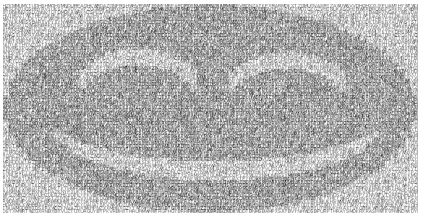


Proyecto Criptografía

Esquema de criptografía visual basado en letras

Autores:
David Solís Martín
Carlos Falgueras García

30 de mayo de 2014

Índice

1. Introducción **3**

2. VCS **5**

2.1. DVCS 6

2.1.1. Pseudocódigo 6

2.1.2. Ejemplos 7

2.2. PVCS 8

2.2.1. Pseudocódigo 8

2.2.2. Ejemplos 9

2.3. Matrices base 9

3. LVCS **12**

3.1. Transformación de matrices base 12

3.2. Mejoras 13

3.2.1. Condición de seguridad 13

3.2.2. Reelección de representantes 15

4. Implementación **17**

4.1. Características 18

4.2. Detalles 18

4.2.1. Generación de matrices base 19

4.2.2. Antialiasing y zoom 19

4.2.3. Resolución de resultados 19

5. Aplicaciones **20**

5.1. Mejora para alineación 22

6. Bibliografía **23**

7. Anexos **24**

7.1. Repositorio 24

1. Introducción

En este documento vamos a tratar el "esquema de criptografía visual" propuesto por *Moni Naor* y *Adi Shamir*[4] y una extensión del mismo donde cambiamos píxeles por letras[2]. También veremos algunos detalles de las implementaciones de ambos.

Esta técnica permite encriptar una imagen, y desencriptarla mecánicamente, sin la intervención de un ordenador [6]. Fue propuesta por *Moni Naor* y *Adi Shamir* en 1994 como una representación visual del "esquema de secreto compartido" propuesto por Shamir en 1979 [2][3].

Como se aprecia en el esquema de la figura 1, una imágen se cifra en N sombras. Harán falta un mínimo de K sombras para que, al superponerlas, se revele el secreto. Un número de sombras menor que K no revela ninguna información sobre la imagen original (Figura 2).

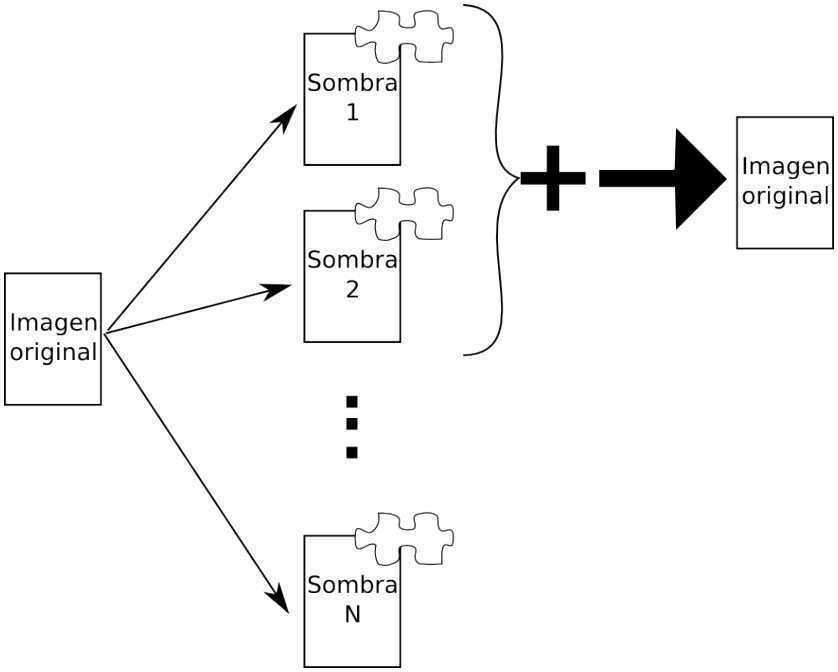
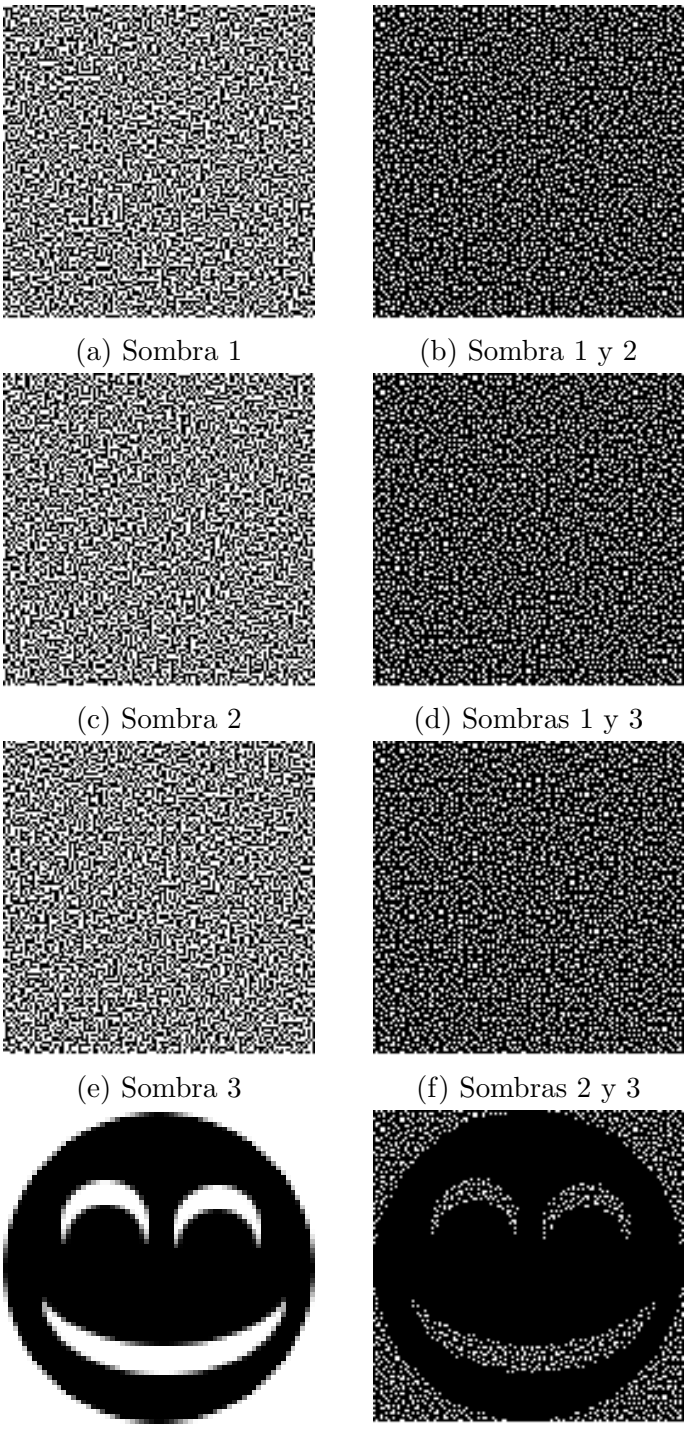


Figura 1: Esquema de cifrado y descrifrado



(a) Sombra 1

(b) Sombra 1 y 2

(c) Sombra 2

(d) Sombras 1 y 3

(e) Sombra 3

(f) Sombras 2 y 3

(g) Resultado (sombras 1, 2 y 3)

(h) Imagen original

Figura 2: Ejemplo de supersposición de sombras en un esquema (3,3)

2. Visual Cryptographic Scheme

En esta sección detallaremos el funcionamiento de los dos algoritmos de VCS: **DVCS**(*Deterministic Visual Cryptographic Scheme*) y **PVCS**(*Probabilistic Visual Cryptographic Scheme*).

Así como las condiciones que han de cumplir las *matrices base* sobre las cuales estriban estos algoritmos.

Como ya comentamos, ambos algoritmos se estiban en un par matrices denominadas "matrices base" para transformar cada píxel de la imagen original en un **superpíxel** de la sombra. Denominamos *superpíxel* al los píxeles que representarán al píxel original en las sombras y la imagen recuperada. Éste está compuesto por varios píxeles normales, negros y blancos (de ahí su nombre).

Dependiendo del número de píxeles negros y blancos que tenga un superpíxel de la imagen resultado, se verá más negruzco o más blancuzco. Los superpíxeles negruzcos representarán a los píxeles negros de la imagen original, y viceversa. Un ejemplo de esto puede verse en la figura 3.

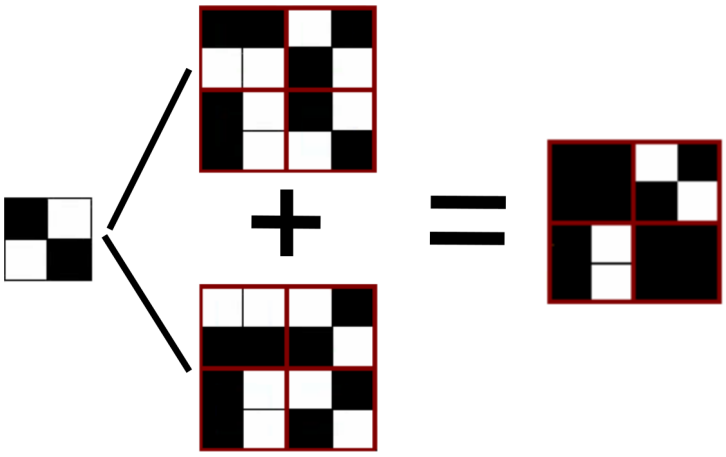


Figura 3: Ejemplo que ilustra como la superposición de dos sombras da como resultado una imagen que imita los píxeles negros y blancos de la original

2.1. Deterministic Visual Cryptographic Scheme

El algoritmo DVCS recibe al adjetivo "Determinista" porque, en contraposición a PVCS, los superpíxeles de la imagen resultante siempre tiene el mismo número de píxeles negros y blancos, dependiendo de que sea un superpíxel negro o blanco.

También se caracteriza por deformar las dimensiones de la imagen resultante, ya que cada píxel de la imagen original se sustituye por m píxeles en las sombras. Estos m píxeles se pueden distribuir en el superpíxel en una forma cuadrática, siempre y cuando m sea potencia de dos, así se conseguiría no distorsionar el aspecto de la imagen original.

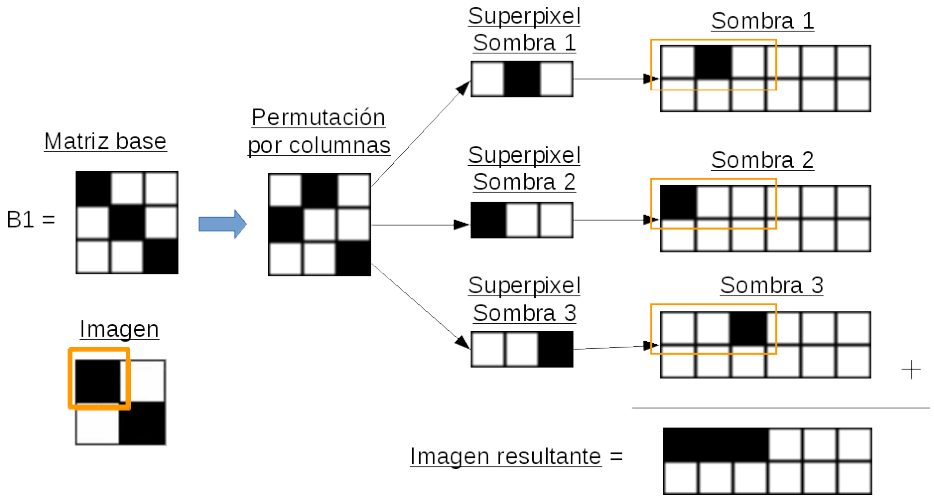
2.1.1. Pseudocódigo

Input:
Una imagen I secreta de $W \times H$
Las matrices base B_0 y B_1
Result:
N sombras S_i de tamaño $m \times W \times H$

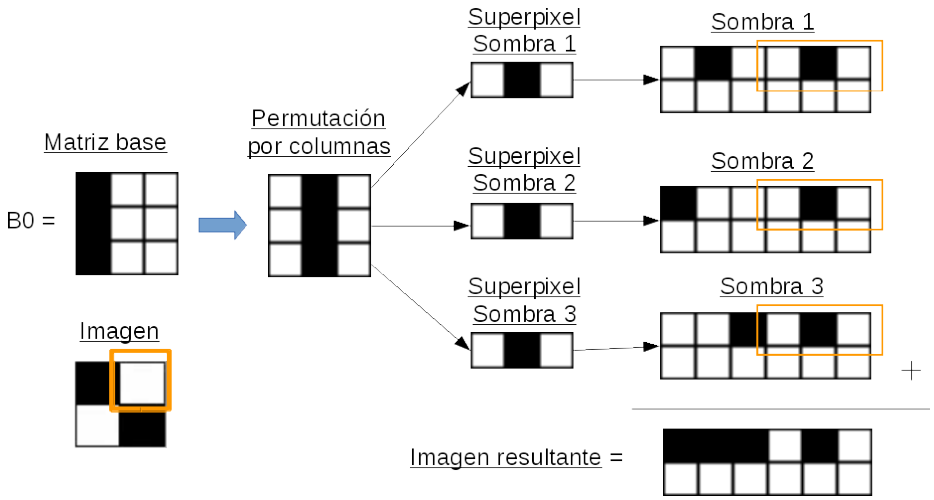
```
for all pixel px in I do
    if px == blanco then
        1. Seleccionar una permutación B_0^p aleatoria de B_0;
        2. Añadir la columna i-ésima de B_0^p (m píxeles) al super-píxel
        de la i-ésima sombra S_i;
    else
        1. Seleccionar una permutación B_1^p aleatoria de B_1;
        2. Añadir la columna i-ésima de B_0^p (m píxeles) al super-píxel
        de la i-ésima sombra S_i;
    end
end
return [S_1, S_2, ··· , S_N]
```

Algoritmo 1: DVCS

2.1.2. Ejemplos



(a) Cifrado de un píxel negro



(b) Cifrado de un píxel blanco

Figura 4: Ejemplos de cifrado con DVCS

2.2. Probabilistic Visual Cryptographic Scheme

El algoritmo PVCS es una variante de VCS en la cual el número de píxeles negros y blancos de un superpíxel no es determinista, sino que viene dado por una probabilidad (de ahí su nombre). Esto quiere decir que en un superpíxel negro de la imagen resultado habrá **probablemente** más píxeles negros y blancos, y al contrario en el caso de un superpíxel blanco.

Esto tiene sus ventajas e inconvenientes. Por un lado permite que la imagen resultante tenga la misma resolución que la original, y por otro lado es más difícil conseguir un contraste adecuado para apreciar con claridad el secreto, debido a que siempre hay cierta probabilidad de que un píxel blanco termine siendo negro o de que un píxel negro termine siendo blanco. En la figura 5b se muestra un ejemplo en el que ocurre esto último.

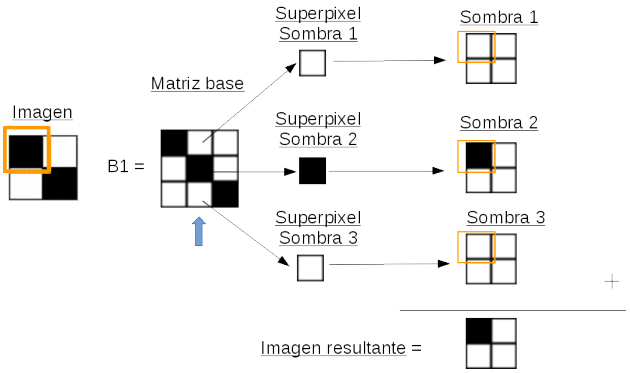
2.2.1. Pseudocódigo

Input:
Una imagen I secreta de $W \times H$
Las matrices base B_0 y B_1
Result:
N sombras S_i de tamaño $W \times H$

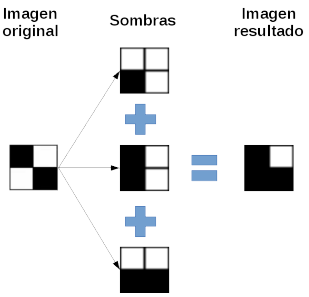
```
for all pixel px in I do
    if px == blanco then
        1. Seleccionar una columna C_j aleatoria de B_0 (con
        j ∈ [0, m − 1]);
        2. Añadir el píxel i-ésimo p_i de C_j (n píxeles) al superpíxel (1
        píxel) de la i-ésima sombra S_i;
    else
        1. Seleccionar una columna C_j aleatoria de B_0 (con
        j ∈ [0, m − 1]);
        2. Añadir el píxel i-ésimo p_i de C_j (n píxeles) al superpíxel (1
        píxel) de la i-ésima sombra S_i;
    end
end
return [S_1, S_2, ··· , S_N]
```

Algoritmo 2: PVCS

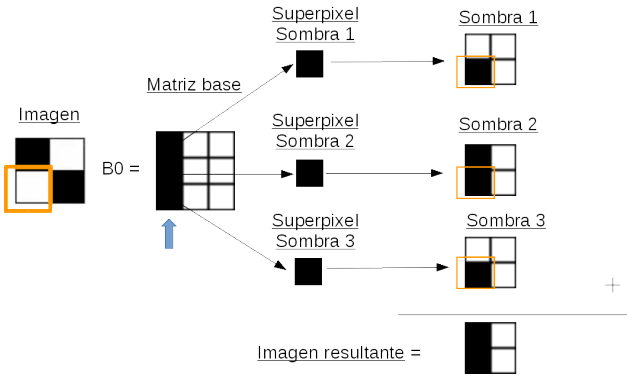
2.2.2. Ejemplos



(a) Cifrado de un píxel negro. Hay una probabilidad de 3/3 de escoger un píxel negro.



(b) Resultado de la superposición de las tres sombras. Se observa que un píxel blanco ha terminado siendo negro



(c) Cifrado de un píxel blanco. Hay una probabilidad de 2/3 de coger un píxel blanco (1/3 de equivocarse).

Figura 5: Ejemplos de cifrado con PVCS

2.3. Matrices base

Como hemos comentado anteriormente, las matrices base son unas matrices binarias que determinan el comportamiento de los algoritmos VCS. Éstas matrices tienen que tener dimensión $N \times m$, dónde N es el número

de sombras en un esquema (K, N) y m es el *factor expansión*, que determinará por cuantos píxeles se representará un píxel de la imagen original en los superpíxeles de la imagen resultado, en el caso del algoritmo DVCS.

Además de esto, han de cumplir dos condiciones:

1. **Condición de contraste**[1] *El número de píxeles negros de un superpíxel negro es mayor que el de un superpíxel blanco.* Esto asegura que en la imagen resultante se puedan distinguir los superpíxeles negros de los blancos.
2. **Condición de seguridad**[2]: *El número de píxeles negros es el mismo para cualquier superpixel resultante de la superposición de $K-1$ sombras o menos.* Esto asegura que si se tienen menos de K sombras, no se obtiene ninguna información sobre el secreto.

Si definimos la operación $OR(B|r)$ como el vector resultante de realizar la operación *or* por columnas, a r filas cualesquiera de la matriz B. Y definimos la operación $H(\cdot)$ como la distancia de hamming (el número de 1s del vector). Podemos definir las restricciones anteriores mediante las siguientes ecuaciones:

$$\begin{cases} H(OR(B_0|r)) \geq (m-l) \\ H(OR(B_0|r)) \leq (m-h) \end{cases} \text{, si } r = K \tag{1}$$
$$\text{Con: } 0 \leq l < h \leq m$$

$$H(OR(B_0|r)) = H(OR(B_1|r)) \text{ , si } r < K \tag{2}$$

Las constantes l y m determinan respectivamente:

- l: Número máximo de píxeles blancos en un superpíxel negro
- m: Número mínimo de píxeles blancos en un superpíxel blanco

Por lo que variando las mismas podemos ajustar la calidad da la imagen resultante.

Como veremos en la sección 4.2.1-Generación de matrices base, la generación de estas matrices es un problema muy complejo (vease [8]).

3. Letter based Visual Cryptographic Scheme

Éste algoritmo es el que se propone en *Hsiao-Ching Lin et al*[2]. Es idéntico a los algoritmos DVCS y PVCS que hemos visto, solo que cambian los píxeles por letras, de manera que dos letras distintas superpuestas van a enegrecer el superpíxel, mientras que dos letras superpuestas iguales se verán más blancuzcas.

3.1. Transformación de matrices base

Para sustituir los píxeles por letras hay que modificar las matrices base cambiando los 0s y 1s por letras. Para ello vamos a seguir el algoritmo 3.

Input:
Una matriz base binaria B de $N \times m$
Result:
Una matriz de letras L de $N \times m$ letras
1 1. Elegir un vector R de m letras, que representará los 0s de nuestra matriz
`/* Recorremos los elementos de la matriz base.` (i, j) `es la posición de` $e_{i,j}$ `en la matriz` B `*/`
2 **for** *all elements* $e_{i,j}$ *in* B **do**
3 **if** $e_{i,j} == 0$ **then**
`/* Sustituimos por la misma letra que el representante de esa columna */`
4 $L[i][j] = R[j]$
5 **else**
`/* Sustituimos por una letra aleatoria a todas las de la columna (incluido el representante) */`
6 $L[i][j] = L|L! = L[k][j] \wedge L! = R[j] conk \in [0, N-1]$
7 **end**
8 **end**
9 **return** L

Algoritmo 3: Algoritmo de transformación de matrices base binarias a matrices base de letras

A continuación mostramos un ejemplo de aplicación del algoritmo:

$$\text{Vector representante} = \begin{bmatrix} A & B & C \end{bmatrix}$$
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} D & B & C \\ A & F & C \\ A & B & G \end{bmatrix}$$

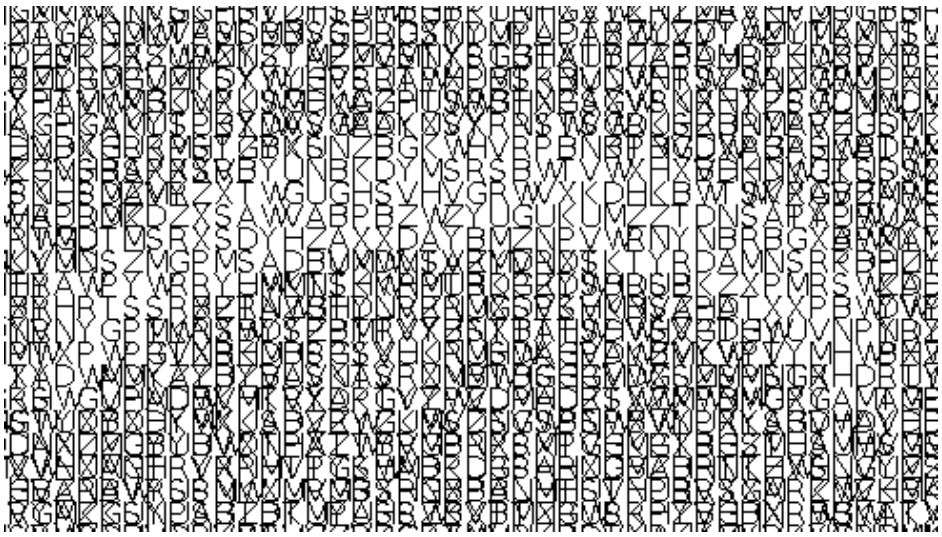
3.2. Mejoras

El artículo de *Hsiao-Ching et al*[2] propone cierta mejora en la calidad del resultado del algoritmo, basada en la elección de determinadas matrices base. Además, nosotros hemos introducido otra mejora en la transformación de matrices base binarias a matrices de letras. Estas dos mejoras las explicamos en las siguientes secciones.

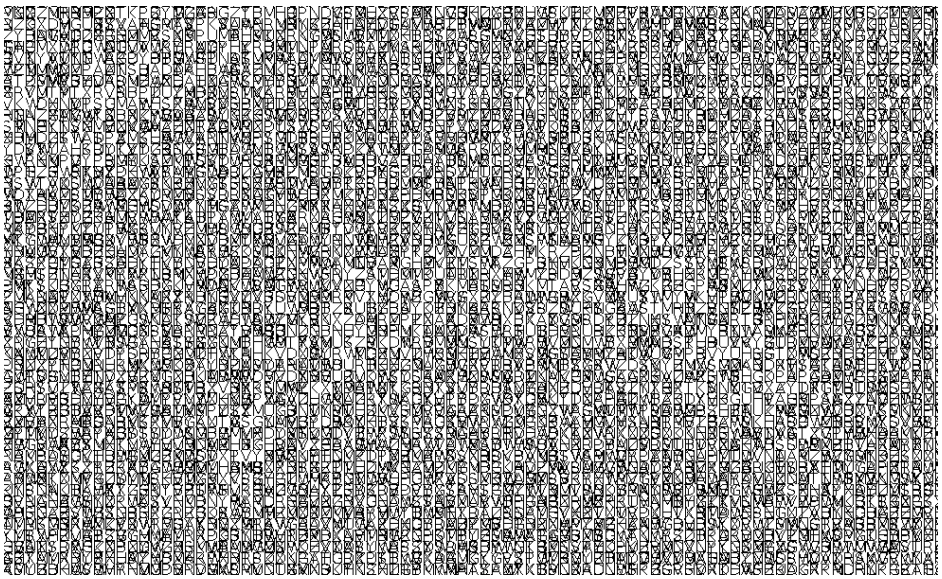
3.2.1. Condición de seguridad

La condición de seguridad que explicábamos en 2.3-Matrices base (equación 2) tiene una excepción al pasar las matrices binarias a matrices de letras. Esto es debido a que no es posible calcular la distancia de hamming a una fila de letras, mientras que sí lo es en una de píxeles. Un "píxel" negro con letras viene representado por una superposición de letras distintas, y uno blanco por letras iguales, así que no tiene sentido el concepto de "píxel" cuando no hay superposición. Por lo tanto podemos admitir que la condición de seguridad siempre se cumple para $r = 1$, ya que una sola sombra de letras no puede revelar información sobre la imagen encriptada (a excepción de lo que explicamos en la próxima sección).

La mejora que proponen se basa en esta propiedad para construir matrices base en las que B_0 es una matriz de "0s", es decir, con letras iguales en todas sus columnas. Aunque esto no cumpliría la restricción de seguridad para los algoritmos DVCS y PVCS, si lo hace para LVCS, y además mejora notablemente la calidad de las imágenes resultantes en esquemas $(2, N)$, ya que los blancos son representados en su totalidad por letras superpuestas iguales. En las figuras 6 se muestra la diferencia entre una imagen resultado con la mejora y sin ella.



(a) Resultado con mejora



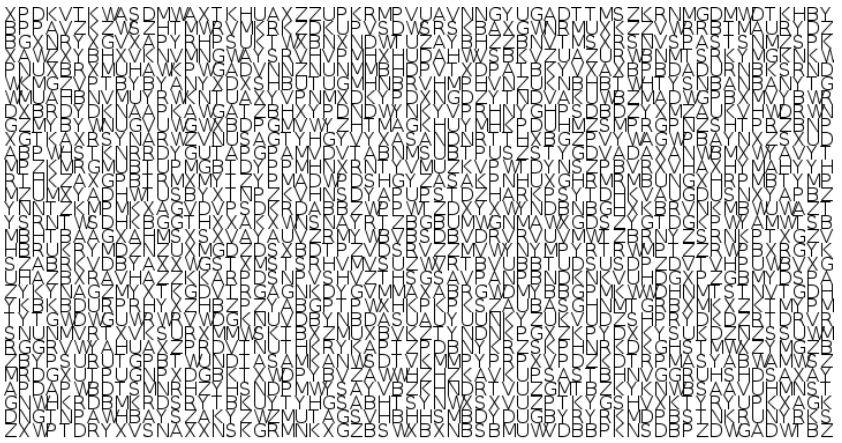
(b) Resultado sin mejora

Figura 6: Comparación de un cifrado con matrices base mejoradas y sin mejorar.

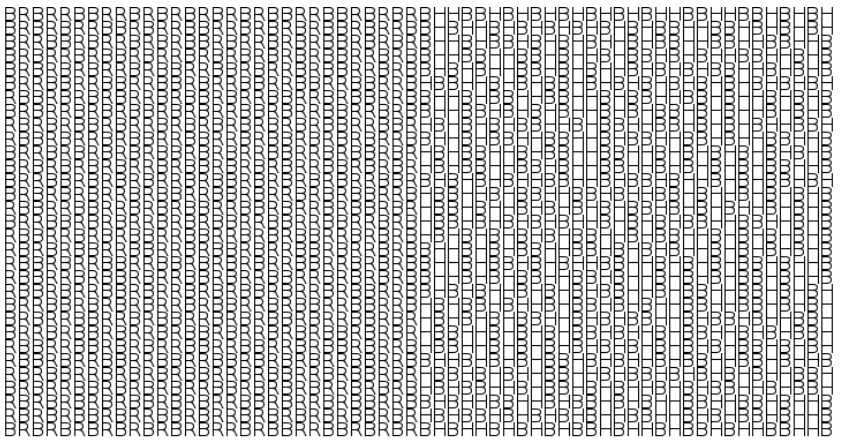
3.2.2. Reelección de representantes

Al transformar una matriz base binaria a una de letras, si para todos los superpíxeles que generamos elegimos el mismo representante para B_0 y otro distinto para B_1, el usar letras diferentes para representar el negro y el blanco, hace que podamos apreciar la imagen encriptada con una sola sombra. Esto se soluciona eligiendo un representante aleatorio distinto cada vez que tenemos que generar un nuevo superpíxel, siempre y cuando usemos el mismo para todas las sombras. En las figuras 7 se muestra el problema.

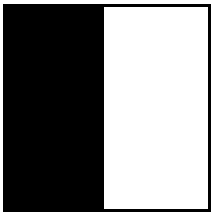
Aunque las imágenes que se muestran en el artículo [2] parecen tener en cuenta este hecho, el algoritmo que se propone no lo describe.



(a) Resultado con mejora



(b) Resultado sin mejora



(c) Imagen original

Figura 7: Comparación de dos sombras, una con reelección de representantes y otra sin reelección

4. Implementación

Hemos realizado una implementación de los algoritmos anteriormente descritos. Esta implementación tiene la función meramente didáctica de comprobar los resultados de los distintos algoritmos bajo la elección de distintas matrices base.

Para la implementación nos hemos decantado por **Python** como lenguaje de programación, junto con el framework **Kivi** que facilita la implementación de interfaces gráficas y permite correr la aplicación en varias plataformas fácilmente sin modificaciones en el código (en concreto Linux, Windows, iOS y Android).

4.1. Características

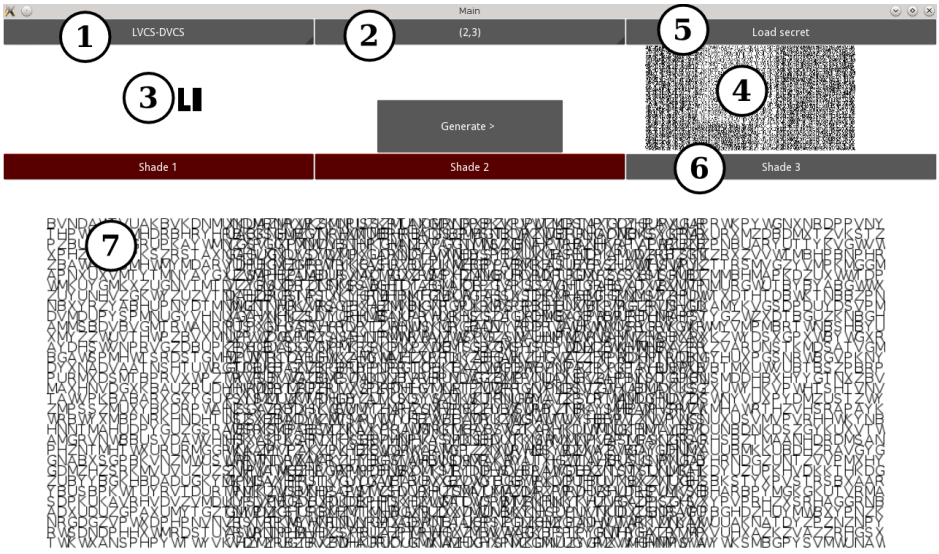


Figura 8: Captura del programa. Se enumeran las características que se describen a continuación.

1. **Selección de algoritmo a utilizar** (DVCS, LVCS-DVCS, LVCS-PVCS).
2. **Selección de las matrices base a utilizar** ((2,2), (2,3), (3,3))
3. **Miniatura de la imagen original**
4. **Miniatura de la imagen resultado**
5. **Carga de la imágen original**
6. **Elección de sombras mostradas:** Permite seleccionar las sombras que se mostraran en la zona de composición de sombras.
7. **Zona de composición de sombras:** Permite arrastra las sombras horizontalmente para superponerlas de forma manual.

4.2. Detalles

A continuación vamos a explicar ciertos detalles de la implementación, así como algunos inconvenientes encontrados durante su realización.

4.2.1. Generación de matrices base

Como ya hemos comentado, la generación de matices base que cumplan correctamente las restricciones es un problema complejo (ver [8]).

Otro problema añadido a la búsqueda de esta matrices es que la mayoría, aunque cumplan las restricciones, muestran una diferencia entre los superpíxeles blancos y negros (contraste) muy leve. Esto hace que las imágenes resultantes del algoritmo LVCS sean difícilmente apreciables si se utilizan estas matrices. A partir de $K = 3$ nos ha sido imposible encontrar matrices que presenten una diferencia entre superpíxeles blancos y negros de más de un píxel (ie. $h - l = 1$ ver 2.3-Matrices base).

Nosotros hemos optado por algoritmos de búsqueda en profundidad con poda para buscar matrices alternativas a las que se nos proporcionaban en los artículos. Estos algoritmos nos han proporcionado algunas matrices pequeñas que cumplen las restricciones, pero resultan inviables a la hora de buscar matrices mayores o que otorguen un mayor contraste.

4.2.2. Antialiasing y zoom

El framework que hemos utilizado realiza antialiasing a las imágenes automáticamente. Esto ha creado divertidos efectos al superponer las sombras, pero también impide que se aprecien los resultados si el zoom no es el adecuado. Hay que tener en cuenta que el zoom digital conlleva una pérdida de información, si se realiza antes de superponer las sombras, la superposición no será efectiva.

4.2.3. Resolución de resultados

La resolución de las imágenes resultantes por el algoritmo LVCS crece mucho en relación a la imagen original. Esto es debido a que cada píxel de la imagen original se sustituye por varias letras que van a ocupar varios píxeles cada una en la imagen resultado. Esto provoca realentizaciones y agotamiento de memoria para imágenes grandes en nuestra implementación.

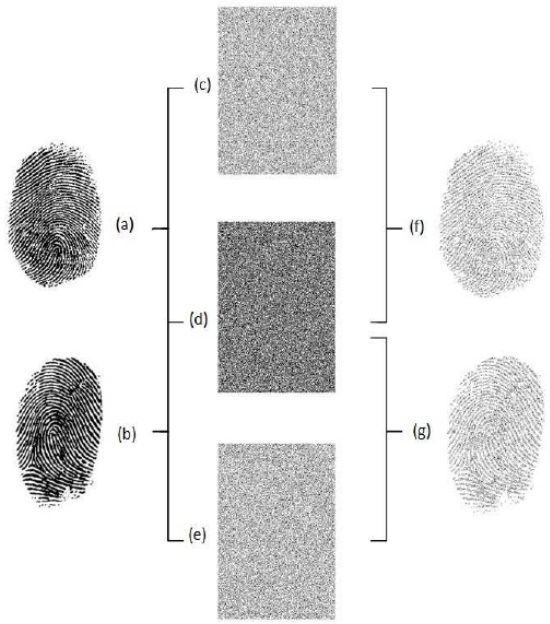
5. Aplicaciones

Las únicas aplicaciones que hemos encontrado para este tipo de algoritmo son de identificación. El algoritmo LVCS no tiene unas aplicaciones claras, se habla de generar sombras con texto con significado [2], pero no se indica cómo conseguirlo y no se pueden aplicar los mismos métodos que para algoritmos con píxeles. Por ello, las aplicaciones que detallamos a continuación se limitan únicamente a los algoritmos DVCS y PVCS.

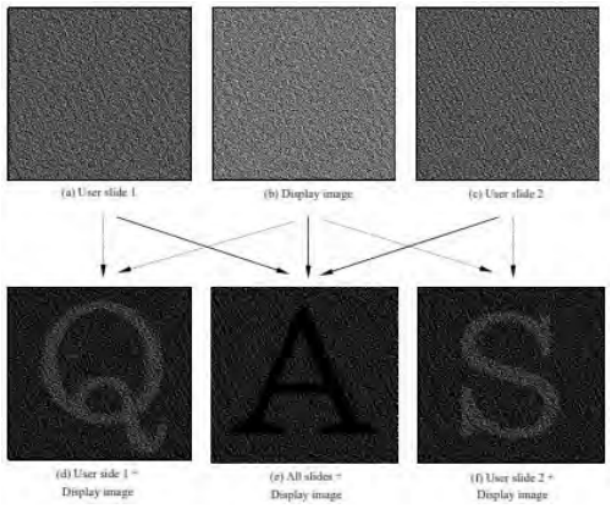
Todas las aplicaciones de identificación siguen un esquema extendido de VCS (ver [1]), en el cual una par de imágenes se pueden cifrar en tres sombras distintas. Una de estas sombras es compartida, de forma que al superponerla con alguna de las otras dos, revela un secreto distinto para cada una (ver figura 9).

Este esquema se pretende utilizar como refuerzo para sistemas de identificación biométrica, como identificación de usuarios o como sistema de autentificación de transacciones bancarias:

- **Identificación biométrica:** El usuario se identifica con su sombra, la cual se superpone con la sombra compartida que tiene almacenado el sistema para recuperar la imagen de la muestra biométrica del usuario (huella dactilar por ejemplo). Después se piede al usuario que vuelva a identificarse biométricamente para cercionar su identidad.
- **Identificación de usuarios:** Se muestra al usuario la sombra compartida que está almacenada en el sistema, el usuario superpone su sombra para obtener la clave con la que se autentifica en el sistema.
- **Transacciones bancarias:** Antes de hacer efectiva una transacción bancaria se muestra la sombra al usuario, que es el único que dispone de la otra sombra que revela la clave necesaria para validar la transacción.



(a) Ejemplo de identificación biométrica

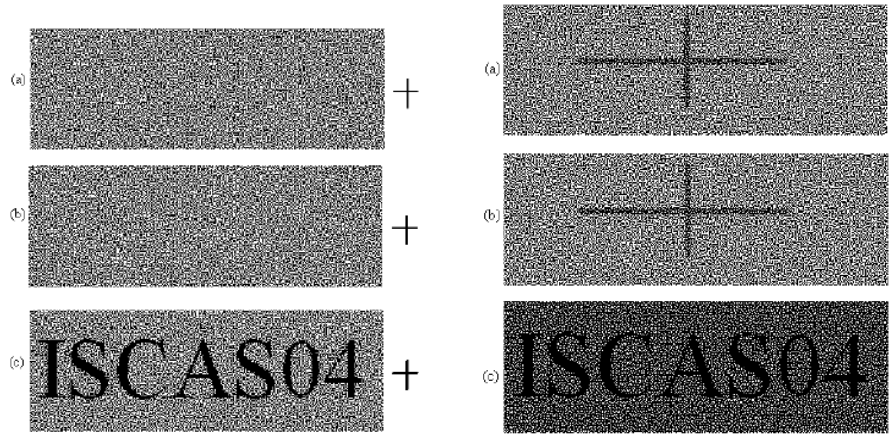


(b) Ejemplo de esquema extendido de VCS

Figura 9: Ejemplos de cifrado con DVCS

5.1. Mejora para alineación

Esto no es una aplicación propiamente, pero la hemos encontrado interesante. En aplicaciones dónde la superposición de sombras se hace de forma manual, y estas son de gran resolución, es difícil alinearlas correctamente y una pequeña desviación impediría revelar el secreto. Las soluciones que implican añadir una marca al lado de las sombras para ayudar a alinearlas son propensas a deformaciones por impresión o escaneado. Este artículo [7] propone añadir estas marcas a las mismas sombras, reduciendo así los problemas por deformaciones en la impresión o escaneado.



(a) Esquema de alineación con marcas de ayuda en un lateral de las sombras

(b) Esquema de alineación con marcas de ayuda embebidas en las misas sombras

Figura 10: Alineaciones por los dos métodos mencionados

6. Bibliografía

Referencias

[1] Nazanin Askari, Cecilia Moloney, and Howard M. Heys. Application of visual cryptography to biometric authentication.

[2] Hsiao-Ching Lin, Ching-Nung Yang, Chi-Sung Laih, and Hui-Tang Lin. Natural language letter based visual cryptography scheme.

[3] Mariko Nakano, Enrique Escamilla, Héctor Pérez, and Mitsugu Iwamoto. Criptografía visual basada en el esquema de umbral: Una revisión tutorial.

[4] Moni Naour and Adi Ahamir. Visual cryptography.

[5] Doug Stinson's. Doug stinson's visual crytography page. http://cacr.uwaterloo.ca/~dstinson/visual.html.

[6] Wikipedia. Visual cryptography. https://en.wikipedia.org/wiki/Visual_cryptography.

[7] Wei-Qi Yan, Duo Jin, and Mohan S Kankanhalli. Visualcryptography for print and scan applications.

[8] Ching-Nung Yang, Chih-Cheng Wu, and Feng Liu. Construction of general (k,n) probabilistic visual cryptography scheme.

7. Anexos

7.1. Repositorio

Esta es la dirección del repositorio público Git que contiene tanto el código del programa, como el código LaTeX de la documentación.

Dirección del repositorio: https://github.com/FarK/CryptoLVCS