k}}$$

Teorema 1.2. *(Fórmula de interpolación de Lagrange). Sea p primo y supongamos $x_1, x_2, \ldots, x_{m+1}$ elementos distintos en $\mathbb{Z}_p$ y $y_1, y_2, \ldots, y_{m+1}$ elementos (no necesariamente distintos) en $\mathbb{Z}_p$. Entonces, existe un único polinomio $a(x) \in \mathbb{Z}_p[x]$ de grado a lo más m, tal que $a(x_i) = y_i$, $1 \leqslant i \leqslant m+1$. Tal polinomio $a(x)$ es de la forma:*

$$a(x) = \sum_{j=1}^{m+1} y_j \prod_{1 \leqslant h \leqslant m+1, h \neq j} \frac{x - x_h}{x_j - x_h}$$

Demostración. Sean $x_1, x_2, \ldots, x_{m+1}$, $m+1$ escalares distintos en $\mathbb{Z}_p$. Defínanse los polinomios $p_1(x), p_2(x), \ldots, p_{m+1}(x)$ como

$$p_i(x) = \frac{(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_{m+1})}{(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_{m+1})},$$

estos son llamados los polinomios de Lagrange asociados a $x_1, x_2, \ldots, x_{m+1}$.
Veamos que

$$p_i(x_j) = \begin{cases} 0 \text{ si } i \neq j \\ 1 \text{ si } i = j \end{cases} \tag{1}$$

siempre que $1 \leqslant i, j \leqslant m+1$. Evaluando $p_i(x)$ en x_j se obtiene:

CASO 1: Si $i \neq j$, supongamos, sin pérdida de generalidad que $i < j$, entonces

$$p_i(x_j) = \frac{(x_j-x_1)\cdots(x_j-x_{i-1})(x_j-x_{i+1})\cdots(x_j-x_j)\cdots(x_j-x_{m+1})}{(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_j)\cdots(x_i-x_{m+1})} = 0,$$

dado que el producto en el numerador contiene al término $x_j - x_j = 0$. Obsérvece que el producto en el denominador es distinto de cero ya que por hipótesis los $x_1, x_2, \ldots, x_{m+1}$ son todos distintos.

CASO 2: $i = j$,

$$p_i(x_i) = \frac{(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_{m+1})}{(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_{m+1})} = 1,$$

por lo tanto, la ecuación (1) es verdadera siempre que $1 \leqslant i, j \leqslant m+1$.

Sea $\mathbb{P}_m = \{p(x) \in \mathbb{F}[x] : grad(p(x)) \leqslant m\}$. Veamos que $p_1(x), p_2(x), \ldots, p_{m+1}(x)$ son l.i.s en $\mathbb{P}_m$. Supongamos que

$$c_1 p_1(x) + c_2 p_2(x) + \cdots + c_{m+1} p_{m+1}(x) = 0, \tag{2}$$

para algunos $c_1, c_2, \ldots, c_{m+1} \in \mathbb{Z}_p$. A partir de (2) se tiene que:

$$\begin{aligned} c_1 p_1(x_1) + c_2 p_2(x_1) + \cdots + c_{m+1} p_{m+1}(x_1) &= 0 \\ c_1 p_1(x_2) + c_2 p_2(x_2) + \cdots + c_{m+1} p_{m+1}(x_2) &= 0 \\ &\vdots \\ c_1 p_1(x_{m+1}) + c_2 p_2(x_{m+1}) + \cdots + c_{m+1} p_{m+1}(x_{m+1}) &= 0, \end{aligned}$$

y a consecuencia de (1) y el sistema de ecuaciones anterior se concluye que:

$$\begin{aligned} c_1 &= 0 \\ c_2 &= 0 \\ &\vdots \\ c_{m+1} &= 0, \end{aligned}$$

por lo que, $c_1 p_1(x) + c_2 p_2(x) + \cdots + c_{m+1} p_{m+1}(x) = 0$ implica que $c_1 = c_2 = \cdots = c_{m+1} = 0$ y, entonces, $\{p_1(x), p_2(x), \ldots, p_{m+1}(x)\}$ es un conjunto l.i. en $\mathbb{P}_m$. Dado que $dim(\mathbb{P}_m) = m+1$ y que $\{p_1(x), p_2(x), \ldots, p_{m+1}(x)\}$ es un conjunto l.i. en $\mathbb{P}_m$ de cardinalidad $m+1$, entonces $\beta = \{p_1(x), p_2(x), \ldots, p_{m+1}(x)\}$ es una base para $\mathbb{P}_m$. Además, cada vector en $\mathbb{P}_m$ tiene una única representación como combinación lineal de los vectores de la base, por lo que podemos concluir que la combinación lineal para $a(x) \in \mathbb{P}_m$ con respecto a la base β es única.

Tomando a $y_i = a(x_i)$ y dado que $a(x_i) = c_i$ por (1), se tiene que $c_i = y_i$, por lo que

$$\begin{aligned} a(x) &= c_1 p_1(x) + c_2 p_2(x) + \cdots + c_{m+1} p_{m+1}(x) \\ &= \sum_{j=1}^{m+1} y_j \prod_{1 \leqslant h \leqslant m+1, h \neq j} \frac{x - x_h}{x_j - x_h}, \end{aligned}$$

y la fórmula de interpolación de Lagrange queda demostrada. □

Es fácil de verificar la exactitud de la fórmula sustituyendo $x = x_{ij}$. Todos los términos de la sumatoria desaparecerán excepto el j-*ésimo* término, el cual es precisamente y_{ij}. Por lo tanto, se tiene un polinomio de grado a lo más $t-1$ el cual interpola a los t pares ordenados (x_{ij}, y_{ij}), $1 \leqslant j \leqslant t$.

Un grupo B de t participantes puede calcular $a(x)$ usando la fórmula de interpolación; sin embargo, recuérdese que dicho grupo está especialmente interesado en el valor de la constante $K = a(0)$. Dicho de otra forma, para encontrar el valor de interés K se sustituye $x = 0$ en la fórmula de interpolación de Lagrange y se obtiene:

$$\begin{aligned} K &= \sum_{j=1}^{t} y_{i_j} \prod_{1 \leqslant k \leqslant t, k \neq j} \frac{0 - x_{i_k}}{x_{i_j} - x_{i_k}} \\ &= \sum_{j=1}^{t} y_{i_j} \prod_{1 \leqslant k \leqslant t, k \neq j} \frac{x_{i_k}}{x_{i_k} - x_{i_j}} \end{aligned}$$

Se define a b_j como:

$$b_j = \prod_{1 \leqslant k \leqslant t, k \neq j} \frac{x_{i_k}}{x_{i_k} - x_{i_j}},$$

$1 \leqslant j \leqslant t$, (nótese que estos valores b_j pueden ser precalculados y sus valores no serían secretos). Entonces, se tiene que:

$$K = \sum_{j=1}^{t} b_j y_{i_j},$$

por lo tanto, la llave es una combinación lineal de las t partes.

Ejemplo 1.2. *Usando los mismos datos que en el ejemplo 1.1, se ilustrará a continuación cómo los participantes* $\{P_1, P_3, P_5\}$ *pueden tener acceso a la llave. En primer lugar, se deben calcular los valores de* b_1, b_3 *y* b_5 *de acuerdo a la fórmula dada arriba:*

$$\begin{aligned} b_1 &= \frac{x_3 x_5}{(x_3 - x_1)(x_5 - x_1)} \ mod\ 17 \\ &= 3 \times 5 \times (2^{-1}) \times (4^{-1}) \ mod\ 17 \\ &= 15 \times 9 \times 13 \ mod\ 17 \\ &= 1755 \ mod\ 17 \\ &= 4 \end{aligned}$$

$$\begin{aligned} b_3 &= \frac{x_1 x_5}{(x_1 - x_3)(x_5 - x_3)} \ mod\ 17 \\ &= 1 \times 5 \times (-2^{-1}) \times (2^{-1}) \ mod\ 17 \\ &= 5 \times 8 \times 9 \ mod\ 17 \\ &= 360 \ mod\ 17 \\ &= 3 \end{aligned}$$

$$\begin{aligned} b_5 &= \frac{x_1 x_3}{(x_1 - x_5)(x_3 - x_5)} \ mod\ 17 \\ &= 1 \times 3 \times (-4^{-1}) \times (-2^{-1}) \ mod\ 17 \\ &= 3 \times 4 \times 8 \ mod\ 17 \\ &= 96 \ mod\ 17 \\ &= 11. \end{aligned}$$

Entonces, dadas sus partes $8, 10$ *y* 11 *respectivamente, ellos obtendrían* $K = 4 \times 8 + 3 \times 10 + 11 \times 11 \ mod\ 17 = 13$, *como antes.*

1.1.1 Un (t,t)-esquema umbral simplificado

Una construcción simplificada para el esquema umbral es el caso especial en el que $w = t$. Esta construcción funciona para cualquier conjunto de llaves $\mathcal{K} = \mathbb{Z}_m$ con $S = \mathbb{Z}_m$ (para este esquema no se requiere que m sea primo y no es necesario que $m \geqslant w + 1$). Si D quiere compartir la llave $K \in \mathbb{Z}_m$, entonces debe llevar a cabo el procedimiento descrito en el algoritmo 2.

Algoritmo 2 (t,t)-esquema umbral simplificado

1: D elige secretamente $t - 1$ elementos de $\mathbb{Z}_m$, $y_1, \ldots, y_{t-1}$.
2: D calcula $y_t = K - \sum_{i=1}^{t-1} y_i \ mod\ m$.
3: D da y_i a P_i, $1 \leqslant i \leqslant t$.

Obsérvece que los t participantes pueden calcular a K con la fórmula:

$$K = \sum_{i=1}^{t} y_i \ mod\ m$$

¿Pueden $t-1$ participantes calcular a K?
Los primeros $t-1$ participantes no pueden hacerlo, ya que ellos recibieron $t-1$ números aleatorios e independientes como sus partes. Ahora, considerar a $t-1$ participantes en el conjunto $\mathcal{P} \setminus \{P_i\}$, donde $1 \leqslant i \leqslant t-1$. Entonces, los $t-1$ participantes poseen las partes

$$y_1, y_2, \ldots, y_{i-1}, y_{i+1}, \ldots, y_{t-1}$$

y

$$K - \sum_{i=1}^{t-1} y_i.$$

Si suman todas sus partes, ellos pueden calcular el valor de $K - y_i$. Sin embargo, no conocen el valor aleatorio de y_i, por lo que no pueden obtener el valor de K. En consecuencia, se tiene un (t, t)—esquema umbral.

1.2 ESTRUCTURAS DE ACCESO Y COMPARTICIÓN GENERAL SECRETA

Considerese el siguiente ejemplo: en un banco, hay una bóveda que debe ser abierta cada día. El banco emplea tres cajeros de alto nivel y dos elementos de seguridad, pero no confía, de manera particular, la totalidad de la combinación para abrir la bóveda, a ninguno de ellos. Sin embargo, el banco desea un sistema de seguridad, donde dos de los tres cajeros de alto nivel, junto con uno de los elementos de seguridad, tengan acceso a la bóveda, ¿cómo se puede llevar a cabo dicho sistema de seguridad de modo que con menos de los elementos necesarios no se abra la bóveda?

En el esquema umbral de Shamir, se deseaba que cualesquiera t de w participantes fueran capaces de calcular a la llave, sin embargo, esta cualidad no es suficiente para resolver el problema planteado anteriormente. Para resolver dicho problema se deben especificar, exactamente, cuáles subconjuntos de participantes son capaces de calcular a la llave y cuáles no.

Sea $\Gamma = \{B \subseteq \mathcal{P} \mid$ Los participantes de B, al juntar sus partes, son capaces de calcular la llave$\}$. Γ es llamada una *estructura de acceso* y los elementos de Γ *subconjuntos autorizados*. Sea $\mathcal{K}$ el conjunto de todas las posibles llaves y sea S el conjunto de todas las posibles partes. Como se vio previamente, cuando un distribuidor D quiere compartir una llave $K \in \mathcal{K}$, este da a cada participante una *parte* de S. Porteriormente, un subconjunto de participantes tratará de calcular a K con las partes que posee.

Definición 1.3. *Un esquema de compartición de secretos perfecto para una estructura de acceso* Γ *es un método para compartir una llave* K *entre un conjunto de* w *participantes (denotado por* $\mathcal{P}$*), de tal manera que se satisfacen las siguientes dos propiedades.*

i) *Si los participantes de un subconjunto autorizado* $B \subseteq \mathcal{P}$ *juntan sus partes; entonces, ellos pueden determinar el valor de* K*.*

ii) *Si los participantes de un subconjunto no autorizado* $B \subseteq \mathcal{P}$ *juntan sus partes; entonces, ellos no pueden determinar nada acerca del valor de K.*

Estas dos propiedades son equivalentes a decir que si $B \in \Gamma$, entonces el conjunto de participantes B puede calcular el valor de K; y, si $B \notin \Gamma$ no puede obtener información alguna acerca de K. Es importante mencionar que la seguridad del esquema debe ser incondicional, es decir, no debe depender de la cantidad de cálculos que puedan llevarse a cabo.

Supongamos que $B \in \Gamma$ y $B \subseteq C \subseteq \mathcal{P}$ y que el subconjunto C quiere determinar el valor de la llave K. Dado que B es un subconjunto autorizado, este puede calcular a K. Entonces, el subconjunto C puede calcular a K, ignorando las partes de los participantes que están en $C \setminus B$. Dicho de otra forma: un subconjunto que contenga a un conjunto autorizado es nuevamente un conjunto autorizado. Esto significa que la estructura de acceso debe satisfacer la propiedad *monótona*:

$$\text{Si } B \in \Gamma \text{ y } B \subseteq C \subseteq \mathcal{P}, \text{ entonces } C \in \Gamma.$$

Obsérvese que a un (t, w)—esquema umbral corresponde la estructura de acceso

$$\{B \subseteq \mathcal{P} : |B| \geqslant t\}.$$

Dicha estructura de acceso es llamada *estructura de acceso umbral*. Nótese que un esquema umbral de Shamir es también un esquema perfecto para una estructura de acceso umbral. En el resto de este capítulo, se asumirá que todas las estructuras de acceso son monótonas.

Si Γ es una estructura de acceso, entonces $B \in \Gamma$ es un subconjunto minimal autorizado, si $A \notin \Gamma$ para todo $A \subseteq B$, $A \neq B$. La familia de subconjuntos minimales de Γ es denotado por Γ_0 y es llamado la *base* de Γ. Dado que Γ consiste de todos los subconjuntos de P que son conjuntos que contienen a un subconjunto en la base Γ_0, Γ está determinado de manera única en función de Γ_0. Expresado matemáticamente, se tiene que

$$\Gamma = \{C \subseteq \mathcal{P} : B \subseteq C, B \in \Gamma_0\}.$$

Se dice que Γ es la *cerradura* de Γ_0 y se escribe

$$\Gamma = \mathrm{cl}(\Gamma_0).$$

Ejemplo 1.3. *Sea* $\mathcal{P} = \{P_1, P_2, P_3, P_4\}$ *y* $\Gamma_0 = \{\{P_1, P_2, P_4\}, \{P_1, P_3, P_4\}, \{P_2, P_3\}\}$*, entonces*

$$\Gamma = \Gamma_0 \cup \{\{P_1, P_2, P_3\}, \{P_2, P_3, P_4\}, \{P_1, P_2, P_3, P_4\}\}.$$

Por otro lado, Γ *es una familia de subconjuntos minimales (por la propiedad monótona), por lo que si se quisiera calcular a* Γ_0 *a partir de* Γ*, solo restaría verificar cuáles subconjuntos son minimales y cuáles no.*

En este ejemplo, dado que

$$\Gamma = \{\{P_1, P_2, P_4\}, \{P_1, P_3, P_4\}, \{P_2, P_3\}, \{P_1, P_2, P_3\}, \{P_2, P_3, P_4\}, \{P_1, P_2, P_3, P_4\}\},$$

obsérvese que a partir de la cardinalidad de los elementos de Γ*, es claro que no existe un elemento de* Γ *que esté contenido propiamente en* $\{P_2, P_3\}$*, por lo que,* $\{P_2, P_3\}$ *es un subconjunto minimal autorizado. Además, véase que* $\{P_2, P_3\} \subset \{P_1, P_2, P_3\}$*,* $\{P_2, P_3\} \subset \{P_2, P_3, P_4\}$ *y* $\{P_2, P_3\} \subset \{P_1, P_2, P_3, P_4\}$*, por lo que* $\{P_1, P_2, P_3\}$*,* $\{P_2, P_3, P_4\}$ *y* $\{P_1, P_2, P_3, P_4\}$ *no son subconjuntos minimales. Ahora, obsérvese que los subconjuntos propios de* $\{P_1, P_2, P_4\}$ *son* $\{P_1\}, \{P_2\}, \{P_4\}, \{P_1, P_2\}, \{P_1, P_4\}, \{P_2, P_4\}$*, los cuales no son subconjuntos autorizados, es decir, no están en* Γ*, por lo que* $\{P_1, P_2, P_4\}$ *es un subconjunto minimal autorizado. Análogamente, se verifica que* $\{P_1, P_3, P_4\}$ *también lo es. Por tanto, los subconjuntos minimales autorizados coinciden con los elementos de* Γ_0*.*

1.3 CIRCUITOS MONÓTONOS

1.3.1 Circuitos, polinomios y su representación Booleana

En 1854, George Boole introdujo una clase de estructuras algebraicas en conexión con su investigación en lógica matemática. Su objetivo era encontrar un modelo matemático para el razonamiento humano. En su honor, estas estructuras se han llamado álgebras Booleanas, que son un tipo especial de retículos. E. Schroder, en 1890, consideró el concepto de retículo en el sentido actual. Por esa época, R. Dedekind desarrolló un concepto similar en su trabajo sobre grupos e ideales. Dedekind definió, en la terminología moderna, los retículos modulares y distributivos, que son de suma importancia en las aplicaciones. El desarrollo avanzado de la teoría de retículos comenzó propiamente alrededor de 1930, cuando G. Birkhoff hizo contribuciones importantes a la teoría.

Antes de hablar de circuitos, polinomios y su representación Booleana es importante mencionar la conexión entre los mismos. Para ello, se iniciará con las definiciones básicas que permiten, en principio, definir al álgebra Booleana y, con esta información, fundamentar el por qué de la validez de sus aplicaciones en el estudio de circuitos Booleanos. Una vez establecida dicha conexión, se pondrá en práctica el uso de la misma.

Definición 1.4. *Un conjunto parcialmente ordenado* $(L, \leqslant)$ *es llamado retículo si para cada par* x, y *de elementos en* L*, su supremo y su ínfimo existe.*

Existe un enfoque equivalente el cual no usa relaciones de orden, si no operaciones algebraicas.

Definición 1.5. *Un retículo algebraico* $(L, \wedge, \vee)$ *es un conjunto* L *con dos operaciones binarias* $\wedge$ *(conjunción)* y $\vee$ *(unión), también llamadas intersección o producto y unión o suma, respectivamente. El cual satisface las siguientes leyes para todo* $x, y, z \in L$*:*

(L1) $x \wedge y = y \wedge x$, $x \vee y = y \vee x$

(L2) $x \wedge (y \wedge z) = (x \wedge y) \wedge z$, $x \vee (y \vee z) = (x \vee y) \vee z$

(L3) $x \wedge (x \vee y) = x$, $x \vee (x \wedge y) = x$.

Dos aplicaciones de (L3), a saber, $x \wedge x = x \wedge (x \vee (x \wedge x)) = x$, conducen a la siguiente ley adicional:

(L4) $x \wedge x = x$, $x \vee x = x$.

(L1) es la ley conmutativa; (L2), ley asociativa; (L3), ley de absorción y (L4), ley de idempotencia.

La conexión entre retículos ordenados y retículos algebraicos se muestra a continuación.

Teorema 1.3. (i) *Sea* $(L, \leqslant)$ *un retículo ordenado, si se define*

$$x \wedge y := \inf(x, y), \qquad x \vee y := \sup(x, y),$$

entonces $(L, \wedge, \vee)$ *es un retículo algebraico.*

(ii) *Sea* $(L, \wedge, \vee)$ *un retículo algebraico. Si se define*

$$x \leqslant y \Leftrightarrow x \wedge y = x \quad (\textit{o bien,} \quad x \leqslant y \Leftrightarrow x \vee y = y),$$

entonces $(L, \leqslant)$ *es un retículo ordenado.*

Demostración. (i) Sea $(L, \leqslant)$ un retículo ordenado. Para cada $x, y, z \in L$ se tiene que:

(L1) $x \wedge y = \inf(x, y) = \inf(y, x) = y \wedge x$,, $x \vee y = \sup(x, y) = \sup(y, x) = y \vee x$.

(L2) $x \wedge (y \wedge z) = x \wedge \inf(y, z) = \inf(x, \inf(y, z)) = \inf(x, y, z) = \inf(\inf(x, y), z) = \inf(x, y) \wedge z = (x \wedge y) \wedge z$,
y similarmente se prueba que $x \vee (y \vee z) = (x \vee y) \vee z$.

(L3) $x \wedge (x \vee y) = x \wedge \sup(x, y) = \inf(x, \sup(x, y)) = x$,
$x \vee (x \wedge y) = x \vee \inf(x, y) = \sup(x, \inf(x, y)) = x$.

(L4) $x \wedge x = \inf(x, x) = x$,
$x \vee x = \sup(x, x) = x$.

(ii) Sea $(L, \wedge, \vee)$ un retículo algebraico. Claramente, para todo $x, y, z \in L$:

- $x \wedge x = x$ por (L4); por lo que $x \leqslant x$, es decir, $\leqslant$ es reflexiva.
- Si $x \leqslant y$ y $y \leqslant x$, entonces $x \wedge y = x$ y $y \wedge x = y$ y por (L1) $x \wedge y = y \wedge x$, por lo que $x = y$, es decir, $\leqslant$ es antisimétrica.
- Si $x \leqslant y$ y $y \leqslant z$, entonces $x \wedge y = x$ y $y \wedge z = y$ por lo que
$x = x \wedge y = x \wedge (y \wedge z) = (x \wedge y) \wedge z = x \wedge z$, luego $x = x \wedge z$, de ahí que $x \leqslant z$ (L2), es decir, $\leqslant$ es transitiva.

Sea $x, y \in L$. $x \wedge (x \vee y) = x$ implica que $x \leqslant x \vee y$ y similarmente $y \leqslant x \vee y$. Si $z \in L$ con $x \leqslant z$ y $y \leqslant z$, entonces $(x \vee y) \vee z = x \vee (y \vee z) = x \vee z = z$ y $x \vee y \leqslant z$. Así que $\sup(x, y) = x \vee y$. Similarmente, $\inf(x, y) = x \wedge y$. Por lo tanto, $(L, \leqslant)$ es un retículo ordenado. □

Definición 1.6. *Si un retículo contiene al elemento más pequeño (o más grande) respecto a* $\leqslant$*, entonces este elemento determinado de manera única es llamado el elemento cero (o elemento unidad, respectivamente), denotado por* 0 *(o por* 1*). Los elementos* 0 *y* 1 *son llamados las cotas universales. Si ellos existen se dice que* L *es acotado.*

Todo retículo finito L es acotado. Si un retículo es acotado (por 0 y 1), entonces para todo $x \in L$ se satisface que $0 \leqslant x \leqslant 1$, $0 \wedge x = 0$, $0 \vee x = x$, $1 \wedge x = x$, $1 \vee x = 1$.

Los retículos Booleanos (o el álgebra Booleana) pueden ser descritos como los retículos más ricos y, al mismo tiempo, los más importantes para las aplicaciones. Puesto que se definen como retículos distributivos y complementados; es natural considerar primero la definición de los mismos y algunas de sus propiedades.

Definición 1.7. *Un retículo* L *es llamado distributivo si las leyes*

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z),$$

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z),$$

se satisfacen para todo $x, y, z \in L$. *Estas igualdades son llamadas leyes distributivas.*

Teorema 1.4. *Un retículo* L *es distributivo si y solo si la regla de cancelación*

$$x \wedge z = y \wedge z, x \vee z = y \vee z \Rightarrow x = y$$

se cumple para todo $x, y, z \in L$

Demostración. $\Rightarrow$) Supóngase que el retículo es distributivo y sean $x, y, z \in L$ tales que $x \wedge z = y \wedge z$ y $x \vee z = y \vee z$. Entonces:

$$\begin{aligned}
x &= x \wedge (x \vee z) && \text{(por absorción)} \\
&= x \wedge (y \vee z) && \text{(por hipótesis)} \\
&= (x \wedge y) \vee (x \wedge z) && \text{(por distributividad)} \\
&= (x \wedge y) \vee (y \wedge z) && \text{(por hipótesis)} \\
&= y \wedge (x \vee z) && \text{(por distributividad)} \\
&= y \wedge (y \vee z) && \text{(por hipótesis)} \\
&= y && \text{(por absorción)}
\end{aligned}$$

$\Leftarrow$) La segunda parte de la demostración se puede consultar a detalle en la referencia [CC16, cap. 2]. □

Definición 1.8. *Un retículo* L *con* 0 *y* 1 *es llamado complementado, si para cada* $x \in L$ *existe al menos un elemento* y *tal que* $x \wedge y = 0$ *y* $x \vee y = 1$. *Dicho elemento* y *es llamado complemento de* x.

Teorema 1.5. *Si* L *es un retículo distributivo con* 0 *y* 1, *entonces cada* $x \in L$ *tiene a lo más un complemento.*

Demostración. Suponga que $x \in L$ tiene dos complementos y_1 y y_2. Entonces $x \vee y_1 = 1 = x \vee y_2$ y $x \wedge y_1 = 0 = x \wedge y_2$; luego $y_1 = y_2$ por el teorema 1.4 □

Definición 1.9. *Un retículo distributivo complementado, es llamado retículo Booleano o un álgebra Booleana.*

Un álgebra Booleana en la que se tiene especial interés es $(\mathbb{B}, \wedge, \vee, 0, 1, ')$, donde $\mathbb{B}$ denotará al conjunto cuyos elementos son: el elemento cero 0 y el elemento unidad 1. Para el cual se definen dos operaciones binarias $\wedge$ y $\vee$, además de una operación unaria de complementación $'$, como sigue:

$\wedge$	0	1
0	0	0
1	0	1

$\vee$	0	1
0	0	1
1	1	1

	$'$
0	1
1	0

Definición 1.10. *Sea* $X = \{x_1, \ldots, x_n\}$ *un conjunto de* n *símbolos (llamados indeterminantes o variables), los cuales son distintos a los símbolos* 0 *y* 1. *Los polinomios Booleanos en* X *son* $x_1, x_2, \ldots, x_n, 0$ *y* 1 *y aquellos que se pueden obtener por un número finito de aplicaciones sucesivas de:*

- *Si* p *y* q *son polinomios Booleanos, entonces* $p \wedge q$, $p \vee q$ *y* p' *también lo son*

Se dice que dos polinomios son iguales si la secuencia de sus símbolos es idéntica. Bajo esta definición nótese que no siempre será cierto que $x_1 \wedge x_2$ sea igual a $x_2 \wedge x_1$.

Ejemplo 1.4. *Algunos polinomios Booleanos de* $\{x_1, x_2\}$ *son* $0, 1, x_1, x_1 \vee 1, x_1 \wedge x_2, x_1 \vee x_2, x_1', x_1' \wedge x_2, x_1' \wedge (x_2 \vee x_1)$.

Al conjunto de todos los polinomios Booleanos en $\{x_1, \ldots, x_n\}$ se le denotará por P_n. Dado que cada polinomio Booleano sobre $\{x_1, x_2, \ldots, x_n\}$ puede ser considerado como un polinomio Booleano sobre $\{x_1, x_2, \ldots, x_n, x_{n+1}\}$, se tiene que

$$P_1 \subset P_2 \subset \cdots \subset \cdots P_n \subset P_{n+1} \subset \cdots$$

Nótese que P_n no es un álgebra Booleana, es más, no es un retículo algebraico ya que $x_1 \wedge x_2$ no siempre es lo mismo que $x_2 \wedge x_1$, violando el apartado L1) de su definición. A continuación se introduce el concepto de función polinomial como sigue:

Definición 1.11. *Sea A un álgebra Booleana, sea A^n un producto directo de n copias de A , y sea p un polinomio Booleano en P_n. Entonces:*

$$\begin{aligned} \overline{p}_A : A^n &\longrightarrow A; \\ (a_1, \ldots, a_n) &\longrightarrow \overline{p}_A(a_1, \ldots, a_n) \end{aligned}$$

es llamada una función polinomial Booleana inducida por p en A. Aquí $\overline{p}_A(a_1, \ldots, a_n)$ es el elemento en A que se obtiene de p, reemplazando cada x_i por $a_i \in A, 1 \leqslant i \leqslant n$.

El siguiente ejemplo muestra que dos polinomios Booleanos diferentes pueden tener la misma función polinomial Booleana. Nuevamente, $\mathbb{B}$ denota al álgebra Booleana $\{0, 1\}$ con las operaciones usuales.

Ejemplo 1.5. *Sea $n = 2$, $p(x_1, x_2) = x_1 \wedge x_2$, $q(x_1, x_2) = x_2 \wedge x_1$. Entonces*

$$\begin{aligned} \overline{p}_{\mathbb{B}} : \mathbb{B}^2 &\longrightarrow \mathbb{B} : \\ (0,0) &\longrightarrow 0, \\ (0,1) &\longrightarrow 0, \\ (1,0) &\longrightarrow 0, \\ (1,1) &\longrightarrow 1 \end{aligned} \qquad \begin{aligned} \overline{q}_{\mathbb{B}} : \mathbb{B}^2 &\longrightarrow \mathbb{B} : \\ (0,0) &\longrightarrow 0 \\ (0,1) &\longrightarrow 0 \\ (1,0) &\longrightarrow 0 \\ (1,1) &\longrightarrow 1 \end{aligned}$$

Por tanto $\overline{p}_{\mathbb{B}} = \overline{q}_{\mathbb{B}}$.

Sea A un álgebra Booleana. Si se usa la notación introducida anteriormente, se define:

Definición 1.12. $P_n(A) = \{\overline{p}_A \mid p \in P_n\}$

Definición 1.13. *Dos polinomios Booleanos p y q son equivalentes (en símbolos $p \sim q$), si sus funciones polinomiales Booleanas en $\mathbb{B}$ son iguales, es decir,*

$$p \sim q \iff \overline{p}_{\mathbb{B}} = \overline{q}_{\mathbb{B}}$$

El propósito principal del álgebra de circuitos booleanos es describir circuitos booleanos electrónicos en una forma matemática o diseñar un diagrama de un circuito con propiedades dadas. Aquí, se combinan interruptores electrónicos en circuitos de conexión en serie y en circuitos de conexión paralela. Tales interruptores son elementos que pueden cambiar de estado (abierto-cerrado); por ejemplo, un contacto eléctrico. En cuanto a los dos estados mencionados, su naturaleza dependerá de sus elementos de cambio; estos se consideran elementos conductores-no conductores, cargados-descargados, etc.

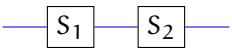


Figura 1: Conexión en serie.

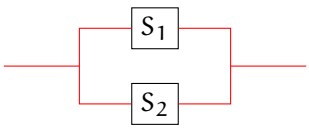


Figura 2: Conexión paralela.

Dado que un *switch* tiene dos estados, "abierto" o "cerrado", esto puede ser simbolizado de la siguiente manera:

Figura 3: Circuito abierto y circuito cerrado.

Si se parte del supuesto básico, para que la corriente fluya a través de un interruptor o para que un contacto se establezca, es necesario que el interruptor esté cerrado. El símbolo $\boxed{S_i'}$ (complemento) indica que un interruptor está abierto, si y solo si $\boxed{S_i}$ (un interruptor que aparecerá en otra parte del circuito) está cerrado; en otras palabras, S_i' y S_i están relacionados de forma que cuando uno está abierto, el otro está cerrado. Si S_i aparece en dos lugares separados en el circuito, significa que hay dos interruptores vinculados, para garantizar que están ambos abiertos o ambos cerrados. De este modo, se puede observar que en la figura 1 se tendrá corriente, si y solo si ambos, S_1 y S_2, están cerrados. Mientras que, en la figura 2 se tendrá corriente, si y solo si al menos uno de S_1 y S_2 está cerrado.
Estas propiedades eléctricas motivan la siguiente definición, que proporciona la conexión entre interruptores eléctricos y elementos de un álgebra Booleana.

Definición 1.14. *Sea* $X_n := \{x_1, \ldots, x_n\}$

i) *Cada* $x_1, \ldots, x_n \in X_n$ *es llamado interruptor.*

ii) *Cada* $p \in P_n$ *es llamado un circuito booleano.*

iii) x_i' *es llamado el complemento booleano de* x_i.

iv) $x_i x_j$ *es llamada una conexión en serie de* x_i *y* x_j.

v) $x_i + x_j$ *es llamada una conexión paralela de* x_i *y* x_j.

vi) *Para* $p \in P_n$, *la correspondiente función polinomial* $\overline{p} \in P_n(\mathbb{B})$ *es llamada la función booleana de p.*

vii) $\overline{p}(a_1, \ldots, a_n)$ *es llamado el valor del circuito booleano* p *evaluado en* $a_1, \ldots, a_n \in \mathbb{B}$. *Las* a_i'*s son llamadas variables de entrada.*

Los circuitos booleanos pueden ser representados gráficamente por diagramas. Para ello, se hará uso de la representación de conexiones en serie y paralelas propuesta en las figuras 1 y 2, y una combinación de las mismas. Por ejemplo, el circuito representado por el polinomio $x_1x_2 + x_1x_3$ puede ser representado como:

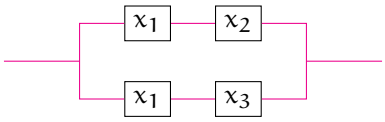
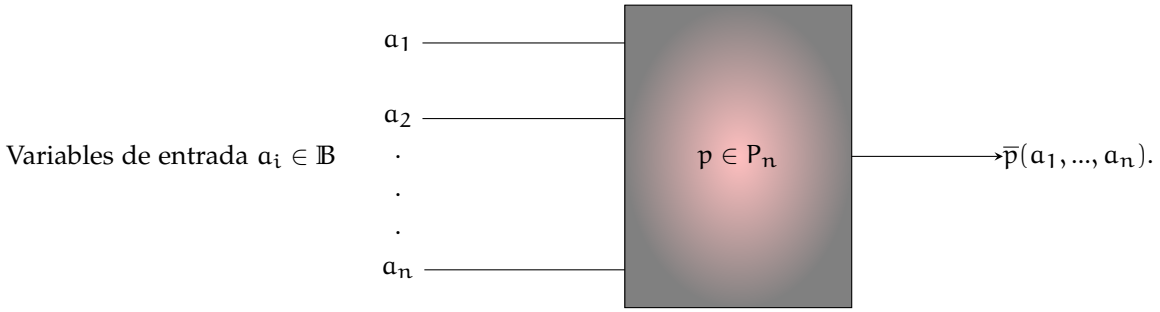
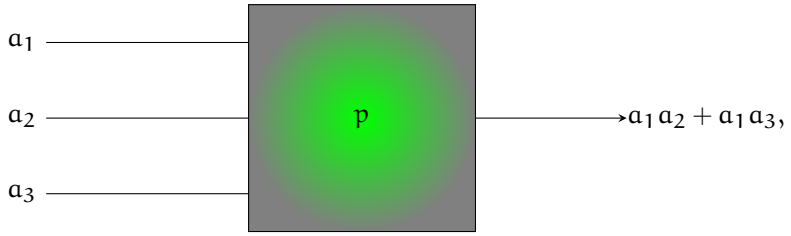


Figura 4: Circuito representado por el polinomio $x_1x_2 + x_1x_3$.

Otra forma de representar a los circuitos booleanos es por medio de una caja que convierte a las variables de entrada en un valor:



Para el ejemplo anterior, se tiene el siguiente diagrama:



donde $\overline{p}(a_1, ..., a_n) = 1$ ($o = 0$) significa que el circuito p conduce corriente (o no conduce corriente).

Es importante mencionar que los circuitos eléctricos pueden ser modelados mediante el uso de polinomios booleanos y se dice que circuitos eléctricos diferentes funcionan idénticamente, si sus valores de salida son los mismos para todas las posibles combinaciones en sus variables de entrada. Esto significa que para los correspondientes polinomios p, q, $\overline{p}_{\mathbb{B}} = \overline{q}_{\mathbb{B}}$, esto es, $p \sim q$. Esta idea es fundamental, porque muchas veces se desea encontrar una simplificación a un circuito eléctrico que conserve las propiedades del circuito original, pero que sea más sencillo de estudiar. El método sugerido en esta ocasión, para llevar a cabo dicha simplificación, es representar al circuito por medio de un polinomio booleano y aplicar la teoría estudiada para minimizar polinomios de esa naturaleza.

A continuación, se profundizará en el estudio de la simplificación de los polinomios Booleanos, especialmente, en la reducción de los mismos a una forma mínima, con respecto a una conveniente condición de minimilidad elegida. Las consideraciones serán restringidas a una condición mínima especial para expresiones tipo suma-de-productos.

Se define a una literal como cualquier variable x_i, sea complementada o no (0 o 1). Sea d_f el número total de literales diferentes en una expresión tipo suma-de-productos de f. Sea e_f el número de sumandos en dicha expresión. Se dice que f es más simple que una expresión suma-de-productos g si $e_f < e_g$, o si $e_f = e_g$ y $d_f < d_g$. A su vez, f es mínima si no existe una expresión más simple de suma-de-productos equivalente a f. En otras palabras, se busca la expresión más corta de suma-de-productos con el menor número posible de literales y que sea equivalente a f. Esta forma mínima no siempre es única.

A continuación, se describirá un método de simplificación. Este método se basa en el trabajo de Quine [LG98]. A partir de este momento, se utilizará el término "expresión" para hacer referencia a un polinomio Booleano.

Definición 1.15. *Una expresión* p *implica una expresión* q *si para todos los* $b_1, \ldots, b_n \in \mathbb{B}$, $\overline{p}_{\mathbb{B}}(b_1, \ldots, b_n) = 1 \Rightarrow \overline{q}_{\mathbb{B}}(b_1, \ldots, b_n) = 1$. *En este caso* p *es llamado un implicante de* q.

Definición 1.16. *Una expresión producto (brevemente producto)* α *es una expresión en la que* $+$ *no ocurre. Un implicante primo para una expresión* p *es una expresión producto* α *la cual implica* p*, pero no implica* p *si uno de sus factores en* α *es eliminado. Un producto cuyos factores forman un subconjunto de los factores de otro producto es llamado subproducto del primero.*

Ejemplo 1.6. x_1x_3 *es un subproducto de* $x_1x_2x_3$ *y de* $x_1x_2'x_3$. *También se tiene que* x_1x_3 *implica a la expresión* $p = x_1x_2x_3 + x_1x_2'x_3 + x_1'x_2'x_3'$, *ya que* $\overline{x_1x_3}(1,0,1) = 1$ *y* $\overline{p}(1,0,1) = 1$, $\overline{x_1x_3}(1,1,1) = 1$ *y* $\overline{p}(1,1,1) = 1$; $\overline{x_1x_3}$ *es igual a* 0 *para otros argumentos. Obsérvese que* $\overline{x_1}(1,1,0) = 1$ *pero* $\overline{p}(1,1,0) = 0$ *y que* $\overline{x_3}(0,1,1) = 1$ *pero* $\overline{p}(1,1,0) = 0$. *Por lo tanto, ni* x_1 *ni* x_3 *implican a* p. *Luego,* x_1x_3 *es un implicante primo.*

Teorema 1.6. *Un polinomio* $p \in P_n$ *es equivalente a la suma de todos los implicantes primos de p.*

Demostración. Sea q la suma de todos los implicantes primos p_a de p. Si $\overline{q}(b_1, \ldots, b_n) = 1$ para $(b_1, \ldots, b_n) \in \mathbb{B}^n$, entonces existe un implicante primo p_a para el cual $\overline{p}_a(b_1, \ldots, b_n) = 1$. Dado que p_a implica p, se tiene también que $\overline{p}(b_1, \ldots, b_n) = 1$. Recíprocamente, supongáse que $\overline{p}(b_1, \ldots, b_n) = 1$ y $s := x_1^{b_1} \cdots x_n^{b_n}$, donde s es un implicante de p. En s, se eliminan a todos los $x_i^{b_i}$ para los cuales $\overline{p}_a(b_1, \ldots, b_{i-1}, b_i', b_{i+1}, \ldots, b_n) = 1$. El producto que queda, r, aún implica a p, pero no implica a p si otro factor es eliminado. Por lo tanto, r es un implicante primo de p con $\overline{r}(b_1, \ldots, b_n) = 1$ y, finalmente, $\overline{q}(b_1, \ldots, b_n) = 1$. □

La suma de los implicantes primos de p es llamada *irredundante* si es equivalente a p, pero no lo es si se omite uno de sus sumandos. Una expresión suma-de-producto mínima debe ser irredundante. Por tanto, para calcular a la expresión mínima se determina al conjunto de expresiones irredundantes y entre ellas se busca a las que tengan menor número de literales.

Los implicantes primos se obtienen comenzando con la forma normal disyuntiva d para polinomios Booleanos p y aplicando la regla

$$yz + yz' \sim y$$

(de izquierda a derecha) cuando sea posible. En general, se usa

$$\alpha\beta\gamma + \alpha\beta'\gamma \sim \alpha\gamma \tag{3}$$

donde α, β, y γ son expresiones producto. El conjunto de todas las expresiones que no pueden ser más simplificadas con este procedimiento es equivalente al conjunto de implicantes primos. La suma de estos implicantes primos es equivalente a p, por el teorema 1.6.

Ejemplo 1.7. *Sea p el polinomio Booleano (ahora en w, x, y, z en lugar de x_1, x_2, x_3, x_4), cuya forma normal disyuntiva está dada por:*

$$p = wxyz' + wxy'z' + wx'yz + wx'yz' + w'xyz + w'x'yz + w'x'y'z$$

se usa la ley idempotente y la regla (3) para analizar a los $\binom{7}{2} = 21$ pares de productos en d (tan rápido como sea posible), de esta manera se busca acortar los productos. Por ejemplo, la primera y la segunda expresiones producto se simplifican a wxz', cuando se usa la regla (3). Si una expresión producto es usada una o más veces para alguna simplificación, es marcada. Dado que $+$ es idempotente, una expresión puede ser usada cualquier número de veces y una marca es suficiente. De este modo, se sabe que todas las expresiones marcadas contienen a otras expresiones subproducto que implican a p y que, por tanto, no pueden ser implicantes primos. El proceso conduce a las siguientes simplificaciones:

de $wxyz'$ *y* $wxy'z'$ *a* wxz'
de $wx'yz$ *y* $wx'yz'$ *a* $wx'y$
de $wxyz'$ *y* $wx'yz'$ *a* wyz'
de $w'x'yz$ *y* $w'x'yz'$ *a* $w'x'y$
de $w'x'yz$ *y* $w'x'y'z$ *a* $w'x'z$
de $wx'yz$ *y* $w'x'yz$ *a* $x'yz$
de $wx'yz'$ *y* $w'x'yz'$ *a* $x'yz'$

en este caso, todas las expresiones han sido sometidas a simplificación y deben ser marcadas. Con las expresiones obtenidas a partir de este procedimiento se lleva a cabo la segunda vuelta de simplificación:

de $wx'y$ *y* $w'x'y$ *a* $x'y$
de $x'yz$ *y* $x'yz'$ *a* $x'y$

las expresiones $wx'y$, $w'x'y$,$x'yz$ y $x'yz'$, que se usaron para esta última simplificación, se marcan y se concluye que las otras expresiones wxz', wyz' y $w'x'z$, al igual que la expresión obtenida $x'y$, no pueden ser simplificadas. Por tanto, p es equivalente a la suma de sus implicantes primos:

$$p \sim wxz' + wyz' + w'x'z + x'y$$

Ejemplo 1.8. *Sea p el polinomio Booleano cuya forma normal disyuntiva está dada por:*

$$p = w'x'yz' + w'x + wx'z' + wz' + w'xyz' + w'yz + wz + wx + wx'z + wxyz'$$

se usa la regla (3) para llevar a cabo las siguientes simplificaciones:

de $w'x'yz'$ *y* $w'xyz'$ *a* $w'yz'$
de $w'x$ *y* wx *a* x
de $wx'z'$ *y* $wx'z$ *a* wx'
de wz' *y* wz *a* w
de $w'xyz'$ *y* $wxyz'$ *a* xyz'

y se marcan a las expresiones que se han utilizado:

$$\begin{array}{lll} p_1 = & w'x'yz' & \surd \\ p_2 = & w'x & \surd \\ p_3 = & wx'z' & \surd \\ p_4 = & wz' & \surd \\ p_5 = & w'xyz' & \surd \\ p_6 = & w'yz & \\ p_7 = & wz & \surd \\ p_8 = & wx & \surd \\ p_9 = & wx'z & \surd \\ p_{10} = & wxyz' & \surd \end{array}$$

Obsérvese que la expresión p_6 no ha sido simplificada y no se puede concluir que no sea implicante primo. Posteriormente, la expresión p_6 será analizada en conjunto con las expresiones obtenidas de la simplificación anterior. La segunda vuelta arroja el siguiente resultado:

de $w'yz$ *y* $w'yz'$ *a* $w'y$

y se marcan las expresiones utilizadas:

$$\begin{array}{lll} p_6 = & w'yz & \surd \\ q_1 = & w'yz' & \surd \\ q_2 = & x & \\ q_3 = & wx' & \\ q_4 = & w & \\ q_5 = & xyz' & \end{array}$$

las expresiones restantes ya no se pueden simplificar por lo que se han encontrado a todos los implicantes primos. Se concluye que:

$$p \sim w'y + x + wx' + w + xyz'$$

McCluskey mejoró este método que conduce al algoritmo general Quine-McCluskey [LG98]. A continuación, se hará uso del ejemplo 1.7 para describirlo.

i) Se representan todas las expresiones producto en términos de ceros y unos, tales que x_i' y x_i son denotadas por 0 y 1, respectivamente. Las variables faltantes son indicadas con un guión. Por ejemplo, $w'x'y'z$ es 0001 y $w'x'z$ es $00-1$.

ii) Las expresiones producto, consideradas como n-tuplas binarias, son particionadas en clases. La partición se lleva a cabo de acuerdo a la cantidad de unos que tienen, de menor a mayor. En el ejemplo:

$w'x'y'z$	0001
$w'x'yz'$	0010
$w'x'yz$	0011
$wx'yz'$	1010
$wxy'z'$	1100
$wx'yz$	1011
$wxyz'$	1110

iii) Este paso busca aplicar la regla (3) para simplicar expresiones con la clase de equivalencia vecina. Para llevar a cabo este proceso se compara a cada una de las expresiones con n unos con cada una de las expresiones con $n+1$ unos que tengan guiones en la misma posición. Si dos de esas expresiones difieren en exactamente una posición, entonces son de la forma $p_a = i_1, i_2, \ldots, i_r, \ldots, i_n$ y $p_b = i_1, i_2, \ldots, i_r', \ldots, i_n$, en donde todos los $i_k \in \{0, 1, -\}$, e $i_r \in \{0, 1\}$, respectivamente. Entonces, la regla (3) reduce p_a y p_b a $i_1, i_2, \ldots, i_{r-1}, i_{r+1}, \ldots, i_n$, y p_a y p_b son marcadas. Obsérvese que las expresiones marcadas no son implicantes primos, por lo que son sujetas a la reducción antes mencionada y en un futuro descartadas.

En el ejemplo, se tienen a las siguientes expresiones:

$p_1 = 0001$ $p_2 = 0010$
$p_3 = 0011$ $p_4 = 1010$ $p_5 = 1100$
$p_6 = 1011$ $p_7 = 1110$

En este caso, las expresiones no tienen guiones por lo que se tendrá que comparar a cada una de las expresiones con n unos con cada una de las expresiones con $n+1$ unos, para encontrar a las que difieran en una sola posición y poder aplicar la regla (3). Se procede a comparar a las expresiones con un uno con las expresiones con dos unos. $p_1 = 0001$ y $p_3 = 0011$ difieren en una sola posición por lo que se aplica la regla (3), las expresiones p_1 y p_3 son reducidas a $00-1$ y p_1 y p_3 son marcadas. Posteriormente, se compara a la expresión $p_1 = 0001$ con la expresión $p_4 = 1010$, pero se observa que difieren en más de una posición, por lo que no es posible aplicar la regla (3). El mismo resultado se obtiene al comparar p_1 con p_5.

Hasta ahora se ha obtenido a la expresión $00-1$ y solo p_1 y p_3 han sido marcadas:

$p_1 = 0001$ $p_2 = 0010$	$\surd$
$p_3 = 0011$ $p_4 = 1010$ $p_5 = 1100$	$\surd$
$p_6 = 1011$ $p_7 = 1110$	

Se prosigue a comparar a la expresión p_2 con p_3, p_4 y p_5, de donde se obtienen a las expresiones $001-$ y -010, por lo que p_2, p_3 y p_4 son marcadas. En este caso p_2 y p_5 difieren en más de una posición.

$p_1 = 0001$ $p_2 = 0010$	$\surd$ $\surd$
$p_3 = 0011$ $p_4 = 1010$ $p_5 = 1100$	$\surd$ $\surd$
$p_6 = 1011$ $p_7 = 1110$	

El siguiente paso es comparar cada una de las expresiones con 2 unos con cada una de las expresiones con 3 unos.

En el caso de que al final del proceso existieran expresiones comparadas que no hayan sido marcadas, estas y las expresiones obtenidas bajo reducción deben ser ordenadas nuevamente como en el inciso ii) y comparadas nuevamente, hasta que no haya reducción posible; es decir, hasta que todas las expresiones sean implicantes primos.

En el ejemplo, se tiene que todas las expresiones p_i con $i \in \{1, 2, \ldots, 7\}$ han sido marcadas por lo que son descartadas. La tabla de las expresiones obtenidas y ya clasificadas de acuerdo con el inciso ii) queda de la siguiente manera:

$q_1 = 00-1$ $q_2 = 001-$ $q_3 = -010$
$q_4 = -011$ $q_5 = 101-$ $q_6 = 1-10$ $q_7 = 11-0$

En este caso, las expresiones $q_2 = 001-$ y $q_5 = 101-$, $q_3 = -011$ y $q_4 = -010$ son las únicas expresiones con el guión en la misma posición y que difieren en exactamente una posición, por lo que son marcadas y sometidas a una futura reducción:

$q_1 = 00-1$	
$q_2 = 001-$	$\surd$
$q_3 = -010$	$\surd$
$q_4 = -011$	$\surd$
$q_5 = 101-$	$\surd$
$q_6 = 1-10$	
$q_7 = 11-0$	

Las expresiones q_2 y q_5, q_3 y q_4, conducen de igual forma a la expresión $-01-$.

Obsérvese que las expresiones restantes (las expresiones que no fueron marcadas y la última expresión obtenida) ya no pueden ser sometidas a reducción, por lo que se concluye que se han encontrado a todos los implicantes primos:

$00-1$	$w'x'z$
$1-10$	wyz'
$11-0$	wxz'
$-01-$	$x'y$

La suma de los implicantes primos no necesariamente es la forma mínima (algunos sumandos pueden ser superfluos). A continuación, se llevará a cabo el último paso del procedimiento.

iv) Dado que la suma de todos los implicantes primos de p es equivalente a p, por el teorema 1.6, para cada expresión producto en la forma normal disyuntiva d de p debe haber un implicante primo, el cual es un subproducto de esta expresión producto. Esto se determina mediante el establecimiento de una tabla de implicantes primos.

Los elementos del encabezado para las columnas son las expresiones producto en d, al comienzo de las filas están los implicantes primos calculados en el paso iii). Se pone un $\times$ en la intersección de la i-ésima fila y j-ésima columna, si el implicante primo en la i-ésima fila es un subproducto de la expresión producto en la j-ésima columna. Se dice que una expresión producto cubre a otra expresión producto, si es un subproducto de la última.

Con el fin de encontrar una suma mínima de implicantes primos que sea equivalente a d, se elige a un subconjunto del conjunto de implicantes primos; de tal manera que, cada expresión producto de d esté cubierta por al menos un implicante primo del subconjunto. Un implicante primo es llamado *término principal*, cuando cubre a una expresión producto que no está cubierta por ningún otro implicante primo. La suma de los términos principales es llamada núcleo.

Si los sumandos del núcleo en conjunto cubren a todas las expresiones producto de d, entonces, el núcleo es ya la (única) forma mínima de d. De otra forma, se denota por $q_1, \ldots, q_k$ a las expresiones producto sin cubrir (por el núcleo). Los implicantes primos que no están en el núcleo se denotan por $p_1, \ldots, p_m$ y se forma una segunda tabla con los elementos denotados q_j para las columnas y los elementos denotados p_i para las filas. La marca $\times$ es colocada en la entrada (i, j), que indica que p_i cubre a q_j. Entonces, se trata de encontrar el subconjunto mínimo de $p_1, \ldots, p_m$ que cubra a todos los $q_1, \ldots, q_k$ y se suma al núcleo. Si la segunda tabla es pequeña, entonces, el subconjunto mínimo se encuentra fácilmente, de lo contrario se tendrá que acudir a algún otro algoritmo de ayuda. Finalmente, esto conduce a la forma mínima de d, la cual no es necesariamente única.

En el ejemplo, se tiene la siguiente tabla de implicantes primos:

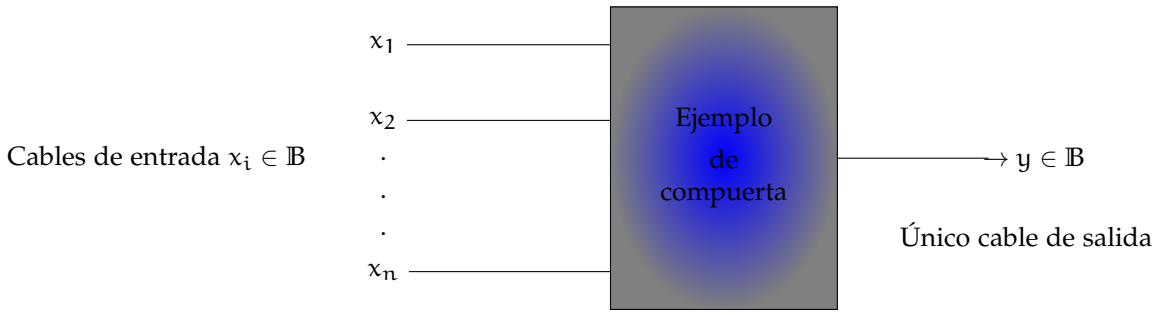
	0001	0010	0011	1010	1100	1011	1110
$00-1$	×		×				
$1-10$				×			×
$11-0$					×		×
$-01-$		×	×	×		×	

El núcleo está dado por la suma de los implicantes primos que son los únicos que cubren a un sumando en d, es decir, por la suma de $00-1$, $-01-$ y $11-0$. Esta suma ya cubre a todos los sumando en d, así que la forma mínima de d está dada por $w'x'z + y'z + wxz'$. Nótese que el implicante primo wyz' fue superficial.

1.3.2 La construcción del Circuito Monótono

A continuación se dará una construcción simple y elegante, cuyo planteamiento se debe a Benaloh y Leichter, que muestran que cualquier estructura monótona de acceso puede ser realizada por un esquema de compartición de secretos perfecto. La idea es construir primero un circuito monótono que "reconozca" la estructura de acceso y luego construir el esquema de compartición del secreto a partir de la descripción del circuito. A esto se le llama la *construcción del circuito monótono*.

Sea C un circuito booleano con w entradas $x_1, \ldots, x_w \in \mathbb{B}$ (correspondientes a los w participantes $P_1, \ldots, P_w$) y una salida booleana $y \in \mathbb{B}$. El circuito consiste de compuertas lógicas *or* y *and*; no permite ninguna compuerta tipo *not*. Tal circuito es llamado *circuito monótono booleano*. La razón para esta nomenclatura es que al cambiar cualquier entrada x_i de "0"(falso) a "1"(verdadero) nunca puede resultar que en la salida y cambie de "1" a "0". El circuito está diseñado para tener entradas arbitrarias y salidas iguales a 1. Es decir, cada compuerta pueda tener muchos cables de entrada pero solo un cable de salida:



Si se especifican los valores lógicos para las w entradas de dicho circuito monótono, se puede definir

$$B(x_1, \ldots, x_w) = \{P_i : x_i = 1\},$$

es decir, el subconjunto de $\mathcal{P}$ que corresponde a las entradas verdaderas. Supóngase que C es un circuito monótono, entonces

$$\Gamma(C) = \{B(x_1, \ldots, x_w) : C(x_1, \ldots, x_w) = 1\},$$

donde $C(x_1, \ldots, x_w)$ denota a la salida de C, dadas las entradas $x_1, \ldots, x_w$. Dado que el circuito C es monótono, se sigue que $\Gamma(C)$ es un conjunto monótono de subconjuntos de $\mathcal{P}$.

Si Γ es un conjunto monótono de subconjuntos de $\mathcal{P}$, entonces es fácil construir un circuito monótono C tal que $\Gamma(C) = \Gamma$. Un camino para realizar este procedimiento es el siguiente: Sea Γ_0 una base de Γ. Al hacer uso de la fórmula booleana en la forma normal disyuntiva

$$\bigvee_{B \in \Gamma_0} \Big(\bigwedge_{P_i \in B} \Big) P_i,$$

en el ejemplo 1.3, donde

$$\Gamma_0 = \{\{P_1, P_2, P_4\}, \{P_1, P_3, P_4\}, \{P_2, P_3\}\},$$

se obtiene la fórmula booleana:

$$(P_1 \wedge P_2 \wedge P_4) \vee (P_1 \wedge P_3 \wedge P_4) \vee (P_2 \wedge P_3). \tag{4}$$

Cada cláusula en la fórmula booleana corresponde a una compuerta *and* del circuito monótono asociado; la disyunción final corresponde a una compuerta *or*. El número de compuertas en el circuito es $|\Gamma_0| + 1$.

Supóngase que C es cualquier circuito monótono que reconoce a Γ (nótese que C no necesita ser el circuito descrito anteriormente). A continuación, se describe un algoritmo que permite a D, el distribuidor, construir un esquema de compartición de secretos perfecto correspondiente a Γ. Este esquema utilizará como base al (t,t)-esquema construido con el algoritmo 2. Por lo tanto, el conjunto de llaves será $\mathcal{K} = \mathbb{Z}_m$ para algún entero m.

El algoritmo procede a asignar un valor $f(W) \in \mathcal{K}$ a cada cable W en el circuito C. En un inicio, al cable de salida W_{out} del circuito se le asigna el valor de K, la llave. El algoritmo itera un número de veces, hasta que cada cable tiene un valor asignado. Finalmente, a cada participante P_i se les da la lista de valores $f(W)$, tales que W es un cable de entrada del circuito el cual recibe la entrada x_i. Una descripción de la construcción está dada en el algoritmo 3.

Algoritmo 3 Construcción del circuito monótono (C)

1: $f(W_{out}) \leftarrow K$
2: **while** Exista un cable W tal que $f(W)$ no esté definida **do**
3: Se encuentra una compuerta G de C tal que $f(W_G)$ esté definida, donde W_G es el cable de salida de G pero $f(W)$ no está definida para ningún cable de entrada de G.
4: **if** G es una compuerta "or" **then**
5: $f(W) \leftarrow f(W_G)$ para cada cable de entrada W de G
6: **else**
7: Se denotan a los cables de entrada de G como $W_1, ..., W_t$
8: Se eligen (aleatoriamente) $t-1$ elementos de $\mathbb{Z}_m$ denotados por $y_{G,1}, ..., y_{G,t-1}$
9: $y_{G,t} \leftarrow f(W_G) - \sum_{i=1}^{t-1} y_{G,i}$ *mod* m
10: **for** $i \leftarrow 1$ to t **do**
11: $f(W_i) \leftarrow y_{G,i}$
12: **end for**
13: **end if**
14: **end while**

Nótese que siempre que una compuerta G es una compuerta *and* con t cables de entrada, se compartirá la "llave" $f(W_G)$ de la compuerta G entre los cables de entrada, usando un (t,t)-esquema umbral.

A continuación, se llevará a cabo este procedimiento para la estructura de acceso del ejemplo 1.3, usando el circuito correspondiente a la fórmula booleana 4.

Ejemplo 1.9. *Supóngase que* K *es la llave. El valor de* K *es dado a cada uno de los tres cables de entrada de la última compuerta or. Luego se considera la compuerta and, correspondiente a la claúsula* $(P_1 \wedge P_2 \wedge P_4)$. *A estos tres cables de entrada se les asignan los valores* $a_1, a_2, K - a_1 - a_2$, *respectivamente, donde la aritmética se hace en* $\mathbb{Z}_m$. *De forma similar a los tres cables correspondientes a* $(P_1 \wedge P_3 \wedge P_4)$ *se les asignan los valores* $b_1, b_2, K - b_1 - b_2$. *Finalmente, a los dos cables de entrada correspondientes a* $(P_2 \wedge P_3)$ *se les asignan los valores* $c_1, K - c_1$. *Nótese que* a_1, a_2, b_1, b_2, c_1 *son valores independientes y aleatorios en* $\mathbb{Z}_m$. *Por lo tanto, se tiene lo siguiente:*

i) P_1 *recibe* $(y_1^1, y_1^2) = (a_1, b_1)$

ii) P_2 *recibe* $(y_2^1, y_2^2) = (a_2, c_1)$

iii) P_3 *recibe* $(y_3^1, y_3^2) = (b_2, K - c_1)$

iv) P_4 *recibe* $(y_4^1, y_4^2) = (K - a_1 - a_2, K - b_1 - b_2)$.

De este manera, cada participante recibe dos elementos de $\mathbb{Z}_m$ como su parte. A continuación la demostración de que el esquema es perfecto.

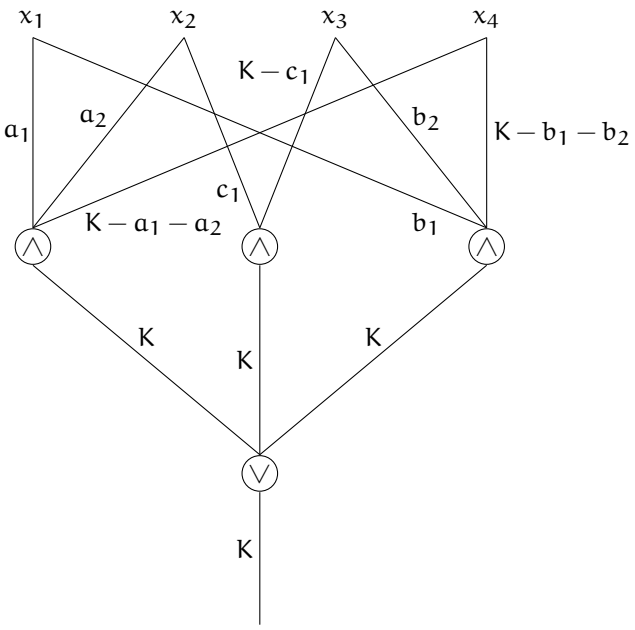


Figura 5: Circuito correspondiente al ejemplo 1.9

Demostración. Primero, se verifica que cada subconjunto de la base pueda calcular a K. El subconjunto autorizado $\{P_1, P_2, P_4\}$ puede calcular $y_1^1 + y_2^1 + y_4^1 \ mod\ m = a_1 + a_2 + (K - a_1 - a_2) \ mod\ m = K$. El subconjunto $\{P_1, P_3, P_4\}$ puede calcular $y_1^2 + y_3^1 + y_4^2 \ mod\ m = b_1 + b_2 + (K - b_1 - b_2) \ mod\ m = K$. Finalmente, el subconjunto $\{P_2, P_3\}$ puede calcular $y_2^2 + y_3^2 \ mod\ m = c_1 + (K - c_1) \ mod\ m = K$.

Por tanto, cada subconjunto autorizado es capaz de calcular a K. A continuación, se deben examinar los subconjuntos no autorizados. Nótese que no es necesario analizar a cada uno de los subconjuntos no autorizados. Por ejemplo, si B_1 y B_2 son ambos subconjuntos no autorizados, $B_1 \subseteq B_2$, y B_2 no puede calcular a K, entonces B_1 tampoco puede calcular a K. A continuación, se define a un subconjunto $B \subseteq \mathcal{P}$ como subconjunto no autorizado *maximal*, si para todo $B_1 \supseteq B$, $B_1 \neq B$, entonces $B_1 \in \Gamma$. Es suficiente verificar que ningún subconjunto no autorizado maximal pueda determinar información de K. En este caso, los subconjuntos no autorizados maximales son $\{P_1, P_2\}$, $\{P_1, P_3\}$, $\{P_1, P_4\}$, $\{P_2, P_4\}$ y $\{P_3, P_4\}$. Por ejemplo, $\{P_1\}$ no es maximal ya que $\{P_1, P_2\} \supseteq \{P_1\}$ y $\{P_1, P_2\} \notin \Gamma$. Ahora, obsérvese que para cada uno de estos conjuntos maximales K no puede ser calculado, ya sea porque alguna pieza necesaria de la información aleatoria hace falta o porque todas las partes que se tienen por el subconjunto son aleatorias. Por ejemplo, el subconjunto $\{P_1, P_2\}$ posee solo los valores aleatorios a_1, b_1, a_2 y c_1. Otro ejemplo es el subconjunto $\{P_3, P_4\}$ que posee a los valores $b_2, K - c_1, K - a_1 - a_2$ y $K - b_1 - b_2$. Dado que los valores a_1, a_2, b_1 y c_1 son valores aleatorios desconocidos, nuevamente K no puede ser calculada. Con un análisis similar se verifica que ningún subconjunto no autorizado maximal pueda obtener información acerca del valor de K. □

Es importante resaltar que se pueden obtener diferentes esquemas para una misma estructura de acceso usando circuitos equivalentes. Lo anterior se ilustra a continuación, retomando la estructura de acceso del ejemplo 1.3.

Ejemplo 1.10. *Si se lleva la fórmula (4) a una forma booleana normal conjuntiva se obtiene:*

$$(P_1 \vee P_2) \wedge (P_1 \vee P_3) \wedge (P_2 \vee P_3) \wedge (P_2 \vee P_4) \wedge (P_3 \vee P_4). \tag{5}$$

A continuación, se implementa el algoritmo de construcción del circuito monótono, usando el circuito correspondiente a la fórmula (5).

Supóngase que K *es la llave. Dado que la última compuerta del circuito es del tipo and, se asignan a sus cinco cables de entrada los valores* a_1, a_2, a_3, a_4 *y* $K - a_1 - a_2 - a_3 - a_4$*, respectivamente, donde la aritmética se hace en* $\mathbb{Z}_m$*. Enseguida, se considera a la compuerta or correspondiente a* $(P_1 \vee P_2)$*, a cuyos dos cables de entrada se les asigna por igual el valor* a_1*. De forma similar, a los dos cables de entrada correspondientes a* $(P_1 \vee P_3)$ *se les asigna el valor* a_2*, a los de* $(P_2 \vee P_3)$ *el valor de* a_3*, a los de* $(P_2 \vee P_4)$ *el valor de* a_4 *y, finalmente, a los de* $(P_3 \vee P_4)$ *el valor de* $K - a_1 - a_2 - a_3 - a_4$*. Así se obtiene lo siguiente:*

i) P_1 *recibe* $(y_1^1, y_1^2) = (a_1, a_2)$

ii) P_2 *recibe* $(y_2^1, y_2^2, y_2^3) = (a_1, a_3, a_4)$

iii) P_3 *recibe* $(y_3^1, y_3^2, y_3^3) = (a_2, a_3, K - a_1 - a_2 - a_3 - a_4)$

iv) P_4 *recibe* $(y_4^1, y_4^2) = (a_4, K - a_1 - a_2 - a_3 - a_4)$.

En lugar de probar que este esquema de compartición de secretos también es perfecto, se dará paso a un resultado más general e interesante.

Teorema 1.7. *Sea C cualquier circuito booleano monótono. Entonces, la construcción del circuito monótono produce un esquema de compartición de secretos perfecto para la estructura de acceso* $\gamma(C)$.

Demostración. Se procede por inducción sobre el número de compuertas en el circuito C. Si C contiene solo una compuerta, el resultado es bastante trivial: Si se trata de una compuerta *or*, entonces, cada participante tendría la llave (este esquema corresponde a la estructura de acceso que consiste de todos los subconjuntos no vacíos de participantes). Alternativamente, si se trata de una compuerta *and* con t entradas, el esquema correspode al (t, t)—esquema umbral encontrado con el algoritmo 2.

Ahora, como hipótesis inductiva, supóngase que existe un entero $j > 1$ tal que, para todos los circuitos C con menos de j compuertas, la construcción produce un esquema correspondiente a $\gamma(C)$. Sea C un circuito con j compuertas. Considérese a la última compuerta G en el circuito, donde G podría ser una compuerta *or* o *and*. Primero, se considera el caso en el que G es una compuerta *or*. Se denotan a los cables de entrada de G por W_i, $1 \leqslant i \leqslant t$. Estos t cables de entrada son las salidas de t sub-circuitos de C, los cuales denotamos C_i, $1 \leqslant i \leqslant t$. Correspondiente a cada C_i, se tiene un sub-esquema que corresponde a la estructura de acceso γ_{C_i}, por inducción. Ahora es fácil ver que

$$\gamma(C) = \bigcup_{i=1}^{t} \gamma_{C_i}.$$

A cada W_i se le asigna la llave K. Finalmente, el esquema corresponderá a $\gamma(C)$, como se deseaba. El análisis es similar si G es una compuerta *and*. En este caso se tiene que

$$\gamma(C) = \bigcap_{i=1}^{t} \gamma_{C_i}.$$

Se hace uso de un (t, t)—esquema umbral para compartir la llave K entre los t cables W_i. Nuevamente, el esquema que se obtiene corresponde a $\gamma(C)$ y la prueba queda completada. □

Por supuesto, cuando un subconjunto autorizado B quiere calcular a la llave, los participantes en B necesitan conocer el circuito usado por D para distribuir las partes y saber qué partes corresponden a qué cables del circuito. Toda esta información será pública. Los valores actuales de las partes son las que deben permanecer secretos.

El algoritmo para la reconstrucción de la llave involucra la combinación de las partes de acuerdo con la estructura del circuito, con la estipulación de que una compuerta *and* corresponde a la suma módulo m de los valores de los cables de entrada (suponiendo que se conocen todos los valores necesarios) y una compuerta *or* implica elegir el valor de cualquier cable de entrada (con el entendimiento de que todos los valores son idénticos).

Tómese como referencia al ejemplo 1.9 y considérese nuevamente al conjunto autorizado $\{P_1, P_2, P_4\}$. Ya se mostró, cómo este conjunto puede calcular a K. La observación anterior busca enfatizar que este procedimiento se debe llevar a cabo en una forma sistemática. En este caso, se deben asignar valores a 6 de los 8 cables de entrada; esto es, a todos los que emanan de x_1, x_2 y x_4 en el circuito correspondiente al ejemplo 1.9. Se puede ver que en el extremo izquierdo la compuerta *and* tiene valores asignados para sus tres cables de entrada. La suma de los valores de estos cables de entrada es K. Este cómputo es, de hecho, el mismo que se ha descrito antes.

1.3.3 Ejemplo de aplicación

El Sr. Espinosa ha muerto, pero, preocupado por el porvenir de su familia dejó instrucciones claras a su abogado y amigo de confianza, el Lic. Vázquez, para llevar a cabo la lectura de su testamentento. El Sr. Espinosa tuvo 5 hijos y las opciones que estipuló para que se lleve a cabo la ceremonia de lectura son las siguientes:

i) Que se encuentren presentes sus 5 hijos

ii) Que se encuentren presentes 4 de sus hijos

iii) Que se encuentren presentes su hijo mayor y 2 hijos más.

El Lic. Vázquez desea llevar a cabo un sistema de seguridad automatizado, donde los hijos del Sr. Espinosa tengan acceso al testamento solo cuando las peticiones de su querido amigo sean cubiertas.

Esta situación puede ser resuelta como un problema de compartición de secretos. Obsérvese que si la opción iii) fuera que cualesquiera 3 de sus hijos estuvieran presentes, entonces el problema podría ser tratado como un $(3,5)-$esquema umbral, pero no es así.

Considerérese al conjunto de participantes $\{P_1, P_2, P_3, P_4, P_5\}$ correspondiente a los hijos del Sr. Espinosa, en el orden que han nacido. A cada P_i corresponde un x_i o un x_i' donde x_i representa al suceso en el que P_i se encuentra presente y x_i' al suceso en el que P_i está ausente, para $i \in \{1, \ldots, 5\}$. Entonces, la expresión $x_1x_2x_3x_4x_5$ representa el evento en el que se encuentran presentes todos los hijos del Sr. Espinosa y $x_1'x_2x_3'x_4x_5$ representa al evento en el que P_1 y P_3 están ausentes y P_2, P_4 y P_5 están presentes.

Los eventos que satisfacen los deseos del Sr. Espinosa se muestran en la siguiente tabla que está ordenada de acuerdo a las condiciones que él pidió:

i)	$x_1x_2x_3x_4x_5$
ii)	$x_1x_2x_3x_4x_5'$
	$x_1x_2x_3x_4'x_5$
	$x_1x_2x_3'x_4x_5$
	$x_1x_2'x_3x_4x_5$
	$x_1'x_2x_3x_4x_5$
iii)	$x_1x_2x_3x_4'x_5'$
	$x_1x_2x_3'x_4x_5'$
	$x_1x_2x_3'x_4'x_5$
	$x_1x_2'x_3x_4x_5'$
	$x_1x_2'x_3x_4'x_5$
	$x_1x_2'x_3'x_4x_5$.

El polinomio

$$p = x_1x_2x_3x_4x_5 + x_1x_2x_3x_4x_5' + x_1x_2x_3x_4'x_5 + x_1x_2x_3'x_4x_5 + x_1x_2'x_3x_4x_5 + x_1'x_2x_3x_4x_5 + x_1x_2x_3x_4'x_5' + x_1x_2x_3'x_4x_5' + x_1x_2x_3'x_4'x_5 + x_1x_2'x_3x_4x_5' + x_1x_2'x_3x_4'x_5 + x_1x_2'x_3'x_4x_5, \quad (6)$$

describe a un circuito booleano, donde $\overline{p}(x_1, x_2, x_3, x_4, x_5)$ arroja resultados verdaderos $(= 1)$ cuando el subconjunto en función es habilitado para tener acceso al testamento y será falso $(= 0)$ si ocurre lo contrario. Además, la evaluación de cada expresión producto contiene información sobre la asistencia o no de los participantes. Por ejemplo, si $q = x_1x_2x_3x_4x_5'$ y $\overline{q}(x_1, x_2, x_3, x_4, x_5) = 1$, se puede asumir que el subconjunto $\{P_1, P_2, P_3, P_4\}$ está presente.

Los elementos de la tabla definen a los subconjuntos autorizados que forman a una estructura de acceso que además es una estructura de acceso monótona, ya que cuando un subconjunto de hijos es habilitado para tener acceso al testamento no puede ocurrir que al agregar a otro elemento, el conjunto se deshabilite. La estructura de acceso definida en su forma booleana normal disjuntiva es la siguiente:

$$(P_1 \wedge P_2 \wedge P_3 \wedge P_4 \wedge P_5) \vee (P_1 \wedge P_2 \wedge P_3 \wedge P_4) \vee (P_1 \wedge P_2 \wedge P_3 \wedge P_5) \vee (P_1 \wedge P_2 \wedge P_4 \wedge P_5) \vee (P_1 \wedge P_3 \wedge P_4 \wedge P_5) \vee (P_2 \wedge P_3 \wedge P_4 \wedge P_5) \vee (P_1 \wedge P_2 \wedge P_3 \wedge) \vee (P_1 \wedge P_2 \wedge P_4) \vee (P_1 \wedge P_2 \wedge P_5) \vee (P_1 \wedge P_3 \wedge P_4) \vee (P_1 \wedge P_3 \wedge P_5) \vee (P_1 \wedge P_4 \wedge P_5). \quad (7)$$

La fórmula 7 describe a un circuito monótono con $|12| + 1 = 13$ compuertas y dicha descripción será utilizada para construir el esquema de compartición de secretos de acuerdo al algoritmo 3. Dicho algoritmo conduce a lo siguiente:

Supóngase que K es la llave. El valor de K es dado a cada uno de los 12 cables de entrada de la última compuerta *or*. Luego se considera la compuerta *and*, correspondiente a la claúsula $(P_1 \wedge P_2 \wedge P_3 \wedge P_4 \wedge P_5)$. A estos 5 cables de entrada se les asignan los valores a_1, a_2, a_3, a_4 y $K - a_1 - a_2 - a_3 - a_4$, respectivamente, donde la aritmética se hace en $\mathbb{Z}_m$. De forma similar, a los 4 cables correspondientes a $(P_1 \wedge P_2 \wedge P_3 \wedge P_4)$ se les asignan los valores b_1, b_2, b_3 y $K - b_1 - b_2 - b_3$. El proceso continúa para cada una de las 12 compuertas tipo *and* y, finalmente, los participantes recibirán las siguientes partes:

i) P_1 recibe $(y_1^1, y_1^2, y_1^3, y_1^4, y_1^5, y_1^6, y_1^7, y_1^8, y_1^9, y_1^{10}, y_1^{11}) = (a_1, b_1, c_1, d_1, e_1, g_1, h_1, i_1, j_1, k_1, l_1)$

ii) P_2 recibe $(y_2^1, y_2^2, y_2^3, y_2^4, y_2^5, y_2^6, y_2^7, y_2^8) = (a_2, b_2, c_2, d_2, f_1, g_2, h_2, i_2)$

iii) P_3 recibe $(y_3^1, y_3^2, y_3^3, y_3^4, y_3^5, y_3^6, y_3^7, y_3^8) = (a_3, b_3, c_3, e_2, f_2, K - g_1 - g_2, j_2, k_2)$

iv) P_4 recibe $(y_4^1, y_4^2, y_4^3, y_4^4, y_4^5, y_4^6, y_4^7, y_4^8) = (a_4, K - b_1 - b_2 - b_3, d_3, e_3, f_3, K - h_1 - h_2, K - j_1 - j_2, l_2)$

v) P_5 recibe $(y_5^1, y_5^2, y_5^3, y_5^4, y_5^5, y_5^6, y_5^7, y_5^8) = (K - a_1 - a_2 - a_3 - a_4, K - c_1 - c_2 - c_3, K - d_1 - d_2 - d_3, K - e_1 - e_2 - e_3, K - f_1 - f_2 - f_3, K - i_1 - i_2, K - k_1 - k_2, K - l_1 - l_2)$.

De esta manera, el participante P_1 recibe 11 elementos de $\mathbb{Z}_m$ como su parte y los otros participantes 8, con las cuales cada subconjunto autorizado podría recuperar a la llave. Por ejemplo, el subconjunto autorizado $\{P_1, P_4, P_5\}$ puede calcular $y_1^{11} + y_4^8 + y_5^8 \ mod\ m = l_1 + l_2 + (K - l_1 - l_2) \ mod\ m = K$.

Hasta ahora, el problema ha sido resuelto. Sin embargo, se hará uso de la teoría estudiada para encontrar otra solución que sea equivalente pero más eficiente. La idea es encontrar un circuito equivalente al descrito por el polinomio 6, pero más simple. Para ello, se hará uso del algoritmo Quine-McCluskey para minimizar dicho polinomio.

i) Las expresiones producto deben estar representadas en términos de ceros y unos, tales que x_i' y x_i son denotadas por 0 y 1, respectivamente. En este caso no hay variables faltantes que deban ser indicadas con guión.

$x_1x_2x_3x_4x_5$	11111
$x_1x_2x_3x_4x_5'$	11110
$x_1x_2x_3x_4'x_5$	11101
$x_1x_2x_3'x_4x_5$	11011
$x_1x_2'x_3x_4x_5$	10111
$x_1'x_2x_3x_4x_5$	01111
$x_1x_2x_3x_4'x_5'$	11100
$x_1x_2x_3'x_4x_5'$	11010
$x_1x_2x_3'x_4'x_5$	11001
$x_1x_2'x_3x_4x_5'$	10110
$x_1x_2'x_3x_4'x_5$	10101
$x_1x_2'x_3'x_4x_5$	10011

ii) Las expresiones producto, consideradas como n-tuplas binarias, son particionadas en clases. La partición se lleva a cabo de acuerdo con la cantidad de unos que tienen, de menor a mayor:

$p_1 =$	11100
$p_2 =$	11010
$p_3 =$	11001
$p_4 =$	10110
$p_5 =$	10101
$p_6 =$	10011
$p_7 =$	11110
$p_8 =$	11101
$p_9 =$	11011
$p_{10} =$	10111
$p_{11} =$	01111
$p_{12} =$	11111

iii) El siguiente paso es aplicar la regla (3) para simplicar expresiones con la clase de equivalencia vecina. Por ejemplo, de p_1 y p_7 se obtiene la expresión $q_1 = 111-0$, de p_1 y p_8 se obtiene la expresión $q_2 = 1110-$ y así sucesivamente. Al final de este proceso, todas las expresiones han sido marcadas, por lo que no son implicantes primos. La tabla de las expresiones obtenidas y ya clasificadas de acuerdo al inciso ii) queda de la siguiente manera:

$q_1 =$	$111-0$
$q_2 =$	$1110-$
$q_3 =$	$11-10$
$q_4 =$	$1101-$
$q_5 =$	$11-01$
$q_6 =$	$110-1$
$q_7 =$	$1-110$
$q_8 =$	$1011-$
$q_9 =$	$1-101$
$q_{10} =$	$101-1$
$q_{11} =$	$1-011$
$q_{12} =$	$10-11$
$q_{13} =$	$1111-$
$q_{14} =$	$111-1$
$q_{15} =$	$11-11$
$q_{16} =$	$1-111$
$q_{17} =$	-1111

Se continúa con la segunda vuelta de simplificaciones. Por ejemplo, de q_1 y q_{14} se obtiene $r_1 = 111--$, de q_2 y q_{13} se obtiene $r_1 = 111--$, de q_3 y q_{15} se obtiene $r_2 = 11-1-$ y así sucesivamente. Al final de este proceso, se marcan las expresiones utilizadas y se construye la nueva tabla con las expresiones simplificadas (si se encuentra una misma expresión, solo se escribe una vez) y las expresiones que no fueron marcadas en esta vuelta, bajo los requerimientos del inciso ii):

$q_1 =$	$111-0$	$\surd$
$q_2 =$	$1110-$	$\surd$
$q_3 =$	$11-10$	$\surd$
$q_4 =$	$1101-$	$\surd$
$q_5 =$	$11-01$	$\surd$
$q_6 =$	$110-1$	$\surd$
$q_7 =$	$1-110$	$\surd$
$q_8 =$	$1011-$	$\surd$
$q_9 =$	$1-101$	$\surd$
$q_{10} =$	$101-1$	$\surd$
$q_{11} =$	$1-011$	$\surd$
$q_{12} =$	$10-11$	$\surd$
$q_{13} =$	$1111-$	$\surd$
$q_{14} =$	$111-1$	$\surd$
$q_{15} =$	$11-11$	$\surd$
$q_{16} =$	$1-111$	$\surd$
$q_{17} =$	-1111	

$r_1 =$	$111--$
$r_2 =$	$11-1-$
$r_3 =$	$11--1$
$r_4 =$	$1-11-$
$r_5 =$	$1-1-1$
$r_6 =$	$1--11$
$q_{17} =$	-1111

Obsérvese que las expresiones restantes ya no pueden ser sometidas a reducción, por lo que se concluye que se han encontrado a todos los implicantes primos:

$111--$	$x_1x_2x_3$
$11-1-$	$x_1x_2x_4$
$11--1$	$x_1x_2x_5$
$1-11-$	$x_1x_3x_4$
$1-1-1$	$x_1x_3x_5$
$1--11$	$x_1x_4x_5$
-1111	$x_2x_3x_4x_5$

iv) Se establece la tabla de implicantes primos:

	11100	11010	11001	10110	10101	10011	11110	11101	11011	10111	01111	11111
$111--$	$\times$						$\times$	$\times$				$\times$
$11-1-$		$\times$					$\times$		$\times$			$\times$
$11--1$			$\times$					$\times$	$\times$			$\times$
$1-11-$				$\times$			$\times$			$\times$		$\times$
$1-1-1$					$\times$			$\times$		$\times$		$\times$
$1--11$						$\times$			$\times$	$\times$		$\times$
-1111											$\times$	$\times$

Todos los implicantes primos resultaron ser términos principales por lo que se concluye que

$$p \sim x_1x_2x_3 + x_1x_2x_4 + x_1x_2x_5 + x_1x_3x_4 + x_1x_3x_5 + x_1x_4x_5 + x_2x_3x_4x_5.$$

El polinomio $p' = x_1x_2x_3 + x_1x_2x_4 + x_1x_2x_5 + x_1x_3x_4 + x_1x_3x_5 + x_1x_4x_5 + x_2x_3x_4x_5$ describe a un nuevo circuito monótono equivalente al circuito manejado anteriormente. Su estructura de acceso definida en su forma booleana normal disjuntiva es la siguiente:

$$(P_2 \wedge P_3 \wedge P_4 \wedge P_5) \vee (P_1 \wedge P_2 \wedge P_3) \vee (P_1 \wedge P_2 \wedge P_4) \vee (P_1 \wedge P_2 \wedge P_5) \vee (P_1 \wedge P_3 \wedge P_4) \vee (P_1 \wedge P_3 \wedge P_5) \vee (P_1 \wedge P_4 \wedge P_5). \quad (8)$$

La fórmula 8 describe a un circuito monótono con $|7| + 1 = 8$ compuertas y dicha descripción será utilizada para construir el esquema de compartición de secretos de acuerdo al algoritmo 3. Dicho algoritmo conduce a lo siguiente:

Sea K la llave. El valor de K es dado a cada uno de los 7 cables de entrada de la última compuerta *or*. Luego se considera la compuerta *and*, correspondiente a la claúsula $(P_2 \wedge P_3 \wedge P_4 \wedge P_5)$. A estos 4 cables de entrada se les asignan los valores a_1, a_2, a_3 y $K - a_1 - a_2 - a_3$, respectivamente, donde la aritmética se hace en $\mathbb{Z}_m$. De forma similar a los 3 cables correspondientes a $(P_1 \wedge P_2 \wedge P_3)$ se les asignan los valores b_1, b_2 y $K - b_1 - b_2$. El proceso continúa para cada una de las 7 compuertas tipo *and* y, finalmente, los participantes recibirán las siguientes partes:

i) P_1 recibe $(y_1^1, y_1^2, y_1^3, y_1^4, y_1^5, y_1^6) = (b_1, c_1, d_1, e_1, f_1, g_1)$

ii) P_2 recibe $(y_2^1, y_2^2, y_2^3, y_2^4) = (a_1, b_2, c_2, d_2)$

iii) P_3 recibe $(y_3^1, y_3^2, y_3^3, y_3^4) = (a_2, b_3, K - b_1 - b_2, e_2, f_2)$

iv) P_4 recibe $(y_4^1, y_4^2, y_4^3, y_4^4) = (a_3, K - c_1 - c_2, K - e_1 - e_2, g_2)$

v) P_5 recibe $(y_5^1, y_5^2, y_5^3, y_5^4) = (K - a_1 - a_2 - a_3, K - d_1 - d_2, K - f_1 - f_2, K - g_1 - g_2)$.

De este manera, el participante P_1 recibe, únicamente, 6 elementos de $\mathbb{Z}_m$ como su parte y los otros participantes, 4. Nuevamente, cada subconjunto autorizado es capaz de recuperar la llave. Por ejemplo, el subconjunto autorizado $\{P_1, P_4, P_5\}$ puede calcular $y_1^6 + y_4^4 + y_5^4 \ mod\ m = g_1 + g_2 + (K - g_1 - g_2) \ mod\ m = K$.

Este ejemplo de aplicación ilustra que para un mismo problema se pueden tener diferentes esquemas, pero no todos son igual de convenientes. Por ello, es importante utilizar un circuito monótono adecuado. En este capítulo, se mostró que un circuito booleano puede ser representado por un polinomio booleano, para el cual existen diferentes métodos de minimización. En este caso, se hizo uso del algoritmo de minimización Quine-McCluskey. Esta idea es útil porque ayuda a encontrar circuitos equivalentes, que sean más sencillos de estudiar y usar en la práctica.

2 COMPARTICIÓN DE SECRETOS VISUALES

La Criptografía Visual es una técnica de codificación donde el secreto es una imagen. Esta fue propuesta por primera vez por Naor y Shamir en la *Eurocrypt* 1994 *Conference*. Uno de sus principales intereses es la implementación de esquemas que no necesitan ningún tipo de conocimiento o cómputo criptográfico para su decodificación, de tal manera que esta última quede a cargo del sistema visual humano.

En este capítulo se presentará el tema de esquemas de compartición de secretos visuales donde el secreto es una imagen binaria. Posteriormente, se proponen una serie de esquemas solución para los casos $(2,2)$, $(2,3)$, $(2,4)$, $(2,n)$, $(3,3)$, $(3,4)$, $(3,n)$ y $(4,4)$. Dichas soluciones son analizadas y, finalmente, programadas en el sofware matemático MATLAB®.

2.1 CONCEPTO DE IMAGEN DIGITAL

Un tipo de imagen es la fotografía monocromática, es decir, en tonos de grises. Dicha imagen puede ser considerada como una función bidimensional $f(x,y)$, donde los valores de la función dan el valor de la intensidad o nivel de gris de la imagen en cada punto dado (ver figura 6). Se puede asumir que los valores del nivel de gris de la imagen pueden ser cualquier número real en el rango de 0,0 (negro) a 1,0 (blanco). Los rangos de x y y dependerán de la imagen, tomando cualquier número real entre su mínimo y su máximo. El concepto de imagen como una función es de vital importancia en el desarrollo de la implementación de técnicas de procesamiento de imágenes.

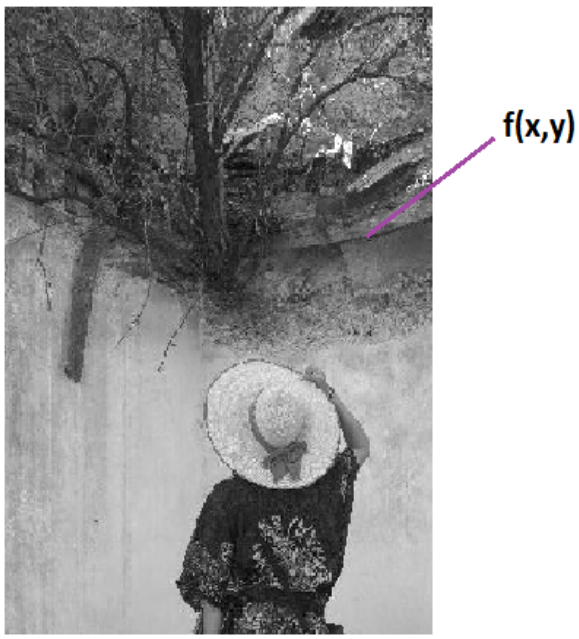


Figura 6: Imagen como función bidimensional

El concepto de imagen digital difiere del de una fotografía en que x, y y $f(x,y)$ son valores discretos. Por ejemplo, la imagen digital (ver figura 7) tiene como rango para x y para y a los números enteros entre 1 y 21 y para $f(x,y)$ a los números enteros entre 0 (negro) y 255 (blanco). Una imagen digital puede ser representada matemáticamente como una matriz de puntos de muestreo de una imagen continua, en donde las entradas de dicha matriz cuantifican el valor del nivel de gris de los puntos

muestreados. Estos valores son los pixeles, que componen a su vez la imagen digital. En la figura 8, se presentan los valores de la matriz que corresponden a la figura 7. Esta idea se extiende para imágenes digitales a color, donde los pixeles pasan a ser los valores asignados para el color de la imagen en cada punto de muestreo.

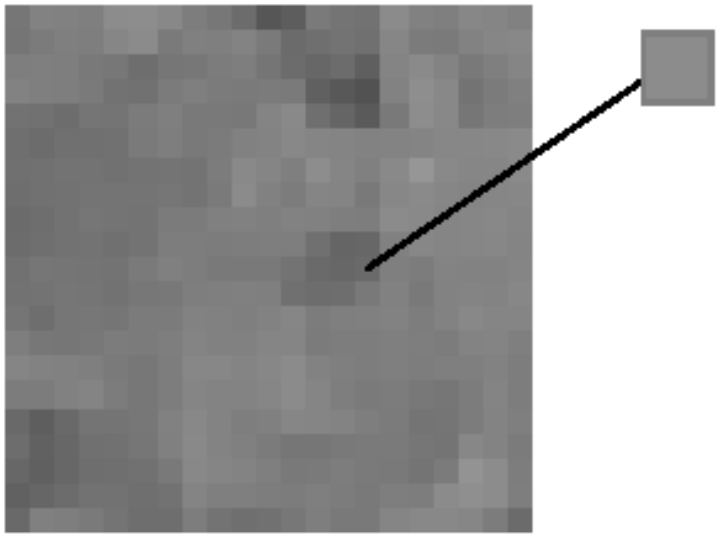


Figura 7: Imagen digital

118	130	129	127	140	141	136	125	119	108	86	93	122	116	114	139	129	127	135	133	129
117	123	127	126	134	139	122	128	127	131	122	107	111	115	112	134	123	122	130	134	136
126	127	126	121	119	110	118	119	124	126	117	108	103	108	95	132	138	131	121	127	133
130	127	126	122	114	113	123	127	123	122	123	122	96	88	82	130	142	136	115	123	128
114	113	109	117	116	123	127	121	121	120	132	137	111	101	90	121	141	136	118	125	126
111	108	112	112	113	122	126	121	123	129	131	136	133	132	130	131	132	135	135	136	138
112	113	115	115	113	115	116	115	124	137	133	127	141	133	128	137	150	137	134	134	138
110	113	114	116	116	116	118	115	122	139	129	124	133	132	121	135	138	134	132	134	135
107	110	114	118	116	115	123	123	128	129	127	130	129	126	125	139	137	133	134	136	133
108	111	114	115	112	115	125	125	124	125	126	126	112	101	103	129	135	133	134	135	132
113	118	117	116	115	122	128	125	121	121	121	115	106	103	106	127	124	131	130	130	133
114	115	118	117	114	119	120	126	127	128	127	112	109	110	121	129	121	130	132	131	135
117	110	118	119	117	123	123	130	129	127	130	134	124	127	128	130	125	130	136	133	133
126	119	118	120	124	123	126	132	129	129	133	133	123	125	125	127	126	124	131	132	127
133	130	129	127	122	119	125	135	135	132	135	139	130	127	125	127	125	127	127	130	125
124	124	129	132	124	119	125	133	131	131	134	139	133	126	122	124	120	118	124	132	125
105	94	105	116	116	119	129	136	131	126	128	134	133	131	126	123	116	117	124	131	127
109	97	103	112	113	118	128	137	131	128	123	118	118	121	122	122	118	116	135	131	123
103	96	103	110	111	113	121	136	133	131	127	121	122	125	122	121	116	123	147	140	125
97	102	108	114	114	111	114	132	127	127	125	124	127	124	121	126	125	139	143	140	126
105	128	126	122	116	110	110	125	115	111	113	119	126	121	121	133	136	136	134	124	107

Figura 8: Entradas de la matriz correspondiente a la figura 7

Otra manera de representar una imagen digital es a través de un mapa de bits, que es una rejilla rectangular de pixeles o puntos de color. A una imagen digital representada por un mapa de bits se le pude definir por su tamaño, es decir, por su altura y anchura (en pixeles) y por su profundidad de color (en bits por pixel). Esto último determina el número de colores distintos que se pueden almacenar en cada punto individual; por lo tanto, determinan la calidad de la imagen en gran medida. A mayor profundidad de bits (más bits de información por pixel), la imagen tiene más colores disponibles y puede representar su color de forma más precisa. Sin embargo, mayor será el tamaño de la imagen. Dado que cada bit puede tener solo dos valores (0 y 1), la fórmula matemática para hallar el número de colores posibles es elevar 2 a la potencia del número de bits por pixel que se tengan (ver cuadro 1).

En esta ocasión se trabajará con imagenes binarias, que son aquellas cuyos pixeles son negros y blancos y estos serán representados con los valores 0 y 1, respectivamente. Dado que hay solo dos

Nombre	Bits por pixel	Colores posibles
Blanco y Negro	1	2
Pantalla Windows	4	16
Escala de Grises	8	256
Color Indexado	8	256
Color Alta Densidad	16	65,536
Color verdadero RGB	24	16,777,216
Color CMYK alta calidad	32	4,294,967,296

Cuadro 1: Tabla de profundidad de color

posibles valores para cada pixel, solo se necesita 1 bit por pixel, por lo que las imágenes binarias son muy eficientes en términos de almacenamiento.

2.2 SOLUCIÓN A UN PROBLEMA DE COMPARTICIÓN DE SECRETOS PARA UNA IMAGEN BINARIA

A partir de ahora se hablará de un (k, n)-esquema visual umbral que es un (k, n)-esquema umbral, con la diferencia de que el secreto es una imagen y no un número como fue considerado en el capítulo anterior. Dicho problema, cuando se trata de una imagen binaria, fue tratado y resuelto por Moni Naor y Adi Shamir en [NS95] y se tomará como referencia su trabajo para el estudio de dicho caso. La idea es, básicamente, generar n transparencias de forma que la imagen original sea visible si se sobreponen k (o más) transparencias, pero que la imagen permanezca invisible si menos de k transparencias son sobrepuestas o analizadas por cualquier otro método. A continuación el planteamiento del modelo.

Considérese a la imagen como una colección de pixeles blancos y negros y que cada pixel puede ser manipulado separadamente. Es decir, si una imagen es de tamaño $l \times a$, se tendrán que manipular a $l \times a$ pixeles. Cada pixel original aparecerá en cada una de las n versiones modificadas (llamadas sombras), una para cada transparencia. Cada sombra será una colección de m subpixeles negros y blancos, los cuales estarán impresos de forma tal que el sistema visual humano sea capaz de promediar sus aportaciones individuales (de los subpixeles negros y blancos). El resultado de esta estructura puede ser descrita por una matriz Booleana $S = [s_{ij}]$ de tamaño $n \times m$, donde $s_{ij} = 1$ si y solo si el j-ésimo subpixel en la i-ésima sombra es negro. Cuando las sombras $i_1, i_2, ..., i_r$ son sobrepuestas entre sí, de forma que los subpixeles queden adecuadamente alineados, entonces se obtiene una sombra combinada cuyos subpixeles negros están representados por la forma *or* de las filas $i_1, i_2, ..., i_r \in S$. El nivel de gris de esta sombra combinada es proporcional al peso de Hamming $H(V)$ del *or* $m-$vector V, es decir, al número de elementos distintos a 0 en el vector V. Este nivel de gris es interpretado por el sistema visual de los usuarios como negro si $H(V) > d$ y como blanco si $H(V) < d - \alpha m$ para algún umbral fijo $1 \leqslant d \leqslant m$ y diferencia relativa $\alpha > 0$. Con este modelo, el efecto de un subpixel negro en una de las transparencias no se puede deshacer por el color de otro subpixel (de otra sombra) que ha sido sobrepuesto al mismo. Por lo tanto, en lugar de representar a un pixel blanco por una colección de subpixeles blancos y a un pixel negro por una colección de subpixeles negros, se usa a un umbral d y a la diferencia relativa α para distinguir entre estos dos colores.

Definición 2.1. *Una solución a un* k *de* n *esquema de compartición de secretos visuales consiste de dos colecciones de* $n \times m$ *matrices Booleanas* C_0 *y* C_1*. Para compartir a un pixel blanco, el distribuidor* D *elige aleatoriamente a una de las matrices en* C_0 *y para compartir a un pixel negro, D elige aleatoriamente a una de las matrices en* C_1*. La matriz que ha sido elegida define el color de los* m *subpixeles en cada una de las* n *sombras. La solución es considerada válida si se cumplen las siguiente tres condiciones:*

i) Para cualquier matriz S *de* C_0*, el or-vector* V *de cualesquieras* k *de sus* n *filas satisface que* $H(V) < d - \alpha m$*.*

ii) Para cualquier matriz S *de* C_1*, el or-vector* V *de cualesquieras* k *de sus* n *filas satisface que* $H(V) \geqslant d$*.*

iii) *Para cada subconjunto* $\{i_1, i_2, ..., i_q\}$ *de* $\{1, 2, ..., n\}$ *con* $q < k$, *las dos colecciones de tamaño* $q \times m$, D_t *con* $t \in \{1, 0\}$, *que se obtienen de restrigir las* $n \times m$ *matricez de* C_t *con* $t \in \{1, 0\}$ *a las filas* $\{i_1, i_2, ..., i_q\}$. *Estas son indistinguibles entre sí, en el sentido de que contienen a matrices con la misma frecuencia.*

La tercera condición indica que si alguien llegase a inspeccionar menos de k sombras, sin importar el poder del criptoanálisis al que pueda recurrir, no podrá obtener ninguna ventaja que le ayude a decidir si el pixel compartido fue blanco o negro. En la mayoría de estas construcciones, existe una función de probabilidad uniforme f, de tal manera que las partes combinadas de $q < k$ sombras consisten en todos los V's con $H(V) = f(q)$, independientemente de si la matriz fue tomada de C_0 o de C_1. Tal esquema es llamado uniforme. A las primeras dos condiciones se les conoce como de contraste y a la tercera como condición de seguridad.

Los parámetros importantes a considerar en este esquema son los siguientes:

m El número de pixeles en una sombra. Este número representa la pérdida de resolución de la imagen original a la imagen compartida. Se desea que m sea tan pequeña como sea posible.

α La diferencia relativa en el peso entre la combinanción de las sombras que vienen de un pixel blanco y de un pixel negro en la imagen original. Esto representa la pérdida de constraste. Se desea hacer a α tan grande como sea posible.

r El tamaño de las colecciones C_0 y C_1, las cuales no son necesariamente del mismo tamaño en todos los casos. $\log r$ representa el número de *bits* aleatorios necesarios para generar las sombras. Esto no afecta la calidad de la imagen.

2.3 SOLUCIONES TEÓRICAS, ¿SOLUCIONES PRÁCTICAS?

2.3.1 Una solución al (2,2)-esquema visual umbral

El problema original de la criptografía visual fue, precisamente, este caso especial en el que se desea que un conjunto de dos participantes recuperen a la imagen secreta bajo la sobreposición de las partes que serán asignadas a cada uno de manera secreta y separada. La solución propuesta por Naor y Shamir [NS95] para llevar a cabo dicho procedimiento considera a las siguientes colecciones de matrices de tamaño 2×4:

$C_0 = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}\}$

$C_1 = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}\}$

Las sombras quedan de la siguiente manera:

i) Cuando el pixel de la imagen original de la imagen binaria es blanco, las dos sombras correspondientes tendrán el mismo patrón de pixeles, es decir, son idénticas.

ii) Cuando el pixel de la imagen original de la imagen binaria es negro, las dos sombras correspondientes serán complementarias, es decir, en la posición que se tiene a un subpixel blanco para una sombra, en la otra sombra se tendrá uno negro, y viceversa.

A continuación se exhibe que este modelo es una solución.

Dado que las matrices son de tamaño 2×4, se sigue que cada sombra está formada por 4 subpixeles, es decir, $m = 4$. Si se considera a $d = 4$ y $0 < \alpha < 1/2$, obsérvese que:

i) Para cualquier matriz construida a partir de C_0, el *or*-vector V de sus 2 filas satisface que $H(V) < 2 - \alpha n$, dado que $H(V)$ será siempre igual a 2 y $2 < 4 - \alpha 2$.

ii) Para cualquier matriz construida a partir de C_1, el *or*-vector V de sus 2 filas satisface que $H(V) \geqslant 4$, dado que $H(V)$ será siempre igual a 4.

iii) Cualquier fila tomada de una matriz construida a partir de C_0 o de C_1 es indistingible, en el sentido en el que su peso de Hamming será siempre igual a 2 y la probabilidad de que corresponda a una sombra de un pixel blanco o de un pixel negro es exactamente la misma.

Este modelo realiza una expansión de pixel de 1 a 4; es decir, un pixel de la imagen original ahora estará representado por cuatro pixeles. Además, cada sombra deberá ser reestructurada de forma que sea una matriz de 2×2, con la finalidad de que en la práctica la imagen original no sea distorcionada. En las figuras 9 y 10, se muestran gráficamente a los 6 pares de sombras que corresponden a las matrices obtenidas de permutar a las columnas enunciadas con anterioridad y que codifican a un pixel blanco y a uno negro, respectivamente.

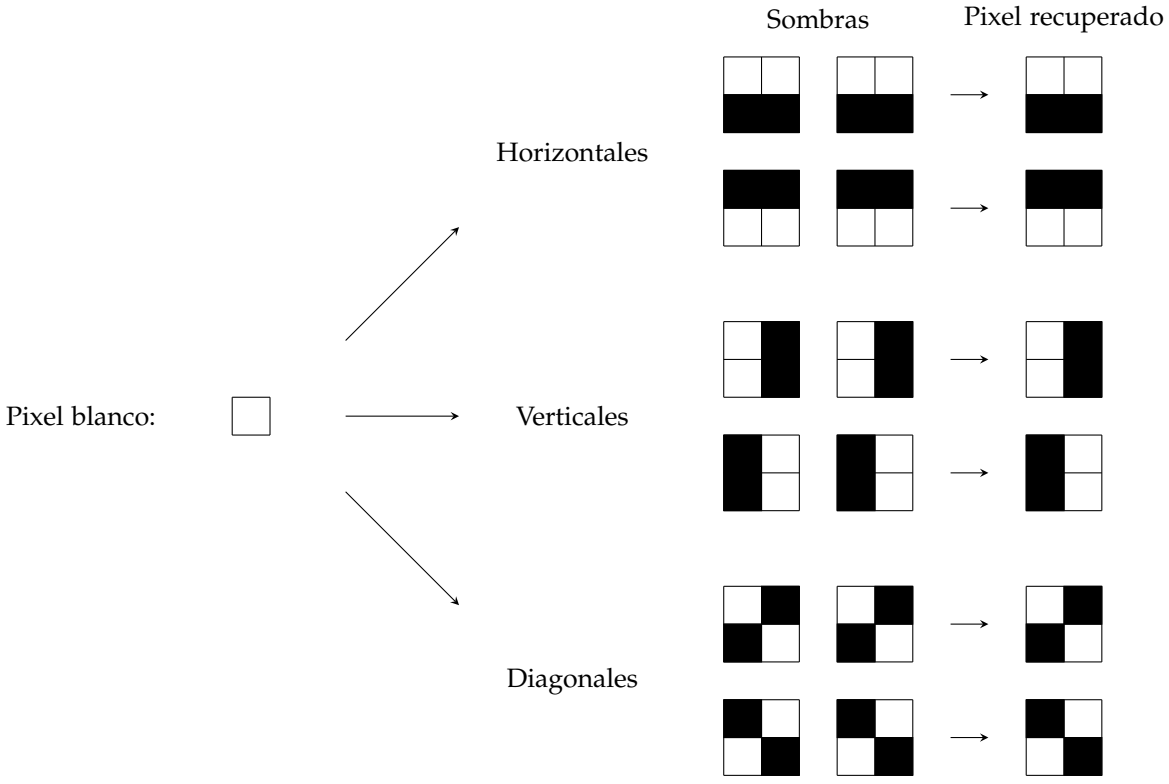


Figura 9: Codificación de un pixel blanco

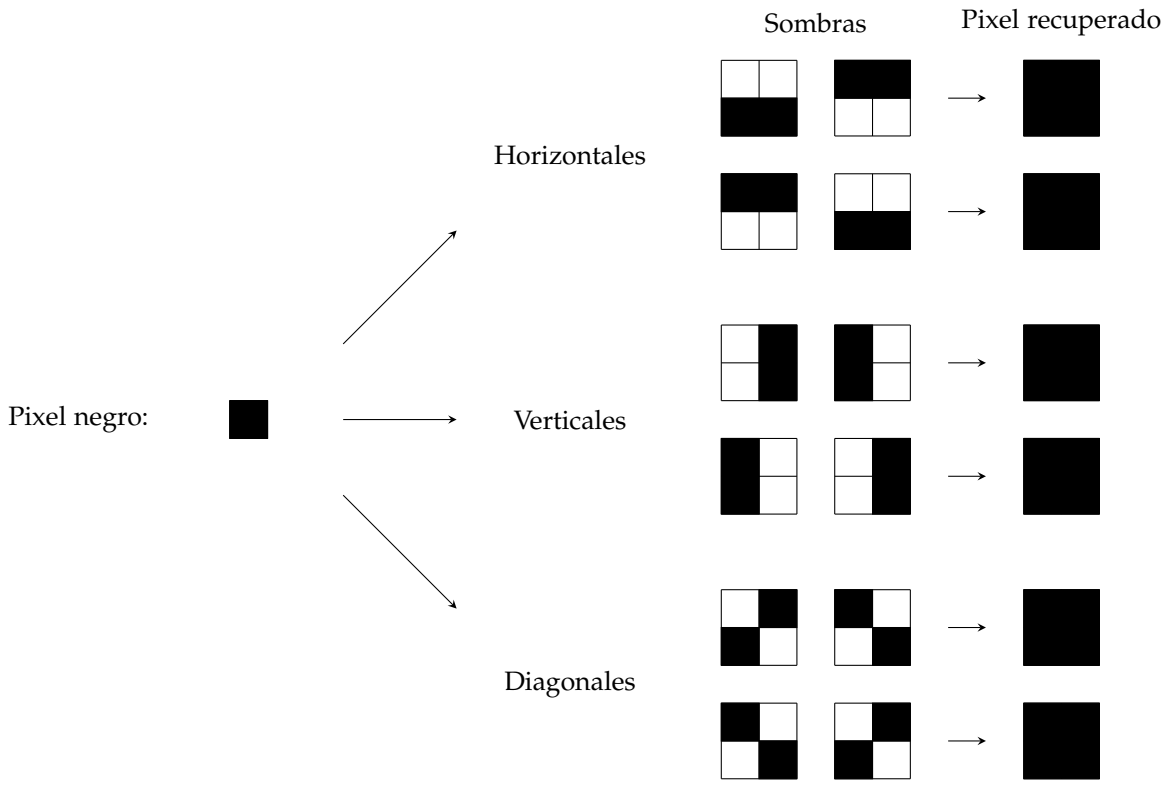
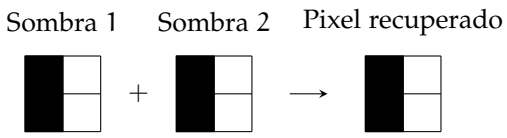
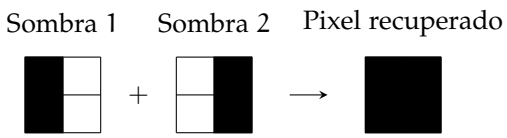


Figura 10: Codificación de un pixel negro

Ejemplo 2.1. *Si se desea compartir un pixel blanco y se considera a la matriz* $C_0 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$ *ambas sombras serían* 1010 *y el or-vector V obtenido al sobreponer dichas sombras sería nuevamente* 1010. *Esto gráficamente se representa:*



Si se desea compartir a un pixel negro y se considera a la matriz $C_1 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$ *las sombras serían* 1010 *y* 0101. *El or-vector V obtenido al sobreponer dichas sombras sería* 1111. *Esto gráficamente se representa:*



Al final del proceso un pixel blanco se recupera como un pixel $\frac{2}{4}$ negro y uno negro como $\frac{4}{4}$ negro. El resultado de aplicar esta solución es ilustrado a continuación:

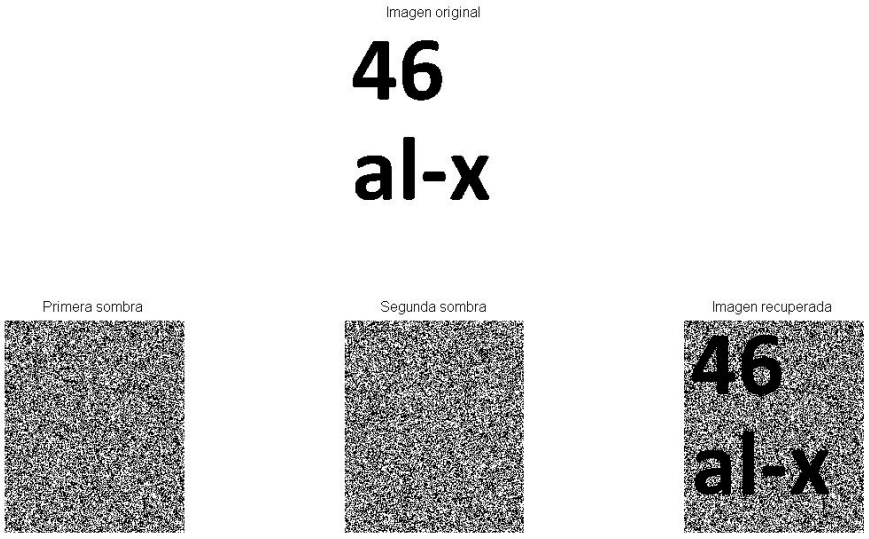


Figura 11: $(2,3)$-esquema visual umbral.

2.3.2 Dos soluciones al (2,3)-esquema visual umbral

Una solución al problema $(2,3)$-esquema visual umbral está dada por las siguientes colecciones de matrices de tamaño 3×4:

$$C_{0_1} = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \}$$

$$C_{1_1} = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \}$$

Las sombras quedan de la siguiente manera:

i) Cuando el pixel de la imagen original de la imagen binaria es blanco, las dos sombras correspondientes tendrán el mismo patrón de pixeles, es decir, son idénticas.

ii) Cuando el pixel de la imagen original de la imagen binaria es negro, las dos sombras correspondientes tendrán un pixel negro en común, un pixel blanco en común y dos pixeles serán complementarios.

A continuación se exhibe que este modelo es una solución.

Dado que las matrices son de tamaño 3×4, se sigue que cada sombra está formada por 4 subpixeles, es decir, $m = 4$. Si se considera a $d = 3$ y $0 < \alpha < 1/3$, obsérvese que:

i) Para cualquier matriz construida a partir de C_0, el *or*-vector V de 2 de sus filas satisface que $H(V) < 2 - \alpha m$, dado que $H(V)$ será siempre igual a 2 y $2 < 3 - \alpha 3$.

ii) Para cualquier matriz construida a partir de C_1, el *or*-vector V de 2 de sus filas satisface que $H(V) \geqslant 3$, dado que $H(V)$ será siempre igual a 3.

iii) Cualquier fila tomada de una matriz construida a partir de C_0 o de C_1 es indistingible, en el sentido de que su peso de Hamming será siempre igual a 2 y la probabilidad de que corresponda a una sombra de un pixel blanco o de un pixel negro es exactamente la misma.

Este modelo realiza una expansión de pixel de 1 a 4; es decir, un pixel de la imagen original ahora estará representado por cuatro pixeles. Además, nuevamente cada sombra deberá ser reestructurada de forma que sea una matriz de 2×2, con la finalidad de que en la práctica la imagen original no sea distorcionada. Al final del proceso un pixel blanco se recupera como un pixel $\frac{1}{2}$ negro y uno negro como $\frac{3}{4}$ negro. El resultado de aplicar esta solución es ilustrado a continuación:

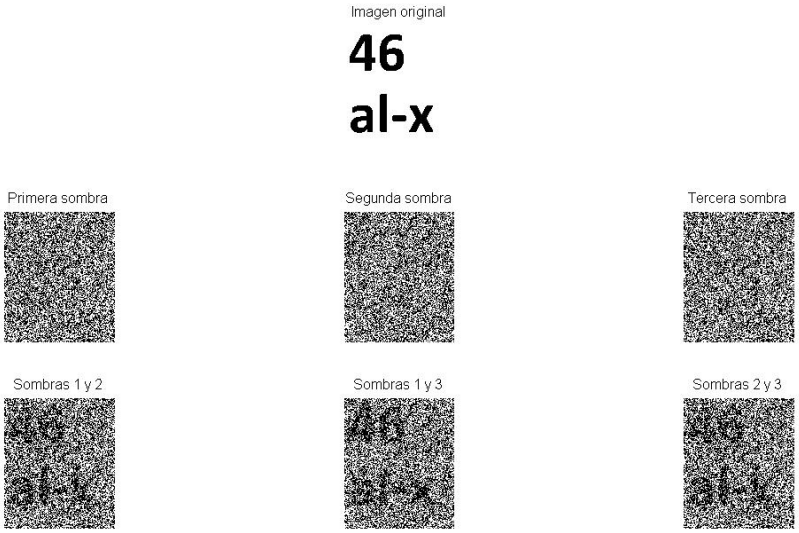


Figura 12: $(2,3)$-esquema visual umbral. Primer conjunto de matrices solución.

De forma análoga se puede demostrar que las siguientes colecciones de matrices de tamaño 3×4 son también una solución al problema $(2,3)$-esquema visual umbral.

$$C_{0_2} = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} \}$$

$$C_{1_2} = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \}$$

Al final del proceso un pixel blanco se recupera como un pixel $\frac{3}{4}$ negro y uno negro como $\frac{4}{4}$ negro. El resultado de aplicar esta solución es ilustrado a continuación:

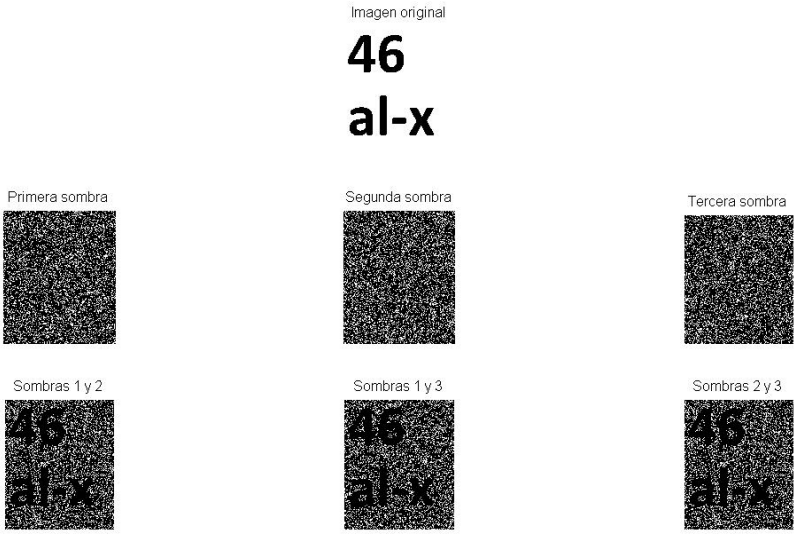


Figura 13: $(2,3)$-esquema visual umbral. Segundo conjunto de matrices solución.

2.3.3 Seis soluciones al (2,4)-esquema visual umbral

El primer modelo de solución al problema $(2,4)$-esquema visual umbral está dada por las siguientes colecciones de matrices de tamaño 4×9:

$$C_{0_1} = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \}$$

$$C_{1_1} = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \}$$

Las sombras quedan de la siguiente manera:

i) Cuando el pixel de la imagen original de la imagen binaria es blanco, las dos sombras correspondientes tendrán cuatro pixeles negros en común, tres pixeles blancos en común y dos pixeles serán complementarios.

ii) Cuando el pixel de la imagen original de la imagen binaria es negro, las dos sombras correspondientes tendrán dos pixeles negro en común, un pixel blanco en común y seis pixeles serán complementarios.

A continuación se exhibe que este modelo es una solución.

Dado que las matrices son de tamaño 4×9, se sigue que cada sombra está formada por 9 subpixeles, es decir, $m = 9$. Si se considera a $d = 8$ y $0 < \alpha < 1/2$, obsérvese que:

i) Para cualquier matriz construida a partir de C_0, el *or*-vector V de 2 de sus filas satisface que $H(V) < 7 - \alpha n$, dado que $H(V)$ será siempre igual a 6 y $6 < 7 - \alpha 4$.

ii) Para cualquier matriz construida a partir de C_1, el *or*-vector V de 2 de sus filas satisface que $H(V) \geqslant 7$, dado que $H(V)$ será siempre igual a 7.

iii) Cualquier fila tomada de una matriz construida a partir de C_0 o de C_1 es indistingible, en el sentido de que su peso de Hamming será siempre igual a 5 y la probabilidad de que corresponda a una sombra de un pixel blanco o de un pixel negro es exactamente la misma.

Este modelo realiza una expansión de pixel de 1 a 9; es decir, un pixel de la imagen original ahora estará representado por nueve pixeles. Además, nuevamente cada sombra deberá ser reestructurada de forma que sea una matriz de 3×3, con la finalidad de que en la práctica la imagen original no sea distorcionada. Al final del proceso un pixel blanco se recupera como un pixel $\frac{6}{9}$ negro y uno negro como $\frac{7}{9}$ negro. El resultado de aplicar esta solución es ilustrado a continuación:

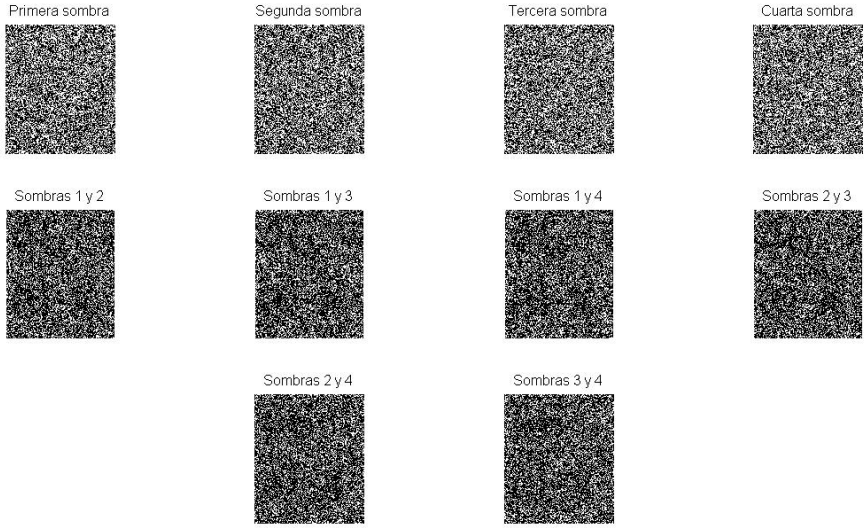


Figura 14: $(2,4)$-esquema visual umbral. Primer conjunto de matrices solución.

De forma análoga se puede demostrar que las siguientes colecciones de matrices de tamaño 4×9 son también una solución al problema $(2,4)$-esquema visual umbral.

$$C_{0_2} = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \}$$

y

$$C_{1_2} = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \}$$

Al final del proceso un pixel blanco se recupera como un pixel $\frac{4}{9}$ negro y uno negro como $\frac{6}{9}$ negro. El resultado de aplicar esta solución es ilustrado a continuación:

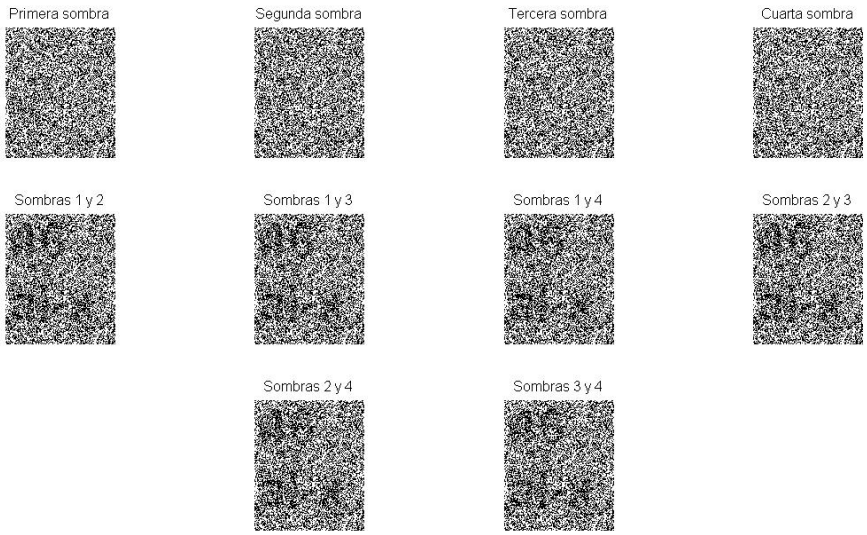


Figura 15: $(2,4)$-esquema visual umbral. Segundo conjunto de matrices solución.

$$C_{0_3} = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \}$$

y

$$C_{1_3} = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \}$$

Al final del proceso un pixel blanco se recupera como un pixel $\frac{5}{9}$ negro y uno negro como $\frac{7}{9}$ negro. El resultado de aplicar esta solución es ilustrado a continuación:

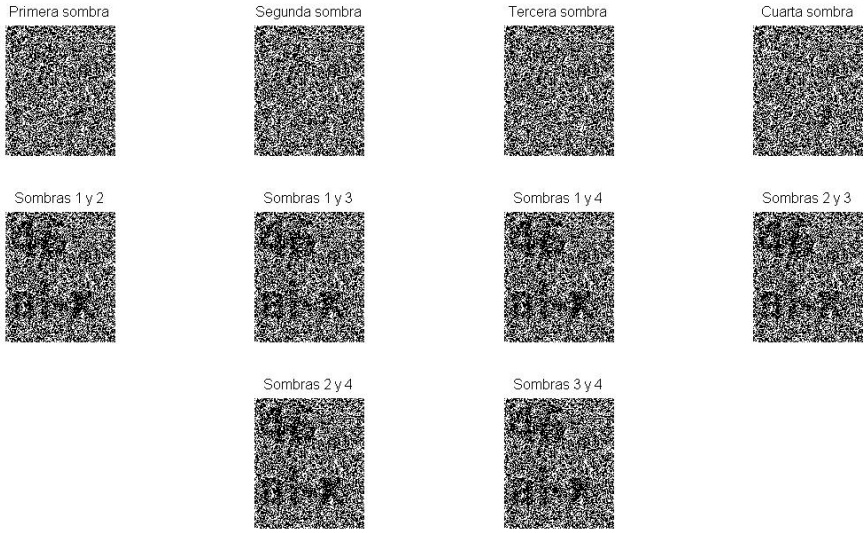


Figura 16: $(2,4)$-esquema visual umbral. Tercer conjunto de matrices solución.

$$C_{0_4} = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \}$$

y

$$C_{1_4} = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \}$$

Al final del proceso un pixel blanco se recupera como un pixel $\frac{6}{9}$ negro y uno negro como $\frac{8}{9}$ negro. El resultado de aplicar esta solución es ilustrado a continuación:

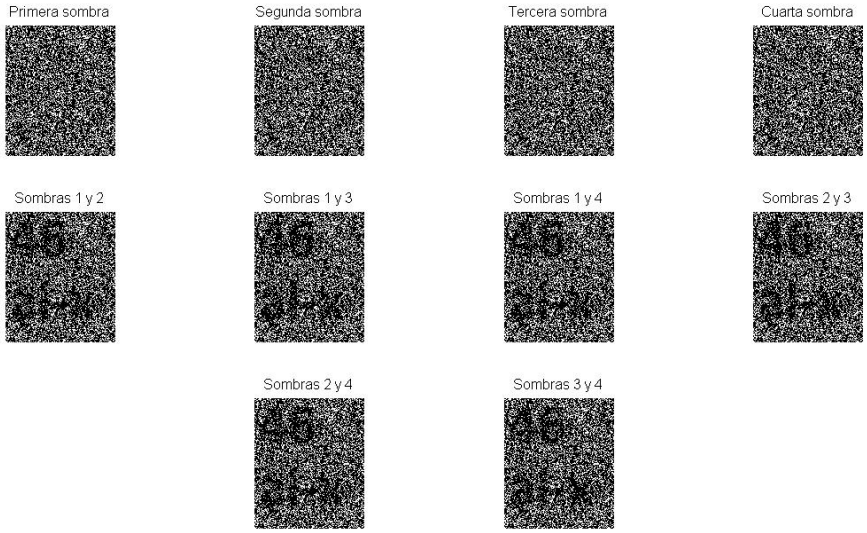


Figura 17: $(2,4)$-esquema visual umbral. Cuarto conjunto de matrices solución.

C_{0_5} ={Todas las matrices obtenidas de permutar a las columnas de $\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$ }

y

C_{1_6} ={Todas las matrices obtenidas de permutar a las columnas de $\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$ }

Al final del proceso un pixel blanco se recupera como un pixel $\frac{7}{9}$ negro y uno negro como $\frac{9}{9}$ negro. El resultado de aplicar esta solución es ilustrado a continuación:

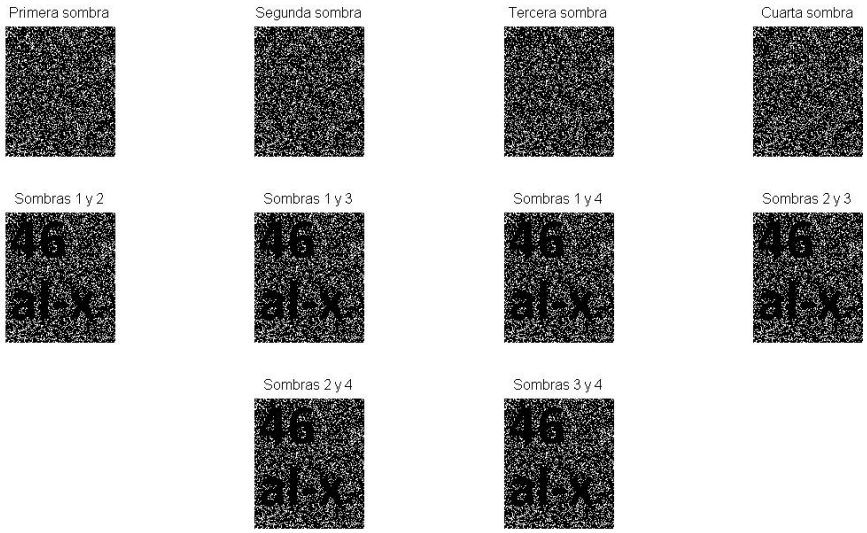


Figura 18: $(2,4)$-esquema visual umbral. Quinto conjunto de matrices solución.

La sexta solución propuesta al problema $(2,4)$-esquema visual umbral está dada por las siguientes colecciones de matrices de tamaño 4×6:

$$C_0 = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \}$$

$$C_1 = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \}$$

Las sombras quedan de la siguiente manera:

i) Cuando el pixel de la imagen original de la imagen binaria es blanco, las dos sombras correspondientes serán idénticas.

ii) Cuando el pixel de la imagen original de la imagen binaria es negro, las dos sombras correspondientes tendrán un pixel negro y uno blanco en común y cuatro pixeles serán complementarios.

A continuación se exhibe que este modelo es una solución.

Dado que las matrices son de tamaño 4×6, se sigue que cada sombra está formada por 6 subpixeles, es decir, $m = 6$. Si se considera a $d = 5$ y $0 < \alpha < 1/2$, obsérvese que:

i) Para cualquier matriz construida a partir de C_0, el *or*-vector V de 2 de sus filas satisface que $H(V) < 5 - \alpha n$, dado que $H(V)$ será siempre igual a 3 y $3 < 5 - \alpha 4$.

ii) Para cualquier matriz construida a partir de C_1, el *or*-vector V de 2 de sus filas satisface que $H(V) \geqslant 5$, dado que $H(V)$ será siempre igual a 5.

iii) Cualquier fila tomada de una matriz construida a partir de C_0 o de C_1 es indistingible, en el sentido de que su peso de Hamming será siempre igual a 3 y la probabilidad de que corresponda a una sombra de un pixel blanco o de un pixel negro es exactamente la misma.

Este modelo realiza una expansión de pixel de 1 a 6; es decir, un pixel de la imagen original ahora estará representado por seis pixeles. Sin embargo, dichas sombras no pueden ser reestructuradas de forma que sean matrices cuadradas y que su representación gráfica no distorcione a la imagen original. Por lo tanto, esta solución no funciona adecuadamente cuando es llevada a la práctica.

2.3.4 Una solución al (2,n)-esquema visual umbral

El problema $(2, n)$-esquema visual umbral puede ser resuelto por las siguientes dos colecciones de matrices de tamaño $n \times n$:

$$C_0 = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix} \}$$

$$C_1 = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \}$$

A continuación se exhibe que el modelo propuesto es una solución:

Dado que las matrices son de tamaño $n \times n$, se sigue que cada sombra está formada por n subpixeles, es decir, $m = n$. Si se considera a $d = 2$ y $0 < \alpha < 1/n$, obsérvese que:

i) Para cualquier matriz construidas a partir de C_0, el *or*-vector V de cualesquiera 2 de las n filas satisface que $H(V) < 2 - \alpha n$, dado que $H(V)$ será siempre igual a 1 y $1 < 2 - \alpha n$.

ii) Para cualquier matriz construidas a partir de C_1, el *or*-vector V de cualesquiera 2 de las n filas satisface que $H(V) \geqslant d$, dado que $H(V)$ será siempre igual a 2.

iii) Cualquier fila tomada de una matriz construida a partir de C_0 o de C_1 es indistingible, en el sentido de que su peso de Hamming será siempre igual a 1 y la probabilidad de que corresponda a una sombra de un pixel blanco o negro es exactamente la misma.

Hasta ahora se tiene que, en efecto, el modelo propuesto es una solución al problema de un $(2, n)$-esquema visul umbral. Sin embargo, obsérvese que una vez que se pone en práctica, se tiene que al sobreponer cualesquiera de las dos sombras correspondientes a un pixel blanco, la frecuencia de subpixeles negros o bien el peso de Hamming del *or*-vector obtenido será $H(V) = 1$; mientras que, para cualesquiera de las dos sombras correspondientes a un pixel negro se tendrá $H(V) = 2$. Considérese el caso para $n = 16$, en donde el sistema visual humano tendría que interpretar a un pixel como blanco cuando 15 de 16 subpixeles del mismo son blancos y como negro cuando 14 de cada 16 subpixeles son blancos y solo 2 son negros. Esta tarea es prácticamente imposible para el sistema visual humano. Por otro lado, si se considera a $n = 2$, las sombras obtenidas bajo esta solución no pueden ser reestructuradas de forma que sean matrices cuadradas, por lo que su representación gráfica distorcionaría a la imagen. En resumen, esta solución teórica llevada a la práctica podría no ser tan buena como se desea para n grandes o bien, cuando n no es un número cuadrado perfecto.

A continuación se presentará una técnica que resuelve el problema de una solución de un $(2, n)$-esquema umbral cuando n no es un número cuadrado perfecto.

2.3.5 Técnica de replicación para obtener una sombra sin distorción en un (2,n)-esquema visual umbral

Solución al (2,2)-esquema visual umbral

Obsérvese que si se considera al modelo mencionado anteriormente, para dar soluciones a $(2, n)$-esquemas visuales umbrales, el problema del $(2, 2)$-esquema visual umbral puede ser resuelto considerando a 2 subpixeles para cada pixel de la imagen. De esta forma la solución quedaría definida por las siguientes colecciones de matrices de tamaño 2×2:

C_0 ={Todas las matrices obtenidas de permutar a las columnas de $\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$ }

C_1 ={Todas las matrices obtenidas de permutar a las columnas de $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ }

Ya se ha demostrado de manera general que esta es una solución. Sin embargo, si se toma en cuenta que la imagen secreta es una imagen digital binaria cuyos pixeles son estructuras cuadradas, este modelo, en la práctica, distorcionaría a la imagen original. Por ello, la solución sugerida es utilizar 4 subpixeles por cada pixel. Tomando en cuenta lo anterior, una solución posible a este problema es duplicar la sombra y, posteriormente, reestructurarla en una matriz de 2×2. A continuación se ilustra esta idea con un ejemplo.

Ejemplo 2.2. *Si se desea compartir un pixel blanco y se considera a la matriz* $C_0 = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$ *ambas sombras serían* 01 *y duplicadas serían* 0101. *El or-vector V obtenido al sobreponer dichas sombras duplicadas sería nuevamente* 0101. *Esto gráficamente se representa:*

Si se desea compartir a un pixel negro y se considera a la matriz $C_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ *las sombras serían* 10 *y* 01 *que duplicadas quedarían como* 1010 *y* 0101. *El or-vector V obtenido al sobreponer dichas sombras duplicadas sería* 1111. *Esto gráficamente se representa:*

Debido al proceso de duplicación que se realiza, se puede observar que las sombras correspondientes son siempre verticales. Por lo tanto, esta solución resulta ser un caso especial de la primera solución propuesta, en el caso en el que las sombras son verticales. Si se lleva a cabo una reestructuración de las sombras en un orden diferente, se tendrían los otros dos casos en los que las sombras son horizontales o diagonales.

El resultado de aplicar esta solución es ilustrado a continuación:

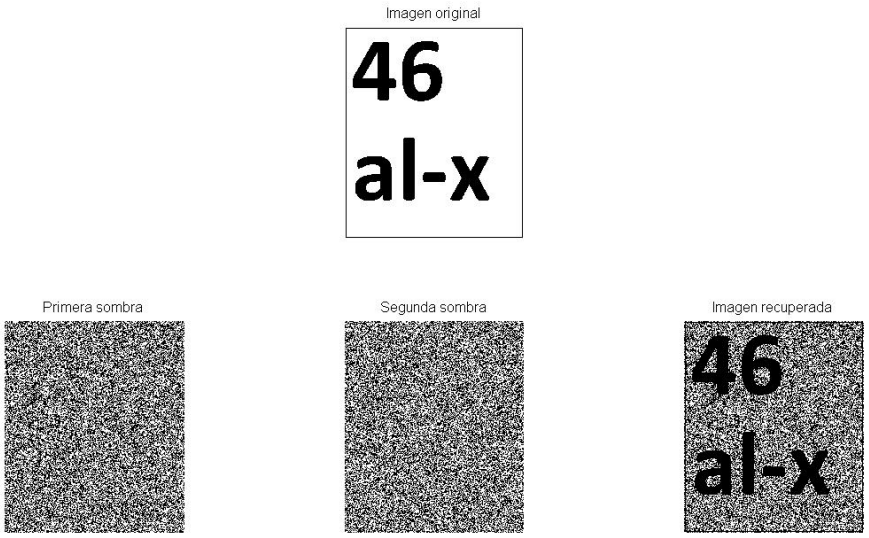


Figura 19: Técnica de replicación para obtener una sombra sin distorción en un $(2,2)$-esquema visual umbral

Solución al (2,3)-esquema visual umbral

El modelo mencionado anteriormente, para dar soluciones a $(2,n)$-esquemas umbrales, resuelve este problema. En esta ocasión se consideran a 3 subpixeles para cada pixel de la imagen. De esta forma la solución quedaría definida por las siguientes colecciones de matrices de tamaño 3×3:

$$C_0 = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \}$$

$$C_1 = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \}$$

Sin embargo, las sombras no pueden ser reestructuradas en matrices cuadradas y en la práctica esto distorcionaría a la imagen. Una solución a esta problemática es tripiclar a las sombras y, posteriormente, reestructurarlas en matrices de 3×3. En esta ocasión, la técnica provocaría que, finalmente, cada pixel fuera expandido de 1 a 9.

El resultado de aplicar esta solución es ilustrado a continuación:

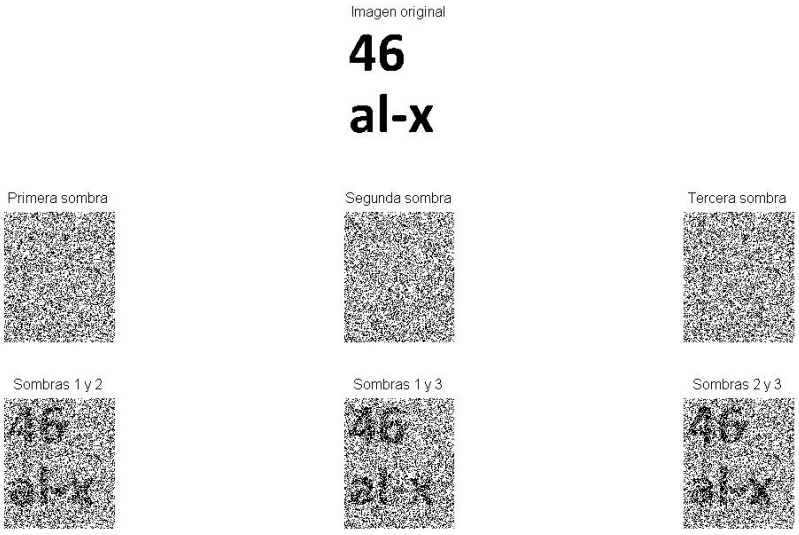


Figura 20: Técnica de replicación para obtener una sombra sin distorción en un $(2,3)$-esquema visual umbral

Solución al (2,4)-esquema visual umbral

En el caso de $n = 4$, la solución quedaría definida por las siguientes colecciones de matrices de tamaño 4×4:

$$C_0 = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \}$$

$$C_1 = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \}$$

En este caso, obsérvese que cada sombra podría ser reestructurada en una matriz de 2×2, por lo que la solución puede ser llevada a la práctica sin distorcionar a la imagen original y la técnica de replicación es innecesaria.

El resultado de aplicar esta solución es ilustrado a continuación:

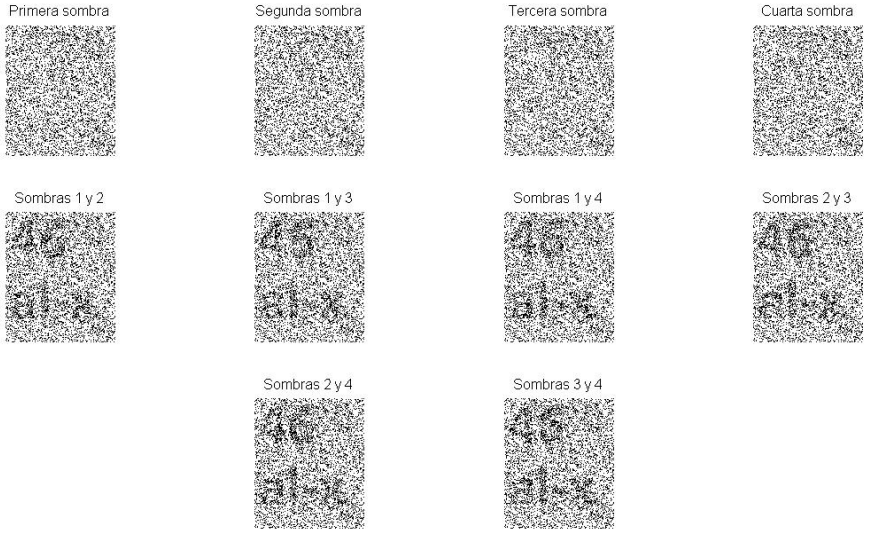


Figura 21: $(2,4)$-esquema visual umbral. No fue necesario usar la técnica de replicación.

En resumen, siempre que n no es un número cuadrado perfecto, la idea general es replicar la sombra n veces y luego reestructurarla en una matriz de $n \times n$. Es importante tener en cuenta que aunque esta técnica resuelve el problema de la distorción, la expansión del pixel pasa a ser de 1 a n^2.

2.3.6 Una solución al (3,3)-esquema visual umbral

Una solución al problema $(3,3)$-esquema visual umbral está dada por las siguientes colecciones de matrices de tamaño 3×4:

$$C_0 = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \}$$

$$C_1 = \{\text{Todas las matrices obtenidas de permutar a las columnas de } \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \}$$

A continuación se exhibe que este modelo es una solución.

Dado que las matrices son de tamaño 3×4, se sigue que cada sombra está formada por 4 subpixeles, es decir, $m = 4$. Si se considera a $d = 4$ y $0 < \alpha < 1/3$, obsérvese que:

i) Para cualquier matriz construida a partir de C_0, el *or*-vector V de las 3 filas satisface que $H(V) < 4 - \alpha n$, dado que $H(V)$ será siempre igual a 3 y $3 < 4 - \alpha 3$.

ii) Para cualquier matriz construida a partir de C_1, el *or*-vector V de las 3 filas satisface que $H(V) \geqslant 4$, dado que $H(V)$ será siempre igual a 4.

iii) Cualquier fila tomada de una matriz construida a partir de C_0 o de C_1 es indistingible, en el sentido de que su peso de Hamming será siempre igual a 2. Cualesquiera dos filas no aportan ninguna información acerca de su procedencia, ya que sin importar si fueron tomadas de una matriz construida a partir de C_0 o C_1, el *or*-vector V_2 de las 2 filas tendrá peso de Hamming igual a 3, por lo tanto, la probabilidad de que dichas sombras correspondan a un pixel negro o blanco es la misma.

Este modelo realiza una expansión de pixel de 1 a 4; es decir, un pixel de la imagen original ahora estará representado por cuatro pixeles. Además, cada sombra deberá ser reestructurada de forma que sea una matriz de 2×2, con la finalidad de que en la práctica la imagen original no sea distorcionada.

El resultado de aplicar esta solución es ilustrado a continuación:

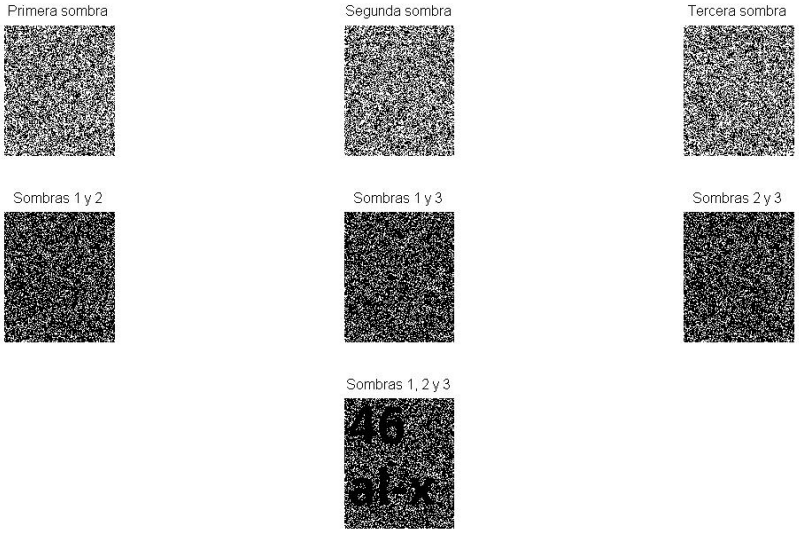


Figura 22: $(3,3)$-esquema visual umbral.

2.3.7 Una solución al (3,n)-esquema visual umbral

Este propuesta da solución al $(3,n)$-esquema visual umbral para un $n \geqslant 3$ arbitrario. Sea B la $n \times (n-2)$ matriz que contiene solo $1's$, y sea I la matriz identidad de tamaño $n \times n$. Sea BI la matriz $n \times (2n-2)$ obtenida de concatenar B e I, y sea $c(BI)$ el complemento Booleano de la matriz BI. Se propone la siguiente solución:

C_0 ={Todas las matrices obtenidas de permutar a las columnas de $c(BI)$ }

C_1 ={Todas las matrices obtenidas de permutar a las columnas de BI }

A continuación se exhibe que este modelo es una solución.

Dado que las matrices son de tamaño $n \times (2n-2)$, se sigue que cada sombra está formada por $(2n-2)$ subpixeles, es decir, $m = (2n-2)$. Si se considera a $d = n+1$ y $0 < \alpha < 1/m$, obsérvese que:

i) Para cualquier matriz construida a partir de C_0, el *or*-vector V de las 3 filas satisface que $H(V) < d - \alpha m$, dado que $H(V)$ será siempre igual a n y $n < d - \alpha m$.

ii) Para cualquier matriz construida a partir de C_1, el *or*-vector V de las 3 filas satisface que $H(V) \geqslant d$, dado que $H(V)$ será siempre igual a $n+1$.

iii) Cualquier fila tomada de una matriz construida a partir de C_0 o de C_1 es indistingible, en el sentido en el que su peso de Hamming será siempre igual a $n-1$. Cualesquiera dos filas no aportan ninguna información acerca de su procedencia, ya que sin importar si fueron tomadas de una matriz construida a partir de C_0 o C_1, el *or*-vector V_2 de las 2 filas tendrá peso de Hamming igual a n, por lo tanto, la probabilidad de que dichas sombras correspondan a un pixel negro o blanco es la misma.

Una solución al (3,4)-esquema visual umbral

Si se considera a la solución anterior y a $n = 4$ se tiene que $B_{(4\times 2)} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$ y $I_{(4\times 4)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

Por lo que la solución está dada por las siguientes colecciones de matrices de tamaño 4×6:

C_0 ={Todas las matrices obtenidas de permutar a las columnas de $c(BI) = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$ }

C_1 ={Todas las matrices obtenidas de permutar a las columnas de $BI = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$ }

Sin embargo, las sombras no pueden ser reestructuradas en matrices cuadradas y en la práctica esto distorcionaría a la imagen. Una solución a esta problemática es usar la técnica de replicación. En este caso se tendría que sextuplicar a las sombras y, posteriormente, reestructurarlas en matrices de 6×6. En esta ocasión, la técnica provocaría que, finalmente, cada pixel fuera expandido de 1 a 36.

El resultado de aplicar esta solución es ilustrado a continuación:

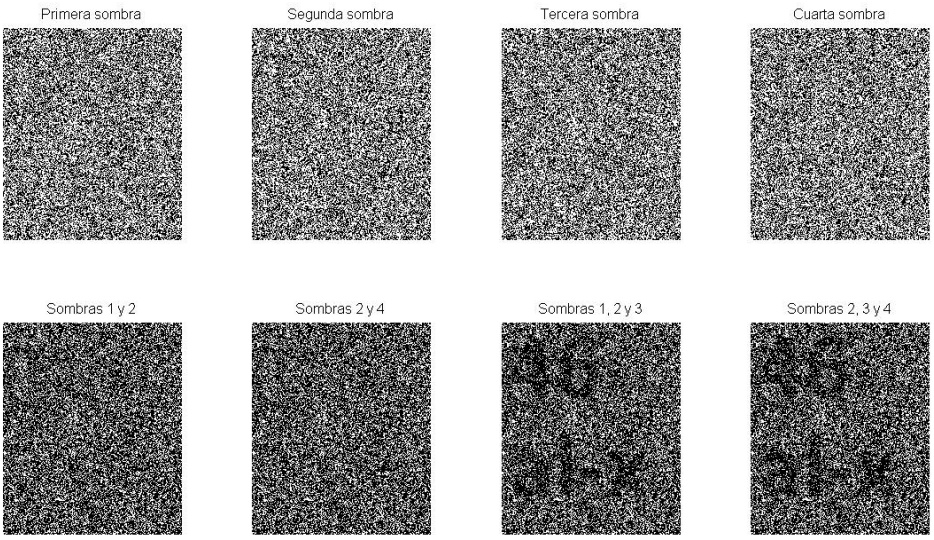


Figura 23: Técnica de replicación para obtener una sombra sin distorción en un $(3,4)$-esquema visual umbral

2.3.8 Una solución al (4,4)-esquema visual umbral

Una solución al problema $(4,4)$-esquema visual umbral está dada por las siguientes colecciones de matrices de tamaño 4×9:

C_0 ={Todas las matrices obtenidas de permutar a las columnas de $\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$ }

C_1 ={Todas las matrices obtenidas de permutar a las columnas de $\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$ }

A continuación se exhibe que este modelo es una solución.

Dado que las matrices son de tamaño 4×9, se sigue que cada sombra está formada por 9 subpixeles, es decir, $m = 9$. Si se considera a $d = 9$ y $0 < \alpha < 1/9$, obsérvese que:

i) Para cualquier matriz construida a partir de C_0, el *or*-vector V de las 4 filas satisface que $H(V) < 9 - \alpha m$, dado que $H(V)$ será siempre igual a 8 y $8 < 9 - \alpha 9$.

ii) Para cualquier matriz construida a partir de C_1, el *or*-vector V de las 4 filas satisface que $H(V) \geqslant 9$, dado que $H(V)$ será siempre igual a 9.

iii) Cualquier fila tomada de una matriz construida a partir de C_0 o de C_1 es indistingible, en el sentido de que su peso de Hamming será siempre igual a 5. Cualesquiera dos filas no aportan ninguna información acerca de su procedencia, ya que sin importar si fueron tomadas de una matriz construida a partir de C_0 o C_1, el *or*-vector V_2 de las 2 filas tendrá peso de Hamming igual

a 7, por lo tanto, la probabilidad de que dichas sombras correspondan a un pixel negro o blanco es la misma. Cualesquiera tres filas no aportan ninguna información acerca de su procedencia, ya que sin importar si fueron tomadas de una matriz construida a partir de C_0 o C_1, el *or*-vector V_3 de las 3 filas tendrá peso de Hamming igual a 8, por lo tanto, la probabilidad de que dichas sombras correspondan a un pixel negro o blanco es la misma.

Este modelo realiza una expansión de pixel de 1 a 9; es decir, un pixel de la imagen original ahora estará representado por nueve pixeles. Además, cada sombra deberá ser reestructurada de forma que sea una matriz de 3×3, con la finalidad de que en la práctica la imagen original no sea distorcionada.

El resultado de aplicar esta solución es ilustrado a continuación:

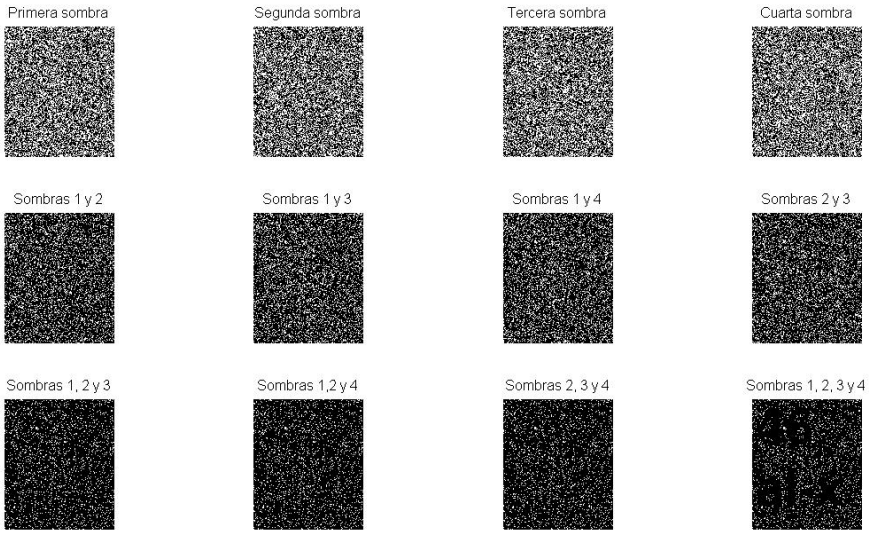


Figura 24: $(4,4)$-esquema visual umbral.

2.3.9 Análisis de resultados

Soluciones teóricas, ¿soluciones prácticas? No siempre. A lo largo de este capítulo se ha hecho notar que hay soluciones que satisfacen la defínción para ser solución a un problema de compartición de secretos visuales, sin embargo, cuando son llevadas a la práctica no funcionan. También se puede notar que si bien α (la pérdida de contraste) es un factor importante para la recuperación de la imagen por medio del sistema visual humano, también existen otros factores. Por ejemplo, de las dos soluciones propuestas al $(2,3)-$esquema visual umbral, para ambas soluciones $\alpha = 1$, pero la segunda solución es más fácil de visualizar. El mismo fenómeno puede ser observado para las seis soluciones propuestas para el $(2,4)-$esquema visual umbral donde se observa que cuando mayor es el peso de Hamming del pixel negro recuperado, la visibilidad de la imagen recuperada mejora.

Como parte de la problemática de la distorción fue propuesta la técnica de replicación, que puede ser utilizada sin sacrificar la definición de la imagen recuperada. Sin embargo, la parte que sí se ve afectada es la de la memoria computacional. Esta medida podría no parecer muy importante, pero considérese a una fotografía estándar tomada con una cámara Nikon D4S. La medida de esta fotografía es de $4\,928 \times 3\,280$ pixeles y la medida de su correspondiente imagen digital binaria es la misma. Si se desea compartir esta imagen entre un conjunto de 6 participantes de modo que cuando 2 de ellos junten sus partes puedan tener acceso a la imagen secreta, la técnica de replicación sería necesaria para obtener sombras sin distorción. Esto significa que para la imagen que consta de 16 163 840 pixeles, las sombras sin la técnica de replicación tendrían $16\,163\,840 \times 6 = 96\,983\,040$ pixeles y con la técnica de replicación tendrían $16\,163\,840 \times 6 \times 6 = 581\,898\,240$ pixeles. Este aumento en la memoria es bastante significativo.

También se puede observar que mientras mayor sea el número de partipantes, la calidad de la imagen recuperada empeora. Tal es el caso del $(4,4)-$esquema visual umbral, que como se muestra en la figura 24, la imagen recuperada es apenas distinguible.

2.4 PROGRAMAS EN MATLAB

La programación de los algoritmos se desarrolló en el lenguaje de MATLAB®.

A continuación se anexan cuatro funciones que serán utilizadas, más adelante, en los programas:

```
function Ab=blancoynegro(An)
%Funci\'on que recibe a una imagen An y devuelve a Ab que es la versión de
%An en blanco y negro
[x,y,z]=size(An);
for i=1:x
    for j=1:y
        a=An(i,j,1);
        if(a<254)
            Ab(i,j)=0;
        else
            Ab(i,j)=1;
        end
    end
end
```

```
function A=negativo(Ab)
%Funci\'on que recibe a una imagen Ab y devuelve a A que es la imagen en
%negativo de Ab
[x,y]=size(Ab);
for i=1:x
    for j=1:y
        a=Ab(i,j);
        if(a==0)
            A(i,j)=1;
        else
            A(i,j)=0;
        end
    end
end
```

```
function P1=permutacion(P)
%Esta funci\'on recibe a una matriz P y devuelve a P1 que es una matriz
%obtenida de permutar aleatoriamente a las columnas de P
[x,y]=size(P);
v2=randperm(y);
for k=1:y
P1(:,k)=P(:,(v2(k)));
end
```

```
function I=unirsombras(S1,S2)
%Funci\'on que une a las sombras S1 y S2
[x,y]=size(S1);
for i=1:x
    for j=1:y
        a=S1(i,j)||S2(i,j);
        if(a==1)
        I(i,j)=1;
        else
         I(i,j)=0;
        end
    end
```

```
end
```

2.4.1 Programa para un (2,2)-esquema visual umbral

El siguiente programa genera dos sombras, mismas que deben ser repartidas entre los dos participantes. Cuando las sombras son sobrepuestas, un pixel blanco es recuperado como un pixel $\frac{1}{2}$ negro y uno negro es completamente negro.

```
%2.3.1 Programa para un (2,2)-esquema visual umbral

%Se obtiene la matriz correspondiente a la imagen.ext y la de su versi\'on
%en blanco y negro:
An=imread('imagen.ext');
Ab=blancoynegro(An);

%Se calcula el negativo de la imagen, ya que en el algoritmo estudiado el 0
%es blanco y el 1 es negro pero en Matlab dichos valores representan lo
%contrario. Por tanto, en este programa se trabajar\'a con el negativo de
%la imagen:
A=negativo(Ab);

%Se calcula el tamaño de la matriz A:
[x,y]=size(A);

%Matrices soluci\'on:
C0=[1 1 0 0; 1 1 0 0];
C1=[1 1 0 0; 0 0 1 1];

%Construcci\'on de las sombras (en negativo):
for i=1:x
    for j=1:y
if (A(i,j)==0)
    C=permutacion(C0);
    M1=reshape(C(1,:),[2,2]);
    M2=reshape(C(2,:),[2,2]);

else
    C=permutacion(C1);
    M1=reshape(C(1,:),[2,2]);
    M2=reshape(C(2,:),[2,2]);
end

p=0;
for n=(i*2)-1:(i*2)
    p=p+1;
    q=0;
    for m=(j*2)-1:(j*2)
        q=q+1;
S1n(n,m)=M1(p,q);
S2n(n,m)=M2(p,q);
    end
end
    end
end

%Sombras listas para ser impresas. Para visualizarlas util\'icese la
```

```
%instrucci\'on imshow(sombra):
S1=negativo(S1n);
S2=negativo(S2n);

%Sobreposici\'on de las dos sombras. Para visualizar dicha sobreposici\'on
%util\'icese imshow(I).
In=unirsombras(S1n,S2n);
I=negativo(In);
```

2.4.2 Programa para un (2,3)-esquema visual umbral

El siguiente programa genera tres sombras, mismas que deben ser repartidas entre los tres participantes. La sobreposición de dos de estas tres sombras revela información acerca de la imagen original. El programa funciona para los dos conjuntos de matrices solución propuestos. Sin embargo, la selección del conjunto solución a la hora de ejecutar el programa se ve reflejado en la resolución de la imagen recuperada.

```
%2.3.2 Programa para un (2,3)-esquema visual umbral

%Se obtiene la matriz correspondiente a la imagen.ext y la de su versi\'on
%en blanco y negro:
An=imread('imagen.ext');
Ab=blancoynegro(An);

%Se calcula el negativo de la imagen, ya que en el algoritmo estudiado el 0
%es blanco y el 1 es negro pero en Matlab dichos valores representan lo
%contrario. Por tanto, en este programa se trabajar\'a con el negativo de
%la imagen:
A=negativo(Ab);

%Se calcula el tamaño de la matriz A:
[x,y]=size(A);

%A continuaci\'on se proponen dos conjuntos de matrices soluci\'on:

%Con estas matrices un pixel blanco se recupera como un pixel 1/2 negro y
%uno negro como 3/4 negro.
C0=[1 1 0 0; 1 1 0 0; 1 1 0 0];
C1=[1 1 0 0; 1 0 1 0; 0 1 1 0];

%Con estas matrices un pixel blanco se recupera como un pixel 3/4 negro y
%uno negro como 4/4 negro.
%C0=[1 1 1 0; 1 1 1 0; 1 1 1 0];
%C1=[1 1 1 0; 1 1 0 1; 1 0 1 1];

%Construcci\'on de las sombras (en negativo):
for i=1:x
    for j=1:y
if (A(i,j)==0)
    C=permutacion(C0);
    M1=reshape(C(1,:),[2,2]);
    M2=reshape(C(2,:),[2,2]);
    M3=reshape(C(3,:),[2,2]);

else
    C=permutacion(C1);
```

```
    M1=reshape(C(1,:),[2,2]);
    M2=reshape(C(2,:),[2,2]);
    M3=reshape(C(3,:),[2,2]);
end

p=0;
for n=(i*2)-1:(i*2)
    p=p+1;
    q=0;
    for m=(j*2)-1:(j*2)
        q=q+1;
S1n(n,m)=M1(p,q);
S2n(n,m)=M2(p,q);
S3n(n,m)=M3(p,q);
    end
end
    end
end

%Sombras listas para ser impresas. Para visualizarlas util\'icese la
%instrucci\'on imshow(sombra):
S1=negativo(S1n);
S2=negativo(S2n);
S3=negativo(S3n);

%Sobreposici\'on de 2 de las 3 sombras. Para visualizar dicha
%sobreposici\'on util\'icese imshow(I12), imshow(I13), imshow(I23).
I12n=unirsombras(S1n,S2n);
I12=negativo(I12n);

I13n=unirsombras(S1n,S3n);
I13=negativo(I13n);

I23n=unirsombras(S2n,S3n);
I23=negativo(I23n);
```

2.4.3 Programa para un (2,4)-esquema visual umbral

El siguiente programa genera cuatro sombras, mismas que deben ser repartidas entre los cuatro participantes. La sobreposición de dos de estas cuatro sombras revela información acerca de la imagen original. El programa funciona para los cuatro conjuntos de matrices solución propuestos. Sin embargo, la selección del conjunto solución a la hora de ejecutar el programa se ve reflejado en la resolución de la imagen recuperada.

```
%2.3.3 Programa para un (2,4)-esquema visual umbral

%Se obtiene la matriz correspondiente a la imagen.ext y la de su versi\'on
%en blanco y negro:
An=imread('imagen.ext');
Ab=blancoynegro(An);

%Se calcula el negativo de la imagen, ya que en el algoritmo estudiado el 0
%es blanco y el 1 es negro pero en Matlab dichos valores representan lo
%contrario. Por tanto, en este programa se trabajar\'a con el negativo de
%la imagen:
A=negativo(Ab);
```

```
%Se calcula el tamaño de la matriz A:
[x,y]=size(A);

%A continuaci\'on se proponen cinco conjuntos de matrices soluci\'on:

%Con estas matrices un pixel blanco se recupera como un pixel 6/9 negro y
%uno negro como 7/9 negro.
%C0=[1 1 1 1 1 0 0 0 0; 1 1 1 1 0 1 0 0 0; 1 1 1 1 0 0 1 0 0; 1 1 1 1 0 0 0 1 0];
%C1=[1 1 1 1 1 0 0 0 0; 1 1 1 0 0 1 1 0 0; 0 1 1 1 0 0 1 1 0; 0 0 1 1 1 1 1 0 0];

%Con estas matrices un pixel blanco se recupera como un pixel 4/9 negro y
%uno negro como 6/9 negro.
%C0=[1 1 1 1 0 0 0 0 0; 1 1 1 1 0 0 0 0 0; 1 1 1 1 0 0 0 0 0; 1 1 1 1 0 0 0 0 0];
%C1=[1 1 1 1 0 0 0 0 0; 1 1 0 0 1 1 0 0 0; 0 0 1 1 1 1 0 0 0; 0 1 1 0 0 1 1 0 0];

%Con estas matrices un pixel blanco se recupera como un pixel 5/9 negro y
%uno negro como 7/9 negro.
%C0=[1 1 1 1 1 0 0 0 0; 1 1 1 1 1 1 0 0 0; 1 1 1 1 1 1 0 0 0; 1 1 1 1 1 1 0 0 0];
%C1=[1 1 1 1 1 0 0 0 0; 1 1 1 0 0 1 1 0 0; 0 1 1 1 0 0 1 1 0; 0 0 1 1 1 1 1 0 0];

%Con estas matrices un pixel blanco se recupera como un pixel 6/9 negro y
%uno negro como 8/9 negro.
%C0=[1 1 1 1 1 1 0 0 0; 1 1 1 1 1 1 1 0 0; 1 1 1 1 1 1 1 0 0; 1 1 1 1 1 1 1 0 0];
%C1=[1 1 1 1 1 1 0 0 0; 1 1 1 0 0 1 1 1 0; 0 1 1 1 0 1 1 0 1; 0 0 1 1 1 1 1 1 0];

%Con estas matrices una pixel blanco se recupera como un pixel 7/9 negro y
%uno negro como 9/9 negro.
C0=[1 1 1 1 1 1 1 0 0; 1 1 1 1 1 1 1 1 0 0; 1 1 1 1 1 1 1 1 0 0; 1 1 1 1 1 1 1 1 0 0];
C1=[1 1 1 1 0 1 1 1 0; 1 1 1 0 1 1 1 0 1; 1 1 0 1 1 1 0 1 1; 1 0 1 1 1 0 1 1 1];

%Construcci\'on de las sombras (en negativo):
for i=1:x
    for j=1:y
if (A(i,j)==0)
    C=permutacion(C0);
    M1=reshape(C(1,:),[3,3]);
    M2=reshape(C(2,:),[3,3]);
    M3=reshape(C(3,:),[3,3]);
    M4=reshape(C(4,:),[3,3]);

else
    C=permutacion(C1);
    M1=reshape(C(1,:),[3,3]);
    M2=reshape(C(2,:),[3,3]);
    M3=reshape(C(3,:),[3,3]);
    M4=reshape(C(4,:),[3,3]);
end

p=0;
for n=(i*2)-1:(i*2)+1
    p=p+1;
    q=0;
    for m=(j*2)-1:(j*2)+1
        q=q+1;
S1n(n,m)=M1(p,q);
```

```
S2n(n,m)=M2(p,q);
S3n(n,m)=M3(p,q);
S4n(n,m)=M4(p,q);
    end
end
    end
end

%Sombras listas para ser impresas. Para visualizarlas util\'icese la
%instrucci\'on imshow(sombra):
S1=negativo(S1n);
S2=negativo(S2n);
S3=negativo(S3n);
S4=negativo(S4n);

%Sobreposici\'on de 2 de las 3 sombras. Para visualizar dicha
%sobreposici\'on util\'icese imshow(I12), imshow(I13), imshow(I14), etc.
I12n=unirsombras(S1n,S2n);
I12=negativo(I12n);

I13n=unirsombras(S1n,S3n);
I13=negativo(I13n);

I14n=unirsombras(S1n,S4n);
I14=negativo(I14n);

I23n=unirsombras(S2n,S3n);
I23=negativo(I23n);

I24n=unirsombras(S2n,S4n);
I24=negativo(I24n);

I34n=unirsombras(S3n,S4n);
I34=negativo(I34n);
```

2.4.4 Programa para un (2,n)-esquema visual umbral con técnica de replicación cuando es necesaria

El siguiente programa genera n sombras, mismas que deben ser repartidas entre los n participantes. La sobreposición de dos de estas n sombras revela información acerca de la imagen original. El programa decide si es necesario acudir a la técnica de replicación para llevar a cabo dicha tarea.

```
%2.3.4 y 2.3.5 Programa para un (2,n)-esquema visual umbral con t\'ecnica
%de replicaci\'on cuando es necesaria

%Se obtiene la matriz correspondiente a la imagen.ext y la de su versi\'on
%en blanco y negro:
An=imread('imagen.ext');
Ab=blancoynegro(An);

%Se calcula el negativo de la imagen, ya que en el algoritmo estudiado el 0
%es blanco y el 1 es negro pero en Matlab dichos valores representan lo
%contrario. Por tanto, en este programa se trabajar\'a con el negativo de
%la imagen:
A=negativo(Ab);

%Se calcula el tamaño de la matriz A:
```

```
[x,y]=size(A);

%El programa solicita el n\'umero de participantes:
n=input('Ingrese el número de participantes: ');

%Generaci\'on de las matrices soluci\'on:
C0=[ones(n,1) zeros(n,n-1)];
C1=eye(n);

%Se calcula la ra\'iz de n. Dicha informaci\'on se utiliza para decidir si
%la t\'ecnica de replicaci\'on es necesaria o no.
r=nthroot(n,2);

%Construcci\'on de las sombras (en negativo):
if mod(r,1)==0 %No necesita replicaci\'on
 for i=1:x
    for j=1:y
        if (A(i,j)==0)
            C=permutacion(C0);
            for k=1:n
                M(:,:,k)=reshape(C(k,:),[r,r]);
            end
        else
            C=permutacion(C1);
            for k=1:n
                M(:,:,k)=reshape(C(k,:),[r,r]);
            end
        end
        for k=1:n
            p=0;
            for f=(i*r)-(r-1):(i*r)
                p=p+1;
                q=0;
                for g=(j*r)-(r-1):(j*r)
                    q=q+1;
                    Sn(f,g,k)=M(p,q,k);
                end
            end
        end
    end
 end

else %Necesita replicación
for i=1:x
    for j=1:y
        if (A(i,j)==0)
            C=permutacion(C0);
            CD=C;
            for k=1:n-1
                CD=[CD C];
            end
            for k=1:n
                M(:,:,k)=reshape(CD(k,:),[n,n]);
            end
        else
            C=permutacion(C1);
```

```
            CD=C;
            for k=1:n-1
                CD=[CD C];
            end
            for k=1:n
                M(:,:,k)=reshape(CD(k,:),[n,n]);
            end
        end

        for k=1:n
            p=0;
            for f=(i*n)-(n-1):(i*n)
                p=p+1;
                q=0;
                for g=(j*n)-(n-1):(j*n)
                    q=q+1;
                    Sn(f,g,k)=M(p,q,k);
                end
            end
        end
    end
 end

end

%La generaci\'on de las sombras se hace manual dependiendo del valor de n.
%Para visualizarlas util\'icese la instrucci\'on imshow(sombra):
S1=negativo(Sn(:,:,1));
S2=negativo(Sn(:,:,2));
%S3=negativo(Sn(:,:,3));
%S4=negativo(Sn(:,:,4));
%S5=negativo(Sn(:,:,5));
%...
%Sn=negativo(Sn(:,:,n));

%Para visualizar la sobreposici\'on de dos sombras util\'cese la
%instrucci\'on:
%imshow(negativo(unirsombras(Sn(:,:,p),Sn(:,:,q)))), donde p es el n\'umero
%de la primera sombra y q el de la segunda sombra.
```

2.4.5 Programa para un (3,3)-esquema visual umbral

El siguiente programa genera tres sombras, mismas que deben ser repartidas entre los tres participantes. La sobreposición de estras tres sombras generadas revela información acerca de la imagen original. Sin embargo, sobreponer dos de las tres sombras no revela ninguna información acerca de la misma.

```
%2.3.6 Programa para un (3,3)-esquema visual umbral

%Se obtiene la matriz correspondiente a la imagen.ext y la de su versi\'on
%en blanco y negro:
An=imread('imagen.ext');
Ab=blancoynegro(An);

%Se calcula el negativo de la imagen, ya que en el algoritmo estudiado el 0
%es blanco y el 1 es negro pero en Matlab dichos valores representan lo
%contrario. Por tanto, en este programa se trabajar\'a con el negativo de
```

```
%la imagen:
A=negativo(Ab);

%Se calcula el tamaño de la matriz A:
[x,y]=size(A);

%Matrices soluci\'on:
C0=[1 1 0 0; 1 0 1 0; 0 1 1 0];
C1=[1 1 0 0; 1 0 1 0; 1 0 0 1];

%Construcci\'on de las sombras (en negativo):
for i=1:x
    for j=1:y
if (A(i,j)==0)
    C=permutacion(C0);
    M1=reshape(C(1,:),[2,2]);
    M2=reshape(C(2,:),[2,2]);
    M3=reshape(C(3,:),[2,2]);

else
    C=permutacion(C1);
    M1=reshape(C(1,:),[2,2]);
    M2=reshape(C(2,:),[2,2]);
    M3=reshape(C(3,:),[2,2]);
end

p=0;
for n=(i*2)-1:(i*2)
    p=p+1;
    q=0;
    for m=(j*2)-1:(j*2)
        q=q+1;
S1n(n,m)=M1(p,q);
S2n(n,m)=M2(p,q);
S3n(n,m)=M3(p,q);
    end
end
    end
end

%Sombras listas para ser impresas. Para visualizarlas util\'icese la
%instrucci\'on imshow(sombra):
S1=negativo(S1n);
S2=negativo(S2n);
S3=negativo(S3n);

%Sobreposici\'on de las tres sombras. Para visualizar dicha
%sobreposici\'on util\'icese imshow(I).
[x,y]=size(S1n);
for i=1:x
    for j=1:y
        a=S1n(i,j)||S2n(i,j)||S3n(i,j);
        if(a==1)
        In(i,j)=1;
        else
         In(i,j)=0;
```

```
        end
    end
end
I=negativo(In);

%Para visualizar que la sobreposici\'on de dos sombras no revela ninguna
%informaci\'on util\'icense las instrucciones:
%imshow(negativo(unirsombras(S1n,S2n))),
%imshow(negativo(unirsombras(S1n,S3n))),
%imshow(negativo(unirsombras(S2n,S3n))).
```

2.4.6 Programa para un (3,n)-esquema visual umbral con técnica de replicación cuando es necesaria

El siguiente programa genera n sombras, mismas que deben ser repartidas entre los n participantes. La sobreposición de tres de estas n sombras revela información acerca de la imagen original. Sin embargo, sobreponer dos de las tres sombras no revela ninguna información acerca de la misma. El programa decide si es necesario acudir a la técnica de replicación para llevar a cabo dicha tarea.

```
%2.3.7 Programa para un (3,n)-esquema visual umbral con t\'ecnica de
%replicaci\'on cuando es necesaria.

%Se obtiene la matriz correspondiente a la imagen.ext y la de su versi\'on
%en blanco y negro:
An=imread('imagen.ext');
Ab=blancoynegro(An);

%Se calcula el negativo de la imagen, ya que en el algoritmo estudiado el 0
%es blanco y el 1 es negro pero en Matlab dichos valores representan lo
%contrario. Por tanto, en este programa se trabajar\'a con el negativo de
%la imagen:
A=negativo(Ab);

%Se calcula el tamaño de la matriz A:
[x,y]=size(A);

%El programa solicita el n\'umero de participantes:
n=input('Ingrese el número de participantes: ');

%Generaci\'on de las matrices soluci\'on:
BI=[ones(n,n-2) eye(n)];
%cBI
for i=1:n
    for j=1:(2*n-2)
        if (BI(i,j)==0)
            cBI(i,j)=1;
        else
            cBI(i,j)=0;
        end
    end
end

%Se calcula a la ra\'iz de 2*n-2. Dicha informaci\'on se utiliza para
%decidir si la t\'ecnica de replicaci\'on es necesaria o no.
r=nthroot(2*n-2,2);

%Construcci\'on de las sombras (en negativo):
```

```
if mod(r,1)==0 %No necesita replicaci\'on
 for i=1:x
    for j=1:y
        if (A(i,j)==0)
            C=permutacion(cBI);
            for k=1:n
                M(:,:,k)=reshape(C(k,:),[r,r]);
            end
        else
            C=permutacion(BI);
            for k=1:n
                M(:,:,k)=reshape(C(k,:),[r,r]);
            end
        end
        for k=1:n
            p=0;
            for f=(i*r)-(r-1):(i*r)
                p=p+1;
                q=0;
                for g=(j*r)-(r-1):(j*r)
                    q=q+1;
                    Sn(f,g,k)=M(p,q,k);
                end
            end
        end
    end
 end

else %Necesita replicaci\'on
for i=1:x
    for j=1:y
        if (A(i,j)==0)
            C=permutacion(cBI);
            CD=C;
            for k=1:(2*n-2)-1
                CD=[CD C];
            end
            for k=1:n
                M(:,:,k)=reshape(CD(k,:),[2*n-2,2*n-2]);
            end
        else
            C=permutacion(BI);
            CD=C;
            for k=1:(2*n-2)-1
                CD=[CD C];
            end
            for k=1:n
                M(:,:,k)=reshape(CD(k,:),[2*n-2,2*n-2]);
            end
        end

        for k=1:n
            p=0;
            for f=(i*(2*n-2))-((2*n-2)-1):(i*(2*n-2))
                p=p+1;
                q=0;
```

```
                for g=(j*(2*n-2))-((2*n-2)-1):(j*(2*n-2))
                    q=q+1;
                    Sn(f,g,k)=M(p,q,k);
                end
            end
        end
    end
end

end

%La generaci\'on de las sombras se hace manual dependiendo del valor de n.
%Para visualizarlas util\'icese la instrucci\'on imshow(sombra):
S1=negativo(Sn(:,:,1));
S2=negativo(Sn(:,:,2));
S3=negativo(Sn(:,:,3));
S4=negativo(Sn(:,:,4));
%S5=negativo(Sn(:,:,5));
%...
%Sn=negativo(Sn(:,:,n));

%Para visualizar que la sobreposici\'on de dos sombras no revela ninguna
%informaci\'on util\'icense la instrucc\'on:
%imshow(negativo(unirsombras(Sn(:,:,p),Sn(:,:,q)))), donde p es el n\'umero
%de la primera sombra y q el de la segunda sombra.

%Para visualizar la sobreposici\'on de 3 o m\'as sombras util\'icese la
%siguiente instrucci\'on:
[x,y]=size(Sn(:,:,1));
    for i=1:x
        for j=1:y
            a=Sn(i,j,1)||Sn(i,j,2)||Sn(i,j,3);%||Sn(i,j,4)||Sn(i,j,5)||
            %Sn(i,j,6)||... seg\'un las sombras que se deseen sobreponer
            if(a==1)
                In(i,j)=1;
            else
                In(i,j)=0;
            end
        end
    end
I=negativo(In);
```

2.4.7 Programa para un (4,4)-esquema visual umbral

El siguiente programa genera cuatro sombras, mismas que deben ser repartidas entre los cuatro participantes. La sobreposición de estras cuatro sombras generadas revela información acerca de la imagen original. Sin embargo, sobreponer menos de cuatro sombras no revela ninguna información acerca de la misma.

```
%2.3.8 Programa para un (4,4)-esquema visual umbral

%Se obtiene la matriz correspondiente a la imagen.ext y la de su versi\'on
%en blanco y negro:
An=imread('imagen.ext');
Ab=blancoynegro(An);
```

```
%Se calcula el negativo de la imagen, ya que en el algoritmo estudiado el 0
%es blanco y el 1 es negro pero en Matlab dichos valores representan lo
%contrario. Por tanto, en este programa se trabajar\'a con el negativo de
%la imagen:
A=negativo(Ab);

%Se calcula el tamaño de la matriz A:
[x,y]=size(A);

%Matrices soluci\'on:
C0=[1 1 1 1 1 0 0 0 0; 1 1 1 0 0 1 1 0 0; 1 1 0 1 0 1 0 1 0; 1 1 0 0 1 0 1 1 0];
C1=[1 1 1 1 0 1 0 0 0; 1 1 1 0 1 0 1 0 0; 1 1 0 1 1 0 0 1 0; 1 0 1 1 1 0 0 0 1];

%Construcci\'on de las sombras (en negativo):
for i=1:x
    for j=1:y
if (A(i,j)==0)
    C=permutacion(C0);
    M1=reshape(C(1,:),[3,3]);
    M2=reshape(C(2,:),[3,3]);
    M3=reshape(C(3,:),[3,3]);
    M4=reshape(C(4,:),[3,3]);

else
    C=permutacion(C1);
    M1=reshape(C(1,:),[3,3]);
    M2=reshape(C(2,:),[3,3]);
    M3=reshape(C(3,:),[3,3]);
    M4=reshape(C(4,:),[3,3]);
end

p=0;
for n=(i*2)-1:(i*2)+1
    p=p+1;
    q=0;
    for m=(j*2)-1:(j*2)+1
        q=q+1;
S1n(n,m)=M1(p,q);
S2n(n,m)=M2(p,q);
S3n(n,m)=M3(p,q);
S4n(n,m)=M4(p,q);
    end
end
    end
end

%Sombras listas para ser impresas:
S1=negativo(S1n);
S2=negativo(S2n);
S3=negativo(S3n);
S4=negativo(S4n);

%Sobreposici\'on de dos sombras. Para visualizar que la sobreposici\'on de
%dos sombras no revela ninguna informaci\'on util\'icense las instrucciones:
%imshow(I12), imshow(13), imshow(14), etc.
I12n=unirsombras(S1n,S2n);
```

```
I12=negativo(I12n);

I13n=unirsombras(S1n,S3n);
I13=negativo(I13n);

I14n=unirsombras(S1n,S4n);
I14=negativo(I14n);

I23n=unirsombras(S2n,S3n);
I23=negativo(I23n);

I24n=unirsombras(S2n,S4n);
I24=negativo(I24n);

I34n=unirsombras(S3n,S4n);
I34=negativo(I34n);

%Sobreposici\'on de tres sombras. Para visualizar que la sobreposici\'on de
%dos sombras no revela ninguna informaci\'on util\'icense las instrucciones:
%imshow(I123), imshow(124) y imshow(234).
[x,y]=size(S1n);
    for i=1:x
        for j=1:y
            a=S1n(i,j)||S2n(i,j)||S3n(i,j);
            if(a==1)
                I123n(i,j)=1;
            else
                I123n(i,j)=0;
            end
        end
    end
    I123=negativo(I123n);

    for i=1:x
        for j=1:y
            a=S1n(i,j)||S2n(i,j)||S4n(i,j);
            if(a==1)
                I124n(i,j)=1;
            else
                I124n(i,j)=0;
            end
        end
    end
    I124=negativo(I124n);

    for i=1:x
        for j=1:y
            a=S2n(i,j)||S3n(i,j)||S4n(i,j);
            if(a==1)
                I234n(i,j)=1;
            else
                I234n(i,j)=0;
            end
        end
    end
    I234=negativo(I234n);
```

```
%Sobreposici\'on de cuatro sombras. Para visualizar la sobreposici\'on de
%las cuatro sombras util\'icese la instrucci\'on imshow(I1234).
[x,y]=size(S1n);
    for i=1:x
        for j=1:y
            a=S1n(i,j)||S2n(i,j)||S3n(i,j)||S4n(i,j);
            if(a==1)
                I1234n(i,j)=1;
            else
                I1234n(i,j)=0;
            end
        end
    end
    I1234=negativo(I1234n);
```

CONCLUSIONES

Como se ha expuesto a lo largo de este trabajo, los esquemas de compartición de secretos tienen la cualidad de ser un método de compartición de información seguro; en el sentido de que, si no se tienen las partes necesarias para su decodificación, no hay cálculo que averigüe información alguna del secreto, sin importar el poder computacional o analítico que sea puesto en práctica. Uno de sus grandes beneficios es que, para los esquemas de compartición de secretos visuales, la decodificación de la imagen secreta es visual. Es decir, una vez que fueron entregadas las sombras a cada participante, un subconjunto autorizado podrá tener acceso a la imagen recuperada por medio de la sobreposición de sus sombras y, para ello, no es necesario ningún tipo de conocimiento criptográfico.

A lo largo del desarrollo de esta tesis pude observar que a pesar de que el tema ha ganado popularidad en los últimos años, se cuenta con muy pocos registros de soluciones planteadas a problemas concretos de compartición de secretos visuales. En la mayoría de los textos, se proponen las mismas soluciones y se plantean incluso los mismos ejemplos. Por ello, fue interesante estudiar a fondo la teoría matemática que está detrás de estas soluciones y encontrar que existe un mundo de posibilidades para el desarrollo y mejora de las mismas.

Como estudiante de la licenciatura en Matemáticas Aplicadas, considero que hay en el estudio de este tema una oportunidad para llevar a cabo la implementación de conceptos algebraicos abstractos a problemas del mundo real. Este trabajo representó para mí un reto: el reto de llevar la teoría matemática a lo aplicable, a lo programable. Espero que esta insistencia de vincular a la teoría con la práctica sea una invitación a mis compañeros a dirigir sus esfuerzos en el desarrollo de soluciones matemáticas que, además de teóricas, puedan ser aplicadas.

BIBLIOGRAFÍA

[AM08] José de Jesús Ángel and Guillermo Morales. Breve descripción de la criptografía en la revolución mexicana. *Revista Digital Universitaria*, 9, 2008.

[CC16] Germán Cejudo Castilla. Introducción a las retículas modulares. (tesis de licenciatura), Benemérita Universidad Autónoma de Puebla, 2016.

[CE49] Shannon Claude E. Communication theory of secret systems. *Bell System Technical Journal*, 28:656–715, 1949.

[DR92] Stinson Douglas R. An explication of secret sharing schemes. *Designs, Codes and Cryptography*, 2:357–390, 1992.

[DR06] Stinson Douglas R. *Cryptography: Theory and Practice*. CRC press, 3 edition, 2006.

[GW02] Rafael C. González and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 2 edition, 2002.

[LG98] Rudolf Lidl and Pilz Günter. *Applied Abstrac Algebra*. Springer, 2 edition, 1998.

[McA04] Alasdair McAndrew. *An introduction to digital image processing with MATLAB*. Brooks/Cole, 1 edition, 2004.

[NS95] Moni Naor and Adi Shamir. Visual cryptography. *Advances in Cryptology – EUROCRYPT'94*, 950:1–12, 1995.

[PC03] Gil Pino C. *Introducción a la Criptografía*. Alfaomega, 2 edition, 2003.

[Poo06] David Poole. *Linear Algebra: A Modern Introduction*. Brooks/Cole, 2 edition, 2006.

[WY12] Jonathan Weir and WeiQi Yan. *Visual Cryptography and Its Applications*. bookboon.com, 1 edition, 2012.