



# COMPARTICIÓ DE SECRETS

Memòria del projecte que presenta Juan Rivas  
amb la direcció de la Dra. Montserrat Alsina  
a l'Escola Politècnica Superior d'Enginyeria de Manresa  
per assolir el grau d'Enginyer en Sistemes TIC

## Resum

Aquest projecte tracta de la compartició de secrets, sobre els mètodes i sistemes que hi ha actualment per compartir informació de manera secreta entre un grup d'usuaris. Hi ha diversos mètodes matemàtics que expliquen com fer aquesta compartició de manera correcta. En aquest projecte estudiarem alguns mètodes i els implementarem de manera que es pugui veure fins a quin nivell es pot millorar els actuals mètodes de compartició de secrets.

Utilitzarem el mètode visual que ens ajuda a entendre, interpretar i desenvolupar els coneixements de manera pràctica i continuarem analitzant l'esquema de compartició de secrets de Blakley, que té un cert component visual de geometria per tancar les conclusions del projecte.

## Abstract

This project deals with the secret sharing information so when these information it's shared it can not be easily obtained by a single user. There are some mathematical methods that explain how to do this sharing correctly. In this project, we will study some methods and implement them in such a way that we can see the potential of the method and what kind of secrets can be shared and how many users can interact on one secret.

We will use the visual method that helps us to understand, interpret and develop knowledge in a practical way and we will continue to analyze the Blakley secret sharing scheme, which has a certain geometrical visual component to close the project conclusions.

## Agraïments

Aquest projecte no hauria estat possible sense l'ajuda de la meva professora de matemàtiques, Montserrat Alsina, que un dia em va parlar sobre aquest tema on vaig trobar la motivació com per treballar-hi i fer un treball final de grau. M'ha aportat molts coneixements i m'ha ajudat a crear un projecte del qual em sento orgullós de presentar.

Gràcies a aquest projecte puc dir que li he tornat a agafar el gust a les matemàtiques.

# Índex

<b>Resum</b>	<b>i</b>
<b>Abstract</b>	<b>i</b>
<b>Agraïments</b>	<b>ii</b>
<b>1 Contextualització del projecte</b>	<b>1</b>
1.1 Introducció a la compartició de secrets . . . . .	2
1.2 Objectius . . . . .	3
<b>2 Compartició de secrets visuals</b>	<b>4</b>
2.1 Partició d'imatges per trossos . . . . .	4
2.2 Compartició visual d'imatges per capes . . . . .	6
2.3 Compartició digital d'imatges . . . . .	9
2.4 Programació de compartició de secrets visuals . . . . .	13
2.4.1 Dealer . . . . .	14
2.4.2 Combiner . . . . .	15
2.4.3 Possible ampliació . . . . .	16
2.5 Implementació de compartició visual de secrets de text . . . . .	18
2.5.1 Plantejament general . . . . .	18
2.5.2 Noves funcions . . . . .	19

<i>ÍNDIX</i>	iv
2.5.3 Interfície per l'usuari . . . . .	21
2.5.4 Codi de la interfície . . . . .	23
2.5.5 Procés . . . . .	23
2.6 Conclusions . . . . .	25
<b>3 Esquema vectorial</b>	<b>27</b>
3.1 Plantejament general de la compartició de secrets . . . . .	27
3.2 Descripció de l'esquema vectorial . . . . .	29
3.3 Construcció de l'esquema vectorial . . . . .	30
3.3.1 Interpretació de l'esquema vectorial per $m=2$ . . . . .	30
3.3.2 Interpretació de l'esquema vectorial per $m=3$ . . . . .	34
3.4 Construcció dels hiperplans . . . . .	38
3.4.1 Construcció de plans en dimensió 3 . . . . .	38
3.4.2 Construcció de l'esquema vectorial en general . . . . .	40
3.5 Programació de l'esquema general per Vandermonde . . . . .	41
3.5.1 Interfície . . . . .	41
3.5.2 Dealer . . . . .	42
3.5.3 Combiner . . . . .	46
3.6 Matrius de Hei-Du-Song . . . . .	48
3.6.1 Plantejament general . . . . .	48
3.6.2 Construcció efectiva de les matrius . . . . .	50
3.7 Comparació dels mètodes . . . . .	52
3.8 Introducció del líder i repartició de poders . . . . .	55
3.9 Conclusions . . . . .	56
<b>4 Valoració personal</b>	<b>57</b>
<b>Llistat de figures</b>	<b>60</b>

<i>ÍNDIX</i>	v
<b>Bibliografia</b>	<b>61</b>
<b>Annexos</b>	<b>63</b>
A.1 Llibreria per treballar amb secrets visuals . . . . .	64
A.2 Codi font de la interfície de secrets visuals . . . . .	71
A.3 Llibreria per treballar amb Vandermonde . . . . .	74
A.4 Codi font de la interfície de Vandermonde . . . . .	78
A.5 Llibreria per treballar amb Hei-Du-Song . . . . .	82
A.6 Llibreria per treballar amb Hei-Du-Song . . . . .	86

# Capítol 1

## Contextualització del projecte

Des de temps immemorials, la compartició de secrets, ha estat un tema important. La informació i missatges valuosos s'han intentat preservar sempre. La criptografia s'ocupa de transmetre i mantenir la informació en secret perquè la transmissió d'aquesta sigui segura. La compartició de secrets són esquemes criptogràfics que busquen mantenir una informació secreta però que aquesta sigui repartida entre molta gent ja que la seva seguretat no depèn de la potència computacional dels adversaris.

En algunes èpoques s'ha entès com a segur un secret quan només estava en unes soles mans, altres quan estava repartit entre més mans.

Per una banda el secret que només sap una persona, morirà si aquella persona mor sense explicar-ho a ningú. En cas de secrets medicinals, per exemple, era important que es mantingués el secret però que no es perdés mai i passés de generació en generació.

Per altra banda, també en casos en què el pare d'una família volgués mantenir la unitat de la seva família una vegada mort, podia repartir un secret entre els seus fills per assegurar que estiguessin junts si volien heretar l'herència.

Un esquema de compartició de secrets és un mètode que consisteix a dividir un secret en dues o més parts, de manera que cap d'aquestes parts reveli informació sobre el secret. Ajuntant aquests trossos es pot recuperar el secret sencer.

Per exemple un cas típic és en una sucursal d'un banc on s'han de moure grans quantitats de diners. És interessant mantenir sota control aquestes transaccions. La clau de la caixa forta ha d'estar compartida de manera que

per obrir-la és necessari més d'una persona, entre elles el director o càrrecs importants del banc.

Aquest projecte es divideix en 2 grans temes, un és la compartició de secrets visuals i l'altre la compartició de secrets numèrics. Al final del projecte trobem els annexos amb el codi font del programari creat per aquest projecte, aquest ha sigut programat amb Python.

En aquest primer capítol estem introduint el tema, plantejant els fonaments de la compartició de secrets, el context general i explicant els objectius.

En el segon capítol del projecte treballem amb exemples on el secret que volem compartir és una imatge física, ja sigui, un mapa del tresor o la imatge d'un compte bancari. Després d'estudiar el sistema de compartició de secrets visuals es presentaran les aplicacions programades específicament per aquest projecte.

En el tercer capítol deixem les imatges i s'estudia la compartició de secrets utilitzant l'esquema vectorial de Blakley. Els secrets passen a ser numèrics i es tracten amb matrius i equacions.

## 1.1 Introducció a la compartició de secrets

Cada peça d'informació s'anomena share, el responsable de la repartició del secret,  $S$ , en diem dealer. Una col·lecció d'accions,  $n$ , que contenen el secret sencer en alguna forma s'anomena, allowed coalition. El procés responsable de la recuperació del secret s'anomena combiner, aquest necessita mínim shares per recuperar-lo.

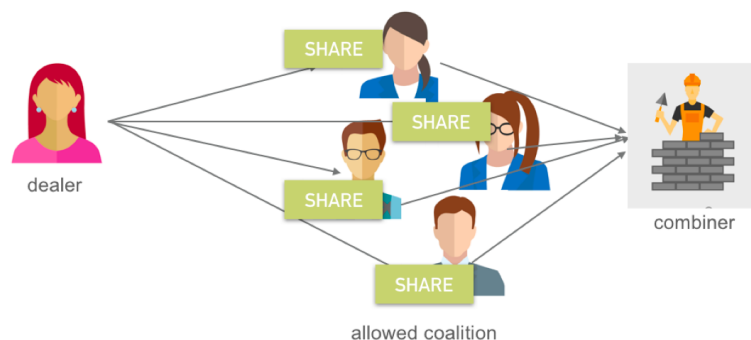


Figura 1.1: Esquema de compartició de secrets



Cada share conté lògicament part de la informació del secret, però no és de cap utilitat. Quan el dealer ha repartit els shares el secret inicial es podria destruir. Així el secret només pot ser accessible quan tenim un cert nombre,  $k$ , de shares d'un allowed coalition.

## 1.2 Objectius

En aquest context els objectius plantejats en aquest treball són els següents.

### Objectius generals

- Entendre i donar a conèixer el sistema de compartició de secrets.
- Identificar sistemes per compartir secrets
- Aportar funcionalitats noves a la compartició de secrets

Per tal de dur-ho a terme, ens hem plantejat com a **Objectius específics**:

- Estudiar sistemes de compartició de secrets de diferents tipus.
- Comparar-los entre ells i fer-ne una explicació senzilla.
- Dissenyar una aplicació que produeixi els shares de manera efectiva i els combini per exemplificar-ho.
- Analitzar els resultats extrets dels diferents mètodes

## Capítol 2

# Compartició de secrets visuals

Tenint en compte que una imatge val més que mil paraules, en aquest capítol les utilitzarem per explicar en què consisteix la compartició de secrets i quines condicions ha de complir un correcte esquema de compartició de secrets.

En la primera secció es parla de la partició d'imatges a trossos, un mètode per repartir una imatge entre diferents usuaris sent aquesta la primera aproximació a una compartició de secrets. A continuació la compartició visual d'imatges per capes, en aquesta secció seguim repartint imatges però en comptes de repartir trossos ho fem a capes. En la tercera secció tenim la compartició digital d'imatges. Al llarg d'aquestes seccions hem anat veient de manera constructiva que és una compartició de secrets i que no ho és, de manera que ens hi ha conduït a la implementació de compartició de secrets visuals que es presenta a la secció 4. A continuació la secció 5 és una extensió de la implementació per aplicar-ho a secrets de text. Finalment s'acaba amb una secció de conclusions.

### 2.1 Partició d'imatges per trossos

Compartir un secret visual pot ser tan senzill com agafar la imatge, partir-la en trossos i repartir aquests trossos en diferents persones, o això és el que es pensava al principi. A la figura 2.1 posem un exemple clarificador.



Figura 2.1: Mapa a trossos

En aquest exemple veiem un mapa del tresor ja partit en trossos que seran els secrets a repartir. Si ens hi fixem, podem veure que en una de les peces es veu el secret principal que és la creu del tresor. Amb la informació que surt en aquest tros de secret es pot intuir informació suficient que permet trobar el tresor. En algun altre només tenim els punts cardinals que per si sols no ens ajudarien gaire a trobar el tresor. És a dir, hi ha trossos molt reveladors i altres que gens. Aquesta característica no és gens desitjable i no està permès en els esquemes de compartició de secrets.

Un altre exemple podria ser aquest:



Figura 2.2: Fragment d'una imatge més gran

En aquest cas només tenim un dels trossos del secret. Podem pensar que la informació ja és suficient per poder intuir el secret final. Ara bé, potser és millor no precipitar-se perquè podria venir de diferents imatges:



Figura 2.3: Banderes que tenen estrelles

La part de secret coneguda, en aquest cas, no ens revela una part prou important del secret global, però segueix contenint massa informació del secret final. Quan repartim un secret no volem de cap manera que hi puguin tenir el mínim accés.

Una resolució a aquest problema seria fer trossos molt més petits perquè ningú tingués cap part suficientment important com per poder intuir res ell sol. Amb aquest nou propòsit de repartir trossos més petits, no estem solucionant un problema anterior, el tros segueix formant part del secret final. L'estrella formava part d'una estrella d'alguna bandera del món, si la trenquem en més parts, fins i tot el tros blau seguirà formant exactament la mateixa part de la imatge original.

El següent pas és partir la imatge en capes, és a dir, trossos que superposats amb els altres shares revelin el secret, però que amb un sol share no puguem revelar el secret. A continuació s'explica aquest concepte.

## 2.2 Compartició visual d'imatges per capes

En aquesta secció presentarem de manera constructiva el que seria la compartició visual d'imatges. Així en acabar quedaran clares algunes de les característiques que té la compartició de secrets.

Si pensem en les imatges de manera digital, és a dir, com a conjunt de píxels, podem extreure una partició d'una manera molt més senzilla. Ara bé, per a tota aquesta part caldrà que intervinguin totes les parts per recuperar el secret. És a dir, que es necessiten tots els trossos per reconstruir la imatge secreta amb seguretat. Després veurem com en general la compartició numèrica de secrets permet més variacions.

Considerem una imatge pixelada en blanc i negre. Podem imaginar una quadrícula que la parteixi, en tants trossos com desitgem. La finalitat és

crear capes que continguin aquests píxels suficientment dispersats com per no reconèixer cap figura en cap capa per si sola.

**Exemple .** Disposem d'una imatge i suposem que ha estat partida en capes, de manera que cada participant només té una d'aquestes capes. Observem que a partir d'una sola no podem reconèixer cap figura estàndard.



Figura 2.4: Partició en 3 capes d'una imatge secreta

Per obtenir el secret que amaguen aquestes senzilles capes només hem de superposar-les. Aquestes capes han de ser transparents per poder fer la superposició. Col·laborant doncs els tres participants recuperarien la imatge secreta, que es mostra a la figura 2.6.

**Procediment implementat.** Per generar les 3 capes, s'ha realitzat un programa especial per nonograms. Aquest programa s'ha programat amb python. En aquest exemple estem treballant de manera aleatòria, és a dir, els punts negres són posats en cada capa en posicions que no segueixen cap patró, el programa tria de manera aleatòria un nombre natural entre 0 i 2, el número que trii serà la capa on anirà a parar el punt negre. Les funcions programades es poden trobar al primer apartat dels codis font de l'annex.

Si analitzem com funciona el programa veiem que en treballar d'aquesta manera no podem controlar el repartiment de punts final i en alguns casos ens pot quedar una disposició de punts que doni molta informació en una capa i poca en una altra. Això li donaria més poder a un usuari, ja que podria arribar a desxifrar el secret per si mateix. En aquest cas si mirem la primera capa podríem arribar a saber més informació que si tenim la tercera. L'input del programa és la imatge a repartir i l'output són les 3 imatges que es guarden en la carpeta on treballa el programa.

Si tornem a executar el programa, podem veure com la disposició dels punts ha canviat completament.



Figura 2.5: Noves capes aleatòries de la mateixa imatge secreta

Té sentit plantejar la possibilitat de mantenir cert control sobre cada una de les capes, de manera que la proporció de punts negres de cadascuna de les imatges que es reparteixen sigui aproximadament la mateixa. Per aconseguir-ho hem retocat el programa fent que el repartiment de punts negres segueixi un patró. El patró és repartir un punt negre a cada capa de manera cíclica, sense escollir res aleatòriament. Ara bé, aquesta regularitat imposada és també un obstacle al manteniment del secret, ja que la regularitat ajuda a preveure i pot revelar el secret. A més, aleshores un usuari pot comptar distàncies entre punts. És a dir, si com a usuari conec que som 2 en la repartició del secret, puc saber que entre un punt negre i un altre punt negre que jo tinc en la meua capa, hi ha un punt negre més, que és el que ha anat a parar a l'altre share. El marge dependrà de les posicions en blanc que hi hagi, i pot ser que provant de col·locar-lo es pugui anar deduint la imatge secreta que s'ha repartit a partir d'una sola imatge. Si s'augmenta el nombre de capes, serà més complicat.

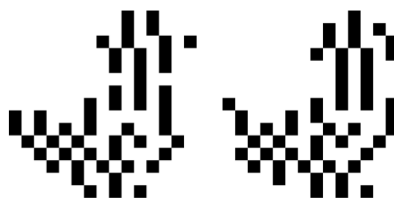


Figura 2.6: Capes repartides de manera no aleatòria.

Si ens hi fixem, els píxels queden repartits més uniformement i el secret pot quedar més a la vista. Per altra banda es pot apreciar el que es comentava de comptar píxels entre negre i negre.

Cada un dels dos mètodes té avantatges i inconvenients. Ara bé, l'inconvenient més gran d'aquest mètode és que estem donant massa informació, tots

els píxels negres acabaran formant part del secret final i per tant els usuaris estan rebent informació real de la imatge secreta que no haurien de tenir, per tant cap dels dos mètodes pot ser bo com a mètode final a implementar.

La imatge que s'ha fet servir en aquest exemple és la següent.

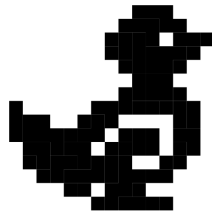


Figura 2.7: Imatge original

Aquesta manera de compartir imatges de manera secreta, ens fa veure el camí a seguir per la digitalització del sistema. Estem tractant les imatges de forma binària, és a dir, cada capa està formada per punts negres o punts blancs, 0 i 1 respectivament. En la secció següent estudiarem la digitalització de la compartició de secrets visuals i com avançar cap a un mètode més fiable.

## 2.3 Compartició digital d'imatges

En la secció anterior hem parlat de capes i superposicions que deixen veure la imatge original. El problema era que cadascuna d'aquestes capes, que estava formada en l'àmbit digital per 1 i 0, ja ens està donant massa informació del secret final. Quan donem una part del secret, l'usuari no ha de ser capaç d'intuir ni un sol punt final del secret. No li podem oferir informació gratuïta només per tenir una part d'aquest secret.

Encertar el número corresponent a una casella que pot ser 1 o 0, té una probabilitat d'encert del 50%, igual que d'error. Aquesta idea és fonamental per la compartició de secrets visuals. Quan compartim una imatge el nostre propòsit és que l'usuari hagi d'encertar totes les caselles, sense cap error, per poder saber el secret. Com que el nombre de píxels d'una imatge és prou elevat, la probabilitat de què encerti tots els píxels de la imatge és tan baixa que ho podem catalogar com a impossible.

Abans de passar a veure-ho en una imatge ho vull exposar d'una manera més senzilla, en l'àmbit digital utilitzem mòdul 2.

Sigui  $S$  un secret, format per una sèrie de  $k$  dígit, on cada dígit és 0 o 1. Volem partir el secret en dos shares  $S_1$  i  $S_2$ , de manera que ajuntant-los donin  $S$ . La manera més natural d'ajuntar dos nombres seria sumar. Així volem que  $S_1 + S_2 = S$ . El que farem justament és utilitzar la suma mòdul 2.

Està clar que el secret  $S$  no es pot deduir de  $S_1$  ni de  $S_2$  per separat. Un share pot ser triat a l'atzar i d'aquest deduir l'altre fent la suma amb el secret,  $S_2 = S + S_1$ .

### Algoritme implementat

- Considerant un secret  $S$
- Triem un  $S_1$  aleatori
- Calculem  $S_2 = S + S_1$  en mòdul 2
- Ja podem repartir els dos shares  $S_1$  i  $S_2$ .

A continuació podem veure un exemple.

Considerem, en aquest cas,  $S = 00000000011000$  el secret a compartir entre dos usuaris. Calculem  $S_1$  aleatòriament, que serà un dels shares a repartir entre aquests dos usuaris, per exemple,  $S_1 = 0001110010011$ . A partir d'aquí el segon share a repartir surt fent la suma en mòdul 2 de  $S_1$  i el secret  $S$ .

$$\begin{array}{r} S : 00000000011000 \\ + S_1 : 0001110010011 \\ \hline S_2 : 00011100111110 \end{array}$$

Com veiem doncs, al donar els shares per separat no estem donant cap tipus d'informació del secret final. A partir d'un sol share, si provem tots els altres possibles shares, l'única manera de trobar el secret és provar els  $2^k$  possibles shares i sumar-los al que tenim. Si treballem amb nombres molt grans, obtindrem totes les solucions possibles i per tant no es pot escollir cap solució.

Aquest mètode es pot ampliar a diversos usuaris.  $U_1, U_2, \dots, U_n$ . Per repartir un secret triem a l'atzar  $S_1, S_2, \dots, S_{n-1}$  i només cal triar  $S_n$  que compleixi que la suma de tots sigui  $S$ . En cas de canviar el secret podríem deixar fixat  $S_1, S_2, \dots, S_{n-1}$  i només recalculat un nou  $S_n$  perquè la suma torni a donar el secret  $S$ .



El mètode de compartició de secrets digitals és escalable als secrets visuals digitals, és a dir, imatges. Tota imatge digital és una matriu de números, freqüentment binaris, que formen els anomenats píxels. Aquesta composició binària de les imatges pot ser tractada de la mateixa manera que hem fet la suma. Una imatge pot ser el resultat de la suma de dos bits binaris.

Establim una codificació binària per representar les imatges. Seguidament volem comparar el mètode numèric amb el mètode visual, és a dir, quan apareixen colors i transparències. A continuació es recorda la suma numèrica en mòdul 2.

$$\begin{aligned} 0 + 0 &= 0 \\ 0 + 1 &= 1 \\ 1 + 0 &= 1 \\ 1 + 1 &= 0 \end{aligned}$$

En el món digital el color negre és el número 0 i el color blanc és l'1. D'aquesta manera interpretem un píxel negre com a 0 i un píxel blanc com a 1. Si això ho interpretem visualment la suma anterior, per poder extrapolarlo a les imatges, tindríem una suma en mòdul 2 d'imatges d'aquest tipus:

MÈTODE DIGITAL A IMATGES						
0 + 0	=	■	+	■	= ■	NEGRE
0 + 1	=	■	+	□	= □	NO NEGRE
1 + 0	=	□	+	■	= □	NO NEGRE
1 + 1	=	□	+	□	= ■	NEGRE

Figura 2.8: Sumes en mòdul 2 versió digital

Com podem veure en el sistema visual tenim un problema, quan volem fer la suma en mòdul 2 de  $1 + 1 = 0$ , ja que blanc i blanc no genera negre. En el cas que superposem dues imatges físiques amb transparències, tampoc funciona. També ens passa amb les sumes  $1 + 0 = 1$ , és a dir, blanc + negre = blanc i  $0 + 1 = 1$  que correspondria a negre + blanc = blanc. Cap d'aquestes sumes segueix el principi visual. Encara que canviéssim els papers del negre i el blanc, de manera que el 0 fos blanc i l'1 fos negre, tampoc aconseguim que es compleixin les 4 igualtats. Per poder dur a terme aquesta suma físicament s'ha tingut d'establir un altre sistema de suma. Un dels mètodes amb el que

treballa equival en partir cada píxel en dues parts iguals i aplicar l'esquema que trobem a la figura 2.9.













EFECTE VISUAL						
0 + 0	=		+		=	 NO NEGRE
0 + 1	=		+		=	 NEGRE
1 + 0	=		+		=	 NEGRE
1 + 1	=		+		=	 NO NEGRE

Figura 2.9: Sumes digitals en mòdul 2 visuals amb transparències

Quan dos píxels són iguals el resultat en sobreposar les dues capes seran iguals, i es visualitzin mig blancs i mig negres i per tant donarà la sensació de no negre. En canvi quan dos píxels són contraposats aquests generen tot negre. Aquesta manera de partir cada píxel en una combinació de dos píxels 0 i 1 suposa un redimensionat de la imatge que no ens interessa.

En aquest projecte ens hem inspirat en aquest mètode però no el fem servir, ja que significaria crear dos píxels addicionals per cada píxel llegit i seria redimensionar la imatge, en aquest cas l'hem simplificat i no hem partit els píxels en dos més. Senzillament hem fet una adaptació de manera que podem utilitzar el mètode visual natural de la figura 2.10.













EFECTE VISUAL						
0 + 0	=		+		=	 NEGRE
0 + 1	=		+		=	 NEGRE
1 + 0	=		+		=	 NEGRE
1 + 1	=		+		=	 NO NEGRE

Figura 2.10: Taula de sumes visuals amb transparències

En aquest mètode simplificat un píxel sempre és un píxel. Quan superposem dues capes, els píxels negres sempre dominaran per davant del blanc per tant en una suma on hi ha un 0, és a dir, píxel negre, al resultat final segur que queda negre. Tan sols quan els dos són blancs el resultat final serà blanc.

El mètode proposat es basa en aplicar l'esquema anterior seguint l'algoritme següent.

### Algoritme

- Analitzem un a un tots els píxels.
- Si el píxel a repartir és de color negre, entendrem que forma part del secret.
- De manera aleatòria posem un 0 en un share i en l'altre li posem un 1.
- Si el píxel és blanc, entendrem que forma part del fons del secret.
- De manera aleatòria triem 0 o 1 i posem la mateixa elecció als dos shares.
- A l'acabar tots els píxels podem repartir els shares.

Així, a cada píxel ens basarem en si en el secret apareix pintat negre, o no. Si forma part del secret, aquest el repartirem en dos shares de manera que en un sigui 1, blanc, i en l'altre sigui 0, negre. Al contrari si el píxel és blanc i per tant no forma part del secret aquest el repartirem en dos shares de manera que en els dos shares els píxels siguin iguals. D'aquesta manera comparant els dos shares sabrem extreure el secret dels píxels que són diferents d'un share a l'altre.

## 2.4 Programació de compartició de secrets visuals

En aquesta secció s'explica com s'ha implementat el procés del dealer i el combiner descrit en la introducció del projecte utilitzant el procés introduït en les seccions anteriors.

Aquest procés ha estat programat amb Python, un llenguatge de programació interpretat d'alt nivell, la seva filosofia és utilitzar sintaxi que faciliti la lectura del codi. També proveeix estructures per permetre programes més entenedors tant a petita com a gran escala. Python suporta diversos paradigmes de programació, incloent-hi programació orientada a objectes, imperativa i també funcional o procedimental. Presenta un sistema dinàmic

i una gestió de la memòria automàtica i té una gran i exhaustiva biblioteca estàndard.

L'objectiu és introduir en el dealer una imatge com a secret i que aquest sigui capaç de partir-la en dues capes anomenades shares de manera que per separat no sigui possible l'obtenció d'aquest secret. A l'apartat del combiner s'explica el pas invers, és a dir com s'aconsegueix que a partir de dos shares que per si soles no aporten informació, es poden unir i extreure el secret original com a una sola imatge.

### 2.4.1 Dealer

El dealer és l'encarregat de generar dos shares per compartir el secret entre els usuaris. El programa que s'ocupa d'això és la funció `dealer(img)`. El primer que fa és crear una còpia de la imatge rebuda. A partir d'aquí es comença un procés de recórrer tota la imatge píxel a píxel.

Cada píxel s'examina, si té un valor menor a 0.8, s'interpreta com a píxel negre. Cada píxel negre interpretem que és part del secret, ja que estem treballant amb imatges que només tenen píxels blancs o negres.

De forma aleatòria triem 0 o 1, això ho fem amb la funció `choice([0, 1])` que ens retorna un 0 o un 1. Aquest píxel formarà part del secret final, per tant ens interessa repartir-ho de manera que sigui recuperable en dos shares. En la imatge que estem consultant li posem el que ens dóna la funció `choice()`, en la mateixa posició de la imatge copiada li posem la resta, `1 - choice()`.

Si el píxel obtingut fos considerat blanc llavors triaríem un número a l'atzar, 0 o 1, però en aquest cas posaríem el mateix número en la mateixa posició de les dues imatges.

Després de crear els dos shares a més a més s'ha programat una funció que el que fa és aplicar-hi transparències a tots els blancs de la imatge per tal que si aquestes imatges es volen sobreposar, quedi demostrada la transparència. El nom d'aquesta funció és `convertToPNG()`.

El resultat d'aquesta funció és el que tenim en les figures 2.11

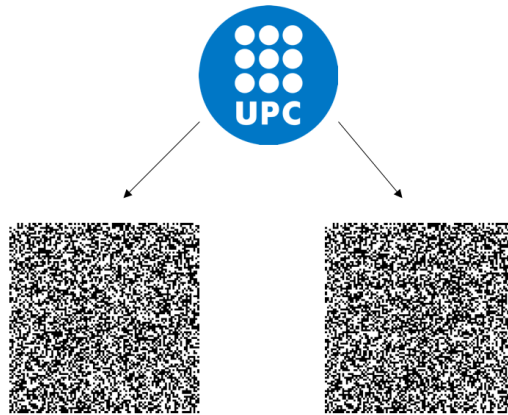


Figura 2.11: Partició de la imatge en 2 shares

En aquest punt ja tenim preparats els dos shares per repartir als dos usuaris. Com podem observar a primera vista és molt complicat extreure un patró o una pista del secret final a partir d'un sol share.

Codi de la funció `dealer()` a l'annex A.1.

### 2.4.2 Combiner

El combiner és l'encarregat de sobreposar els shares i ensenyar el resultat obtingut. Aquest programa és la funció `combinator()`, que llegeix les dues imatges rebudes, obté tots els píxels de cadascuna i fa una comparació píxel a píxel. Si els dos píxels coincideixen, llavors aquell píxel no forma part del secret final, per tant en una imatge creada anteriorment es pintarà un píxel blanc. De l'altra manera si no hi ha coincidència en llegir els dos píxels de cada imatge, s'entén que aquell píxel pertany al secret final i per tant anirà negre.

A la figura 2.12 podem veure el resultat de superposar els dos shares.



Figura 2.12: Superposició dels dos shares per obtenir el secret.

A la implementació se li ha afegit com a millora el resultat final del combiner, a part de superposar els dos shares, també neteja tots els píxels sobrants que no formen part del secret i que no ens aporten cap informació útil, d'aquesta manera podem observar el secret sense cap redundància. A la figura 2.13 podem veure el resultat de la imatge després d'aplicar el combiner.



Figura 2.13: Imatge resultant després del processament del combiner.

Es pot veure que és igual a la imatge que el dealer ha repartit en l'apartat 2.4.1 tot i que hem perdut el color.

Fins ara hem estat parlant i treballant d'imatges, però digitalment no només existeixen imatges, i quan parlem de secrets no són imatges amb què els comuniquem normalment. En la secció següent introduïrem paraules, números i símbols per comunicar altres tipus de secrets.

Codi de la funció `combinator()` a l'annex A.1.

### 2.4.3 Possible ampliació

Després de treballar aquest apartat i veure com podem repartir una imatge en dos shares, se'ns planteja una possible millora a estudiar que no hem pogut

aprofundir en aquest projecte.

Aquest mètode queda molt limitat a dos shares i seria interessant poder ampliar-ho i intentar generar 3 shares de la mateixa manera. En aquest projecte s'ha començat a estudiar alguns camins per arribar-hi.

Tenint en compte que un secret  $S$  és el resultat de la suma de dos shares, si afegim un tercer tindriem aquesta equació:  $S_1 + S_2 + S_3 = S$ .

- Podem fer  $S_1 + (S_2 + S_3) = S$  tal que  $S_1 + S_{23} = S$ . A l'atzar podem agafar  $S_2$  o  $S_3$  tal que  $S_2 + S_3 = S_{23}$ .
- Una altra possibilitat és tenint  $(S_1 + S_2) + S_3 = S$  podem agafar  $S_1$  i  $S_2$  completament aleatoris i després triar  $S_3$  que compleixi la igualtat amb  $S$ . En realitat tenim una equació amb 3 incògnites, dues variables són lliures i una és determinada.

Aquests mètodes estudiats són factibles matemàticament i es poden programar, la pregunta és si es pot implementar gràficament.

**Exemple .** L'objectiu és repartir el secret  $S = 1001011$  entre 3 usuaris,  $S_1, S_2, S_3$ , els shares repartits a aquests usuaris no poden contenir suficient informació com per poder recuperar el secret, però quan els superposem un sobre de l'altre han de deixar veure aquest secret. Per veure clarament del que es vol parlar s'exposen els exemples amb els mètodes descrits anteriorment.

Amb el primer mètode podem escollir  $S_1$  aleatòriament, en aquest cas escollim  $S_1 = 1101000$ , el número  $S_{23}$  és la suma  $S + S_1$ .

$$\begin{array}{r} S \quad : \quad 1001011 \\ + S_1 \quad : \quad 1101000 \\ \hline S_{23} \quad : \quad 0100011 \end{array}$$

A partir d'aquí hem de triar un tercer número corresponent a  $S_2$  o  $S_3$ , en aquest cas triem  $S_2 = 1001001$ . Amb això ens queda definit  $S_3 = S_2 + S_{23}$

$$\begin{array}{r} S_{23} \quad : \quad 0100011 \\ + S_2 \quad : \quad 1001001 \\ \hline S_3 \quad : \quad 1101011 \end{array}$$

Tenint  $S_3$  ja tenim els 3 shares i fent la suma dels 3 ens hauria de donar el secret  $S = 1001011$ .

$$\begin{array}{r} S_1 : 1101000 \\ S_2 : 1001001 \\ + S_3 : 1101010 \\ \hline S : 1001011 \end{array}$$

Fins aquí queda demostrat que és factible, de fet és un concepte senzill. El que hem de tenir en compte és que són secrets visuals i que no els podem tractar de la mateixa manera que tractem els números digitals. Com s'ha dit en la secció 2.3, el 0 és un píxel negre i l'1 és un píxel blanc. Si aquests 3 shares que hem generat els superposéssim un a sobre de l'altre tindríem una imatge que reflectiria aquest secret.

$$\begin{array}{r} S_1 : 1101000 \\ S_2 : 1001001 \\ + S_3 : 1101010 \\ \hline S : 1001000 \end{array}$$

Per portar a terme els mètodes mostrats hem d'entendre la suma com a superposició de píxels físics, és a dir que un píxel blanc amb un altre blanc, no fan negre  $1 + 1 \neq 0$ .

## 2.5 Implementació de compartició visual de secrets de text

### 2.5.1 Plantejament general

En aquest apartat es proposa una nova aplicació a la compartició de secrets visuals utilitzant l'esquema explicat en els apartats anteriors. Fins ara hem estat treballant amb imatges que contenien algun tipus de dibuix o logo, és a dir no tenien cap secret llegible, com pot ser una paraula o un codi d'una caixa forta. L'objectiu d'aquesta nova aplicació és poder compartir també missatges de text, números i símbols entrats per teclats, aquests convertir-los en imatges i procedir amb el sistema explicat en la secció 2.4.



L'usuari pot entrar lletres, números i els símbols acordats, en aquest cas només s'accepten: @#€\$&\_-, distingint també els espais.

Per portar a terme aquesta aplicació hem hagut d'estudiar casos similars que treballin amb text.  $\text{\LaTeX}$  és un sistema de composició de text orientat a la creació de documents com ara aquest projecte.  $\text{\LaTeX}$  treballa enfilant les lletres i té un control absolut sobre la posició d'aquestes dins del document, com a prova el logotip  $\text{\LaTeX}$ .

Estudiant aquest sistema hem vist que les lletres minúscules poden ocupar diferents posicions i altures dins d'una paraula, és a dir, la "g" és una lletra que va cap avall i la "t" en canvi va cap amunt i per tant és complicat enfilar-les en un mateix fil. Aquí és on s'ha de saber tractar les lletres depenent de les propietats de cada una, si són baixes, si són altes o en canvi són centrals com seria la lletra "a".

Nosaltres hem treballat amb lletres majúscules que són més senzilles de tractar i enfilat en el mateix fil, tot i que tenen algunes complicacions com per exemple les amplades. Algunes lletres són més amples que les altres i per tant es poden veure espais no desitjats entre algunes lletres. En la figura 2.14 podem veure un exemple d'aquests espais.

**GG | I**

Figura 2.14: Diferència entre l'espai de lletres amples i estretes.

En aquest projecte no s'han gestionat les amplades variables, cada lletra ocupa un espai de 50x50 fixes.

## 2.5.2 Noves funcions

### Implementació de text a imatge

**str2img(paraula)** Aquesta funció agafa lletra per lletra de la paraula que li passem i busca en el directori local totes les imatges que tinguin el mateix nom que cada lletra de la paraula. Aquestes són posades en una llista i aquesta és obtinguda per la següent funció `join_images(llista)` com a variable.

**join\_images(llista)** La imatge resultant d'aquesta funció té sempre com a nom; "joined.png". Es crea una imatge buida, l'altura és la màxima de totes les imatges a ajuntar i l'amplada és la suma de totes les amplades de les imatges. A partir d'aquí anem enganxant ordenadament totes les lletres, una al costat de l'altre per formar la paraula. Aquí hem trobat alguns problemes, ja que no totes les lletres segueixen el mateix fil. Per solucionar-ho s'ha decidit treballar únicament amb lletres majúscules, ja que són més fàcils d'enfilegar que les minúscules.

## PYTHON

Figura 2.15: Imatge resultant de la funció.

Aquesta imatge es guarda en local i se li passa a la funció `dealer()`. Que com hem descrit abans s'encarrega de crear els shares.

```
1 def str2img(paraula):
2     llista=[]
3     for letter in paraula:
4         llista+=["letter.png"]
5     return join_images(llista)
6
7 def join_images(imatges):
8     name_file="test.png"
9     imatges_aux=[]
10    for im in imatges:
11        imatges_aux+=["im"]
12    images = map(Image.open, imatges_aux)
13    widths, heights = zip(*(i.size for i in images))
14
15    total_width = sum(widths)
16    max_height = max(heights)
17
18    new_im = Image.new('RGB', (total_width, max_height))
19
20    x_offset = 0
21    for im in images:
22        new_im.paste(im, (x_offset,0))
23        x_offset += im.size[0]
24    #new_im=new_im.resize((int(new_im.size[0]*0.1),int(new_im.size
    ↪ [1]*0.1)))
```

```
25 new_im.save(name_file)
26 imatge = plt.imread(name_file)
27 return imatge
```

### Implementació d'imatge a text

**img2str(image)** Aquesta funció agafa la imatge, la llegeix i la parteix en trossos, on cada tros hi ha una lletra. Aquests trossos de lletra es guarden i s'examinen un a un buscant coincidències amb el diccionari de lletres que tenim guardat.

Les imatges al comparar-les anirem sumant tots els píxels que s'han trobat diferents, al final s'agafarà la imatge que ha donat menys diferències al comparar-les, d'aquesta s'agafarà el nom de la imatge que coincideix amb la lletra representada visualment. Així amb totes les lletres fins a poder presentar la imatge en format de text pla.

```
1 def img2str(img1):
2     image=Image.open(img1)
3     size=image.size
4     llarg=size[0]/size[1]
5     lletres=[]
6     i=0
7     while i<llarg:
8         lletres+= [image.crop((0,0,size[1],size[1]))]
9         image2=image.crop((0,0,size[1],size[1]))
10        image=image.crop((size[1],0,size[0],size[1]))
11        image2.save("cropped"+str(i)+".png", "PNG")
12        i+=1
13        paraula=""
14        for fitxer in find_crop():
15            paraula+=search_image(fitxer)
16            os.remove(fitxer)
17        return paraula
```

### 2.5.3 Interfície per l'usuari

La interfície vam considerar dues possibilitats: per una banda una elaborada amb TkInter, eina molt útil de python, o per altra banda fer-ho en format

pàgina web. Posant tots els pros i contres sobre la taula es va decidir fer-ho amb Tkinter, sobretot perquè és una eina que es programa amb Python, el mateix llenguatge de programació que s'està utilitzant per al programa.

TkInter és un binding de la biblioteca gràfica Tcl/Tk per el llenguatge de programació Python. És considerat un estàndard per la interfície gràfica d'usuari (GUI) per python i és el que ve per defecte per Microsoft Windows.

Per aquest projecte s'ha creat una interfície per unir tots els processos descrits i demostrar el funcionament de manera més atractiva.

A continuació es mostren imatges de la interfície.

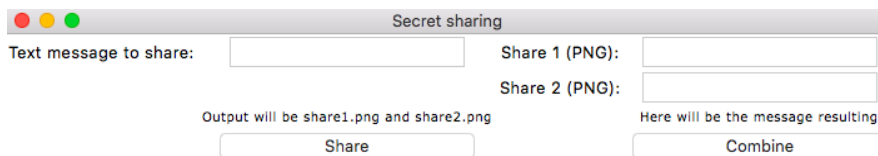


Figura 2.16: Captura de la interfície

La interfície és única i integra les dues funcions: Share i Combine (dealer/-combiner). A la part esquerra hi trobem el dealer, aquí hem d'introduir-hi el missatge secret que volem compartir. A la figura 2.17 hi veiem un exemple. Els inputs estàn marcats en groc i els outputs en blau.

Després d'entrar el secret cliquem el botó Share, això posarà en marxa el procés explicat en les seccions anteriors, fins a retornar de manera local en forma d'imatge els dos shares resultants.

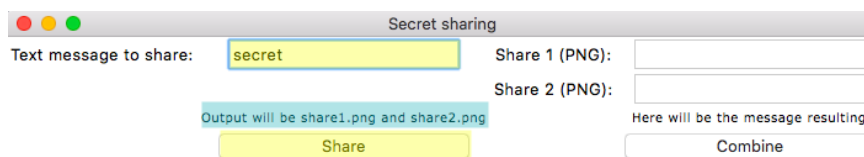


Figura 2.17: Entrar els secret en format text per generar els dos shares.

A la part dreta de la interfície, tenim el combiner. Aquesta part s'encarrega de fer el procés invers, li haurem de passar el nom de dues imatges, aquestes han d'estar a la carpeta local on treballa el programa. El programa serà capaç d'obrir-les i fer el procés descrit també anteriorment. Quan cliquem el botó combine ens apareixerà el secret trobat amb el programa.

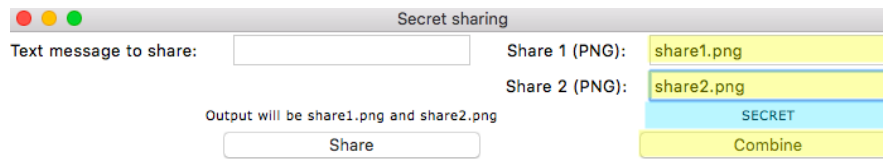


Figura 2.18: Entrar els dos shares per combinar a la interfície.

### 2.5.4 Codi de la interfície

Per programar la interfície s'ha dissenyat un espai senzill on poder tenir les dues funcionalitats juntes. L'usuari no ha de triar entre interfícies per fer un procés o un altre, ho pot triar dins de la interfície. Treballa amb la llibreria dissenyada específicament per aquest projecte (annex A.1) i es comunica a partir de 3 funcions, `share()`, `combine()` i `show()`.

**share()** Funció que és cridada quan l'usuari clica el botó de Share i per tant vol compartir el text que ha entrat per teclat. Aquesta transforma el text en imatge i li passa com a paràmetre a la funció `dealer()` de la llibreria principal.

**combine()** Funció que és cridada quan l'usuari clica el botó de Combine i per tant vol combinar les dues imatges que ha passat com a paràmetre per extreure el possible secret. Aquesta funció agafa el nom de les dues imatges i les passa com a paràmetre a la funció `combinator()` de la llibreria principal del projecte, transforma el secret en text i l'ensenyà per pantalla.

**show()** Funció que ensenya per la interfície el secret trobat en combinar dues imatges.

El codi es pot trobar a l'annex A.2

### 2.5.5 Procés

Ara que tenim presentades totes les funcions que intervenen és interessant veure com treballen dins del procés de `dealer` i `combiner`.

Per començar triem una paraula de pas i que serà el nostre secret,  $S=ABDFM020145D31$ . La introduïm en l'aplicació de la manera explicada en l'apartat 2.5.3.



Figura 2.19: Introducció de la paraula de pas a l'aplicació.

Quan cliquem al botó de Share, el programa agafa el text que hem escrit i li passa a la funció `share()`. Aquesta funció s'encarrega de passar el text escrit a imatge amb la funció `str2img(img)`.

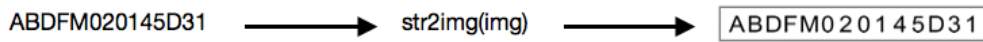


Figura 2.20: Procés de transformar text a imatge

Quan tenim el text passat a imatge, se li passa a la funció `dealer()`. Aquesta funció s'encarregarà de dividir el secret en dos shares i guardar-los a la carpeta on treballa el programa.

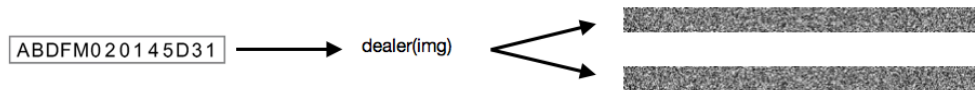


Figura 2.21: Procés de separar el secret en dos shares

En aquest punt ja es poden repartir els shares entre els dos usuaris, en aquest cas.

El procés invers comença amb l'aplicació on haurem d'introduir en l'apartat del combiner el nom dels dos shares.

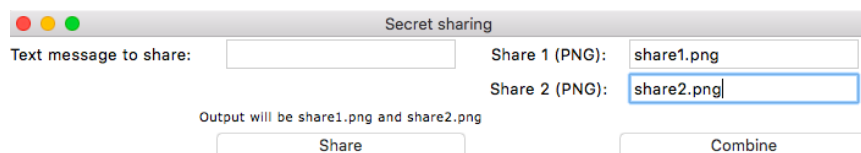


Figura 2.22: Introducció dels shares per obtenir la paraula de pas.

Al clicar en el botó Share es crida la funció `combine()`. Aquesta funció agafa el nom dels dos shares que hem introduït i els hi passa a la funció `combinator(img1, img2)` de la llibreria principal.

La funció `combinator` compara els píxels de cada share i posa en blanc tots aquells que són iguals en dos shares, a la vegada posa en negre tots aquells que són diferents. D'aquesta manera extreu una imatge on hi surt el secret.

El següent pas és passar-li a la funció `img2str(img1)`, la imatge resultant de la funció `combinator(img1, img2)`. Aquesta funció retalla un per un els caràcters de la imatge i els compara amb el diccionari de caràcters local, aquest és el que es fa servir per construir les paraules així que al compararlo les diferències han de ser mínimes. De fet aquesta funció busca la imatge amb la qual hi ha diferència mínima. D'aquesta manera extraïem el missatge que porta la imatge.

Aquest missatge se li passa a l'aplicació i aquesta l'ensenya per pantalla tal com podem veure a la figura 2.23.

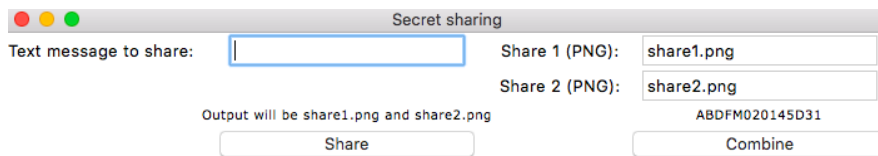


Figura 2.23: Secret sortint per pantalla.

D'aquesta manera queda integrat tot el procés amb totes les funcions programades fent per l'usuari més senzill el procés de compartició de secrets visuals.

## 2.6 Conclusions

Comprendre el funcionament dels secrets visuals ens marca una pauta per poder seguir treballant més en profunditat en la compartició de secrets. La compartició de secrets visuals ens permet compartir qualsevol secret, pot ser imatge, paraules, números o símbols. Aquesta flexibilitat a l'hora de compartir secrets no la tenim en triar el nombre d'usuaris que participen en la compartició. També hem de tenir en compte que aquest esquema ens comporta processar moltes dades per obtenir una quantitat de shares limitats. Per tant no és un sistema adequat per repartir un secret entre molts usuaris.

Una de les particularitats d'aquest esquema de compartició de secrets és la necessitat de tenir tots els shares per recuperar el secret. Cada share que es reparteix és un share necessari per recuperar el secret, veurem que en altres esquemes de compartició pot no ser així necessàriament.

En aquest projecte hem enfocat l'algoritme de dealer i combiner pensant en shares imprimibles amb transparències que quan se superposen deixen veure el secret que amaguen. Això implica que quan una persona té en possessió els dos shares pot ser capaç de desxifrar el secret superposant les imatges o si no és possible sobreposar-les físicament el que hauria de fer és mirar píxel a píxel si cada parell són iguals o diferents, tal com fa l'algoritme programat. El que es vol presentar és que no ha de ser així obligatòriament. Podem acordar un algoritme totalment diferent, és a dir, acordar una clau que no tothom sap. Aquesta clau hauria de ser compartida pel dealer i el combiner. D'aquesta manera seria molt més complicat la resolució del secret tot i tenir tots els shares.

Aquest projecte pot ser punt de partida a altres projectes, ja que encara hi ha moltes aplicacions a descobrir i implementar. Algun exemple seria la possibilitat de repartir els shares de manera instantània per correu electrònic i que amb una tecnologia com el bluetooth o el NFC es pogués ajuntar els dos shares de diferents mòbils i així trobar el secret.

També es podria fer servir per repartir contrasenyes per entrar a espais protegits, on un dels usuaris és la persona que vol accedir i l'altra és l'edifici en si mateix.

Actualment la compartició de secrets visuals s'utilitza en medicina per mantenir en secret les EPR (electron paramagnetic resonance). Aquestes imatges a vegades és necessari el traspàs d'un hospital a l'altra i s'utilitza aquest mètode junt amb la criptografia DNA.



## Capítol 3

# Compartició de secrets via l'esquema vectorial

En aquest capítol tractarem l'esquema de compartició de secrets vectorial. Després d'haver explicat el sistema visual ens ajuda a introduir de manera més senzilla altres esquemes més complexos que no utilitzen imatges i que per tant no s'ha de moure tant volum d'informació.

L'esquema de compartició de secrets vectorial es basa en la repartició d'infinitos punts d'un espai multidimensional, de manera que trobant la intersecció d'aquests punts es determina el secret. Aquests punts infinits que repartim poden ser rectes o plans, de manera que els podem mostrar com una equació. Amb el conjunt d'equacions repartides podem determinar la intersecció, és a dir, el secret.

Per explicar aquest esquema se seguirà una metodologia constructiva, primer s'explicarà l'esquema a un espai de dimensió dos, les rectes, després per espais de dimensió tres, els plans i finalment s'explicarà el cas general per qualsevol dimensió.

### 3.1 Plantejament general de la compartició de secrets

En aquesta secció tractem en general en què consisteix la compartició de secrets, quins són els seus objectius, la notació i quines són les teories generals que hi intervenen.

Un esquema de compartició de secrets es formalitza de la manera següent:

- $S$  és el secret que es vol repartir
- $n$  és el nombre de participants a qui es reparteix el secret
- $S_i$ , anomenat *share*, és el secret repartit a cada participant, per  $i = 1 \dots n$ .
- $k$  és el nombre mínim de participants que han de col·laborar per recuperar el secret
- *dealer* és el responsable, sigui persona o procés, de crear a partir de  $S$  els *shares*  $S_i$  que es reparteixen a cada participant
- *combiner* és el responsable, sigui persona o procés, de combinar els *shares*  $S_i$  de  $k$  participants per recuperar el secret  $S$ .

Els mètodes més coneguts per la compartició de secrets són l'esquema de Shamir i l'esquema vectorial de Blakley.

### Shamir

Adi Shamir va néixer a Tel Aviv, Israel el 1952. Es va llicenciar en Matemàtiques a la Universitat de Tel Aviv, i després va continuar estudiant a l'Institut Weizmann, on va obtenir el màster i el doctorat en Ciències de la Computació. Va treballar com investigador al Massachusetts Institute of Technology (MIT) i més tard com a professor a l'Institut de Massachusetts.

L'esquema de compartició de secrets que porta el seu nom es basa en col·locar el secret com a nombre en el terme independent d'un polinomi. La informació a repartir als usuaris són els punts per on passa la gràfica del polinomi. Per recuperar el secret és necessari interpolar una quantitat  $k$  de punts per recuperar el polinomi.

### Blakley

George Blakley va ser un criptògraf americà i professor de matemàtiques a la Universitat de Texas. Es va llicenciar en física a la Universitat de Georgetown, a Washington DC, i es va doctorar en matemàtiques a la Universitat de Maryland, College Park, el 1960. Després del seu post doc a la Universitat de Cornell, Ítaca, i a la Universitat de Harvard, Boston, va tenir càrrecs

docents a la Universitat D'Illinois al campus Urbana-Champaign i a la Universitat de Buffalo, Nova York. Més tard, va ser director del departament de matemàtiques a la Universitat de Texas, des del 1970 fins al 1978. El 2001 va rebre el títol de Doctor Honoris Causa de la Queensland University of Technology (QUT), i el 2009 va ser nomenat "fellow" de l'IACR. Actualment és professor emèrit del departament de matemàtiques al College of Sciences de la Universitat de Texas, als Estats Units.

L'esquema vectorial de Blakley es basa en la intersecció d'hiperplans. La idea és col·locar el secret en un punt de l'espai de dimensió  $n$ . Llavors repartim hiperplans que interseccionin en el punt de l'espai, de manera que quan s'ajunten  $m$  usuaris, es pot obtenir el secret original. En el cas que s'ajuntin  $m-1$  usuaris no seria possible trobar el secret original.

## 3.2 Descripció de l'esquema vectorial

Per començar, dir que l'esquema de compartició de secrets de Shamir és més popular que el de Blakley. La raó és perquè l'esquema de Blakley no té matrius generals determinades i adequades.

La tècnica de Blakley assumeix que el secret és un punt en un espai de dimensió  $k$ . Hiperplans que interseccionen en aquest punt són els que es fan servir per construir els shares. Els coeficients de  $n$  hiperplans diferents constitueixen els corresponents  $n$  shares. Tant en l'esquema de Shamir com en el de Blakley, un secret és partir entre  $n$  participants. Només si  $k$  shares s'ajunta'n el secret es podrà recuperar.

El secret i els shares de l'esquema de Blakley es poden representar com un sistema lineal  $Ax = y$  on la matriu  $A$  i el vector  $y$  s'obtenen de les equacions dels hiperplans.

L'esquema de Blakley té una aproximació diferent de Shamir basada en la geometria dels hiperplans: Per implementar un  $(t, n)$  esquema, cada un dels  $n$  usuaris se'ls hi és donada una equació d'un hiperplà en una dimensió  $t$ , aquest hiperplà passa per un cert punt no conegut. Quan  $t$  usuaris s'ajunten, poden solucionar el sistema d'equacions i trobar el punt corresponent al secret.

Un hiperplà d'un espai de dimensió  $t$  amb coordenades en un camp  $\mathcal{F}$  es pot descriure amb una equació de la següent forma:

### 3.3 Construcció de l'esquema vectorial

#### 3.3.1 Interpretació de l'esquema vectorial per $m=2$

Suposem que volem repartir un secret a un grup d'usuaris de manera que tan sols dues persones siguin necessàries per reconstruir-lo. Ens estem situant en un espai de dimensió  $m=2$ , és a dir, treballem amb rectes.

Primer fixem el secret  $S$ , aquest és un punt del pla de dimensió  $m=2$ .

Construïm rectes que passin per aquest punt, totes han de ser diferents. A cada usuari repartim una recta  $U_i$  que serà una part del secret,  $S_i$ .

Lavors quan dos usuaris calculin la intersecció dels seus trossos de secret,  $U_j \cap U_k$ .

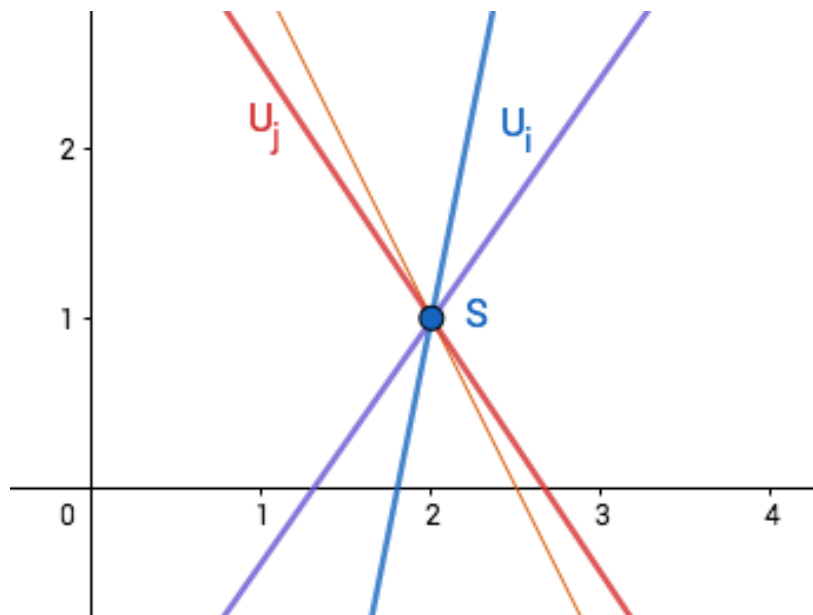


Figura 3.1: Rectes que es tallen en un punt  $S$

En cap cas ens podem trobar amb rectes paral·leles, ja que no passarien pel punt i tampoc podem trobar rectes amb el mateix pendent, ja que seria una recta ja existent i una de les premisses és tenir totes les rectes diferents.

**Exemple .** Repartirem un secret entre 5 persones, les anomenarem  $U_1$ ,  $U_2$ ,  $U_3$ ,  $U_4$  i  $U_5$ .

Triarem un secret  $S$  qualsevol, en aquest cas  $S=(3,5)$ .

De les infinites rectes que hi ha que passen pel punt  $(3,5)$  s'han escollit aquestes:

$$S_1 : y = -2x + 10$$

$$S_2 : y = x + 1$$

$$S_3 : y = \frac{-7}{2}x + \frac{29}{2}$$

$$S_4 : y = \frac{-4}{3}x + \frac{24}{3}$$

$$S_5 : y = \frac{1}{2}x + \frac{15}{6}$$

A cada usuari  $U_i$  se li reparteix un tros de secret  $S_i$ . D'aquesta manera aconseguim compartir un secret.

A la figura 3.2 hi podem veure la gràfica amb les rectes  $S_i$  i la intersecció de totes elles, el secret.

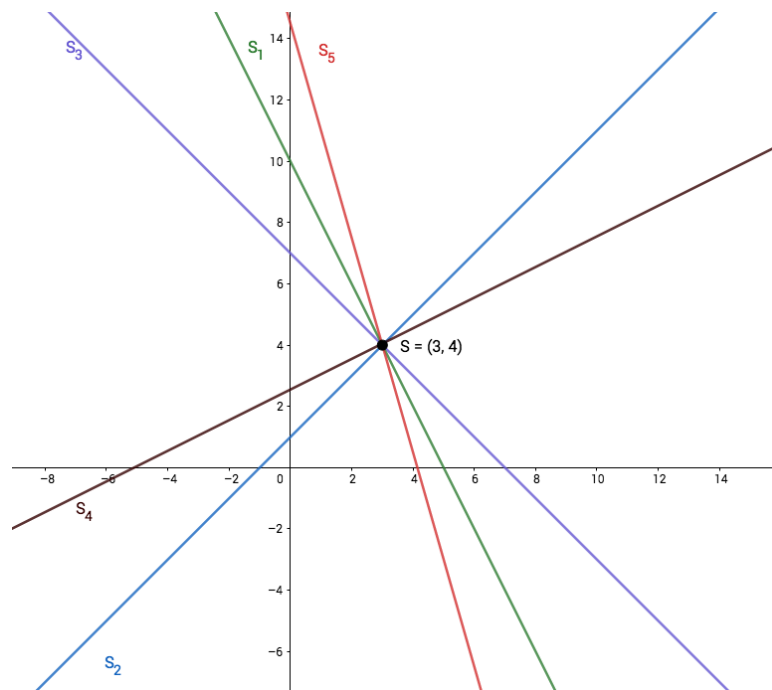


Figura 3.2: Rectes que es tallen en un punt  $S$

### Recuperació del secret:

Estem seguint un esquema de dimensió  $m=2$ , per tant només són necessaris dos usuaris per recuperar el secret original. Aquesta parella d'usuaris pot ser

qualsevol combinació d'entre els 5 que hem repartit.

Per resoldre l'exemple agafem dues equacions qualsevols de les 5 donades:

$$\begin{aligned} S_1 : y &= -2x + 10 \\ S_2 : y &= x + 1 \end{aligned}$$

Aquestes rectes no són paral·leles ni equivalents, per tant assegurem la intersecció en un punt.

Resolem el sistema d'equacions, en aquest cas el resolem per igualació.

$$\begin{cases} y = -2x + 10 \\ y = x + 1 \end{cases} \longrightarrow -2x + 10 = x + 1 \longrightarrow x = 3, y = 4$$

D'aquesta manera obtenim el punt d'intersecció que a la vegada és el secret  $S=(3,4)$ . Ho podem observar a la figura 3.3. Representar-ho gràficament no ens serveix en tots els casos.

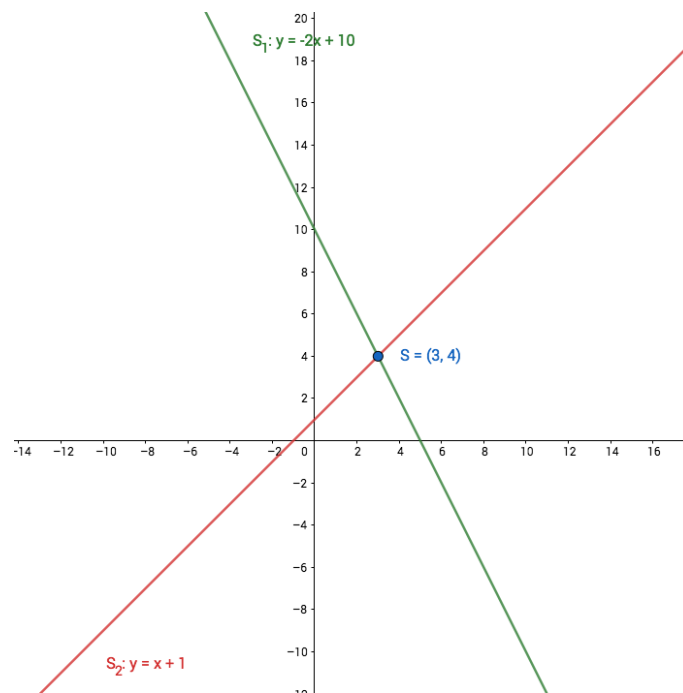


Figura 3.3: Dos shares interseccionant en el punt S, el secret.

Aquesta recuperació és vàlida per qualsevol de les parelles d'usuaris possibles, en aquest cas  $\binom{5}{2} = 10$  parelles. Per comprovar-ho a continuació un segon exemple amb una altra combinació de shares.

En aquest cas hem escollit els següents shares:

$$S_3 : y = \frac{-7}{2}x + \frac{29}{2}$$

$$S_5 : y = \frac{1}{2} + \frac{15}{6}$$

Resolem el sistema d'equacions de la mateixa manera que hem operat anteriorment.

$$\begin{cases} y = \frac{-7}{2}x + \frac{29}{2} \\ y = \frac{1}{2} + \frac{15}{6} \end{cases} \longrightarrow \frac{-7}{2}x + \frac{29}{2} = \frac{1}{2} + \frac{15}{6} \longrightarrow x = 3, y = 4$$

En la figura 3.4 es pot comprovar gràficament que certament la intersecció de les dues rectes és el secret.

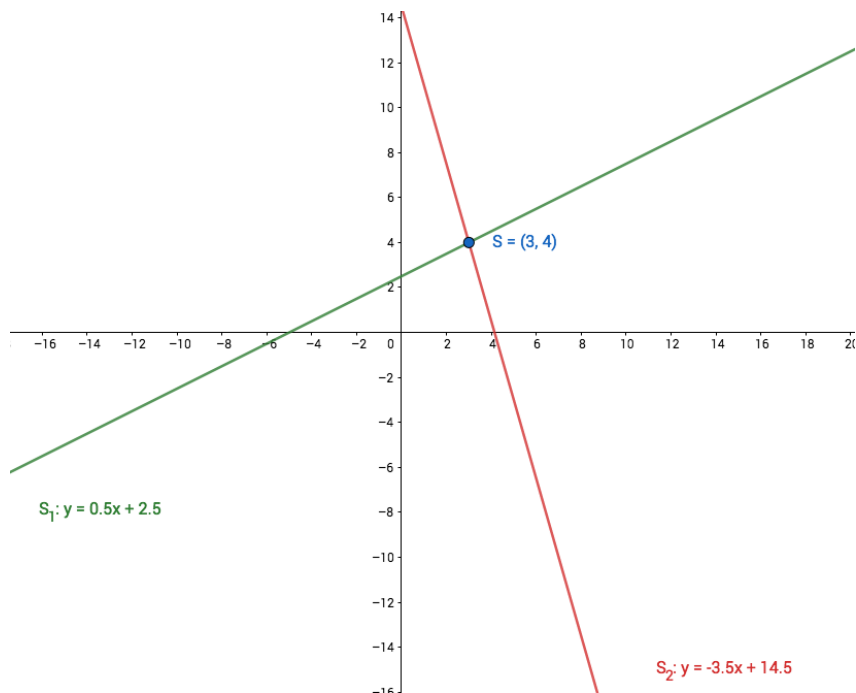


Figura 3.4: Dos nous shares que intersequen en un punt S, el secret.

Com observem una tercera persona seria redundant a l'hora de saber el secret, si volem un tercer usuari per recuperar el secret, hem d'augmentar la dimensió de l'espai.

### 3.3.2 Interpretació de l'esquema vectorial per $m=3$

Fins ara hem seguit l'esquema amb una dimensió  $m=2$ , només necessitàvem 2 usuaris per recuperar el secret. A vegades necessitem més de dues persones per recuperar el secret, això és possible si augmentem la dimensió de l'espai en el qual treballem, en aquest cas treballarem amb dimensió  $m=3$ .

El procés a seguir és el mateix, suposem que volem repartir un secret entre uns usuaris per a després més tard poder recuperar-lo, almenys tres persones hauran d'estar presents amb la seva part de secret.

Primer triem un secret a repartir, en aquest cas es triarà el mateix que abans però afegint una tercera coordenada, fins ara treballàvem amb el parell  $(x,y)$ , ara tindrem  $(x,y,z)$ .

A partir d'aquí s'ha de generar una quantitat de plans igual a la quantitat d'usuaris que participin en la compartició del secret. Els plans generats han de tenir unes condicions entre ells:

- El punt  $S$  ha de formar part de tots els plans.
- La intersecció de tres plans qualsevol només poden donar com a resultat el punt  $S$ .

Els plans a  $\mathbb{R}^3$  vénen donats per equacions del tipus:  $Ax + By + Cz = D$ .

Per aconseguir la primera premissa, que el pla passi pel punt  $S$ , és senzill. Dels quatre valors  $A,B,C,D$ , tres són lliures, els triem nosaltres, el quart sortirà de substituir el punt i resoldre l'equació. A continuació un exemple:

**Exemple .** Volem un pla que passi pel punt  $S = (3, 4, 6)$ , que segueixi la forma d'equació  $Ax + By + Cz = D$ .

Primer de tot triem  $A,B$  i  $C$  aleatòriament.

$$A = 1, B = 2, C = -1$$

Substituïm a l'equació d'aquesta manera:

$$x + 2y - z = D$$

Ara substituïm  $x, y$  i  $z$  pel punt  $S = (3, 4, 6)$ .

$$3 + 2 * 4 - 6 = D$$



Resolem i trobem  $D$ .

$$D = 5$$

Per tant el pla  $x + 2y - z = 5$  passa pel punt  $S = (3, 4, 6)$ .

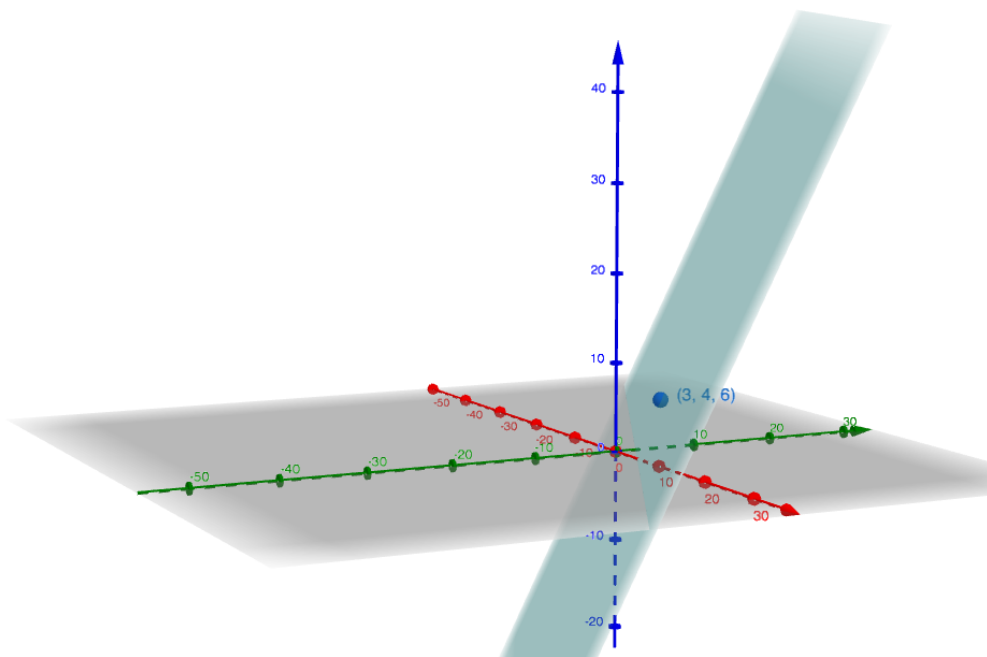


Figura 3.5: Pla que generat a l'exemple que conté el punt  $S = (3, 4, 6)$ .

Per la segona condició, hem d'estudiar les diferents posicions entre plans, aquests, com les rectes poden ser paral·lels o secants, en aquest cas podem descartar els plans paral·lels, ja que només ens interessen plans que comparteixen punts, és a dir, els que tenen intersecció. Els plans paral·lels formen un sistema d'equacions incompatible.

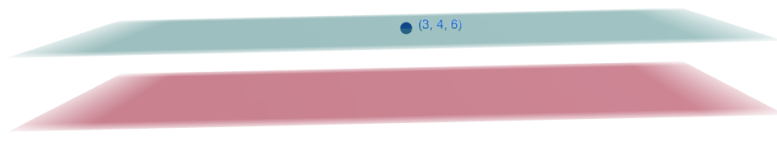


Figura 3.6: Dos plans paral·lels entre ells.

Dels secants extraiem que descriuen un sistema d'equacions compatible. És a dir al resoldre el sistema d'equacions obtenim un resultat que és el punt que comparteixen els plans implicats. L'objectiu és trobar un feix de plans que tinguin un sol punt d'intersecció. En la figura 3.7 podem observar dos plans que interseccionen i que comparteixen una recta de punts, entre ells el punt  $S$ .

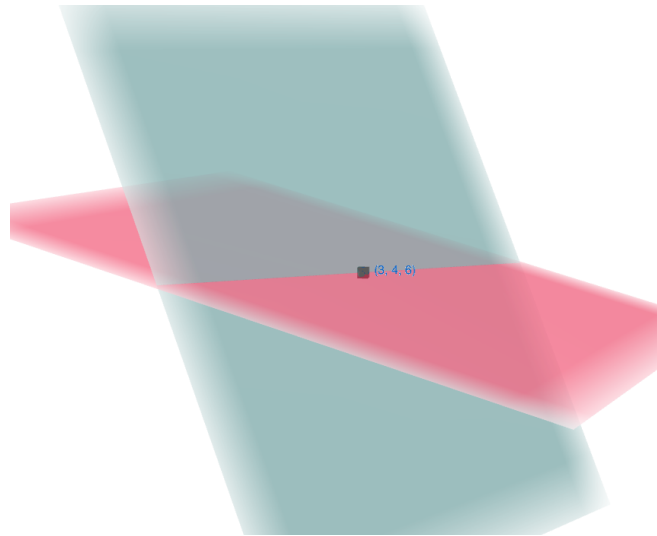


Figura 3.7: Dos plans secants que comparteixen el punt  $S$ .

El punt  $S$  no queda definit amb aquests dos plans, ja que el resultat de la intersecció és una recta d'infinitos punts dels quals només un és el secret. Per entendre-ho millor ho treballarem com un exemple.

**Exemple .** Agafem el mateix punt que hem estat treballant fins ara per donar-li continuïtat,  $S = (3, 4, 6)$ .

Farem 4 particions de secret, és a dir, participaran 4 usuaris  $U_1, U_2, U_3, U_4$ . Fent servir el mètode que hem seguit per trobar el pla de la figura 3.5 trobarem els altres 3 que ens queden.

Aquests són una de les infinites possibilitats:

$$\begin{aligned} S_1 : x + 2y - z &= 5 \\ S_2 : x + 4y - 16z &= -80 \\ S_3 : 2y - z &= 14 \\ S_4 : 2x + y - 2z &= 22 \end{aligned}$$

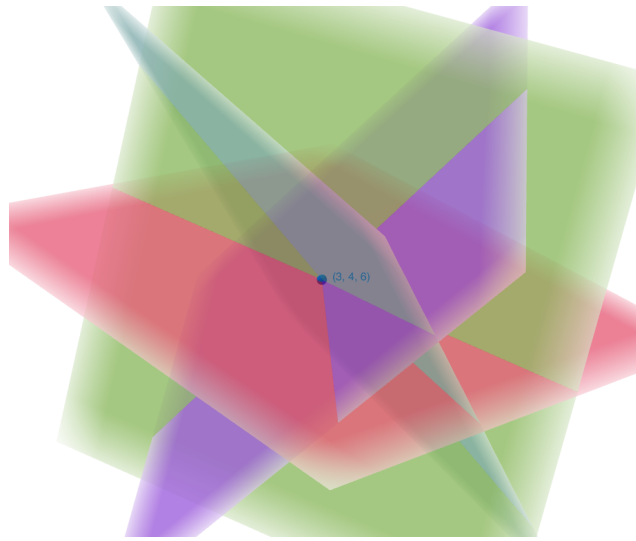


Figura 3.8: Quatre plans generats que comparteixen el punt S.

Es pot veure que els 4 plans coincideixen en un sol punt, aquest és el secret  $S$  que volem compartir. El fet és que qualsevol combinació de 3 plans ens determina el punt. En el cas d'agafar-ne menys l'únic que tindrem serà una recta d'infinits punts o un pla solitari.

El procés per trobar el secret és igual que amb dimensió  $m=2$ . Qualsevol combinació de tres usuaris és vàlida. En aquest exemple hem triat aquests usuaris:

$$\begin{aligned} S_1 : x + 2y - z &= 5 \\ S_3 : 2y - z &= 14 \\ S_4 : 2x + y - 2z &= 22 \end{aligned}$$

Plantegem un sistema d'equacions amb els usuaris que intervenen en el procés.

$$\begin{cases} x + 2y - z = 5 \\ 2y + z = 14 \\ 2x + y - 2z = 22 \end{cases}$$

El podem resoldre de qualsevol mètode, en aquest cas es fa servir substitució.

Aïllem  $z = 14 - 2y$ , substituïm i obtenim les dues equacions resultants.

$$\begin{cases} x + 4y = 19 \\ 2x - 3y = -6 \end{cases}$$

Agafem una de les dues equacions, aïllem una de les incògnites,  $x = 19 - 4y$ , la substituïm en l'altra equació i resollem. D'aquesta manera obtenim el

primer resultat del secret  $y = 4$ . D'aquest resultat obtenim les dues següents incògnites,  $x = 3$  i  $z = 6$ , amb aquestes ja tenim el secret  $S$  descobert.

$$S = (3, 4, 6)$$

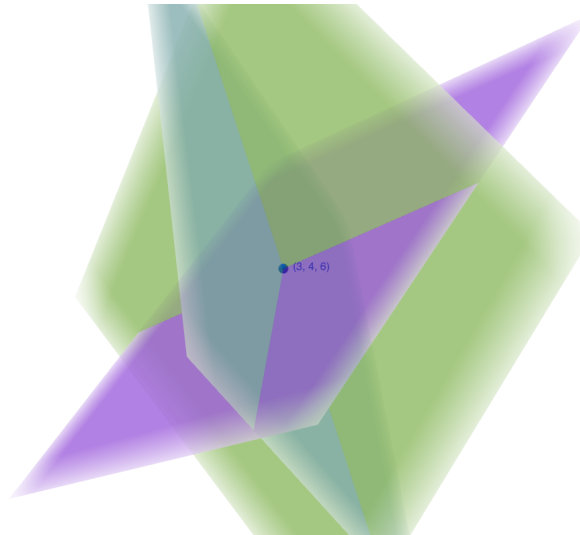


Figura 3.9: Els tres plans interseccionen en el secret  $S$ .

## 3.4 Construcció dels hiperplans

### 3.4.1 Construcció de plans en dimensió 3

Per construir un esquema de repartició és fonamental la construcció dels plans que es repartiran, fins ara ens hem mogut en espais de dimensió dos i tres. Aquests espais poden pujar de dimensió i per tant les equacions es compliquen, Són equacions que no es poden representar gràficament i que compliquen la idealització d'aquestes situacions. Construir plans sense cap manera sistemàtica es complica i es necessita més temps per comprovar que compleixen les condicions necessàries entre ells, ha de formar un sistema determinat, és a dir una matriu de rang 3.

La idea a seguir és triar els 3 coeficients de cada pla com hem fet en els anteriors casos, però seguirem un mètode que assegura que els plans resultants compartiran el secret  $S$  que es busca compartir. A continuació s'explica com arribar a aquest mètode:

Siguin  $A_i x + B_i y + C_i z = D$ ,  $i=1,2,3$ , tres d'aquests plans que volem construir. El sistema d'equacions

$$\begin{cases} A_1 x + B_1 y + C_1 z = D_1 \\ A_2 x + B_2 y + C_2 z = D_2 \\ A_3 x + B_3 y + C_3 z = D_3 \end{cases}$$

ha de ser compatible, perquè tots els plans contenen el punt S per construcció. Perquè el punt S sigui l'única solució d'aquest sistema hem d'estudiar les matrius, la matriu de coeficients ha de ser del mateix rang que la matriu ampliada, en aquest cas ha de ser 3, igual al nombre d'incògnites. Aquest rang depèn del determinant de la matriu de coeficients.

$$\begin{vmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \\ A_3 & B_3 & C_3 \end{vmatrix}$$

Si el determinant és diferent de 0, el rang és 3 i el sistema serà compatible determinat, exactament el que busquem. Si és igual a 0, serà compatible indeterminat, és a dir, compartiran una recta comuna. L'objectiu doncs és construir plans que siguin compatibles determinats. Anar comprovant el determinant de cada trio ens genera un temps de càlcul elevat si treballem amb moltes comparticions.

Una manera més intel·ligent i ràpida és utilitzar el determinant de Vandermonde. Siguin  $x_0, x_1, x_2 \in \mathbb{R}$  tres nombres reals qualsevol. Representem el determinant de Vandermonde de 3x3 d'aquesta manera:

$$\begin{vmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{vmatrix} = (x_1 - x_0)(x_2 - x_0)(x_2 - x_1)$$

El determinant de Vandermonde mai és 0, això ens aporta un patró per generar equacions que ens donin matrius de determinant no nul, és a dir sistemes compatibles determinats. El patró a seguir és el següent:

- El coeficient que companya la  $x$  ha de ser  $A = 1$ .
- El coeficient que acompanya la  $y$  ha de ser un nombre real  $B$  qualsevol (no es pot repetir en cap altra pla).
- El coeficient que acompanya a  $z$  ha de ser el quadrat del coeficient de la  $y$ , és a dir  $C = B^2$ .

- El terme independent  $D$  ha de ser triat de manera que el pla resultant contingui el punt  $S$ .

D'aquesta manera trobem  $n$  plans que entre ells comparteixin el secret  $S$ .

$$\begin{aligned}x + B_1y + B_1^2z &= D_1 \\x + B_2y + B_2^2z &= D_2 \\x + B_3y + B_3^2z &= D_3 \\&\vdots \\x + B_ny + B_n^2z &= D_n\end{aligned}$$

Qualsevol trio que s'agafi ens donarà el punt  $S$ .

$$\begin{vmatrix} 1 & B_0 & B_0^2 \\ 1 & B_1 & B_1^2 \\ 1 & B_2 & B_2^2 \end{vmatrix} \neq 0$$

Aquest sistema es podria complicar si necessitem construir molts plans, ja que cada cop haurem de fer servir números més elevats, de totes maneres podem fer servir qualsevol nombre real, és a dir, també podem utilitzar negatius i fraccions.

No s'ha d'utilitzar mai el cas  $B = 0$ , ja que estaríem donant un pla d'aquest tipus:  $x + 0y + 0z = S_x$ , i l'usuari amb aquest pla sabria part del secret.

### 3.4.2 Construcció de l'esquema vectorial en general

Per fer el cas general, en què el secret sigui recuperable per  $m$  usuaris, cal que la intersecció de  $m$  equacions determini un punt. Per tant necessitem equacions amb  $m$  incògnites, que formen part de l'hiperplà  $\mathbb{R}^m$ . El secret formarà part del mateix hiperplà,  $S = (x_1, x_2, x_3, \dots, x_m)$ . Aquest secret serà repartit entre  $n$  usuaris. Qualsevol combinació de  $m$  usuaris ha de ser capaç de reconstruir el secret  $S$ , mentre que combinacions de  $m - 1$  o menys, no.

Cada usuari ha de tenir una equació que contingui el secret  $S$ , aquesta és de la forma:  $A_1x_1 + A_2x_2 + \dots + A_mx_m = A_0$ . Aquesta equació forma part de l'hiperplà  $\mathbb{R}^m$ .

Perquè el secret  $S$  es pugui recuperar hem d'ajuntar  $m$  equacions que generin una matriu de rang  $m$ , és a dir un determinant no nul. Podem seguir aplicant

el determinant de Vandermonde.

$$\begin{vmatrix} 1 & B_0 & B_1^2 & \dots & B_1^{m-1} \\ 1 & B_1 & B_2^2 & \dots & B_2^{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & B_n & B_n^{m-1} & \dots & B_n^{m-1} \end{vmatrix} \neq 0$$

Triem una  $B_i$ , els coeficients seran els corresponents a les potències de  $B_i$  fins a arribar a  $B_i^{m-1}$ .

$$1 * x_1 + B_i * x_2 + B_i^2 * x_3 + \dots + B_i^{m-1} * x_m = D_i \quad (3.1)$$

La  $D_i$  es calcula substituint el secret a l'equació perquè aquest formi part de l'equació.

A l'hora de recuperar el secret es necessita la col·laboració de  $m$  usuaris amb el seu share, i fer la intersecció. Quan ens trobem amb una  $m$  gran els ordinadors són els responsables de resoldre aquests sistemes.

## 3.5 Programació de l'esquema general per Vandermonde

Per aquesta part s'ha construït una nova interfície capaç de gestionar la compartició de secrets mitjançant l'esquema general exposat en la secció anterior. En el primer apartat ens trobem una explicació de la interfície, com està programada i com està dissenyada. A continuació expliquem com treballa el dealer en aquest sistema i com s'ha programat. Per acabar la secció s'explica la funcionalitat del combiner i el procés que segueix fins a donar l'output.

### 3.5.1 Interfície

La interfície creada segueix el mateix patró que la que utilitza el mètode visual. El software està programat en Python i TkInter és l'eina més útil que ens ofereix aquest llenguatge de programació.

A continuació podem veure l'aspecte d'aquesta interfície.

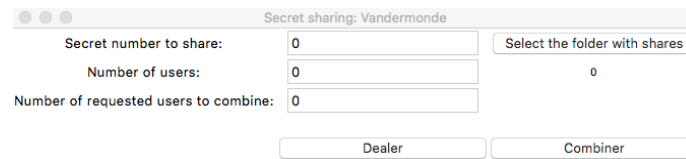


Figura 3.10: Primera vista de la interfície per l'esquema vectorial utilitzant Vandermonde.

Hi podem distingir dues seccions amb les quals interactuar, la part de dealer i la part de combiner.

**share()** Funció que és cridada quan l'usuari clica el botó de Share i per tant vol compartir el número que ha entrat per teclat. Aquesta agafa els paràmetres entrats,  $(n,k,s)$  i li passa a la funció **dealer()** de la llibreria principal de la part vectorial que és l'encarregada de crear els shares corresponents. Aquesta funció també ensenya per la interfície quina és la carpeta on es troben els shares resultants.

**combine()** Funció que és cridada quan l'usuari clica el botó de Combine i per tant vol combinar les imatges que hi ha a la carpeta que ha passat com a paràmetre i així extreure el possible secret que contenen. Aquesta funció agafa tots els arxius .txt i els converteix en una matriu de shares. Quan té la matriu creada li passa a la funció **combiner()** de la llibreria principal de la part vectorial del projecte. Quan ha trobat el missatge l'ensenya a la interfície.

**get\_last\_folder()** Funció que retorna la carpeta amb el número més gran del directory.

**openDirectory()** Funció que obra el navegador per escollir el directori on es troben els shares per combinar.

El codi d'aquestes funcions i el programari de la interfície es pot trobar a l'annex a la secció A.4.

En els següents apartats s'expliquen per separats la manera com treballen aquests dos grans processos, dealer i combiner.

### 3.5.2 Dealer

El dealer és l'encarregat de generar  $n$  equacions o shares que seràn repartides a  $n$  usuaris o participants de l'esquema de repartició.



Per generar els shares necessitem entrar el secret  $S$  a repartir, el nombre de participants  $n$  i el nombre d'usuaris  $k$  necessaris per recuperar el secret. Aquests han de ser nombres naturals.

En l'exemple següent podem veure com introduïm aquests inputs necessaris.

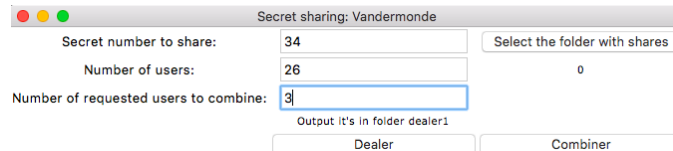


Figura 3.11: Primera vista de la interfície per l'esquema vectorial utilitzant Vandermonde.

En introduir  $S, n, k$  i clicar el botó "Dealer" comença el procés programat en aquest apartat.

El programa crea un secret aleatori i en l'última coordenada hi situa el secret.

$$[x_1, \dots, x_{k-1}, x_s]$$

Seguidament crea una matriu  $n$  files i  $k$  columnes, aquesta matriu és completa d'uns.

$$\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{pmatrix}$$

A aquesta matriu aplicarem Vandermonde, trobarem aleatòriament els coeficients  $B_n$  de cada matriu i quan tinguem l'equació amb els coeficients, substituint el punt  $S$  trobarem el coeficient  $D$ . Després d'això ja tenim la matriu acabada i disposada per ser repartida als  $n$  usuaris.

$$\begin{pmatrix} 1 & B_0 & B_1^2 & \dots & B_1^{m-1} \\ 1 & B_1 & B_2^2 & \dots & B_2^{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & B_n & B_n^{m-1} & \dots & B_n^{m-1} \end{pmatrix}$$

Cada equació  $n$  es guarda en un fitxer .txt i tota la coalició de fitxers es guarda en una mateixa carpeta.

Nombre	Fecha de modificación	Tamaño	Clase
browse.py	anteayer 18:25	544 bytes	Python
combiner	ayer 18:30	--	Carpeta
combiner2	hoy 15:57	--	Carpeta
dealer1	hoy 16:16	--	Carpeta
dealer2	hoy 16:17	--	Carpeta
dealer3	hoy 16:19	--	Carpeta
dealer4	hoy 16:21	--	Carpeta
gui.py	hoy 16:20	3 KB	Python
matrix.py	hoy 16:17	3 KB	Python
matrix.pyc	hoy 16:17	5 KB	Docume

Figura 3.12: Carpets generades pel programa.

Dintre de la carpeta trobem  $n$  arxius de text en format `.txt`, dintre dels arxius hi trobem un share diferent de tots els altres de la carpeta, aquests són de la forma  $a_1x_1 + \dots + a_kx_k = a_k + 1$ . El dealer és l'encarregat d'agafar aquests shares i repartir-ho als  $n$  usuaris. La carpeta on es guarda ens la diu la interfície quan cliquem el botó "Dealer" després d'introduir els paràmetres pertinents.

**Exemple .** En aquest exemple decidirem repartir el secret  $S=21$  amb un total de  $n=15$  usuaris, amb la condició de tenir  $k=4$  usuaris per recuperar el secret.

Secret sharing: Vandermonde

Secret number to share: 21      Select the folder with shares

Number of users: 15      0

Number of requested users to combine: 4

Output it's in folder dealer1

Dealer      Combiner

Figura 3.13: Introducció dels paràmetres de l'exemple.

Al clicar en el botó de "Dealer" ens crea una carpeta "dealer1" on trobarem els 15 shares.

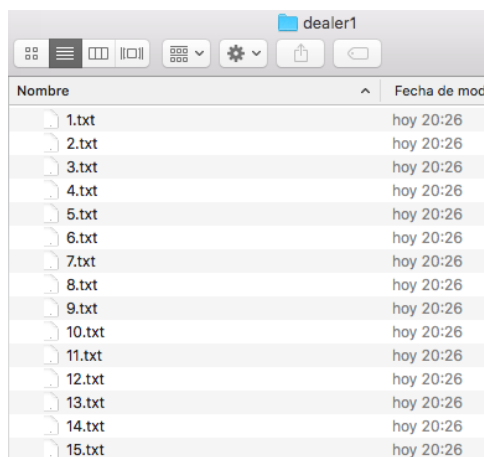


Figura 3.14: Carpeta on estan ubicats els 15 shares.

Aquests shares són el resultat de crear una matriu utilitzant Vandermonde on el primer coeficient és un 1, el segon escollim una  $B$  qualsevol i els següents són les potències de  $B$  desde  $B^2$  fins a  $B^{k-1}$ . A continuació ho podem veure en la matriu 3.5.2.

$$\begin{pmatrix} 1 & 23 & 529 & 12167 & 237795 \\ 1 & 131 & 17161 & 2248091 & 31673031 \\ 1 & 69 & 4761 & 328509 & 4956797 \\ 1 & 73 & 5329 & 389017 & 5824345 \\ 1 & 210 & 44100 & 9261000 & 126656153 \\ 1 & 262 & 68644 & 17984728 & 243536449 \\ 1 & 20 & 400 & 8000 & 164793 \\ 1 & 282 & 79524 & 22425768 & 302808329 \\ 1 & 232 & 53824 & 12487168 & 169972129 \\ 1 & 230 & 52900 & 12167000 & 165679233 \\ 1 & 137 & 18769 & 2571353 & 36103449 \\ 1 & 268 & 71824 & 19248832 & 260419465 \\ 1 & 47 & 2209 & 103823 & 1671339 \\ 1 & 271 & 73441 & 19902511 & 269145931 \\ 1 & 272 & 73984 & 20123648 & 272097489 \end{pmatrix}$$

Cada fila és un share que segueix la forma expressada anteriorment  $a_1x_1 + \dots + a_kx_k = a_k + 1$ , aquesta matriu segueix la norma de què qualsevol conjunt de  $k = 4$  shares conforma un sistema d'equacions compatible determinat del qual el resultat és el punt que conté el secret.

Quan tenim la matriu creada el dealer és l'encarregat de repartir cada share a cada usuari.

### 3.5.3 Combiner

La part de combinar és més senzilla gràficament, no trobem caselles per omplir. L'input del combiner és una carpeta amb shares, igual que l'output del dealer, però amb la diferència que en aquesta carpeta només hi poden haver els  $k$  shares necessaris.

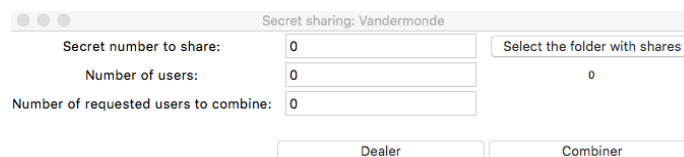


Figura 3.15: Primera vista de la interfície per l'esquema vectorial utilitzant Vandermonde.

Per seleccionar aquesta carpeta on tenim els  $k$  shares, s'ha de clicar el botó "Select folder with shares" i apareixerà un cercador per seleccionar la carpeta que desitgem.

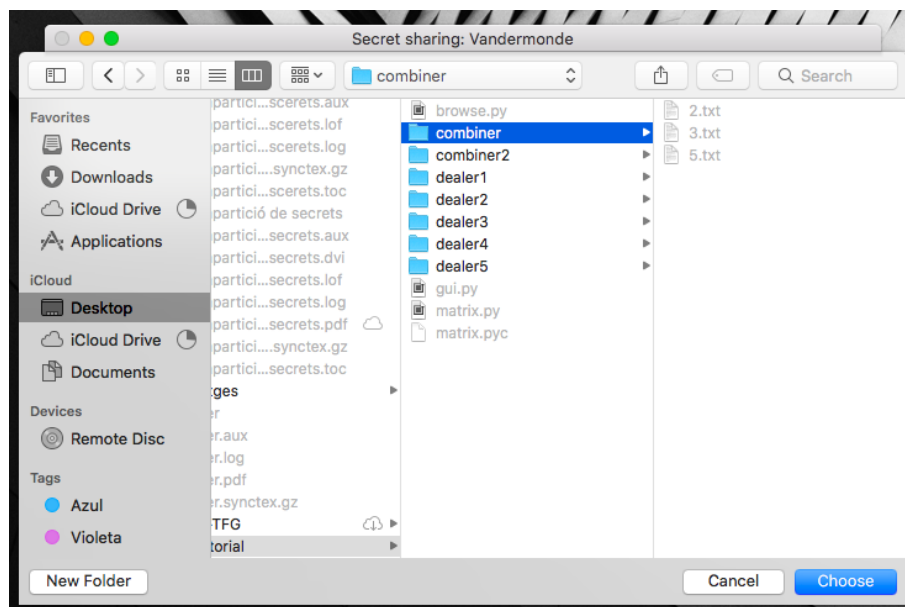


Figura 3.16: Cercador per seleccionar la carpeta.

Després d'això en clicar "Combiner" el programa agafa els shares de tots els fitxers i crea una matriu, si es pot resoldre el sistema i per tant es pot extreure el secret sortirà per pantalla com veiem en la imatge.

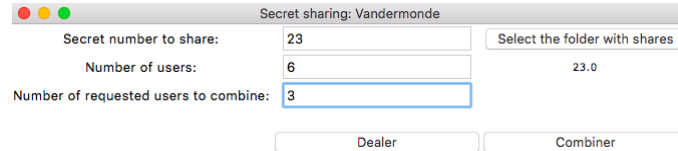


Figura 3.17: Secret sortint per pantalla.

$$\begin{pmatrix} 1 & 23 & 529 & 12167 & 237795 \\ 1 & 131 & 17161 & 2248091 & 31673031 \\ 1 & 69 & 4761 & 328509 & 4956797 \\ 1 & 230 & 52900 & 12167000 & 165679233 \end{pmatrix}$$

El resultat d'aquest sistema el trobem utilitzant la llibreria NumPy de Python, amb la funció `numpy.linalg.solve(A, B)`, on  $A$  és la matriu de coeficients i  $B$  la matriu de termes independents.

$$\begin{pmatrix} 1 & 23 & 529 & 12167 \\ 1 & 131 & 17161 & 2248091 \\ 1 & 69 & 4761 & 328509 \\ 1 & 230 & 52900 & 12167000 \end{pmatrix} = A$$

$$\begin{pmatrix} 237795 \\ 31673031 \\ 4956797 \\ 165679233 \end{pmatrix} = B$$

$$S = [245, 101, 152, 21]$$

Quan tenim el punt hem de recordar que el secret tan sols és una de les coordenades i sempre és l'última, només cal recuperar l'última posició del resultat donat per l'equació.

Quan utilitzem aquest mètode ens trobem que acabem treballant amb números molt grans i que creixen que dificulten el càlcul exacte del secret, ja que en calcular la matriu inversa dividim i com a resultat tenim comes flotants, això ens suposa pèrdua d'informació que podem resoldre aproximant, però fins

a un cert punt. A continuació tenim un share on podem veure el tipus de números als quals ens podem trobar quan volem repartir shares amb una  $k = 8$ .

[1, 174, 30276, 5268024, 916636176, 159494694624, 27752076864576, 4828861374436224, 59615189316690037739]

Fent proves s'ha arribat a un límit en el qual ja no és capaç de tornar el secret de forma correcta, aquest és quan treballem amb shares en dimensió  $k = 25$ . La llibreria Numpy no és capaç de calcular amb la precisió adequada quan tenim condicions de  $k > 24$  i  $n! > k$ . Aquest límit encara és més reduït si shares que agafem per recuperar el secret contenen els números més grans ens pot retornar un fals positiu, és a dir, ens retorna un secret que nosaltres el donem per bo però en realitat no ho és.

Per solucionar això s'ha fet recerca sobre una possible solució o implementació que ens aportí un resultat més bo. En el pròxim capítol es tracta aquest nou mètode.

## 3.6 Matrius de Hei-Du-Song

Quan parlem d'esquemes de compartició de secrets el de Shamir és més popular que el de Blakley. La raó és perquè l'esquema de Blakley manca de matrius determinades, generals i intercanviables. En aquesta secció es presenta dues matrius que es poden fer servir per a l'esquema de compartició de secrets de Blakley. Comparat amb Vandermonde aquesta matriu creix de manera més lenta.

### 3.6.1 Plantejament general

La tècnica de Blakley assumeix que el secret és un punt en una dimensió  $k$  de l'espai. Els plans que interseccionen en aquest punt es fan servir per construir els shares. Els coeficients de  $n$  hiperplans diferents formen els  $n$  shares a repartir. En aquests dos esquemes el secret és partit entre  $n$  participants. A menys que  $k$  shares siguin reunits, el secret no es podrà recuperar.

La proposta de Hei-Du-Song Hei-du-song és utilitzar dues matrius per tal de recuperar de forma exacta el secret, els números amb coma flotant no estan permesos en aquestes matrius, tan sols s'utilitza nombres enters. Tampoc hi ha d'haver zeros.

Aquestes noves matrius funcionen millor que la de Vandermonde en termes d'expansió de coeficients, computació i emmagatzematge. En aquest projecte també s'ha creat un programa amb el mateix estil que Vandermonde, explicat en l'apartat 3.5, però fent servir les matrius estudiades en aquest apartat.

Per crear aquestes matrius tenim  $a_{pq} = a_{p-1q-1} + a_{p-1q}$  on ( $p > 1, q > 1$ ) per tant deduïm que

$$a_{p,q} - a_{1,q} = \sum_{k=1}^{p-1} a_{k,q-1}$$

o

$$a_{p,q} = 1 + \sum_{k=1}^{p-1} a_{k,q-1} = 1 + (p-1) + \sum_{k=1}^{p-1} \sum_{w=1}^{k-1} a_{w,q-2} = p + \sum_{k=1}^{p-1} \sum_{w=1}^{k-1} a_{w,q-2}$$

Per començar l'esquema de compartició de secrets el dealer genera  $(k-1)$  nombres aleatoris  $x_1, x_2, \dots, x_{k-1}$ , i els combina amb un secret  $S$  que serà el nombre  $x_k$  per obtenir el punt  $(x_1, x_2, \dots, x_{k-1}, x_k)$ .

Quan tenim el punt generat el dealer ha de repartir els  $n$  hiperplans de dimensió  $k$  que són els shares.

El primer usuari s'emporta  $y_1 = x_1 + x_2 + \dots + x_k$

L'usuari  $k$  s'emporta  $y_k = x_1 + kx_2 + \frac{k^2-k+2}{2}x_3 + 2^{k-1}x_k$

L'usuari  $k+1$  obté  $y_{k+1} = x_1 + (k+1)x_2 + \frac{k^2+k+2}{2}x_3 + \dots + (k+1 + \sum_{k=1}^{p-1} \sum_{w=1}^{k-1} a_{w,k-3})x_{k-1} + (2^k - 1)x_k$

Per recuperar el secret de  $k$  usuaris, el que hem de fer és solucionar el sistema d'equacions lineals que formen aquests  $k$  shares i així obtenir el punt exacte, sabent que l'última coordenada  $(x_k)$  és el secret.

$$\left\{ \begin{array}{l} y_1 = x_1 + x_2 + \dots + x_k \\ \vdots \\ y_k = x_1 + kx_2 + \frac{k^2-k+2}{2}x_3 + 2^{k-1}x_k \\ y_{k+1} = x_1 + (k+1)x_2 + \frac{k^2+k+2}{2}x_3 + \dots + (k+1 + \sum_{k=1}^{p-1} \sum_{w=1}^{k-1} a_{w,k-3})x_{k-1} + \\ (2^k - 1)x_k \\ \vdots \end{array} \right.$$

Les matrius resultants són les següents.

$$A_1 = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & k-1 & \frac{k^2-k+2}{2} & \dots & 2^{k-1} \end{pmatrix}$$

L'expressió de la matriu  $A_2$  és la següent:

$$\begin{pmatrix} 1 & k+1 & \frac{k^2-k+2}{2} & \dots & k+1 + \sum_{k=1}^{p-1} \sum_{w=1}^{k-1} a_{w,k-3} & 2^k - 1 \\ 1 & k+2 & \frac{k^2+3k+4}{2} & \dots & \dots & k+2 + \sum_{k=1}^{p-1} \sum_{w=1}^{k-1} a_{w,k-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

En aquests sistemes hi trobem propietats que ens faciliten el càlcul del sistema.

### Propietats

- El determinant de la matriu de  $k$  participants qualsevol, sempre és 1. Això facilita el càlcul de la inversa de la matriu.
- El número màxim a la matriu és de  $2^n$ . El número màxim a Vandermonde era  $n^n$ .
- La matriu és fàcilment extensible i dinàmica. Quan  $k$  és fixada podem afegir o eliminar files sense que afecti altres peces.

A continuació s'explica la construcció d'aquestes matrius en aquest projecte.

### 3.6.2 Construcció efectiva de les matrius

Per construir una matriu de 3x3 utilitzant Hei-Du-Song procediríem d'aquesta manera:

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & a_{22} & a_{23} \\ 1 & a_{32} & a_{33} \end{pmatrix}$$



Per trobar el valor de  $a_{2,2}$  aplicariem la fórmula:

$$a_{22} = a_{(2-1)(2-1)} + a_{(2-1)2} = a_{11} + a_{12} = 1 + 1 = 2$$

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & a_{23} \\ 1 & a_{32} & a_{33} \end{pmatrix}$$

Si ens hi fixem el que sumem són el número que tenim a dalt de la posició que volem calcular més el que tenim en diagonal amunt a l'esquerre.

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 3 & 4 \end{pmatrix}$$

Per tant cap fila  $i$  no pot ser representada per cap combinació lineal de les  $(i - 1)$  files, llavors el rang de la matriu sempre serà màxim. Qualsevol  $k$  files de vectors a la matriu  $A$  són linealment independents.

Ara que s'ha explicat com es troben els valors de la matriu, podem veure en el següent exemple una de més gran i comprovar algunes propietats.

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 2 & 2 \\ 1 & 3 & 4 & 4 & 4 & 4 \\ 1 & 5 & 11 & 15 & 16 & 16 \\ 1 & 6 & 16 & 26 & 31 & 32 \\ 1 & 7 & 22 & 42 & 57 & 63 \\ 1 & 8 & 29 & 64 & 99 & 120 \end{pmatrix}$$

En aquest exemple es pot comprovar que el determinant resultant és 1 i el número major no és més gran que  $2^7 = 128$ .

L'objectiu de programar aquestes matrius era veure com podíem arribar a un parell de  $(n,k)$  més grans que amb Vandermonde. El canvi més important del programari en aquest apartat és una nova funció que s'encarrega de generar la nova matriu de Hei-Du-Song en comptes de l'anterior de Vandermonde. La funció `dealer` també s'ha tingut d'adaptar per alguns canvis de computació dels resultats.

`hei_du_song(matrix)` Aquesta funció necessita com a paràmetre una matriu  $n,k$  que pot ser buida i li aplica el procés definit per obtenir una matriu amb les propietats de Hei-Du-Song.

```

1 def hei_du_song(matrix):
2   for p in range(len(matrix)):
3     for q in range(len(matrix[0])):
4       if p==0 or q==0:
5         matrix[p][q]=int(1)
6       else:
7         matrix[p][q]=int(matrix[p-1][q-1]+matrix[p-1][q])
8   return matrix

```

El codi de la llibreria amb la qual es treballa aquest apartat es pot trobar als annexos A.4 i A.5 el codi de la interfície de Hei-Du-Song.

### 3.7 Comparació dels mètodes

Seguidament podem veure alguns exemples en què hi comparem els dos mètodes anteriors per veure quins dels ens aporta més rendiment i amb quina precisió hi podem treballar.

**Exemple 1.** El primer exemple és veure com funcionen els dos amb matrius relativament petites. En aquest cas volem repartir el secret  $S = 123$  en 10 shares amb una  $k = 5$ . És a dir 10 persones de les quals només farà falta 5 per recuperar el secret. (En tots dos programes s'ha tret la funció `round()` que aproxima el secret final a un nombre enter).

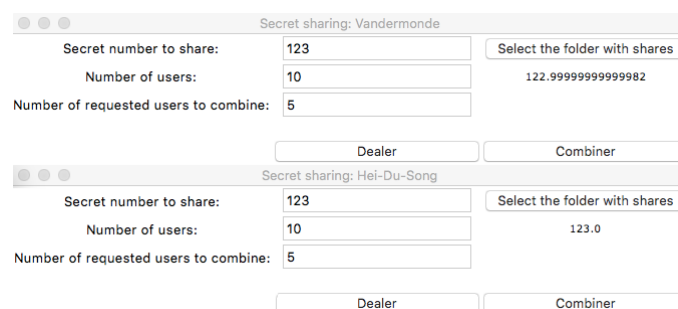


Figura 3.18: Comparació de la precisió dels dos programes.

Com podem observar estem parlant de matrius de  $5 \times 5$  i amb Vandermonde ja no tenim una precisió exacta, en canvi Hei-Du-Song el troba perfectament, sense cap pèrdua d'informació.

**Exemple 2.** El segon exemple és com treballen els programes quan ens trobem al límit establert de  $k = 24$ . Deixant  $S = 123$ , repartint 25 shares i una  $k = 24$ .

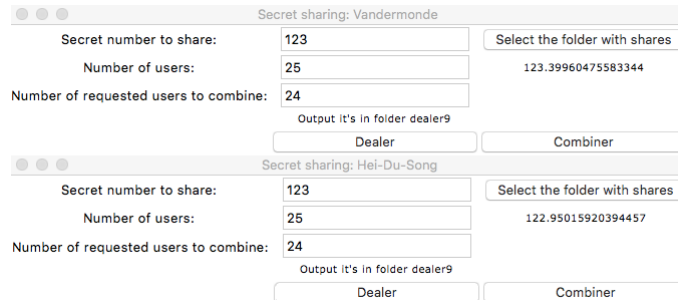


Figura 3.19: Comparació de la precisió dels dos programes, al límit  $k = 24$ .

En cap dels dos s'ha recuperat el secret exacte, Vandermonde l'ha sobrepassat i Hei-Du-Song no hi ha arribat, en els dos casos però la funció `round()` hagués arrodonit al secret correcte. Tot i això es pot veure que Hei-Du-Song ha tingut una desviació més petita que Vandermonde.

**Exemple 3.** Després de veure que tots dos són capaços de donar-nos el secret quan  $k = 24$  és 1 punt més petit que  $n$ , la següent prova és veure que passa quan  $n$  puja de valor.

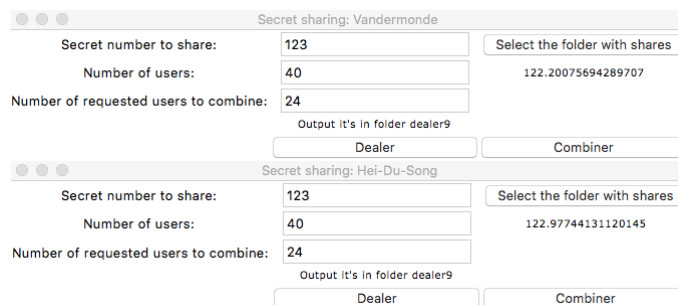


Figura 3.20: Comparació de la precisió dels dos programes, al límit  $k = 24$  i una  $n$  molt superior a  $k$ .

Hem pujat a  $n = 40$  i observem que Vandermonde ja no és capaç de retornar el secret bó ni arrodonint, en canvi Hei-Du-Song no ha variat gaire de l'anterior exemple, això és degut perquè els shares que genera Hei-Du-Song són molt regulars i fins ara hem estat agafant els  $k$  shares primers que generava el

programa. Amb Vandermonde el coeficient B que es tria per cada share en cada repartició és diferent.

**Exemple 4.** En l'exemple anterior hem vist com Hei-Du-Song podia arribar més enllà que Vandermonde, en aquest exemple l'únic que provem és el resultat del programa quan agafem els resultats més alts dels shares repartits per Hei-Du-Song i una combinació diferent de shares de Vandermonde.

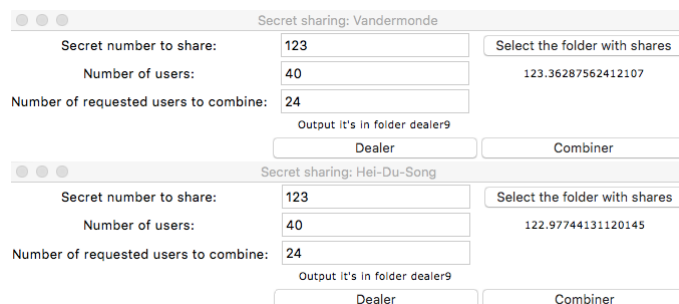


Figura 3.21: Comparació de la precisió dels dos programes, al límit  $k = 24$  i una  $n$  molt superior a  $k$  amb els shares més alts.

En aquest cas veiem que Vandermonde hi ha sapigut retornar el secret correcte, amb un error corregible per la funció `round()`, per l'altra banda Hei-Du-Song ha sigut capaç de calcular el mateix secret que amb els shares de valors baixos i fins i tot es segueix apropant més que Vandermonde.

Com a conclusió d'aquest exemple extreiem que el programa amb Vandermonde, en aquest punt, no el podem agafar com a programa capaç de retornar un secret fiable, tot al contrari que Hei-Du-Song. En tots els casos hem treballat amb un secret relativament petit, de tres xifres, si augmentessim a més xifres els resultats variarien.

De totes maneres arribar a repartir i resoldre matrius d'aquest tamany trobem que ja és una solució suficientment bona, avui dia hi han molts problemes a l'hora de resoldre problemes lineals amb enters i sistemes d'equacions mal condicionats.

### 3.8 Introducció del líder i repartició de poders

En algunes situacions ens podria interessar que per trobar el secret ens hagués de fer falta l'assistència d'una persona a l'hora de fer el combiner, és a dir, en un esquema de compartició de secrets hi pot haver una de les persones a la que se li vol donar més poder.

L'objectiu és generar un pla a l'espai que sigui necessari per determinar un punt comú a tots els plans en l'espai vectorial  $k$ . D'aquesta manera tindriem un usuari a l'esquema que sempre hauria d'estar present en la recuperació del secret.

Si estudiem aquest cas, podria ser un retorn al principi on el secret només el coneixia una persona i quan aquesta moria, aquest es perdia. Llavors tindriem un secret compartit però sense la peça principal del líder tampoc el podríem recuperar.

Això ens porta a estudiar un esquema de repartició de poders. En un esquema on participen  $n = 10$  usuaris, amb un secret recuperable de  $k = 5$  persones mínimes, podem assignar rols i repartir shares d'acord amb aquests rols, per exemple:

- Líder
- Tresorer
- Empleat

Tenint en compte aquests possibles rols podem repartir a l'usuari que farà el paper de líder 4 shares. Aquest com a líder només li farà falta un usuari qualsevol amb mínim 1 share per descobrir el secret.

Un usuari amb el rol de tresorer tindrà 3 shares. Aquest li farà falta més persones per accedir al secret, en total 2 usuaris més que disposin d'1 share cadascun mínim o fins i tot es pot ajuntar amb el líder, entre els dos tindrien 7 shares totals per recuperar el secret.

Els empleats tindran el poder mínim i només obtindran 1 share per usuari, d'aquesta manera s'hauran d'ajuntar 5 usuaris per obtenir el secret revelat o unir-se amb usuaris amb més poder. Esquemes d'aquest estil es poden generar a gust dels interessos del dealer.

## 3.9 Conclusions

Després de treballar en aquesta secció amb el sistema de Blakley implementant Vandermonde i Hei-Du-Song, podem extreure resultats i saber veure que ens aporta més fiabilitat.

Com s'ha comprovat en els exemples, Hei-Du-Song és capaç d'arribar a esquemes de compartició més elevats que Vandermonde. A més la creació de les equacions és molt més senzilla i ens dóna números per treballar molt més còmodes. El càlcul del sistema amb equacions de grau alt és un problema del qual no hi ha solució avui dia, en molts programes es troba aquesta dificultat on a vegades un càlcul que sembla simple no es pot resoldre.

Aquest sistema només ens permet compartir secrets numèrics però a diferència dels secrets visuals ens dóna molta més flexibilitat a l'hora de crear esquemes de compartició, ja que podem compartir el secret fàcilment entre més usuaris i a més donar-hi rols.

Avui dia hi ha pocs sistemes que facin servir la compartició de secrets i pot ser un camp a explorar que pot obrir portes a nous sistemes de comunicació i criptografia. De fet un dels principals objectius d'aquest segle és mantenir tota la informació que es manipula en secret, des de la petita empresa fins al Big Data de les grans empreses.

En els últims anys els esquemes de compartició de secrets també estan participant en la construcció de protocols per mètodes criptogràfics.

# Capítol 4

## Valoració personal

Per seguir evolucionant en la tecnologia es necessita trobar temes nous, conceptes nous, eines útils que serveixin en algun aspecte de la vida.

Quan vaig agafar aquest projecte el que volia era aprendre sobre un tema del qual no se'n parla massa, estudiar-lo i que quan el meu cercle de gent em preguntés que feia el projecte final de grau, els hi pogués explicar una cosa que no haguessin sentit mai.

Estic molt satisfet d'haver escrit aquest projecte i poder haver estudiat un tema tan interessant i entretingut, i sobretot que sigui sobre les matemàtiques. Durant la carrera he estat programant tot tipus de sistemes. Aquest treball no l'he volgut centrar en el programari, el meu objectiu era destacar les matemàtiques amb l'ajuda del software necessari.

Aquest projecte m'ha suposat un repte, ja que he hagut d'aprendre un tema totalment nou per mi i segurament m'he deixat moltes coses per comentar que eren interessants. Animo a qui es llegeixi aquest projecte a continuar-lo, per mi ha sigut una gran aventura.

A part d'haver programat els mètodes matemàtics explicats, aquest projecte també ha estat escrit amb  $\text{\LaTeX}$ , que és una manera d'escriure documents molt relacionat amb la programació. El fet d'haver escrit aquest document tal com està, m'ha suposat pèrdua de temps, ja que he sigut bastant autodidacta, de totes maneres ho tornaria a fer de la manera que ho he fet.

# Índex de figures

1.1	Esquema de compartició de secrets . . . . .	2
2.1	Mapa a trossos . . . . .	5
2.2	Fragment d'una imatge més gran . . . . .	5
2.3	Banderes que tenen estrelles . . . . .	6
2.4	Partició en 3 capes d'una imatge secreta . . . . .	7
2.5	Noves capes aleatòries de la mateixa imatge secreta . . . . .	8
2.6	Capes repartides de manera no aleatòria. . . . .	8
2.7	Imatge original . . . . .	9
2.8	Sumes en mòdul 2 versió digital . . . . .	11
2.9	Sumes digitals en mòdul 2 visuals amb transparències . . . . .	12
2.10	Taula de sumes visuals amb transparències . . . . .	12
2.11	Partició de la imatge en 2 shares . . . . .	15
2.12	Superposició dels dos shares per obtenir el secret. . . . .	16
2.13	Imatge resultant després del processament del combiner. . . . .	16
2.14	Diferència entre l'espai de lletres amples i estretes. . . . .	19
2.15	Imatge resultant de la funció. . . . .	20
2.16	Captura de la interfície . . . . .	22
2.17	Entrar els secret en format text per generar els dos shares. . . . .	22
2.18	Entrar els dos shares per combinar a la interfície. . . . .	23



<i>ÍNDIX DE FIGURES</i>	59
2.19 Introducció de la paraula de pas a l'aplicació. . . . .	24
2.20 Procés de transformar text a imatge . . . . .	24
2.21 Procés de separar el secret en dos shares . . . . .	24
2.22 Introducció dels shares per obtenir la paraula de pas. . . . .	24
2.23 Secret sortint per pantalla. . . . .	25
3.1 Rectes que es tallen en un punt $S$ . . . . .	30
3.2 Rectes que es tallen en un punt $S$ . . . . .	31
3.3 Dos shares interseccionant en el punt $S$ , el secret. . . . .	32
3.4 Dos nous shares que interseccionen en un punt $S$ , el secret. . .	33
3.5 Pla que generat a l'exemple que conté el punt $S = (3, 4, 6)$ . . .	35
3.6 Dos plans paral·lels entre ells. . . . .	35
3.7 Dos plans secants que comparteixen el punt $S$ . . . . .	36
3.8 Quatre plans generats que comparteixen el punt $S$ . . . . .	37
3.9 Els tres plans interseccionen en el secret $S$ . . . . .	38
3.10 Primera vista de la interfície per l'esquema vectorial utilitzant Vandermonde. . . . .	42
3.11 Primera vista de la interfície per l'esquema vectorial utilitzant Vandermonde. . . . .	43
3.12 Carpetes generades pel programa. . . . .	44
3.13 Introducció dels paràmetres de l'exemple. . . . .	44
3.14 Carpeta on estan ubicats els 15 shares. . . . .	45
3.15 Primera vista de la interfície per l'esquema vectorial utilitzant Vandermonde. . . . .	46
3.16 Cercador per seleccionar la carpeta. . . . .	46
3.17 Secret sortint per pantalla. . . . .	47
3.18 Comparació de la precisió dels dos programes. . . . .	52
3.19 Comparació de la precisió dels dos programes, al límit $k = 24$ . .	53

3.20	Comparació de la precisió dels dos programes, al límit $k = 24$ i una $n$ molt superior a $k$ . . . . .	53
3.21	Comparació de la precisió dels dos programes, al límit $k = 24$ i una $n$ molt superior a $k$ amb els shares més alts. . . . .	54

# Fons d'informació

## Bibliografia

Naor, Moni, and Adi Shamir. *Visual cryptography.*' Workshop on the Theory and Application of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1994.

Yang, Ching-Nung. *New visual secret sharing schemes using probabilistic method.* Pattern Recognition Letters 25.4 (2004): 481-494.

Domingo, J., and J. Herrera. *Criptografia: per als serveis telemàtics i el comerç electrònic.* Universitat Oberta de Catalunya, Barcelona (1999).

Arce Muela, María. *Reparto de secretos.* (2016).

Blakley, George Robert. *Safeguarding cryptographic keys.* Proceedings of the national computer conference. Vol. 48. 1979.

Blakley, G. R., and G. A. Kabatianskii. *Linear algebra approach to secret sharing schemes.* Error Control, Cryptology, and Speech Compression. Springer, Berlin, Heidelberg, 1994. 33-40.

Shamir, Adi. *How to share a secret.* Communications of the ACM 22.11 (1979): 612-613.

Ulutas, Mustafa, Güzin Ulutas, and Vasif V. Nabiyev. *Medical image security and EPR hiding using Shamir's secret sharing scheme.* Journal of Systems and Software 84.3 (2011): 341-353.

Lay, David C. *Algebra lineal y sus aplicaciones.* Pearson educación, 2007.

Simmons G.J., *An introduction to shared secret and/or shared control schemes and their applications.*? In Contemporary Cryptology:the Science of Information Integrity, IEEE Press, N.Y., pp.441?497, 1992.

Kikuchi R., Chida K., Ikarashi D., Hamada K., Takahashi K. *Secret Sharing Schemes with Conversion Protocol to Achieve Short Share-Size and Extensibility to Multiparty Computation*. Australasian Conference on Information Security and Privacy. 2013

Hei, Xiali, Xiaojiang Du, and Binheng Song. *Two matrices for Blakley's secret sharing scheme*. ICC. 2012.

## Recursos web

### Geogebra

<https://www.geogebra.org/download?lang=es-ES>

<https://www.geogebra.org/m/m2GbzNqd>

### L<sup>A</sup>T<sub>E</sub>X

<http://www.xmlmath.net/texmaker/download.html>

<http://matematicas.uclm.es/earanda/wp-content/uploads/downloads/2013/10/latex.pdf>

### TkInter

<https://www.lawebdelprogramador.com/cursos/Python/7494-Manual-de-Tkinter-para-Python.html>

### Gestió d'imatges

<http://www.numpy.org>

<https://pillow.readthedocs.io/en/3.1.x/reference/Image.html>

# Annexos

## A.1 Llibreria per treballar amb secrets visuals

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 import matplotlib.pyplot as plt
4 import numpy as np
5 from random import choice
6 import copy
7 from PIL import Image
8 from itertools import izip
9 from os import listdir
10 import os
11
12 MIDA=50
13
14 def search_letter(letter):
15     file=letter+".png"
16     f = plt.imread(file)
17
18     return f
19     #plt.imshow(f[:, :, 0], vmin=0, vmax=1, interpolation='
20     ↪ nearest')
21     #plt.show()
22
23 def search_image(image):
24     llista=['.PNG', '#.PNG', '$.PNG', '&.PNG', '-.PNG', '0.
25     ↪ PNG', '1.PNG', '2.PNG', '3.PNG', '4.PNG', '5.PNG
26     ↪ ', '6.PNG', '7.PNG', '8.PNG', '9.PNG', '@.PNG', '
27     ↪ A.PNG', 'B.PNG', 'C.PNG', 'D.PNG', 'E.PNG', 'F.
28     ↪ PNG', 'G.PNG', 'H.PNG', 'I.PNG', 'J.PNG', 'K.PNG
29     ↪ ', 'L.PNG', 'M.PNG', 'N.PNG', 'O.PNG', 'P.PNG', '
30     ↪ Q.PNG', 'R.PNG', 'S.PNG', 'T.PNG', 'U.PNG', 'V.
31     ↪ PNG', 'W.PNG', 'X.PNG', 'Y.PNG', 'Z.PNG', '_.PNG
32     ↪ ', '?PNG']
33
34     i1 = Image.open(image)
35     dif=10000000.0
36     diferencias=[]
37     i_minor=0
38     i=-1
39     while dif>90000.0 and i!=len(llista)-1:
```

```
30         i+=1
31         i2 = Image.open(llista[i])
32
33         pairs = izip(i1.getdata(), i2.getdata())
34         if len(i1.getbands()) == 1:
35             # for gray-scale jpegs
36             dif = sum(abs(p1-p2) for p1,p2 in pairs)
37         else:
38             dif = sum(abs(c1-c2) for p1,p2 in pairs for
39                 ↪ c1,c2 in zip(p1,p2))
40         differences+=[(i,dif)]
41         print i,dif,str(llista[i][:4])
42         ncomponents = i1.size[0] * i1.size[1] * 3
43         #print "Difference (percentage):", (dif / 255.0
44         ↪ * 100) / ncomponents
45     return str(llista[i][:4])
46
47 def str2img(paraula):
48     llista=[]
49     for letter in paraula:
50         llista+=[letter+".png"]
51     return join_images(llista)
52
53 def join_images(imatges):
54     name_file="test.png"
55     imatges_aux=[]
56     for im in imatges:
57         imatges_aux+=[im]
58     images = map(Image.open, imatges_aux)
59     widths, heights = zip(*(i.size for i in images))
60
61     total_width = sum(widths)
62     max_height = max(heights)
63
64     new_im = Image.new('RGB', (total_width, max_height))
65
66     x_offset = 0
67     for im in images:
68         new_im.paste(im, (x_offset,0))
69         x_offset += im.size[0]
```

```
68     #new_im=new_im.resize((int(new_im.size[0]*0.1),int(
69         ↪ new_im.size[1]*0.1))
70     new_im.save(name_file)
71     imatge = plt.imread(name_file)
72     return imatge
73 def find_crop():
74     fitxers=[]
75     for file in listdir("."):
76         if "cropped" in file:
77             fitxers+= [int(file[7:-4])]
78
79     a=sorted(fitxers,key=int)
80     b=[]
81     for element in a:
82         b+= ['cropped'+str(element)+'.PNG']
83     return b
84
85 def img2str(img1):
86     image=Image.open(img1)
87     size=image.size
88     llarg=size[0]/size[1]
89     lletres=[]
90     i=0
91     while i<llarg:
92         lletres+= [image.crop((0,0,size[1],size[1]))]
93         image2=image.crop((0,0,size[1],size[1]))
94         image=image.crop((size[1],0,size[0],size[1]))
95         image2.save("cropped"+str(i)+".png", "PNG")
96         i+=1
97
98     paraula=""
99     for fitxer in find_crop():
100         print search_image(fitxer)
101         paraula+=search_image(fitxer)
102         os.remove(fitxer)
103     return paraula
104
105
106 def combinator(img1,img2):
107     img1 = Image.open(img1)
```



```
108     img2 = Image.open(img2)
109     size=img1.size
110     imatge_aux = Image.new('RGBA', size, "white")
111     imatge_aux.save('image.png')
112     img = Image.open("image.png")
113     datas1 = img1.getdata()
114     datas2 = img2.getdata()
115     newData=[]
116     i=-1
117     j=-1
118     k=-1
119     for item in datas1:
120         i+=1
121         if item == datas2[i]:
122             newData.append((255, 255, 255, 255))
123
124         else:
125             #print item
126             datas2[i]
127             newData.append((0, 0, 0, 255))
128     i=-1
129     img.putdata(newData)
130     img.save("image.png", "PNG")#converted Image name
131
132
133 def dealer(img1):
134     plt.rcParams['image.cmap']='gray'
135     img2=copy.copy(img1)
136     i=-1
137     j=-1
138     k=-1
139     for tupla in img1:
140         i+=1
141         for llista in tupla:
142             j+=1
143             for number in llista:
144                 k+=1
145                 if number < 0.8:
146                     #print number
147                     num=choice([1,0])
148                     img1[i][j][k]=num
```

```

149         img2[i][j][k]=1-num
150         #print image[i][j][k]
151         elif number >= 0.8:
152             num=choice([1,0])
153             img1[i][j][k]=num
154             img2[i][j][k]=num
155             k=-1
156             j=-1
157     plt.imsave('share1.png',img1[:,:,:])
158     plt.imsave('share2.png',img2[:,:,:])
159     convertToPNG("share1.png","share1.png")
160     convertToPNG("share2.png","share2.png")
161
162 def dealer_nonogram(img1):
163     plt.rcParams['image.cmap']='gray'
164     img2=copy.copy(img1)
165     img3=copy.copy(img1)
166     i=-1
167     j=-1
168     k=-1
169     for tupla in img1:
170         i+=1
171         for llista in tupla:
172             j+=1
173             for number in llista:
174                 k+=1
175                 if number < 0.8:
176                     #print number
177                     num=choice([1,0,2])
178                     random=[1,1,1]
179                     random[num]=0
180                     img1[i][j][k]=random[0]
181                     img2[i][j][k]=random[1]
182                     img3[i][j][k]=random[2]
183                     #print image[i][j][k]
184                 elif number >= 0.8:
185                     img1[i][j][k]=1
186                     img2[i][j][k]=1
187                     img3[i][j][k]=1
188                 k=-1
189             j=-1

```

```
190
191     plt.imshow('share1.png',img1[:,:,:])
192     plt.imshow('share2.png',img2[:,:,:])
193     plt.imshow('share3.png',img3[:,:,:])
194     convertToPNG_nonogram("share1.png","share1.png")
195     convertToPNG_nonogram("share2.png","share2.png")
196     convertToPNG_nonogram("share3.png","share3.png")
197
198 def dealer_nonogram_pattern(img1):
199     plt.rcParams['image.cmap']='gray'
200     img=copy.copy(img1)
201     img2=copy.copy(img1)
202     img3=copy.copy(img1)
203     i=-1
204     j=-1
205     k=-1
206     num=0
207     for tupla in img1:
208         i+=1
209         for llista in tupla:
210             j+=1
211             for number in llista:
212                 k+=1
213                 if number < 0.8:
214                     if num==2:
215                         num=1
216                     else:
217                         num+=1
218                     pattern=[1,1,1]
219                     pattern[num]=0
220                     img[i][j][k]=pattern[0]
221                     img2[i][j][k]=pattern[1]
222                     img3[i][j][k]=pattern[2]
223                 elif number >= 0.8:
224                     #print number
225                     img[i][j][k]=1
226                     img2[i][j][k]=1
227                     img3[i][j][k]=1
228             k=-1
229         j=-1
230     plt.imshow('share1.png',img[:,:,:])
```

```
231 plt.imshow(img2[:,:,:])
232 plt.imshow(img3[:,:,:])
233 plt.imshow(convertToPNG_nonogram("share1.png","share1.png"))
234 plt.imshow(convertToPNG_nonogram("share2.png","share2.png"))
235 plt.imshow(convertToPNG_nonogram("share3.png","share3.png"))
236
237 def convertToPNG_nonogram(img1,name):
238     img = Image.open(img1)#image path and name
239     img=img.resize((int(img.size[0]*30),int(img.size[1]*30)))
240     datas = img.getdata()
241     newData = []
242     for item in datas:
243         if item[0] == 255 and item[1] == 255 and item[2] ==
           ↪ 255:
244             newData.append((0, 0, 0, 0))
245         else:
246             newData.append(item)
247     img.putdata(newData)
248     img.save(name, "PNG")#converted Image name
249
250 def convertToPNG(img1,name):
251     img = Image.open(img1)#image path and name
252     #img=img.resize((int(img.size[0]*30),int(img.size[1]*30)))
253     datas = img.getdata()
254     newData = []
255     for item in datas:
256         if item[0] == 255 and item[1] == 255 and item[2] ==
           ↪ 255:
257             newData.append((0, 0, 0, 0))
258         else:
259             newData.append(item)
260     img.putdata(newData)
261     img.save(name, "PNG")#converted Image name
```

## A.2 Codi font de la interfície de secrets visuals

```
1  -*- coding:utf-8 -*-
2  from Tkinter import *
3  from libshare import *
4
5
6  #functions
7  def print_secret():
8      print secret.get()
9
10 def show():
11     secret_trobat.get()
12
13 def share():
14     secret_img=str2img(secret.get())
15     dealer(secret_img)
16     print "done"
17
18 def combine():
19     sh1=str(url1.get())
20     sh2=str(url2.get())
21     combinator(sh1,sh2)
22     secret_trobat.set(img2str("image.png"))
23     ventana.update_idletasks()
24
25 #Tk
26 ventana = Tk()
27 ventana.title('Secret sharing')
28
29 #vars
30 secret = StringVar()
31 url1 = StringVar()
32 url2 = StringVar()
33 proces = IntVar()
34 proces.set(1)
35 secret_trobat = StringVar()
36
37
38
```

```
39 #widget generation
40 #secret to share
41 etiqueta_secret = Label(ventana,
42     text='Text message to share:')
43 entrada_secret = Entry(ventana,
44     textvariable=secret)
45 etiqueta_secret.grid(row=2, column=1)
46 entrada_secret.grid(row=2, column=2)
47
48 etiqueta_secret_entrat = Label(ventana,
49     text="Output will be share1.png and share2.png",font=("
    ↪ Verdana",10))
50 etiqueta_secret_entrat.grid(row=5, column=2)
51
52 #images to combine
53 etiqueta_combine = Label(ventana,
54     text='Share 1 (PNG): ')
55 entrada_combine = Entry(ventana,
56     textvariable=url1)
57 etiqueta_combine.grid(row=2, column=3)
58 entrada_combine.grid(row=2, column=4)
59
60 etiqueta_combine = Label(ventana,
61     text='Share 2 (PNG): ')
62 entrada_combine = Entry(ventana,
63     textvariable=url2)
64 etiqueta_combine.grid(row=3, column=3)
65 entrada_combine.grid(row=3, column=4)
66
67 etiqueta_secret_trobat = Label(ventana,
68     textvariable=secret_trobat, font=("Verdana",10))
69 etiqueta_secret_trobat.grid(row=5, column=4)
70
71 #button
72 boton1 = Button(ventana,
73     text='Share',
74     command=share, width=20)
75 boton1.grid(row=6, column=2)
76
77 boton2 = Button(ventana,
78     text='Combine',
```

```
79         command=combine, width=20)
80 boton2.grid(row=6, column=4)
81 boton2.configure(command=combine)
82
83
84 #window
85 ventana.mainloop()
```

### A.3 Llibreria per treballar amb Vandermonde

```
1 import numpy as np
2 import random
3 import os
4
5 def create_matrix(numero_filas,numero_columnas):
6     matrix = [1] * numero_filas
7     for i in range(numero_filas):
8         matrix[i] = [1] * numero_columnas
9     return matrix
10
11 def first_ones(matrix):
12     f=0
13     c=0
14     files=np.size(matrix,0)
15     columnes=np.size(matrix,1)
16
17     while f<files:
18         matrix[f,0]=1
19         f+=1
20     while c<columnes:
21         matrix[0,c]=1
22         c+=1
23
24 def coeficient_D(llista,punt):
25     i=0
26     result=0
27     while i<len(llista):
28         result+=llista[i]*punt[i]
29         i+=1
30     llista.append(result)
31     return llista
32
33
34 def vandermonde(share,B):
35     i=1
36     lenght=len(share)
37     while i<lenght:
38         share[i]=B**i
```



```
39         i+=1
40     return share
41
42 def determinant(matriu):
43     return np.linalg.det(matriu)
44
45 def get_D_values(matriu):
46     d_values=[]
47     for llista in matriu:
48         d_values.append(llista[-1])
49     return d_values
50
51 def get_coeficients(matriu):
52     coef=[]
53     for llista in matriu:
54         coef.append(llista[:-1])
55     return np.array(coef)
56
57 def get_Bs(n,llista=[]):
58     i=0
59     while i<n:
60         B=int(random.randint(2,300))
61         if B not in llista:
62             llista.append(B)
63             i+=1
64     return llista
65
66 def dealer(n,k,secret):
67     s=make_secret(k,secret)
68     matriu=create_matrix(n,k)
69     i=0
70     B=get_Bs(n, [])
71     for llista in matriu:
72         matriu[i]=vandermonde(llista,B[i])
73         matriu[i]=coeficient_D(matriu[i],s)
74         i+=1
75     matrix=np.array(matriu)
76     share2file(matriu)
77     return matrix
78
79 def combiner(matriu):
```

```
80     coef=get_coeficients(matriu)
81     d_values=get_D_values(matriu)
82     try:
83         solution=np.linalg.solve(coef,d_values)
84         return solution[-1]
85     except:
86         print "It can't be solved"
87
88 def make_secret(k,s):
89     i=0
90     secret=[]
91     while i<k:
92         secret.append(random.randint(2,300))
93         i+=1
94     secret[-1]=s
95     print secret
96     return secret
97
98 def share2file(matrix):
99     i=1
100    j=1
101    while os.path.exists("dealer"+str(j)):
102        j+=1
103    os.mkdir("dealer"+str(j))
104    for element in matrix:
105        f=open("dealer"+str(j)+"/"+str(i)+".txt","w")
106        f.write(str(element))
107        f.close()
108        i+=1
109
110 def llista2eq(llista):
111     llista_nova=[]
112     i=0
113     abc="xyzabcdefghijklmnopqrstuv"
114     while i<len(llista):
115         llista_nova.append(str(llista[i])+abc[i])
116         i+=1
117     return str(llista_nova)
118
119 def file2matrix(directory):
120     fitxers=os.listdir(directory)
```

```
121     llista=[]
122     llista_aux=[]
123     aux=[]
124     for file in fitxers:
125         if ".txt" in file:
126             L_out(directory+"/"+file)
127             f=open(directory+"/"+file)
128             aux=f.read().split(',')
129             for element in aux:
130                 if '[' in element:
131                     llista.append(int(element
132                                     ↪ [1:]))
133                 elif ']' in element:
134                     llista.append(int(element
135                                     ↪ [1:-1]))
136                     llista_aux.append(llista)
137                     llista=[]
138                 elif element[1:]==' ':
139                     pass
140                 else:
141                     llista.append(int(element
142                                     ↪ [1:]))
143             f.close()
144     return np.array(llista_aux)
145
146 def L_out(a):
147     f=open(a,"r")
148     share=f.read()
149     f.close()
150     f=open(a,"w")
151     share=share.replace("L","")
152     f.write(share)
153     f.close()
```

## A.4 Codi font de la interfície de Vandermonde

```
1 import numpy as np
2 import random
3 import os
4 from libmatrix import *
5
6
7 def create_matrix(numero_filas,numero_columnas):
8     matrix = [1] * numero_filas
9     for i in range(numero_filas):
10         matrix[i] = [1] * numero_columnas
11     return matrix
12
13 def first_ones(matrix):
14     f=0
15     c=0
16     files=np.size(matrix,0)
17     columnes=np.size(matrix,1)
18
19     while f<files:
20         matrix[f,0]=1
21         f+=1
22     while c<columnes:
23         matrix[0,c]=1
24         c+=1
25
26 def coeficient_D(llista,punt):
27     i=0
28     result=0
29     while i<len(llista):
30         #print llista[i],punt[i]
31         result+=int(llista[i]*punt[i])
32         i+=1
33     llista.append(result)
34     return llista
35
36 def hei_song(matrix):
37     for p in range(len(matrix)):
38         for q in range(len(matrix[0])):
```

```
39         if p==0 or q==0:
40             matrix[p][q]=int(1)
41         else:
42             matrix[p][q]=int(matrix[p-1][q-1]+
43                             ↪ matrix[p-1][q])
44
45     return matrix
46
47 def get_D_values(matriu):
48     d_values=[]
49     for llista in matriu:
50         d_values.append(llista[-1])
51     return d_values
52
53 def get_coeficients(matriu):
54     coef=[]
55     for llista in matriu:
56         coef.append(llista[:-1])
57     return coef
58
59 def dealer(n,k,secret):
60     s=make_secret(k,secret)
61     matriu=create_matrix(n,k)
62     i=0
63     matriu=hei_song(matriu)
64     for i in range(len(matriu)):
65         matriu[i]=coeficient_D(matriu[i],s)
66         i+=1
67     share2file(matriu)
68     return np.array(matriu)
69
70 def combiner(matriu):
71     coef=get_coeficients(matriu)
72     d_values=get_D_values(matriu)
73     print np.linalg.det(coef)
74     #print coef
75     #print d_values
76     try:
77         solution=np.linalg.solve(coef,d_values)
78         return solution[-1]
79     except:
80         print "It can't be solved"
```

```
79
80
81 def make_secret(k,s):
82     i=0
83     secret=[]
84     while i<k:
85         secret.append(random.randint(-30,30))
86         i+=1
87     secret[-1]=s
88     return secret
89
90
91 def share2file(matrix):
92     i=1
93     j=1
94     while os.path.exists("dealer"+str(j)):
95         j+=1
96     os.mkdir("dealer"+str(j))
97     for element in matrix:
98         f=open("dealer"+str(j)+"/"+str(i)+".txt","w")
99         f.write(str(element))
100        f.close()
101        i+=1
102
103 def llista2eq(llista):
104     llista_nova=[]
105     i=0
106     abc="xyzabcdefghijklmnopqrstuv"
107     while i<len(llista):
108         llista_nova.append(str(llista[i])+abc[i])
109         i+=1
110     return str(llista_nova)
111
112 def file2matrix(directory):
113     fitxers=os.listdir(directory)
114     llista=[]
115     llista_aux=[]
116     aux=[]
117     for file in fitxers:
118         if ".txt" in file:
119             f=open(directory+"/"+file)
```

```
120     aux=f.read().split(',')
121     for element in aux:
122         if '[' in element:
123             llista.append(int(element
124                             ↪ [1:]))
125         elif ']' in element:
126             llista.append(int(element
127                             ↪ [1:-1]))
128             llista_aux.append(llista)
129             llista=[]
130         elif element[1:]==' ':
131             pass
132         else:
133             llista.append(int(element
134                             ↪ [1:]))
135     f.close()
136     return np.array(llista_aux)
```

## A.5 Llibreria per treballar amb Hei-Du-Song

```
1 import numpy as np
2 import random
3 import os
4 from libmatrix import *
5
6 def create_matrix(numero_filas,numero_columnas):
7     matrix = [1] * numero_filas
8     for i in range(numero_filas):
9         matrix[i] = [1] * numero_columnas
10    return matrix
11
12 def first_ones(matrix):
13     f=0
14     c=0
15     files=np.size(matrix,0)
16     columnes=np.size(matrix,1)
17
18     while f<files:
19         matrix[f,0]=1
20         f+=1
21     while c<columnes:
22         matrix[0,c]=1
23         c+=1
24     #print matrix
25
26 def coeficient_D(llista,punt):
27     i=0
28     result=0
29     while i<len(llista):
30         #print llista[i],punt[i]
31         result+=int(llista[i]*punt[i])
32         i+=1
33     llista.append(result)
34     return llista
35
36 def hei_du_song(matrix):
37     for p in range(len(matrix)):
38         for q in range(len(matrix[0])):
```



```
39         if p==0 or q==0:
40             matrix[p][q]=int(1)
41         else:
42             matrix[p][q]=int(matrix[p-1][q-1]+
43                             ↪ matrix[p-1][q])
44
45     return matrix
46
47 def get_D_values(matriu):
48     d_values=[]
49     for llista in matriu:
50         d_values.append(llista[-1])
51     return d_values
52
53 def get_coeficients(matriu):
54     coef=[]
55     for llista in matriu:
56         coef.append(llista[:-1])
57     return coef
58
59 def dealer(n,k,secret):
60     s=make_secret(k,secret)
61     matriu=create_matrix(n,k)
62     i=0
63     matriu=hei_du_song(matriu)
64     for i in range(len(matriu)):
65         matriu[i]=coeficient_D(matriu[i],s)
66         i+=1
67     share2file(matriu)
68     return np.array(matriu)
69
70 def combiner(matriu):
71     coef=get_coeficients(matriu)
72     d_values=get_D_values(matriu)
73     print np.linalg.det(coef)
74     #print coef
75     #print d_values
76     try:
77         solution=np.linalg.solve(coef,d_values)
78         print solution
79         return solution[-1]
80     except:
```

```
79         print "It can't be solved"
80
81
82 def make_secret(k,s):
83     i=0
84     secret=[]
85     while i<k:
86         secret.append(random.randint(-30,30))
87         i+=1
88     secret[-1]=s
89     #print secret
90     return secret
91
92
93 def share2file(matrix):
94     i=1
95     j=1
96     while os.path.exists("dealer"+str(j)):
97         j+=1
98     os.mkdir("dealer"+str(j))
99     for element in matrix:
100         f=open("dealer"+str(j)+"/"+str(i)+".txt","w")
101         f.write(str(element))
102         #f.write(llista2eq(element))
103         f.close()
104         i+=1
105
106 def llista2eq(llista):
107     llista_nova=[]
108     i=0
109     abc="abcdefghijklmnopqrstuv"
110     while i<len(llista):
111         llista_nova.append(str(llista[i])+abc[i])
112         #print llista[i]
113         i+=1
114     return str(llista_nova)
115
116 def file2matrix(directory):
117     fitxers=os.listdir(directory)
118     llista=[]
119     llista_aux=[]
```

```
120     aux=[]
121     for file in fitxers:
122         if ".txt" in file:
123             f=open(directory+"/"+file)
124             aux=f.read().split(',')
125             #print aux
126             for element in aux:
127                 #print element
128                 if '[' in element:
129                     #print "hey"
130                     #print element[1:]
131                     llista.append(int(element
132                                     ↪ [1:]))
133                     #print int(element[1:]),"
134                     ↪ inici"
135                 elif ']' in element:
136                     llista.append(int(element
137                                     ↪ [1:-1]))
138                     llista_aux.append(llista)
139                     llista=[]
140                 elif element[1:]==' ':
141                     pass
142                     #print "pass"
143                     #print int(element[1:-1]),"
144                     ↪ final"
145                 else:
146                     llista.append(int(element
147                                     ↪ [1:]))
148                     #print int(element[1:]),"
149                     ↪ else"
150             #aux=np.array(aux)
151             f.close()
152     return np.array(llista_aux)
```

## A.6 Codi font de la interfície de Hei-Du-Song

```
1 #-*- coding:utf-8 -*-
2 from Tkinter import *
3 from heisong import *
4 import Tkinter, tkFileDialog, Tkconstants
5
6
7
8
9 #functions
10 def print_secret():
11     print secret.get()
12
13 def show():
14     secret_trobat.get()
15
16 def share():
17     a=dealer(n.get(),k.get(),secret.get())
18     #print a
19     folder_save.set("Output it's in folder "+
20         ↪ get_last_folder())
21     #print folder_save.get()
22     ventana.update_idletasks()
23
24 def combine():
25     #print dirname.get()
26     matriu=file2matrix(dirname.get())
27     secret_trobat.set(combiner(matriu))
28     ventana.update_idletasks()
29
30 def get_last_folder():
31     folders=os.listdir(".")
32     llista=[]
33     try:
34         for element in folders:
35             if "dealer" in element:
36                 llista.append(element)
37     llista=sorted(llista)
38     #print llista
```

```
38         return "dealer"+str(int(llista[-1][-1]))
39     except:
40         return "dealer1"
41
42 def openDirectory():
43     directory = tkFileDialog.askdirectory(parent=ventana,
44         ↪ initialdir='/home/Users/juanrivas/Desktop', title
45         ↪ ='Select your pictures folder')
46     dirname.set(directory)
47     #print dirname.get()
48
49
50 #Tk
51 ventana = Tk()
52 ventana.title('Secret sharing: Hei-Du-Song')
53
54 #vars
55 dirname= StringVar()
56 secret = IntVar()
57 n = IntVar()
58 k = IntVar()
59 url1 = StringVar()
60 proces = IntVar()
61 proces.set(1)
62 secret_trobat = IntVar()
63 folder_save = StringVar()
64 folder_save.set("")
65 ventana.update_idletasks()
66 button_opt = {'fill': Tkconstants.BOTH, 'padx': 5, 'pady': 5}
67
68 etiqueta_secret = Label(ventana,
69     text='Secret number to share:')
70 entrada_secret = Entry(ventana,
71     textvariable=secret)
72 etiqueta_secret.grid(row=2, column=1)
73 entrada_secret.grid(row=2, column=2)
74
75 etiqueta_users = Label(ventana,
76     text='Number of users:')
```

```
77 entrada_users = Entry(ventana,
78     textvariable=n)
79 etiqueta_users.grid(row=3, column=1)
80 entrada_users.grid(row=3, column=2)
81
82 etiqueta_keys = Label(ventana,
83     text='Number of requested users to combine:')
84 entrada_keys = Entry(ventana,
85     textvariable=k)
86 etiqueta_keys.grid(row=4, column=1)
87 entrada_keys.grid(row=4, column=2)
88
89 etiqueta_secret_entrat = Label(ventana,
90     textvariable=folder_save,font=("Verdana",10))
91 etiqueta_secret_entrat.grid(row=5, column=2)
92
93
94 boton0 = Button(ventana,
95     text = 'Select the folder with shares',
96     fg = 'black',
97     command= openDirectory)
98 boton0.grid(row=2,column=4)
99 #images to combine
100
101 etiqueta_secret_trobat = Label(ventana,
102     textvariable=secret_trobat, font=("Verdana",10))
103 etiqueta_secret_trobat.grid(row=3, column=4)
104
105 #button
106 boton1 = Button(ventana,
107     text='Dealer',
108     command=share, width=20)
109 boton1.grid(row=6, column=2)
110
111 boton2 = Button(ventana,
112     text='Combiner',
113     command=combine, width=20)
114 boton2.grid(row=6, column=4)
115 boton2.configure(command=combine)
116
117
```

118	#window
119	ventana.mainloop()