Protecció del copyright

Daniel Mulero Roig
ETIS

Antoni Martínez Ballesté

18/06/04

1. Resum

La protecció del copyright mitjançant un sistema de marca d'aigua, és un dels temes a solucionar el mon actual a on la comunicació i l'intercanvi de material a través de la xarxa està a l'abast de tothom. La tasca a realitzar és fer un estudi dels mètodes possibles per a poder introduir dins una imatge en format BMP un codi que de manera unívoca, permeti assegurar que la propietat d'aquella imatge és nostra. Aquesta marca ha de poder suportar atacs que de forma normal succeirien en la seva vida útil com poden ser la compressió, l'escalat i d'altres, de tal manera que puguem aconseguir algun mètode que malgrat tot, permeti la recuperació de la marca. Fins aquí podem pensar que els atacs poden haver estat amb la finalitat d'aconseguir la imatge per a un ús particular, però alguns usuaris poden voler la imatge per a poder fer-la servir com a pròpia utilitzant copiés amb diferents marques per a poder fer confabulacions que generin imatges amb una marca que podrà fer servir com a pròpia per part de l'atacant. Això suposa que s'ha d'incorporar al mètode de marcat codis que ens protegeixin d'això i pels quals és proposa la utilització de codis duals de Hamming

2. Índex

3. Introducció al treball fi de carrera.

3.1. Justificació.

La justificació del projecte és fer un estudi de les possibilitats que existeixen d'incloure dins d'una imatge determinada informació que permeti assegurar que aquella imatge ha estat obtinguda de forma correcta.

3.2. Objectius.

L'objectiu del projecte és aconseguir la protecció del copyright en imatges informàtiques del tipus bitmap. Aquesta protecció s'aconseguirà mitjançant la incorporació d'una marca d'aigua dins de la imatge de manera que és pugui assegurar de manera fefaent que aquesta imatge és d'un únic i determinat comprador. La robustesa ha d'assegurar que si dos compradors amb la mateixa imatge és confabulen per a aconseguir una tercera que hagi estat estreta de les dues primeres, el venedor pugui arribar a determinar, com a mínim qui ha estat un dels dos compradors. I un objectiu no menys important és mostrar que s'han assolit els objectius d'aprenentatge en l'àmbit general de la Enginyeria Tècnica Informàtica.

3.3. Mètode seguit.

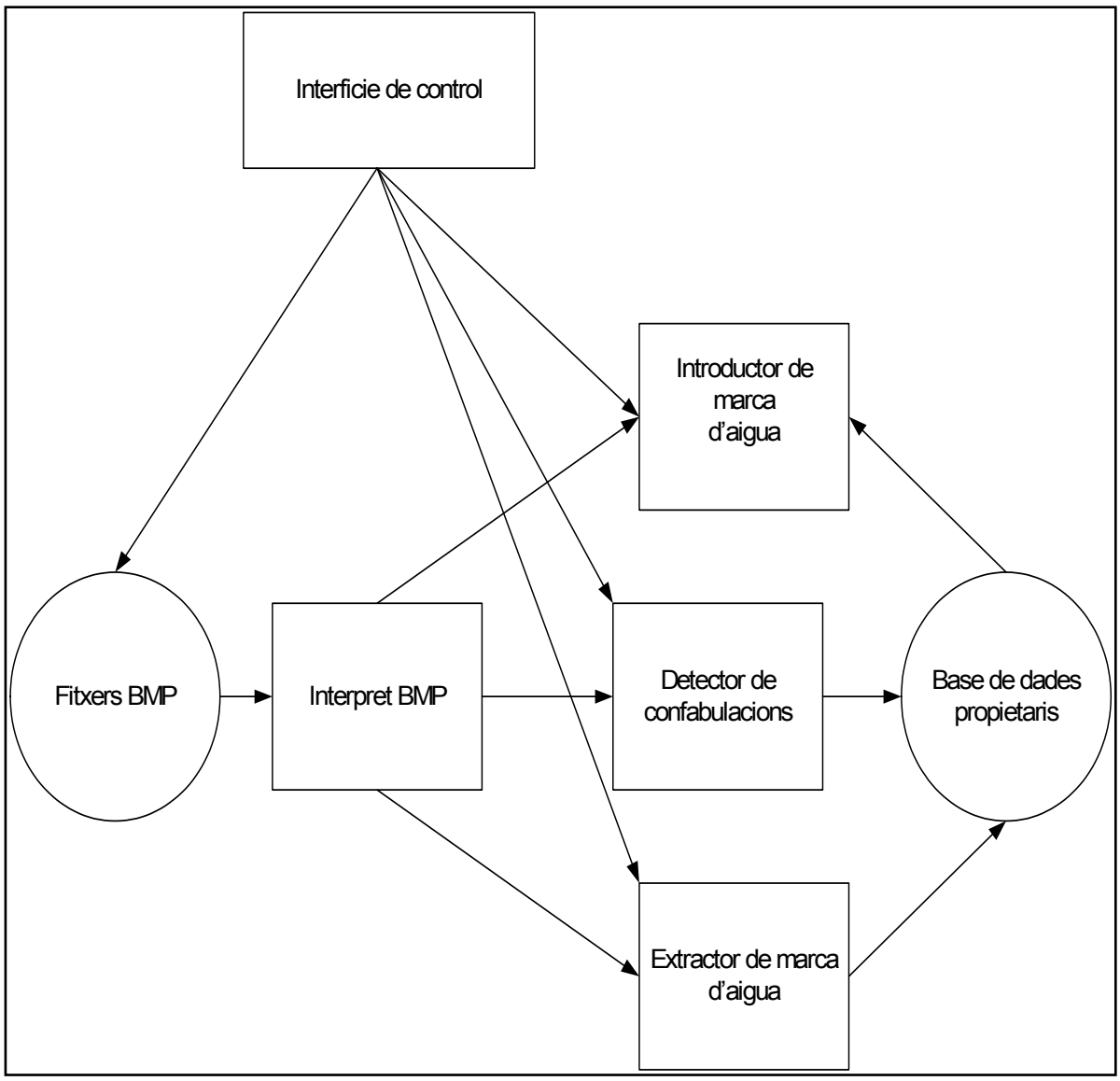
El diagrama de blocs a molt alt nivell és pot veure en la pàgina següent. Podem dividir el programa en quatre
parts, la interfície, els mètodes de gestió de les imatges, la base de dades i la gestió de disc.

La gestió de disc és la que s'encarrega de guardar i recuperar la informació en el sistema. Encara que pugui semblar trivial perquè normalment treballem de manera fàcil, no ho és degut a que caldrà integrar-la amb la interfície de control de forma clara i entenedora per no crear equívocs. Com podem veure en el sistema la definim per unes rodones en les quals marquem que treballem amb els discos tant amb els fitxers bitmap com amb les bases de dades.

La base de dades és possiblement la part més important del sistema. En ella guardarem tota la informació que hem incorporat com a marques d'aigua en els fitxers bitmap. Serà tota aquesta informació la que arribat un moment donat ens permetrà saber si el fitxer que se'ns demana autenticar porta la nostra marca d'aigua i si el que diu que és el comprador l'ha aconseguit comprant-la a nosaltres o be l'ha aconseguit de tercers que s'han confabulat per generar-la. També hem d'assegurar la integritat de les dades i que atenent a la normativa actual sobre la seguretat de les dades, aquest fitxer sigui inaccessible per tercers si no és per autorització expressa del comprador.

Els mètodes per a la gestió de les imatges son el pal de paller del sistema, es a dir sense ells aquest disseny no te sentit. Esta dividit en tres parts, encara que de ben segur tindran una sèrie de parts comunes que faran més fàcil la seva codificació. Esta dividit en tres parts, la part a on pròpiament introduïm la marca d'aigua, la part a on l'extraiem i la part on detectem les possibles confabulacions. Podem assegurar sense por a errar gaire que tindran una part comuna com pot ésser l'intèrpret BMP i una altre que no esta dibuixat però que porten incorporat que serà el codificador i descodificador Hamming dual.

Finalment només queda per veure la interfície, que serà la relació de tot l'anterior o part interior del projecte amb el mon exterior, és a dir la relació home – màquina que intentarem que sigui el més semblant possible al que tothom esta acostumat a utilitzar per a aconseguir un aprenentatge extremadament ràpid i intuïtiu.



3.4. Planificació

Aquesta és la tasca més important de totes i és la que determinarà si estem portant be el desenvolupament o no de tot el TFC.
La divisió en tasques pot ésser la següent,

Tasca	Descripció	Precedent
1.1	Cerca d'informació sobre el format BMP	-
1.2	Cerca d'informació sobre protecció del copyright amb marques d'aigua	-
1.3	Cerca d'informació sobre codis Hamming duals i contra confabulacions	-
2.1	Dissenyar un mètode de prova de marca d'aigua	1.1/1.2/1.3
2.2	Dissenyar l'algorisme d'introducció de marca d'aigua	2.1
2.3	Dissenyar l'algorisme que permetrà llegir la marca i obtenir un comprador	2.2
2.4	Dissenyar un programa que donades dues imatges, generi una confabulada	2.3
2.5	Dissenyar una base de dades per gestionar els compradors	1
3.1	Implementació del mètode de marca d'aigua .	2.4/2.5

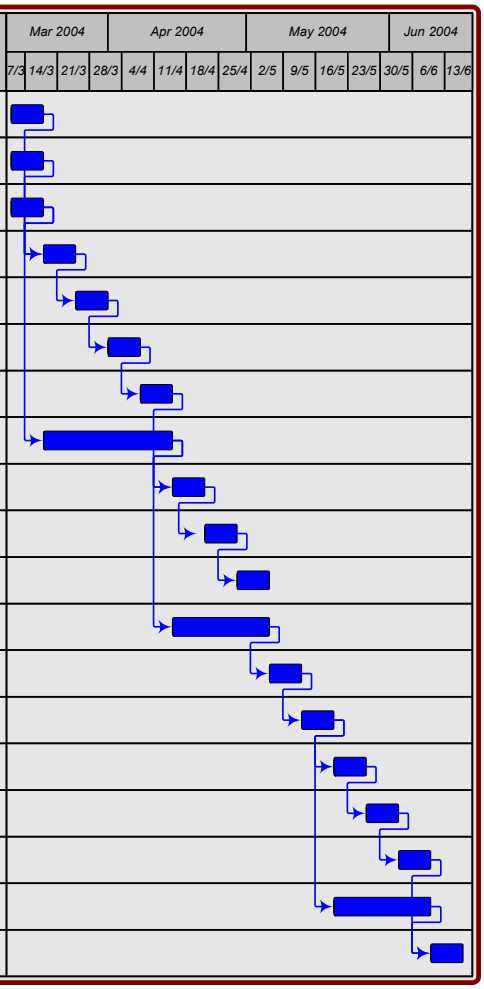
3.2	Implementació del programa d'extracció de marca	3.1
3.3	Implementació del programa de confabulacions	3.2
3.4	Disseny del joc de proves	2.4/2.5
4.1	Creació del joc de proves de la robustesa	3.3/3.4
4.2	Creació del joc de proves per provar la resistència a confabulacions	4.1
5.1	Elaboració de la memòria: estat de l'art	4.2
5.2	Elaboració de la memòria: disseny del programa algorismes	5.1
5.3	Elaboració de la memòria: manual d'usuari	5.2
5.4	Presentació amb PowerPoint	4.2
5.5	Elaboració de la memòria: apèndix amb codi font comentat	5.3/5.4

I la següent taula ens determinarà pel damunt una cronologia tenint en compte les fites de lliurament de les Pacs i els terminis necessaris per a assolir l'acabament del TFC.

Set mana	**Dates**	Activitat	Esdeveniment
1	26 febrer – 3 març	Definir els objectius del projecte i l'abast	Trobada Presencial: dissabte 1 de març
2	4 – 10 març	Definir les diferents tasques. Realitzar la planificació	9 març. Lliurament Planificació Provisional
3	10 – 17 març	Realitzar Tasques 1.1, 1.2 i 1.3 Redacció del capítol corresponent	
4	18 – 24 març	Realitzar Tasca 2.1 i 2.5 Redacció del capítol corresponent	
5	25 – 31 març	Realitzar Tasca 2.2 i 2.5 Redacció del capítol corresponent	
6	1 – 7 abril	Realitzar Tasca 2.3 i 2.5 Redacció del capítol corresponent	
7	8 – 14 abril	Realitzar Tasca 2.1 i 2.5 Redacció del capítol corresponent	13 abril. Lliurament PAC1
8	15 – 21 abril	Realitzar Tasca 3.1 i 3.4 Redacció del capítol corresponent	
9	22 – 28 abril	Realitzar Tasca 3.2 i 3.4 Redacció del capítol corresponent	
10	29 abril – 5 maig	Realitzar Tasca 3.3 i 3.4 Redacció del capítol corresponent	
11	6 – 12 maig	Realitzar Tasca 4.1 Redacció del capítol corresponent	
12	13 – 19 maig	Realitzar Tasca 4.2 Redacció del capítol corresponent	17 maig. Lliurament PAC2
13	20 – 26 maig	Realitzar Tasca 5.1 i 5.4 Redacció del capítol corresponent	
14	27 maig – 2 juny	Realitzar Tasca 5.2 i 5.4 Redacció del capítol corresponent	
15	3 – 9 juny	Realitzar Tasca 5.3 i 5.4 Redacció del capítol corresponent	10 juny. Lliurament presentació provisional
16	10 – 16 juny	Realitzar Tasca 5.5 Redacció del capítol corresponent	18 juny. Lliurament final

Escollint una representació per tasques agafem el Gantt, perquè no hi ha gaires tasques critiques que obliguin a utilitzar un Pert o un Roy i aquest es el resultat,

ID	Nom tasca	Comença	Acaba	Duració
1	Tasca 1.1	11/03/04	17/03/04	1w
2	Tasca 1.2	11/03/04	17/03/04	1w
3	Tasca 1.3	11/03/04	17/03/04	1w
4	Tasca 2.1	18/03/04	24/03/04	1w
5	Tasca 2.2	25/03/04	31/03/04	1w
6	Tasca 2.3	01/04/04	07/04/04	1w
7	Tasca 2.4	08/04/04	14/04/04	1w
8	Tasca 2.5	18/03/04	14/04/04	4w
9	Tasca 3.1	15/04/04	21/04/04	1w
10	Tasca 3.2	22/04/04	28/04/04	1w
11	Tasca 3.3	29/04/04	05/05/04	1w
12	Tasca 3.4	15/04/04	05/05/04	3w
13	Tasca 4.1	06/05/04	12/05/04	1w
14	Tasca 4.2	13/05/04	19/05/04	1w
15	Tasca 5.1	20/05/04	26/05/04	1w
16	Tasca 5.2	27/05/04	02/06/04	1w
17	Tasca 5.3	03/06/04	09/06/04	1w
18	Tasca 5.4	20/05/04	09/06/04	3w
19	Tasca 5.5	10/06/04	16/06/04	1w



3.5. Productes obtinguts.

S'han obtingut com a resultat del treball realitzat un visió actual de l'estat de l'art al voltant de les marques d'aigua i un programa que permet experimentar en la codificació, descodificació o en la generació d'imatges confabulades. Cal afegir que el mateix programa permet l'anàlisi dels fitxers bitmap originals o modificats.

3.6. Descripció de la resta de capítols.

Capítol 4. En aquest capítol mostrarem els requeriments i necessitats per les marques d'aigua
Capítol 5 Mostrarem quines son avui dia les tècniques mes habituals de marca d'aigua
Capítol 6 Farem una ullada a la robustesa de les marques d'aigua en front els possibles atacs.
Capítol 7 Fem una descripció de la codificació del TFC
Capítol 8 Estudi dels atacs suportats per l'aplicació
Capítol 9 Manual d'usuari
Capítol 10 Conclusions
Capítol 11 Glossari
Capítol 12 Bibliografia
Capítol 13 L'annex 1 a on estarà tota la informació trobada a sobre el format BMP
Capítol 14 L'annex 2 on hi trobarem el codi de l'aplicació.

4. Tècniques de marca d'aigua

4.1. Introducció

Abans de res cal aclarir dos conceptes que si be semblen iguals no ho son, la esteganografia i les marques d'aigua. Tant una com l'altre serveixen per enviar informació amagada dins de dades de manera que no és pugui detectar. No obstant l'esteganografia és normalment utilitzada per enviar informació del tipus punt a punt entre dos parts. Els mètodes estenogràfics no son robustos enfront la modificació de les dades i només disposen de mètodes molt senzills per a poder suportar la transmissió i l'emmagatzematge. La marca d'aigua ha de poder resistir tots els intents de destrucció de la informació incorporada
No obstant ambdós mètodes son usats per evitar que tercers accedeixin a les nostres dades si no ens interessa que ho facin.
Les marques d'aigua seran utilitzades de forma molt comuna per controlar la propietat del que ens interessi, en aquest cas imatges.

4.2. Historia i terminologia

4.2.1. Historia

Les primeres marques d'aigua és remunten al segle XIII. Eren utilitzades a Itàlia per afegir una senyal que assegurava la qualitat del paper i de qui era el fabricant. Des d'un punt de vista legal, la seva importància queda remarcada en un cas succeït el 1887 a França. Les marques d'aigua de dues cartes van fer que el President Grévy hagués de dimitir.
Des d'un punt de vista digital, la seva importància creix a mida que l'accés per part de tothom primer als equips informàtics i finalment a Internet s'intensifica. D'acord amb l'INSPEC, les primeres publicacions al voltant d'aquest tema apareixen el 1992, tenint un creixement pràcticament exponencial.

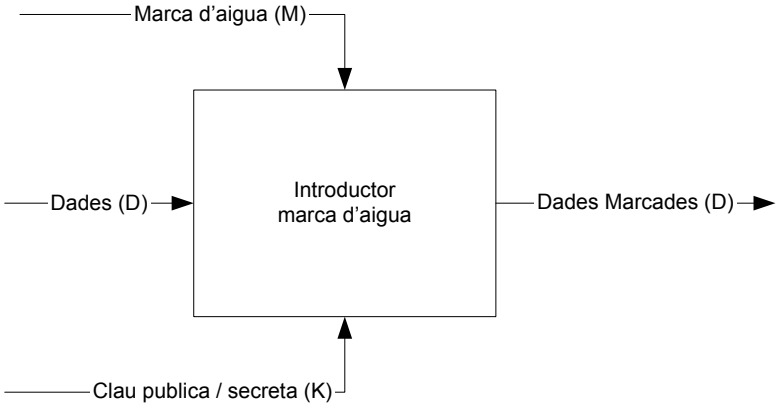
4.2.2. Terminologia

- Marques d'aigua visibles (Visible watermarks). Son patrons visuals inserits o flotant a sobre d'imatges o de vídeo. Avui dia també es pot pensar en fer el mateix amb l'àudio.
- Marques d'aigua (Watermarking). Es incorporar informació amagada extremadament persistent. Es molt robusta a la seva desaparició i incorpora molt poca informació si la comparem amb mètodes esteganogràfics.
- Etiquetatge i signatura (Labeling and Fingerprinting). Etiquetatge és que la marca d'aigua pot contenir qualsevol informació interessant, com identificadors únics. Signatura es que hi ha informació oculta sigui sobre l'autor o del venedor, o un codi únic pertanyent a un conjunt de codis possibles identificant al contenidor on son les dades.
- Marca d'aigua de tires de bits (Bitstream watermarking) Es utilitzada per marcar dades comprimides.
- Signatures contingudes. (Embedded signatures). Nova manera d'anomenar les marques d'aigua en publicacions actuals. Normalment no s'utilitza perquè es pot confondre amb signatures criptogràfiques que tenen la funció de detectar i autenticar dades signades, mentre que les marques d'aigua només serveixen per autenticar. Això si, tenen una major robustesa.
- Marques d'aigua fràgils (Fragile watermarks). Son marques d'aigua amb una robustesa limitada i que només serveixen per a detectar modificacions.

4.3. Principis bàsics de les marques d'aigua

Tots els mètodes de marca d'aigua comparteixen els mateixos blocs, un per a introduir les dades i un per llegir-les.

L'esquema següent ens mostra el diagrama de blocs d'introducció de les dades.

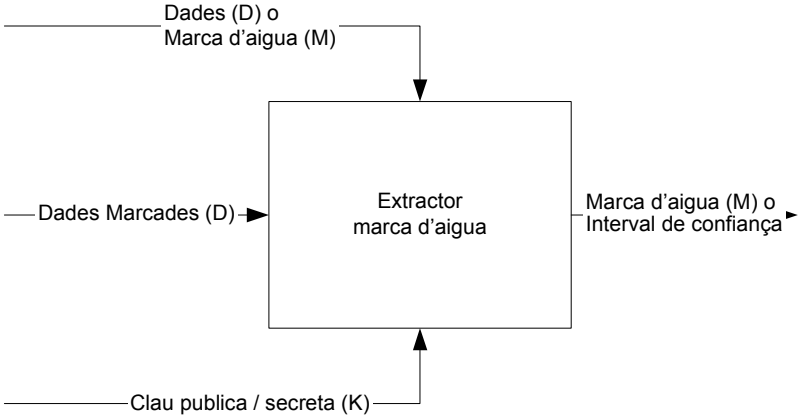


Les entrades al esquema son la marca d'aigua, les dades i una clau opcional. La marca d'aigua pot ésser números, text o imatges. La clau pot fer-se servir per reforçar la seguretat d'accés i manipulació de les marques d'aigua. De fet la majoria de sistemes pràctics utilitzen almenys una clau

Les propietats dels sistemes robustos de marca d'aigua son,

- Imperceptibilitat. Totes les modificacions que comporta la introducció de la marca d'aigua han d'estar per sota d'un llindar que faci que sigui perceptible. Es a dir que hem de quantificar la distorsió acceptable.
- Redundància. Per assegurar la robustesa la marca d'aigua és redundant i està distribuïda, permetent que només amb una part de les dades originals, aquesta pugui ésser recuperada.
- Claus. Son necessàries per assegurar que no es puguin manipular o esborrar les marques d'aigua. Si una persona la pot llegir, segur que la pot modificar o esborrar.

L'esquema següent ens mostra el diagrama de blocs d'extracció de les dades.



De fet en l'esquema anterior podem pensar que doni com a resultat la marca d'aigua o be un marge de confiança, si en lloc d'introduir les dades (D) introduïm una marca que suposem que esta dintre,

En funció d'aquestes combinacions podem trobar aquests tres tipus de sistemes de marca d'aigua.

- Marca d'aigua privada. Es pot dividir en dos tipus, el primer és quan introduïm les dades (D) i extraiem la marca d'aigua ($D \times \check{D} \times K \rightarrow W$). La segona és quan s'introdueix la marca d'aigua (M) i dona com ha resultat que la marca esta dintre o no ($W \times \check{D} \times K \rightarrow \{0,1\}$).
- Marca d'aigua semi privada. No utilitza les dades originals. Només dona com ha resultat si o no. Serveix per exemple en els DVD per saber si allò és pot o no veure en imatges escabroses ($W \times \check{D} \times K \rightarrow \{0,1\}$).

- Marca d'aigua pública. No és necessiten ni les dades originals ni la marca per extreure la marca (Ď x K → W).

4.4. Aplicacions de les marques d'aigua

Sempre s'ha de pensar que els requeriments per a una marca d'aigua donada, venen referits al tipus de mitja on el volem incorporar i que per tant no hi ha mètodes universals.

4.4.1. Marques d'aigua per protegir el copyright

Bàsicament el que es pretén és aconseguir un mètode que ens asseguri que davant d'una disputa sobre qui és el propietari asseguri de manera fefaent que nosaltres ho som. El mes important en aquest cas es la robustesa que eviti possibles manipulacions. Amb l'ús de Internet esta esdevenint capdal en el control de la propietat de les imatges.

4.4.2. Signatures per rastrejar confabulacions

Es la marca que permet com si fos un número de sèrie seguir el seu origen. Ha de permetre que encara que un cracker pugui comparar dos imatges i extreure una tercera, nosaltres siguem capaços de reconèixer que és falsa i de descobrir els seus orígens.

4.4.3. Marques d'aigua per evitar les copies

Son aquelles que eviten que d'un original és puguin fer copies, com les que porten els CD o DVD o be que només permetin fer un determinat nombre de copies i després ja no ho permetin.

4.4.4. Marques d'aigua per autenticar imatges.

Serveixen per assegurar que aquella imatge a estat modificada. Son les menys robustes, es dir, qualsevol manipulació les fa inservibles i per tant ja hem detectat el que volíem.

4.5. Requeriments per a possibles dissenys d'algoritmes.

4.5.1. Imperceptibilitat.

L'únic requeriment necessari per a totes les aplicacions es que sigui imperceptible. .No obstant el mes important és que no es superi el nivell de detecció i que no faci sospitar que aquella imatge o so te quelcom de diferent. Aquest nivell ha de permetre lleugeres modificacions com un escalat d'imatge sense que sigui detectable.

4.5.2. Robustesa.

Evita l'extracció o els atacs malintencionats. Depenen del mitja sobre el que estem treballant, imatges, so,.. . influeix sobre el tipus de robustesa possible. Per exemple si és una imatge hem de permetre que una compressió a JPEG de percentatge elevat no elimini la marca, per tant, podem pensar en el domini de la transformada. També ha de suportar modificacions normals com rotacions podem treballar en mètodes en el domini espacial. Per tant en el primer cas una distorsió no és mes que afegir soroll i en el segon afegir modificacions en la geometria espacial.

4.5.3. Recuperable amb o sense les dades originals.

Si és pot s'aconsegueix un sistema mes robust perquè permet detectar no només el soroll afegit per distorsionar sinó que permet detectar variacions geomètriques. No obstant, actualment s'està treballant mes en el cas que no és disposi de les dades originals.

4.5.4. Extracció o verificació de marques d'aigua donades.

Podem veure des de dos punt. Un en el que la marca esta inclosa dins d'un conjunt de valors possibles i una altra on s'incorporen una sèrie de símbols modulats de forma determinada. Per extreure'ls només hem de fer l'operació a l'inrevés.

4.5.5. Us de claus.

Depèn de l'aplicació que les claus siguin privades o publiques. Podem pensar en dos nivells. Un de mes suau on la marca pugui ésser llegida però no descodificada i un de mes fort on ni és pot llegir ni descodificar perquè per fer-ho és necessitin claus.

4.5.6. Determinació de la propietat

Un cas possible seria el que diferents persones introdueixin dins d'unes mateixes dades la seva marca. Per evitar-ho caldria afegir noves restriccions al disseny com la incorporació de la data de realització.

4.6. Avaluació i benchmarks dels sistemes de marca d'aigua.

No només cal tenir en compte la robustesa per a fer un control d'una marca d'aigua. Realment s'ha de fer una anàlisi entre la robustesa i la perceptibilitat (distorsió introduïda).

4.6.1. Avaluació de les prestacions i la seva representació

- Quantitat d'informació amagada. Contra mes hi ha menys robust és el sistema.
- Resistència de la marca d'aigua. Es la relació entre robustesa i perceptibilitat. Contra mes robusta, mes perceptible.
- Mida i tipus de dada. La mida és molt important, les imatges petites no tenen interès comercial però els programes de marcatge han d'ésser independents de la mida. Normalment a Internet les imatges tenen una menor resolució que les utilitzades per a fer impressions de forma comercial. L'origen de les imatges també importa. Les imatges escanejades del natural amb marca son molt mes robustes que les generades per ordinador amb el mateix tipus de marcatge.
- Informació secreta. La clau oculta no te impacte directe sobre la perceptibilitat, però si el espai de les possibles claus. Com mes gran sigui menys possibilitats hi ha que és pugui trobar per atacs d'exhaustivitat.

Tenint en compte tot l'anterior, els benchmark han de fer les comparatives amagant la mateixa quantitat d'informació.
Com sempre hem de tenir en compte la relació entre perceptibilitat i robustesa. Tenir en compte la perceptibilitat en els benchmark és una tasca difícil perquè depèn de les persones i no de valors numèrics. Normalment aquest test tenen dos passos, el primer, que s'ordenen de major a menor les dades distorsionades i un segon pas és fer la consulta a la persona seleccionada i que valori en funció d'una taula. Es pot utilitzar com a base la ITU-R Rec. 500 quality rating per fer la valoració.

Valor	Percepció	Qualitat
5	Imperceptible	Excel·lent
4	Perceptible. No molesta	Bona
3	Lleugerament molesta	Correcta
2	Molesta	Pobre
1	Molt molesta	Dolenta

El tipus de persona importa molt perquè un fotògraf o una persona normal tenen valoracions diferents.

Per tant és un mètode que és pot aplicar en segons quines situacions però no en un entorn de recerca i desenvolupament. En aquest cas s'acostuma a utilitza mesures en funció de la distorsió en els punts. Aquesta mesura actualment és basa en relacions de senyal soroll (SNR signal noise ratio) o relacions de pic senyal soroll (PSNR peak signal noise ratio). Com a unitat de mesura s'acostuma a fer servir el *dB*. Així doncs $SNR(dB) = 10\log_{10}(SNR)$. Encara que puguem pensar que és un bon mètode aquesta mesura no te en compte la visió humana ni els sistemes aplicats per mostrar-ho i per tant avui dia existeixen mètodes que si que ho contemplen.
Després de tenir la mètrica que aplicarem a la distorsió, farem un petit estudi per representar gràficament la robustesa. Per fer els test hem de fer una taula variant o deixant fixes una sèrie de paràmetres. La següent taula ensenya totes les combinacions possibles.

Tipus gràfica	**Paràmetres**			
	Qualitat visual	Robustesa	Atac	Bits
Robustesa envers Atac	Fix	Variable	Variable	Fix
Robustesa envers Qualitat visual	Variable	Variable	Fix	Fix
Atac envers Visual qualitat	Variable	Fix	Variable	Fix
ROC	Fix	Fix	Fix/variable	Fix

Així entenem atac com qualsevol intent d'eliminar la marca d'aigua i robustesa com la resistència enfront d'aquest atac i que serà mesurada amb bit error rate que es pot definir com el nombre de bit erronis llegits respecte els bits continguts o be l'invers elevada al numero de bits que es l'error de detecció.

Així doncs la gràfica que podríem veure en la robustesa envers l'atac mostra com el mètode de marcatge es comportaria en relació als atacs
La robustesa envers la qualitat visual és la relació entre el error de bit o de detecció respecte la qualitat de la imatge. Amb això és pot determinar el bit error respecte d'un atac determinat.
La gràfica de l'atac envers la qualitat visual representa el màxim atac permès per una qualitat d'imatge establerta.
ROC (Receiver operating characteristic) és donar una ullada general al esquema de marca d'aigua que estem revisant perquè permet mostrar la gràfica per a diferents nivells de decisió. Es a dir que mostra la relació entre falsos positius i falsos negatius, es dir, que diem que és negatiu quan realment és positiu i a l'inrevés.

4.6.2. Software per remoure marques d'aigua i benchmarks.

Com el tema de les marques d'aigua és d'interès general, és poden trobar programes a títol particular per fer proves. Per imatges JPEG podem fer servir el Unzign si volem un de mes general podem utilitzar el StirMark

4.7. Futur i estandardització

L'interès en aquesta tecnologia es gran sigui des d'un punt de vista acadèmic o industrial. A Europa tenim TALISMAN (Tracing Author's Rights by Labeling Image Services and Monitoring Network) que pretén protegir amb un mecanismes estàndard per a protegir el copyright de la pirateria .
La ISO també te interès en aquest tema. En concret ho vol incorporar al nou format MP4 de manera que sigui fàcil incorporar tècniques me copyright i d'encriptació.
La indústria del DVD també esta molt interessada per evitar les copies i permetre o una copia o varies en funció del que és desitgi.
De moment no estem parlant d'una tècnica madura encara que la indústria de l'àudio i el vídeo ho utilitzin habitualment. Encara que protegeixi de l'atac de persones inexpertes, qualsevol expert pot eliminar la marca encara que hi tingui que treballar bastant temps. De moment les marques d'aigua de moment no poden fer-se servir com a prova en un procés judicial però mes

endavant quan la tècnica estigui mes evolucionada poder pugui servir. Per a fer una bona protecció cal combinar-la amb tècniques d'encriptació.

5. Tècniques actuals de marca d'aigua

5.1. Introducció

Els últims anys ha començat a pujar l'interès en in introduir al documents multimedia (àudio, vídeo i imatges) algun tipus de marca que asseguri que allò te un propietari determinat. A mes a mes ha de permetre que les manipulacions habituals com la compressió no afectin a la seva funció. Al menys hem de tenir en compte els següents tres punts,

- Relació entre la informació de la marca d'aigua i la informació que l'acull. Aquesta depèn tant del tipus de marca, si esta codificada o no, com de la informació que l'acull (text, vídeo,..)
- Degradació de la imatge per culpa de la marca d'aigua. Es un dels criteris primordial en l'avaluació d'una marca d'aigua.
- Robustesa. La marca d'aigua resisteix qualsevol atac.

Altres punts serien,

- Elecció del punts a on la informació serà amagada
- Elecció de la forma de treballa, sigui en l'espai en el domini de la transformada,..
- Estratègies per introduir marques codificades, amb control d'errors..
- Maneres de barrejar la marca d'aigua i la informació.
- Tècniques per a millora l'extracció de la marca i que evitin atacs de tipus geomètric.

5.2. Elecció de la posició on inserir la marca dins les dades: aspectes criptogràfics i de percepció

Segons el principi de Kerckhoff que diu que conegut el mètode de xifrat, l'única seguretat que podem tenir és la que proporciona la codificació, aleshores el algoritme de marca d'aigua tindria que ser públic però la marca d'aigua no tindria que ser accessible fàcilment per a prevenir esborrats casuals. Això es pot aconseguir escollint la posició a on la posarem. En moltes implementacions, la posició, s'acostuma a utilitzar un número pseudoaleatori extret d'una clau secreta. El propietari es l'únic que coneix la clau i per tant es l'únic que pot inserir i treure la marca. Al igual que utilitzem claus secretes podem utilitzar claus públiques.

Fins i tot es molt important escollir on introduir la marca d'aigua per que pot afectar a la percepció que el comprador pot tenir de la imatge ja marcada.

5.2.1. El algoritme Patchwork

Com a exemple de tot l'anterior podem comentar aquest. Aquest algoritme no funciona dient a quina posició permet inserir la marca d'aigua si no que simplement respon a la següent pregunta. Aquesta persona coneix la clau que va ser utilitzada per introduir la marca d'aigua?. Aquest algoritme genera un numero pseudoaleatori que ens dona com a resultat la posició on col·locar la marca.

El funcionament bàsic es el següent per inserir la marca al propietari selecciona n punts parells de manera aleatòria segons una clau secreta Ks. Aleshores es modifica la luminància donada segons un parell (a_i , b_i) segons la següent fórmula

$$\tilde{a}_i = a_i + 1$$
$$\tilde{b}_i = b_i - 1$$

El propietari només te que sumar un 1 a la a_i i restar un 1 a la b_i. En el procés d'extracció el n punts parells es tornaran a aconseguir segons la clau secreta Ks. Aleshores la suma es calcula així

$$S = \sum_{i=1}^{n} \tilde{a}_i - \tilde{b}_i$$

Si tot a anat bé i la imatge està marcada la suma valdrà 2n, si no serà aproximadament igual a zero. L'extracció es basa es suposicions estadístiques,

$$E[S] = \sum_{i=1}^{n} E[a_i] - E[b_i] = 0$$

Si ho féssim de manera manual seria extraordinàriament difícil aconseguir sumes igual a 2n i per tant és pot assegurar que només el propietari aconsegueix que $S \approx 2n$

Des de la seva aparició s'han fet millores a aquest algoritme per amagar mes enllà de només un bit i incrementar la robustesa d'aquest esquema. (Es pot verificar en una operació bàsica que la translació d'un punt de la imatge es suficient per atrapar al propietari).

5.2.2. Criptografia de clau i recuperació de marca d'aigua públiques

Els algoritmes de marca d'aigua en clau secreta tenen el seu major problema en que no permeten llegir-les de manera pública, per això s'han creat mètodes que barregen les dues coses la marca d'aigua s'introdueix amb una clau secreta i es pot verificar amb una pública.

Alguns autors han introduït la idea de extreure aquesta clau pública pel mètode dispersió d'espectre (S). La tècnica per aconseguir-ho, requereix la seqüència de dispersió per codificar i descodificar però degut a la robustesa de la codificació es possible llegir la senyal original sense conèixer la totalitat de l'espectre(amb una part hi ha prou S^{pub}). Només és necessària la clau secreta per codificar

5.2.3. Codi predictiu per arranjar la percepció de la marca d'aigua

Els models predictius és fan servir àmpliament. La suposició es que qualsevol punt que esta a prop d'un altre tenen una correlació elevada i en la major part dels casos és així. Els codis predictius calculen primerament els error entre els predit i l'original abans de codificar els errors per fer-ho de forma eficient.
Des de el punt de vista de la marca d'aigua i la percepció els humans detecten malament les zones amb textures o amb arestes i molt be les zones amb canvis de colors suaus. Així doncs el codi ha d'intentar amagar la marca en les zones menys detectables. La distribució de l'error acompleix aquestes condicions i per tant la senyal de l'error es pot fer servir com a portadora de la modulació de la senyal de la marca d'aigua.

5.3. Elecció de la forma de treballar

Com hem vist podem fer-ho en un domini espacial o en el de la transformada.

5.3.1. Transformada discreta de Fourier (TDF)

Es una de les més utilitzades en el camp del tractament de senyal i quan és va començar a treballar en el camp de les marques d'aigua va ésser de les primeres en fer-se servir perquè és poden controlar les freqüències a on col·locar la senyal. Es una de les funcions més adequades

per aconseguir seleccionar les parts de la imatge a on inserir la marca obtenint un major compromís entre visibilitat i robustesa.

En una senyal f(x,y) la TDF és defineix com,

$$F(k_1,k_2)=\beta\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}f(n_1,n_2)\exp(-i2\pi n_1k_1/N_1-i2\pi n_2k_2/N_2)$$

essent $\beta=\frac{1}{\sqrt{N_1N_2}}$ i $i=\sqrt{-1}$

Aleshores, la inversa (ITDF) serà

$$f(n_1,n_2)=\beta\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}F(k_1,k_2)\exp(-i2\pi k_1n_1/N_1-i2\pi k_2n_2/N_2)$$

La TDF el que és fa servir per a fer una modulació de fase entre la marca d'aigua i la imatge. Encara que el mes usual és utilitzar derivades de la TDF com poden ser la DCT o la transformada Mellin-Fourier,

5.3.2. Transformada discreta del cosinus (TDC)

La TDC és una de les mes utilitzades per els generadors de codi perquè és la que s'utilitza per a comprimir en formats MPEG i JPEG i per tant també és va pensar en ella per a introduir marques d'aigua dintre de imatges o de vídeo
Un dels motius per fer servir la TDC és que ja que treballem amb el mateix domini que el JPEG i MPEG, aleshores els atacs que provinguin d'aquí seran tractats mes fàcilment. Com a complement el dispositiu de codificació és fa en el mateix domini comprimit que el JPEG i MPEG i així podem minimitzar el temps de computació.
La TDC pot utilitzar-se per a fer marques d'aigua de diferents formes. Primer és pot fer la suma dels coeficients DFT de la imatge i de la marca. Es poden buscar diferents relacions entre els coeficients de la DFT i els valors dels bits de la marca d'aigua.

5.3.3. Transformada Mellin-Fourier

La majoria dels algoritmes de la marca d'aigua tenen un dels seus punts dèbils en les transformacions geomètriques i per això és fa servir aquesta transformada.
La transformada en l'espai Mellin-Fourier esta basada en la translació de les propietats de la transformada de Fourier,

$$f(x_1+a,x_2+b)\leftrightarrow F(k_1,k_2)\exp\left[-i\left(ak_1+bk_2\right)\right]$$

Podem verificar que només s'altera la fase amb la translació. Si ajustem la zona de treball en una més petita relacionada amb l'amplitud de la transformada de Fourier, farem la imatge insensible als desplaçaments espacials. Per fer-la insensible a les modificacions de mida o a les rotacions podem fer un canvi de coordenades passant les cartesianes a polars.

$$(x,y)\mapsto\begin{cases}x=\exp\rho\cos\theta\\ y=\exp\rho\cos\theta\end{cases}\text{ amb }\rho\in\Re\text{ i }\theta\in[0,2\pi]$$

Aleshores qualsevol canvi és redueix a una translació i per tant fem insensitiva la marca a aquestes modificacions.

5.3.4. Transformada en el domini Wavelet (TDW)

Es una tècnica que apareix a partir de l'aparició del estàndard JPEG-2000. De fet compleix els mateixos criteris que la TDC i a mes afegeix la multiresolució que ens millora la relació robustesa envers visibilitat.
Bàsicament la transformada Wavelet consisteix en una descomposició de la imatge en una multiescala espai – freqüència. Per fer un exemple, pensem en una descomposició d'una imatge amb tres factors d'escala. En el mes baix factor d'escala fem una divisió amb la freqüència més baixa vertical i horitzontal (LL3). A la seva dreta, amb el mateix nivell de resolució fem la mes baixa freqüència horitzontal però la mes alta vertical (HL3). Successivament farem el mateix per les altres combinacions i per els altres dos factors d'escala. Un factor d'escala no és una divisió de la imatge amb parts sinó que cada escala conte informació del total de la imatge, però només a determinades freqüències. Així el esquema següent ens mostraria l'ordenació de totes les escales i de totes les freqüències.

<table>
<tr><td>LL3</td><td>HL3</td><td>HL2</td><td rowspan="2">HL1</td></tr>
<tr><td>LH3</td><td>HH3</td><td></td></tr>
<tr><td colspan="2">LH2</td><td>HH2</td><td></td></tr>
<tr><td colspan="3">LH1</td><td>HH1</td></tr>
</table>

Una manera d'aconseguir aquest diferents nivell de resolució és col·locar en cascada bancs de filtres de dos canals i fer un procés de delmació. Els filtres tenen que ser ortogonals i venen donats pel següent ,

$$H(\omega) = \sum_k h_k \exp(-jk\omega) \_ pasa \_ alts$$

$$G(\omega) = \sum_k g_k \exp(-jk\omega) \_ pasa \_ baixos$$

El procés d'iteració per la descomposició serà,

$$c_{j-1,k} = \sum_n h_{n-2k} c_{j,n}$$

$$d_{j-1,k} = \sum_n g_{n-2k} c_{j,n}$$

I el procés per la reconstrucció serà,

$$c_{j,n} = \sum_k h_{n-2k} c_{j-1,k} + \sum_k g_{n-2k} d_{j-1,k} +$$

5.3.5. Divisió imatges en zones de percepció

Com sempre cal tenir en compte la invisibilitat de la marca d'aigua. L'idea és aprofitar les problemes de la visió humana per a detectar textures i formes i amagar en aquestes zones la marca d'aigua. Així assegurarem que sempre estarem sota dels llindars de percepció de la marca dins la imatge. De fet, en àudio, és treballa de forma similar amagant el que és vol en zones no audibles per a la gran majoria de persones. Una manera d'aprofitar que la visió humana separa els estímuls visuals en diferents components es descriure cada component segons aquests tres paràmetres,

- La localització en el camp visual
- L'amplitud de la transformada de Fourier (o be la freqüència espacial)
- La fase de la transformada de Fourier (o be l'orientació)

5.4. Donar format als bits de la marca d'aigua

5.4.1. Expansió d'espectre

Generalment la marca d'aigua te una mida molt més petita que la imatge. O sigui que és una senyal de banda estreta. Per evitar això, s'intenta que la senyal de la marca concordi amb la de la imatge. També sabem que per a les marques d'aigua una freqüència alta no te sentit, perquè és de les primeres en desaparèixer en una compressió des d'un punt de vista de la robustesa però que és la millor zona per amagar informació si no volem impacte visual. I amb les freqüències baixes passa a l'inrevés. L'expansió d'espectre permet d'amagar una senyal de baixa energia dins de cadascuna de les bandes de freqüència de la imatge.
Hi ha dos mètodes principals,

- Seqüència directa. Consisteix en una modulació en el temps de la senyal original amb una senyal de pseudosoroll de banda ampla. El resultat dona una senyal molt semblant a la senyal del pseudosoroll. La reconstrucció es fa utilitzant la mateixa senyal de pseudosoroll. La senyal original és pot reconstruir encara que s'hagi perdut informació perquè la informació original havia estat amagada en diferents bandes de freqüència.
- Salt de freqüència. Utilitzant un sistema aleatori, la freqüència portadora va desplaçant-se, descrivint un ampli rang de valors de freqüència. El resultat és una senyal de banda ampla.

Per aconseguir la reconstrucció cal conèixer el procés aleatori de modulació. Però, per la mateixa filosofia d'aquest procés, qualsevol atac geomètric afecta molt.

5.4.2. Disseny de marques d'aigua de baixa freqüència

Sabem que la majoria de les vegades, fer una compressió o algun tipus d'escalat provoca un filtrat a on s'eliminen les altes freqüències. Es per això que s'han desenvolupat alguns algoritmes que marquen en la zona de les baixes freqüències mitjançant la transformada de Fourier. Per fer-ho imaginem una marca de la mida de la imatge. Aleshores la reduïm i després li apliquem la transformada de Fourier. Finalment s'afegeixen zeros fins a assolir la mida de la marca original i tornem a invertir la transformada de Fourier de la marca. D'aquesta manera assegurem que la marca només esta en la zona de baixa freqüència.

5.4.3. Codis de correcció d'errors.

Per millorar les marques, s'està estudiant afegir control d'errors. Un símil seria comparar la marca amb la transmissió de senyal dins un entorn ple de soroll. Però en un mitja amb soroll, aquest normalment és Gaussià, però en un entorn com l'estudiat, els "sorolls" o possibles atacs son molt diferents,. També juga en contra que una imatge és una sola cosa i altres sistemes que porten o poden portar control d'errors com l'àudio o el vídeo tenen més canals o mes frames.

5.5. Amagant la marca dins la imatge

5.5.1. Modulació de fase.

Pensem en una F(I) com la TDF de una imatge I. Si I és real, aleshores Fourier ens implica que F(I) és complexa. Perquè F(I) fos real, la senyal tindria d'ésser simètrica. Si pensem en l'anterior, arribarem a la conclusió que la modulació de fase fa que una marca sigui molt robusta. Això és així perquè els components de la fase son molt importants per a la visibilitat de la imatge, molt més enllà que els components de magnitud. Si pensem des del punt de vista de les comunicacions, qualsevol transmissió amb modulació de fase, és molt mes robusta en un entorn sorollós. I un altre punt a considerar és que la fase, és molt mes resistent als canvis de contrast.
Per tot l'anterior, qualsevol intent d'extreure la marca suposa una greu pèrdua de visibilitat de la imatge.

5.5.2. Modulació d'amplitud.

Vist tot l'anterior, no tindríem que pensar en aquesta possibilitat, però podem aprofitar una de les peculiaritats de la vista humana. Els humans quan veiem imatges amb valors alts de lluminositat, perdem agudesa. Aleshores podem pensar en treballar directament en el domini espacial, modulant en amplitud. Per tant si modifiquem el canal del blau, per exemple, en modulació d'amplitud, podem arribar a ocultar la marca aprofitant-se dels defectes de la visió humana. El que si s'ha de contemplar és que el valor que modula ha de tenir en compte la robustesa envers la visibilitat.

5.5.3. Amagar la marca preservant la mitjana de la lluminositat de la imatge

El mètode és basa en la classificació de la imatge en zones homogènies en funció de la informació continguda als punts.
Per tant l'algoritme que ho contempli ha d'acomplir el següent,

- Escollir els blocs de la imatge a on amagarem la informació en funció e la clau secreta.
- Classificar els punts de cada bloc com a membres de una zona d'alt o de baix contrast i per tant hi ha dos zones R_1 i R_2 El valor mig de cada zona s'utilitzarà per a fer càlculs.
- Dividim cadascuna de les regions anteriors en dos zones marcades com A i B d'acord a una trama. Fent combinacions trobem que apareixen 4 subzones: $R_{1,A}$, $R_{1,B}$, $R_{2,A}$,$R_{2,B}$, incloent dins d'elles $n_{1,A}$, $n_{1,B}$, $n_{2,A}$, $n_{2,B}$, punts associats amb valors migs de lluminositat $Y_{1,A}$, $Y_{1,B}$, $Y_{2,A}$, $Y_{2,B}$.
- Aleshores per a cada bit de la marca d'aigua(m) l'operació d'inclusió serà la següent, tenint en compte que l és el nivell on l'incloem.

$$m = 0 \rightarrow \tilde{Y}_{1,A} - \tilde{Y}_{1,B} = \tilde{Y}_{2A} - \tilde{Y}_{2,B} = -l$$
$$m = 1 \rightarrow \tilde{Y}_{1,A} - \tilde{Y}_{1,B} = \tilde{Y}_{2A} - \tilde{Y}_{2,B} = l$$

Hem de tenir en compte que la lluminositat de R_1 i R_2 s'ha de mantenir i per tant definim que,

$$\frac{n_{1,A}\tilde{Y}_{1,A} + n_{1,B}\tilde{Y}_{1,B}}{n_{1,A} + n_{1,B}} = Y_1$$
$$\frac{n_{2,A}\tilde{Y}_{2,A} + n_{2,B}\tilde{Y}_{2,B}}{n_{2,A} + n_{2,B}} = Y_2$$

Aleshores ja podem calcular els valors de $\tilde{Y}_{1,A}, \tilde{Y}_{1,B}, \tilde{Y}_{2A}, \tilde{Y}_{2,B}$ amb els valors del bit de la marca a amagar

- Aleshores per a cada punt de la mateixa regió queda modificat per un offset que és calcula així

$$\delta_{i,j} = \tilde{Y}_{i,j} - Y_{i,j}$$

Per a l'extracció els primers tres passos serien els mateixos. El quart faria el següent

$$\sigma_1 = \tilde{Y}_{1,A} - \tilde{Y}_{1,B}; \sigma_2 = \tilde{Y}_{2,A} - \tilde{Y}_{2,B}$$

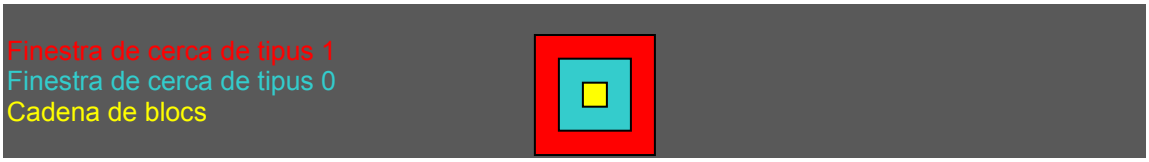
En funció dels seus signes donarien el bit de la marca. Quan parlem de copyright de tots els valors amagats, només una desena de bits son tinguts en compte, aleshores podem pensar en afegir redundància per a corregir possibles errors.

5.5.4. Amagar la marca basant-se en la quantització dels coeficients de la TDC

Una manera de fer servir els coeficients de la TDC és dividir una imatge en blocs de n x n punt en funció de una clau secreta. Aleshores calculem el TDC per a cada punt del bloc. Aleshores. En funció del valor del bit a amagar i per a cadascun dels possibles valors quantificats per la TDC, arriben a una clau. Si el bit de la marca és 1 i $a_{m,n}>a_{n,m}$ o si el bit és 0 i $a_{m,n}<a_{n,m}$ no fem res. En cas contrari els canviem d'ordre. Així assegurem que a cada bloc, haurem guardat el valor d'un bit. Per extreure la marca és faria a l'inrevés.
Malgrat tot, aquest algoritme provoca que la marca sigui visible. S'han proposat solucions com escollir els blocs en funció dels defectes de la visió humana i no marcar-los tots, treballar amb tres coeficients de Fourier en lloc de dos.

5.5.5. Amagar la marca basant-se en la substitució del bloc en la codificació fractal.

Abans d'avançar, explicarem la codificació fractal.
Tenim I_{orig} que és la imatge a ésser comprimida, I_0 una imatge arbitraria inicial, I y J dues imatges genèriques i d(I,J) com una mesura de la distorsió que quantifica aquesta per les dues imatges. Una transformació *T* que mapeja una imatge dins d'una altre és contraent si, $d(T(I),T(j))<\delta d(I,J)$ amb $0<\delta<1$, essent δ la capacitat de contracció. Aleshores $T^n(I_0)$ convergeix cap a una I_a (atraient) si n tendeix a infinit i I_0 es independent de I_a
El teorema del collage diu que si existeix una transformació *T* tal que $d(I_{orig},T(I_{orig}))<\varepsilon$ i *T* és pot contraure essent δ la capacitat de contracció, aleshores $d(I_{orig},I_{a)}< \varepsilon /(1- \delta)$. La feina del codificador és determinar una transformació *T* (codi IFS) de manera que $T(I_{orig})$ sigui tan similar a I_{orig} com sigui possible fent servir el nombre de bits especificats per *T*. Per descodificar-lo s'ha de transmetre el codi IFS .
Des del punt de vista de les marques d'aigua, per codificar-lo tindrem de seguir les següents passes,



- Escollir m com una seqüència per a amagar amb redundància U
- Per cada bit m_k seleccionem U bloc escollits per una clau secreta
  - Si m_k =1 aleshores R_k es codificat cercant el seu candidat bloc de domini D_i fent cerca de tipus 0
  - En cas contrari la cerca es fa en regions de tipus 1
  - Els restants blocs de la cadena son codificats cercant D_k dintre de les zones de tipus 0 i 1 com en el cas clàssic de la codificació de imatges fractals
  - Calculem l'atraient.

El procés d'extracció serà,

- Des de l'atraient i de cada bloc senyalat R_k, indicat per la clau secreta, trobem el bloc de domini associat $\check{D}_k$, si $\check{D}_k$ pertany a una zona de tipus 1 tenim un 1 de la marca, altrament tenim un zero si es de tipus 0
- Per cada bit m_k la decisió final és pren tenint en compte la majoria de 0 i 1 del conjunt de respostes U

L'aproximació explota el fet de que l'atraient és invariant i per tant que pot ésser codificada sense error.

5.6. Optimització del receptor de la marca

Les operacions del receptor son directament derivades del algoritme d'inserció encara que altres son independents d'aquest.

5.6.1. Prefiltratge de la imatge.

S'utilitza en segons quins casos. En la modulació d'amplitud és mes fàcil extreure la imatge de una senyal Gaussiana amb mitja zero i petites variançes. Per a fer-ho farem servir la convolució.

5.6.2. Màxima correlació de fase per reorientació i escalat.

Una manera d'evitar alguns atacs geomètrics com la rotació i el zoom, s'aconseguiria aplicar transformacions inverses a las que aquests atacs provoquen.
Per fer-ho l'algoritme necessita tenir una part de la imatge o de la marca original. Aleshores generem una trama en tres dimensions de la correlació de fase. D'aquí extraurem de manera fàcil el màxim i tindrem el valor de l'escala horitzontal, de l'escala vertical i de l'angle de rotació.

5.6.3. Llindar adaptatiu per a millora la robustesa.

Degut a els possibles atacs, moltes vegades és necessari canviar el nivell per a codificar i descodificar. Alguns algoritmes incorporen marques de bits de valor constant per a poder determinar els llindars d'extracció.

5.7. Problemàtica dels algoritmes per a marcar vídeo

No només ha d'acomplir el mateixos requisits de visibilitat i robustesa que les imatges sinó que ha de suportar tot els possibles atacs temporals. Així la visibilitat en funció del temps és molt mes dificultosa de controlar i apareixen atacs com el de canvi de frames o be els número de frames per segon. I si parlem de vídeo comprimit podem parlar d'un conjunt d'imatges mostrades en el temps en les quals podem modificar la quantització dels coeficients de TDF però que poden afectar al control de la visibilitat de les trames.

6. Robustesa dels sistemes de marca d'aigua

6.1. Necessitats per la robustesa

La robustesa d'una marca d'aigua a la seva extracció vindrà definida per la quantitat de temps que necessitarem per a poder eliminar-la o per a la quantitat de dany fet a la imatge per aconseguir-ho.

Així, per exemple, la IFPI (La federació internacional de la indústria fonogràfica) ha intentat definir-la a fi de aconseguir evidencies de pirateria, seguir l'ús que fan de la musica els mitjans de masses i fer un control de les possibles copies.
Així la robustesa és defineix per,

- El mecanisme de marcat no ha d'afectar la qualitat sonora
- La marca ha d'ésser recuperables fins i tot després de processos de filtrat, incloent dos conversions successives A/D i D/A, de suportar compressions i expansions de fins a un 10%, d'afegir o multiplicar soroll, entre d'altres operacions de modificació de la senyal original.
- No tindria d'haver altre tipus de possible desaparició o alteració de la marca que fent malbé el suport
- Donada una relació senyal – soroll de 20dB o mes, les dades contingudes tindrien que tenir un ample de banda de 20 bps després de cada correcció de l'error independentment del tipus de senyal (clàssica, pop, veu)

Un atacant sempre vol degradar o eliminar qualsevol marca que pugui evitar que el propietari reclami els seus drets. En seguretat un sistema és tan robust com ho pugi ésser el seu punt més dèbil. Es considerarà un atac com a reeixit quan aquest hagi aconseguit destruir qualsevol pas del procés de marca d'aigua. El propietari es qui ha d'assegurar que tots els passos son igual de segurs.

Així doncs, podem dividir els atacs de marca d'aigua en quatre tipus: atacs de robustesa, atacs de presentació, atacs de falsificació i atacs legals. Els atacs de robustesa son els que s'encarreguen d'aconseguir una disminució de la senyal i son els mes intuïtius dels quatre, aquest atacs poden ser tan innocents com una compressió o be utilitzar programes específics per aquesta finalitat. Els atacs de presentació intenten explotar les errades de detecció de marca d'aigua, el que és pretén es que encara que estigui marcat, no es pugui detectar la marca. Els atacs de falsificació el que pretenen és crear situacions en les quals la marca d'aigua original no pugui ser determinada o sigui que no signifiqui res i finalment els atacs legals pretenen aconseguir que la possessió de la marca d'aigua no signifiqui res des de un punt de vista legal.

Malgrat conèixer tots els atacs, la millor protecció es la prevenció. Per a molts consumidors que una imatge estigui mes o menys afectada no suposa cap problema (cas de les pel·lícules en format DVD que hom passa a VCD), mentre que per a un professional, com un fotògraf, pot esdevenir insuportable.

6.2. Disminució de senyal

Aconseguir esborrar una marca d'aigua degradant el contingut és el mes senzill de fer. Encara que per la gran majoria de coses que es poden fer (compressions, escalats i d'altres), està previst que la marca d'aigua ho suporti, sempre pot haver un atac que la destrueixi.

6.2.1. Substitució de marca original

Una de les maneres d'aconseguir una bona destrucció de la marca sense afectar gaire, es afegir una nova marca a sobre, però això només ho poden fer persones que dominin molt el tema.

Com ja sabem hi ha dos tipus de sistemes de marcatge: el públic i el privat. Fer servir aquest mètode per canviar la marca, resulta molt mes fàcil d'aconseguir quan es de tipus públic. En un de privat nosaltres podem incorporar la marca en tot moment però no sabem pas on i per tant no podem estar segurs de que coincideixi plenament. De fet existeixen programes que no permeten tornar a marcar en una determinada posició, sempre i quan estem parlant de codis públics. D'altre manera, el que es podria fer es fer públic el algoritme de marca d'aigua quan la marca és privada, encara que això està prohibit per llei i seria mes fàcil de perseguir. D'una manera senzilla si la imatge pogués ser modificada per qualsevol dels processos habituals i fes desaparèixer la marca seria molt fàcil de rescriure a sobre. Aquest seria el cas del photoshop de Adobe, fent que una imatge marcada és torni borrosa, després fem que és torni lleugerament transparent i aleshores tornar-la a barrejar amb la imatge original. Ja podem pensar que la imatge resultant esta borrosa, però també podem fer comprovacions i veurem que la marca esta eliminada. Només ens queda ja fer un altre filtrat que afini la imatge i tindrem el resultat final, una imatge sense marca i amb poca distorsió.

6.2.2. Compressió

JPEG és un dels mètodes de compressió mes utilitzat per imatges fitxes. Normalment quan volem fer pàgines Web, les imatges acostumen a ésser modificades amb mida i comprimides per ajustar-se al layout de la pàgina i a l'ample de banda requerits. Com sempre, el que s'acostuma a fer desaparèixer son les freqüències altes i mantenir les baixes. Això afecta al sistemes de marca d'aigua per que la gran majoria incorporen tot en les altes freqüències per intentar evitar distorsions. Per això, moltes vegades es recomana utilitzar el marcatge en freqüències baixes encara que es perdi qualitat d'imatge.

6.2.3. Nivells de qualitat en funció de l'usuari.

Com es defineix la qualitat? O quan es considera que s'ha perdut molta qualitat? Aquestes son les preguntes que normalment ens tenim que fer quan parlem de robustesa.

Una de les vulnerabilitats de les marques d'aigua es el grau de degradació de la qualitat que els usuaris estan disposats a acceptar: Conegut això, coneixerem el nivell de danys que una marca d'aigua tindrà que suportar. Els dissenyadors de pàgines Web prefereixen imatges comprimides en format JPEG que s'adeqüin a les seves necessitats o be que els usuaris siguin capaços d'acceptar pel·lícules amb només 250 línies però que ocupen menys recursos

Per aquests motius es difícil de definir la qualitat i per tant la robustesa des de un punt de vista dels creadors de productes de vídeo i de àudio. Els usuaris moltes vegades prefereixen una pitjor qualitat si aquesta s'ha aconseguit de manera gratuïta.

Un altre punt a considerar es la quantitat de degradació suportada causada per la inserció de la marca. Alguns programes de marca d'aigua generen una gran quantitat de soroll (distorsió), això vol dir que els usuaris finals accepten aquest soroll, per tant els que tenen l'ultima paraula son els usuaris que finalment decideixen si es acceptable o no.

6.2.4. Interpolacions

Quan es pot disposar de moltes mostres d'imatge, qualsevol atacant pot agafar-les totes, sumar-les punt a punt i fer el valor mig, aleshores segur que obtindrem una imatge sense marca detectable. Això en casos de vídeos en que tots els frames estiguin marcats es molt senzill d'aconseguir però per evitar-lo de manera fàcil es introduir mes d'una marca i fer aquestes dependents de la imatge perquè no és pugui utilitzar aquest atac.

6.2.5. Atacs específicament dissenyats

Si coneixem els detalls de l'algoritme de marcatge, qualsevol atacant pot crear un atac específic contra aquesta marca. Per exemple, si la marca d'aigua es fa modificant certes freqüències en el domini de Fourier qualsevol atacant que ho conegui pot fer el mateix i eliminar les freqüències a on està la marca d'aigua.

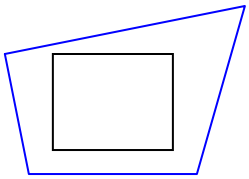
Podem tornar a recordar el principi de Kerckhoff (veure 5.2) que diu que les eines actuals de Enginyeria inversa per a codi son bastant sofisticades, que molta gent te accés al codis fonts a les companyies i que el temps per a desenvolupar noves tècniques de marca d'aigua son bastant grans, per això es fàcil de pensar que quan aquesta tecnologia sigui molt popular podem pensar que qualsevol persona serà capaç de crear cavalls de Troia o coses semblants per aconseguir el que es vol

6.3. Errors de detecció de marca d'aigua

No es necessari remoure la marca d'aigua per fer-la inútil, el contingut pot ser manipulat de tal manera que el lector no la consideri vàlida. Els atacs mes populars d'aquest tipus son les distorsions geomètriques o els atacs de mosaic.

6.3.1. Atacs de distorsió

Encara que el sistema de marcatge d'aigua sobrevisqui normalment a les manipulacions bàsiques, les marques moltes vegades no superen les combinacions d'aquestes o be les manipulacions geomètriques d'ordre menor. Això vol dir que qualsevol imatge que es distorsioni uns graus cap a la dreta cap a l'esquerra la marca d'aigua quedarà inútil però la imatge podrà tornar a ser recuperable. En el dibuix següent és veu una imatge a la que hem modificat els extrems i que podrem fàcilment reparar



Altra possibilitat de distorsió es una lleugera desviació aplicada a cada punt essent mes gran al centre i gaire be nul·la als extrems.

Qualsevol d'aquestes distorsions combinades amb una compressió JPEG no eliminen de fet la marca però si que la fan indetectable. Això el que aconsegueix és que el detector sigui incapaç de detectar el inici de la marca i no es pugui sincronitzar. En el cas del vídeo no només tenim que pensar en dos dimensions si no que també tenim que pensar en els atacs que pugin venir en funció del temps com modificació de frames o coses similars.

Per tant una opció per evitar tot això seria col·locar marques de diferents tipus que evitessin aquest problemes.

6.3.2. Limitacions degudes al bitrate

Per evitar segons quins tipus d'atacs, el millor es col·locar les marques de mida petita. Pot succeir que si el que fem es dividir la imatge inicial en petits blocs podem arribar a trobar que dintre d'aquest blocs no quedi encabida cap i per tant no sigui detectable. Aquest és un dels atacs mes usuals per que molts dels robots Web que baixen imatges el que fan es detectar si aquestes tenen marques i aleshores les tallen a trossos. Quan es volen fer servir l'únic que s'ha de fer es col·locar-les juntes.

6.3.3. Falses alarmes i col·lisions no anticipades

Una falsa alarma ocorre quan un detector troba que aquella imatge està marcada i no es cert. Si aquella imatge ja tenia una marca quan intentem inserir la nostra provoquen el que s'anomena col·lisió de marca d'aigua, això succeeix quan les dues parts utilitzen el mateix sistema de marcatge.

Els dos problemes poden ocasionar que no es puguin resoldre els drets de propietat sobre la imatge. Amb això el que es pot arribar a succeir es que es posi en dubte el propi sistema de marcatge. Però el mes preocupant no es això, donada una imatge i una funció de detecció podem trobar una clau secreta fent proves que ens doni com a resultat que aquesta funció ens doni que es correcta? Sempre hem de pensar si utilitzem un sistema comercial que aquest faci impossible la detecció d'aquesta clau.

6.4. Atacs de falsificació

A part de tot el que pot arribar a passar, ara intentarem seguir tots els passos del procés de marcatge d'aigua per veure, si en algun és pot produir un atac.

6.4.1. Atacs de protocol

Com havíem descrit prèviament moltes marques privades sobreviuen a la inserció d'una segona marca per que la primera marca ha estat introduïda de manera que no sap l'atacant a on està i que aquestes marques normalment son petites i deixen molt espai buit. Aleshores podríem pensar que només inserint una segona marca es podria demanar la seva propietat. Malgrat que això pugui semblar plausible, el propietari sempre ha de tenir el original amagat de possibles modificacions per a poder demostrar que aquella es la imatge inicial.

En general una disputa sobre la propietat es pot solucionar buscant cadascun dels litigants la seva marca sobre l'original de l'altre. Seria molt improbable que cada original fos una versió amb marca d'aigua de l'altre original.

Idealment el creador de la imatge original afegeix una marca w a una imatge I aconseguint una imatge marcada Ĩ=I+w, aleshores aquesta imatge és distribuïda. Si trobem una imatge I' la diferència entre I' menys I tindrà que donar un valor molt similar a w. Una funció de correlació c(w,x) és la que s'utilitza per determinar la similitud entre la marca original i la dada extreta x, de fet el que es fa es buscar la marca d'aigua amb la diferència entre dues imatges
Imaginem que un tercer fa la següent operació amb la marca x

$$\tilde{I}' = \tilde{I} - x = I + w - x$$

Aleshores el propietari i l'usurpador reclamen la propietat, fent operacions tenim que

$$I' - I = w - x; c(w - x, w) = 1$$

$$I - I' = x - w; c(x - w, x) = 1$$

I per tant trobem la marca de l'usurpador a la imatge original!

En altres paraules, el atac treballa restant una marca mentre que l'altre l'està afegint i això be donat per que aquesta marca d'aigua pot ser invertible. Podem arribar a pensar que un mètode per inserir una marca d'aigua només tindria que tenir un sentit d'introducció i així aconseguiríem evitar aquest tipus d'atac.

6.4.2. Atacs d'oracle.

Amb moltes aplicacions un atacant te accés al detector de la marca. Aquest detector pot ser una part de software que be incorporat dintre d'un programa de processament d'imatge o be com una part d'un circuit electrònic d'un dispositiu de electrònica de consum, com un DVD. Encara que el atacant no conegui molt sobre el mètode de marca d'aigua, aquest pot utilitzar

informació retornada pel detector per esborrar la marca aplicant petits canvis a la imatge fins que el descodificador ja no sigui capaç de detectar-la.
La manera de fer aquest atac seria fent servir una imatge que ja sabem que està molt propera al nivell d'acceptació del dispositiu, aleshores només tenim que fer lleugeres modificacions fins que trobem el motiu pel que decideix si es acceptada o no. El segon pas seria arribar a modificar a nivell de punt. Variaríem la seva lluminositat fins que el detector canviés d'estat. Això es tindria que fer per cada punt i l'atacant podria trobar una combinació que aconseguís eliminar la marca sense distorsions.

Per evitar-ho aquest atac és podem incorporar contra mesures com podria ser fer aleatori el procés de detecció. Una cosa molt senzilla seria fer servir dos nivells de detecció en comptes d'un i que cadascun fos independent de l'altre. Altre manera és podria fer buscant algun sistema que fes necessari tant gran el temps de processat que fos físicament inviable la seva resolució. I un altre possible solució encara mes dràstica seria que es permetés fer un número de copies però que a partir d'un punt el propi aparell, que sabia que l'estaven intentant copiar, es deshabilites i fes necessari portar aquest al servei tècnic. Aleshores aquests serien els responsables de notificar-ho a les autoritats

6.4.3. Atacs d'oracle ajustats als sistemes privats.

En l'apartat anterior, encara que no es digués, es podia pensar que treballàvem amb marques d'aigua de tipus públic. Malgrat això, si fos de tipus privat es podríem seguir els mateixos passos ajustant-se al sistema del propietari de la marca.

El atacant insereix la seva pròpia marca una vegada o moltes dintre de la imatge utilitzant la mateixa marca que el propietari del copyright.

El atacant utilitza la seva pròpia marca com a indicador de nivell pel sistema de marca d'aigua privat. Aleshores només tenim que utilitzar el mateix que amb el apartat anterior, considerant també que la marca d'aigua original és torna mes dèbil quan la imatge es canvia de manera aleatòria i això també succeeix quan s'insereixen altres marques d'aigua. Per tant podem assumir que la fortalesa de la nova marca introduïda, proporciona un nivell superior a la fortalesa de la marca d'aigua original. Aleshores quan la nova marca hagi estat remoguda amb tota probabilitat la marca d'aigua original també haurà desaparegut.

6.5. Detecció de marques d'aigua

En els atacs previs és considera que l'algoritme de inserció es desconegut però en alguns casos coneixem detalls sobre el mètode d'inserció i de detecció i per tant la informació amagada pot ser trobada i mes tard feta desaparèixer.

6.5.1. Un atac d'ocultació de eco

Un atac obvi contra aquest tipus d'ocultació es pot fer simplement invertint la formula de convolució. L'únic problema és trobar el eco sense coneixement de l'objecte original o dels paràmetres de l'eco. Sembla que les tècniques utilitzades en els sistema de legitimació poden ser utilitzades pels atacants però requereixen una mica mes de treball.

6.5.2. L'atac per detecció de pics

En alguns casos la imatge que volem marcar te certes condicions que poden ajudar als atacants maliciosos a aconseguir informació sobre la marca. Un cas molt senzill de veure seria una imatge, per exemple procedent d'un dibuix animat, en el qual el nombre de colors es molt petit. Si fem un histograma dels colors en funció del nombre de punts podríem veure uns pics clarament en l'anàlisi de l'espectre que no pertanyen als colors originals i que ens indiquen posicions i valors possibles per a la marca.

6.6. Atac en funció de l'arquitectura del sistema

Aquest atacs son aquells que procedeixen de no haver tingut en compte coses que son pròpies del mon real, com el factor humans, les interfícies d'usuari o les habilitats de la implementació.

6.6.1. Factors humans

Un usuari típic el que pretén és considerar el procés de marcatge d'aigua com una caixa negre que per una banda se li entra la imatge i per l'altre aconsegueix ja la imatge marcada. La majoria d'usuaris no poden perdre, ni els interessa, temps en comprendre o com s'ha fet el procés realitzat.

Normalment les persones que necessiten que el seu treball sigui marcat acostumen a ser artistes. Un artista prefereix que la seva obra quedi tal com es originalment a que aquesta sigui modificada per la introducció d'una marca que asseguri la propietat, perquè aquesta provoca una lleugera modificació al treball original.

6.6.2. Interfície d'usuari

Des de que ens trobem la introducció de la marca d'aigua incorporada en la majoria de aplicacions de processat de imatges, l'usuari pot accidentalment esborrar la marca treballant sobre la imatge. Aquest programes han de evitar que el processos que el usuari habitualment fa, com canviar colors o modificar mides, no destrueixin de manera accidental la marca d'aigua. Una manera de evitar-ho seria introduir un nivell gràfic dintre de la interfície gràfica que ens indiques la fortalesa de la marca, per poder veure com aquesta reacciona a les modificacions. El software tindria que ser capaç de discernir quan pot ser aquesta imatge guardada de manera definitiva a disc. Altra possibilitat seria que quan es carrega la imatge aquesta estigui sense la marca essent aquest procés transparent a l'usuari.

6.6.3. Debilitats de la implementació

La majoria d'atacs sobre sistemes criptogràfics apareixen per la explotació oportunista de bugs trobats de manera accidental. Aquest atacs exploten les debilitats de la implementació en lloc dels algoritmes de marcatge. Imaginem un sistema on cada usuari tingues un identificador i una contrasenya de dos dígits, el primer mètode per trencar aquest sistema seria un debugger per introduir-nos dintre del software i deshabilitar els mecanismes de contrasenya. En aquest atac podem fer servir la documentació que moltes vegades està disponible a Internet. Un segon mètode seria amb el password secret, com aquest només te dos dígits, amb un centenar de combinacions tindríem prou per descobrir-ho.

De la mateixa manera, fent servir el mateix debugger podríem fer que quan el programa fes la comprovació, evites detectar que l'ha trobat i permetés sobre escriure la marca que és vulgues

6.6.4. Limitacions als robots de cerca automàtica d'Internet

Quan nosaltres volem buscar dins Internet si les nostres imatges han estat modificades, ens trobem en un primer problema que es l'ample de banda. Es impossible per un usuari individual trobar totes les infraccions sobre les seves imatges. L'usuari necessita registrar-se en un servei públic de cerca per trobar-les, aleshores també necessitarà si utilitza un sistema de marca privat, registrar el seu sistemes de marcatge i per tant això obre nous problemes.

Imaginem un cas en que dos persones, una es la que ha creat la imatge original i l'altre l'ha modificada es registren en el mateix servidor i utilitzen la mateixa clau secreta, per que tots dos son amics però un esta robant a l'altre. Aleshores per que aquest segon que te imatges modificades acompleixi que els motors de cerca a Internet no les trobin, senzillament ha d'indicar al seu servidor que no serveixi aquestes imatges.

6.7. Atacs als jutjats

6.7.1. Servidors a l'estranger

Es la manera mes senzilla d'utilitzar imatges que no son propietats dels que les tenen. Qualsevol país que no estigui adherit a la convenció de Berna sobre la protecció del copyright, no te que donar cap explicació sobre la informació dels servidor que hi ha en aquell país. Fins ara no hi havia gaires problemes per que aquest països no tenien enllaços d'alta velocitat però ara, que si que els posseeixen el problema comença a esdevenir sagnant.

6.7.2. Atacs de suplantació de DNS

Encara que el país acompleixi la convenció de Berna es molt difícil de demostrar en un judici que aquella imatge pertany a una persona determinada, per a fer-ho es necessiten testimonis imparcial, no podem dir que es nostre pel simple fet de dir-ho.

Un atac bastant senzill seria fer que quan el propietari accedís a una imatge seva robada, ho fes a una adreça d'Internet, però que la DNS que suporta aquesta adreça estigui redirigida a un servidor on realment estigui la imatge amagada, aleshores el propietari pensarà que el culpable serà a on ell la trobat però el culpable real serà un altre.

Un altre possibilitat que tindríem seria que el propietari trobes la imatge i mentre fa la denuncia, demana a un notari de donar fe que la imatge està en aquell servidor, el Webmaster s'ha adonat que l'estan vigilant, esborra la imatge i ja s'han perdut totes les proves.

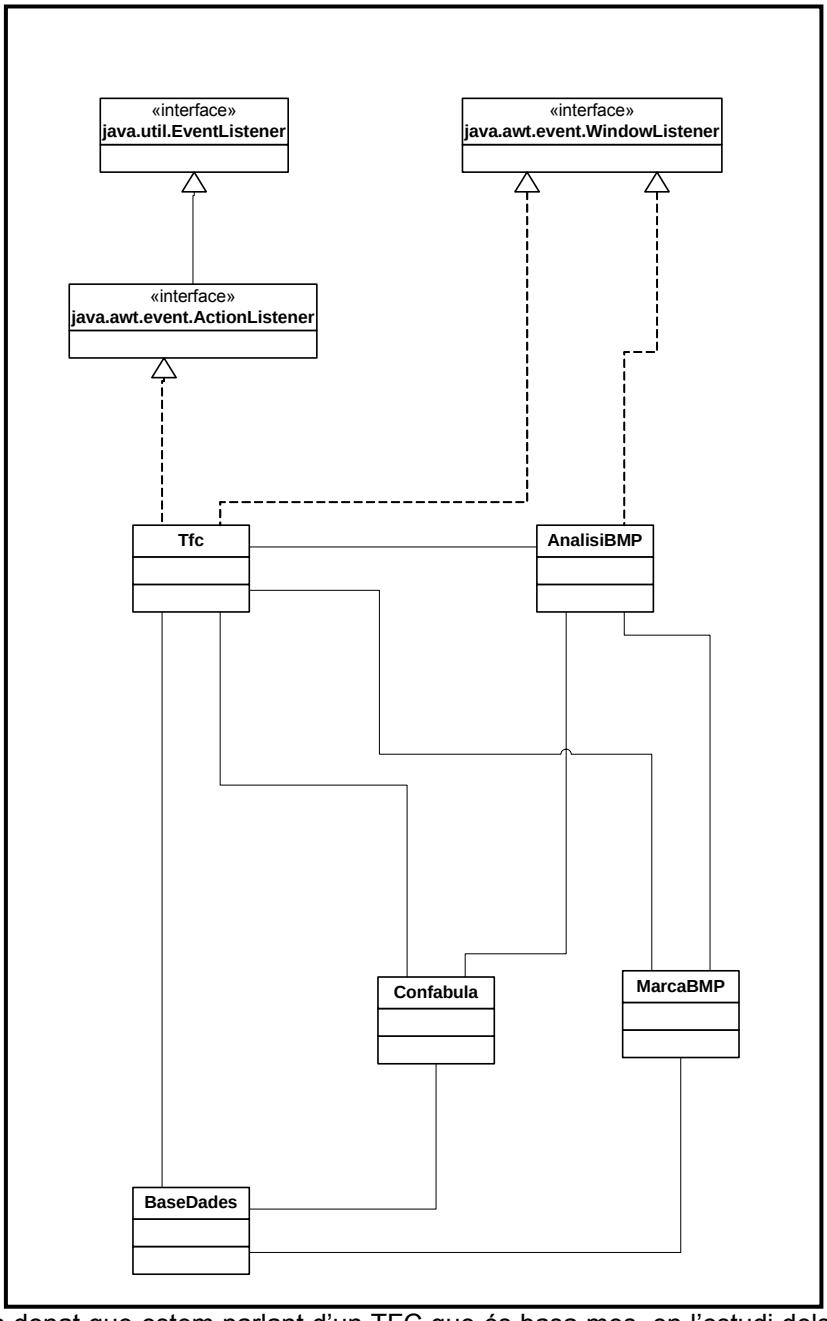
.

7. L'aplicació TFC

7.1. Breu descripció

Se'ns demanava generar dos programes per separat que realitzessin dues tasques clarament diferenciades, per un costat tindríem les funcions del venedor i per l'altre les del atacant.

Per facilitar les tasques de funcionament del sistema, s'ha optat per utilitzar una interfície



gràfica conjunta donat que estem parlant d'un TFC que és basa mes, en l'estudi dels possibles mètodes d'incorporació i extracció de marques d'aigua per part d'un venedor i de l'estudi dels possibles atacs per a eliminar-la, que no d'una finalitat comercial.

L'aplicació te un funcionament molt intuïtiu i de fàcil aprenentatge per l'usuari que esta molt mes acostumat a l'ús d'entorns gràfics que no al de pantalles de text com podrien ésser les de MS-DOS

7.2. Gràfic UML TFC

En el gràfic de la pàgina anterior és mostra el diagrama UML general. El de cada classe en particular és veurà en el apartat que li pertoqui. Tenim tres interfícies, EventListener, ActionListener que estén a EventListener i WindowListener. Aleshores podem veure que TFC implementa les tres i analisiBMP només ho fa amb WindowListener. La resta de classes ensenyen les relacions que mantenen entre si.

7.3. Classe AnalisiBMP

7.3.1. UML de la classe

AnalisiBMP
- String nomFitxer; - BufferedReader entrada; - PrintWriter sortida; - byte []capcalera= new byte [1500]; - byte []word= new byte[4]; - String s=""; - Frame finestraAnalisi; - Panel panel1; - TextArea dades; - byte CRLF[]={0x0d,0x0a}; - boolean retorn=false,ferAnalisi;
+ AnalisiBMP(String nomFitx,boolean bol) + llegeixFitxer():boolean - llegeixWord(int index,int limit): long - creafinestraAnalisi() + windowClosing(WindowEvent e) + windowActivated(WindowEvent e) + windowClosed(WindowEvent e) + windowDeactivated(WindowEvent e) + windowDeiconified(WindowEvent e) + windowIconified(WindowEvent e) + windowOpened(WindowEvent e)

7.3.2. Funcionament

Aquesta classe és a la que, donada l'autorització per part de la classe TFC, la resta de classes passen els paràmetres adients al constructor i al mètode llegeixFitxer per a poder crear en pantalla una finestra a on és vegi l'anàlisi d'un fitxer del tipus BMP.

7.3.3. Mètodes

- Mètode +llegeixFitxer():boolean. És el mes important de tota aquesta classe. El que fa és fer una lectura del fitxer que se li ha passat com a paràmetre al constructor sempre que el bit que se li ha passat també com a paràmetre ho autoritzi. Si tot ha anat bé retorna com a valor cert o fals si no ha pogut.
  Per fer-ho, primer és fa una instanciació del fitxer creant un FileInputstream. Com sempre s'ha d'utilitzar la parella try – catch per al control de possibles incidències. Aleshores fem que és llegeixin a la matriu capçalera fins a 1500 octets. Amb aquest valor assegurem que ens caben totes les possibles combinacions per a la capçalera. Si hem pogut llegir alguna cosa, seguim endavant, sinó retornaríem un error. Es en aquest punt a on la lectura de certs punts concrets de la capçalera o del nom del fitxer, ens

determina que és tracta d'un fitxer del tipus BMP i que emmagatzema la informació en el format de 24 bit sense compressió. L'ultima comprovació abans de crear la pantalla a on mostrar tot el que és troba és vigilar si el bit d'anàlisi està actiu o no.
La creació de la finestra la mostrarem mes endavant, perquè estem parlant d'un altre mètode, però l'únic que cal saber és que la informació és mostra a través d'una variable de tipus TextArea. Per a fer-ho, llegim de forma consecutiva la capçalera llegint segons el format BMP te establertes les posicions per a cada cosa (Veure annex 1). El mètode que farem servir per a la lectura l'explicarem mes endavant. Aleshores, conegudes les dades, és passen com a valor a la finestra fent servir el mètode .append. Les dades que és visualitzaran seran les següents,

- Nom del fitxer:
- Identificador:
- Longitud del fitxer:
- Offset fins a les dades del bitmap:
- Longitud de la capçalera bitmap:
- Amplada horitzontal del bitmap:
- Amplada vertical del bitmap:
- Numero de plans del bitmap:
- Numero de bits per punt:
- Especificació de la compressió:
- Mida del bitmap:
- Resolució horitzontal:
- Resolució vertical:
- Nombre de colors utilitzats en el bitmap:
- Nombre de colors importants utilitzats en el bitmap:

En el cas de que la taula de colors no fos l'estàndard, apareixeria una taula a on és mostraria per a cada color les seves components RGB com està previst en el format BMP i seria mostrada d'aquesta forma,

- TAULA DE COLORS
- COLOR BLAU VERT VERMELL

Col·loquem el cursor a la primera línia per una més fàcil visió de la informació i retornem que hem pogut fer correctament totes les operacions. L'últim a fer-se abans d'abandonar el mètode és tancar el flux de dades procedents del fitxer.

- Mètode - llegeixWord(int index,int limit): long. Amb aquest mètode aconseguim alliberar de tasques repetitives als mètode que tracta la capçalera. És de tipus privat perquè no ens interessa que és pugui accedir des d'altres classes. El que fa és donat un punt inicial i quant valors volem llegir concreta la següent equació,

$$long = \sum_{i=index}^{i=index+lmit} capçalera[i] * 256^{i-index}$$

El fet de que treballi amb 256 és que busquem els valors per a 8 bit, sabent que 2^8=256.

- Mètode - creafinestraAnalisi(). Mètode que genera la pantalla per visualitzar les dades del fitxer. Degut a que normalment és fan dos crides a aquesta classe, tret de quan codifiquem s'ha incorporat una variació aleatòria de la posició d'aparició. Es dir que pot aparèixer des de la posició (150,150) fins a la (250,250) però sempre amb valor iguals per a X i per a Y. Per tant el primer que fem és crear una finestra de 500x150 a la posició que és calculi. Després l'incorporem el mètode necessari per a poder controlar la finestra i fem que aquesta no puguem fer-la a pantalla complerta. Això últim s'ha fet, com és farà en la resta de finestres, per a poder mantenir una uniformitat en la

presentació. Completada la finestra, creem un pannell que farem que els elements que inserim compleixin la disposició de FlowLayout. El pannell tindrà les mateixes mides i posició que la finestra i finalment, inserim un element del tipus TextArea al pannell. Afegim el pannell a la finestra, fent posteriorment que la finestra és faci visible i que s'ajusti als components interiors. En principi, això últim no caldria, perquè ja hem definit el pannell exactament igual a la finestra.

- Mètode + windowClosing(WindowEvent e). Controla l'event de la finestra quan s'està tancant. En aquest cas el que és fa és amagar-la.
- Mètode + windowActivated(WindowEvent e). Controla l'event de la finestra quan és activada. Aquí és posa per a poder implementar la classe WindowListener
- Mètode + windowClosed(WindowEvent e). Controla l'event de la finestra quan està tancada. Aquí és posa per a poder implementar la classe WindowListener
- Mètode + windowDeactivated(WindowEvent e). Controla l'event de la finestra quan és desactivada. Aquí és posa per a poder implementar la classe WindowListener
- Mètode + windowDeiconified(WindowEvent e). Controla l'event de la finestra quan no està minimitzada. Aquí és posa per a poder implementar la classe WindowListener
- Mètode + windowIconified(WindowEvent e). Controla l'event de la finestra quan està minimitzada. Aquí és posa per a poder implementar la classe WindowListener
- Mètode + windowOpened(WindowEvent e). Controla l'event de la finestra quan està oberta. Aquí és posa per a poder implementar la classe WindowListener

7.4. Classe BaseDades

7.4.1. UML de la classe

BaseDades
- String nomFitxer; - RandomAccessFile fitxerDades,fitxerDadesAuxiliar; - byte []dadesFitxer= new byte [1500]; - byte CRLF[]={0x0d,0x0a}; - String s="";
+ BaseDades(String nomFitx) + creaFitxer() + estaFitxerCreat():boolean + escriuDada(String dada): boolean + cercaDada(String dada1,String dada2): boolean + cercaNomComprador(String dada1,String dada2): String + esborraBD()

7.4.2. Funcionament

Gestiona el funcionament general d'una base de dades. Permet de crear un fitxer buit que donarà suport físic a la base de dades o be d'esborrar totes les dades que incorpora. Com a tota base de dades, si se li dona una línia amb un format adequat l'emmagatzema controlant que la dada s'incorpori seguint un ordre establert. Finalment, presenta dos mètodes de cerca diferents en funció de la dada que és vol aconseguir. Cal assenyalar que en aquesta classe tots els mètodes son de tipus públic, és dir que no hi ha cap d'us intern de la classe o del package.

7.4.3. Mètodes

- Mètode + creaFitxer(). El que fa primer és llençar una excepció pels possibles errors, després crea un fitxer de lectura – escriptura amb el nom desitjat, que en aquest cas no es pot canviar i és diu BaseDeDades.txt, i finalment el tanca.
- Mètode + estaFitxerCreat():boolean. Instancia un fitxer amb el nom desitjat i comprova que és pugui llegir retornat cert si ha pogut, fals altrament.
- Mètode +escriuDada(String dada): boolean. Aquest mètode és el mes important perquè no tan sols escriu la dada, sinó que també l'ordena. Novament és llença una excepció pels possibles problemes que puguin aparèixer perquè estem treballant amb dispositius

d'entrada – sortida, en aquest cas un fitxer. Per a poder fer l'ordenament, el que és fa es crear un fitxer auxiliar amb el mateix nom que el fitxer de BD però amb el símbol ~ davant. És van passant les dades a mida que és fa la cerca i la dada a incorporar està mes amunt en el ordenament. Col·loquem el punter de cerca del fitxer al començament i anem llegint línies. El que fem per a cadascuna d'elles, és primer comparar la primera paraula de cada línia. Aquesta correspon amb el nom del fitxer amb adreçament en valor absolut. Si aquest és inferior continuem, si és mes gran col·loquem primer la dada a guardar i després la resta de les que ja estaven emmagatzemades. En el cas que tingues el mateix nom, la cerca ja és fa amb la segona paraula que és la marca i reproduïm el mateix cas que abans però sense la possibilitat de trobar-la igual, perquè des d'on és fa la crida ja s'ha contemplat aquesta possibilitat. S'acaben de passar les dades que hagin pogut quedar sense controlar, per finalment esborrar el fitxer original i canviar a aquest amb el nom pel ja esmentat al damunt com al nom usual de la BD. El retorn serà com sempre cert si ha pogut, fals altrament.

- Mètode + cercaDada(String dada1,String dada2): boolean. Per a fer aquesta cerca el que fem és passar dues dades amb el format de cadena de caràcters, un serà el nom del fitxer i l'altra la marca. Llencem l'excepció usual quan treballem amb fitxers i col·loquem el punter al inici del fitxer. El que primer fem per a evitar errors és llegir la primera línia i comparar-la amb una cadena nul·la o be una buida. Si no és cap de les dos coses, tornem a col·locar el punter al inici i comencem a llegir dades fent iteracions, donant com a límit la longitud del fitxer. Per assegurar que no estem al final del fitxer sense haver trobat la dada buscada, tornem a comparar si el valor de la línia és nul o buit. Ara ja només cal comparar les dues cadenes amb les dues primeres paraules de la línia llegida i retornar cert si l'hem trobat o fals altrament. Com sempre abans de sortir hem de tancar el fitxer amb el que s'ha estat treballant.
- Mètode + cercaNomComprador(String dada1,String dada2): String. El funcionament és el mateix que en el cas anterior, però en lloc de retornar un valor binari, torna el valor del nom que està associat amb les dos dades passades com a paràmetres o bé si no l'ha trobat una cadena buida.
- Mètode + esborraBD(). L'única cosa que fa és esborrar el fitxer que suporta la BD

7.5. Classe Confabula

7.5.1. UML de la classe

Confabula
- String nomFitxer,nomSegonFitxer,marca,nom; - BufferedReader entrada; - byte []dadesFitxer= new byte [1500]; - byte []dadesSegonFitxer= new byte [1500]; - byte []word= new byte[4]; - byte [][]dadesBMP= new byte[1500][4500]; - byte [][]dadesSegonBMP= new byte[1500][4500]; - int []V0=new int[7]; - int []V1=new int[7]; - String s=""; - int offset,longitutBitmap,ampladaHoritzontal,ampladaVertical; - int offset1,longitutBitmap1; - int ampladaHoritzontal1,ampladaVertical1; - int offset2,longitutBitmap2; - int ampladaHoritzontal2,ampladaVertical2; - int longitutTrama=7,quadre=10; - byte profunditat=1; - int quadreExtraccio; - boolean ferAnalisi;
+ Confabula(String nomfitxer,String nomsegonfitxer,boolean bol) + generaNovaMarca():String - llegeixFitxer(String nomfitx,byte []dadesfitx,byte [][] dadesbmp): boolean - llegeixWord(int index,int limit,byte []dadesFitx): long

- confabulaFitxer() - creaFitxerModificat() - estaFitxerCreat(String nomFitxe): boolean

7.5.2. Funcionament

La confabulació és fa quan dos fitxers marcats amb diferents propietaris, els barregem per a obtenir un tercer fitxer. Per a fer-ho el que farem és comparar les dues imatges punt a punt. Si en algun punt difereixen qualsevol dels valor que composa un punt per a les dues imatges, el que farem en la tercera és generar el darrer bit per aquest punt de manera aleatòria per a cadascun dels tres colors que el defineixen. El nom del fitxer confabulat sempre serà de la següent manera nomFitxer.conf.nomComprador1.nomComprador2.bmp i no modificable per l'usuari.

7.5.3. Mètodes

- Mètode + generaNovaMarca():String. El que primer farem és comprovar que el fitxer que generarem de sortida i en el que no podem actuar sobre el nom, no existeixi. Si existeix retornem la cadena d'error. En cas contrari el que fem és analitzar el primer fitxer amb l'ajuda de la classe AnalisiBMP i guardem les dades que ens interessin. Després fem el mateix amb el segon fitxer i comparem per si tenen el mateix format i el mateix fitxer original. Si tot l'anterior s'ha acomplert, fem la crida al mètode confabulaFitxer i novament fent servir la parella try –catch fem la crida al mètode creaFitxerModificat. Si tot ha anat be donarem com ha resposta fet i sinó donarem l'error que correspongui.
- Mètode - llegeixFitxer(String nomfitx,byte []dadesfitx,byte [][] dadesbmp): boolean. Com que tornem a treballar amb fitxers, fem servir com sempre el conjunt try –catch i aleshores fem una lectura de la capçalera del primer fitxer colocant-la en una matriu ja definida al començament de la classe per a aquestes funcions. Llegim un màxim de 54 dades i portem un control per saber si s'ha llegit alguna cosa o en cas contrari retornar l'error. Recull les dades que ens interessa i comença a omplir la matriu que se li ha passat com a paràmetre i a on deixarem les dades. Així doncs, el que és fa és d'un sol fitxer crear dues matrius, la primera conté la capçalera i la segona les dades. La lectura de les columnes és fa sabent que el valor de l'amplada ve donat en punts i nosaltres treballem en bytes, per tant tindrem que multiplicar aquest valor per tres, que son els colors que conformen un punt. Per coses del format BMP, l'amplada ha d'ésser multiple de 4 i si no és així, el que fa és omplir-ho de zeros que és el que s'anomena padding. En aquest mètode, aquestes dades s'obvien i no s'emmagatzemen. La de les files és llegeix amb el valor màxim que ve determinat per l'amplada vertical. Com sempre el que és fa finalment és tancar el fitxer i retornem cert si tot ha anat be, fals altrament.
- Mètode - llegeixWord(int index,int limit,byte []dadesFitx): long. Amb aquest mètode aconseguim alliberar de tasques repetitives als mètodes que l'utilitzen. És de tipus privat perquè no ens interessa que és pugui accedir des d'altres classes. El que fa és donat un punt inicial i quant valors volem llegir concreta la següent equació, havent passat com a paràmetre la matriu que ens interessa llegir,

$$long = \sum_{i=index}^{i=index+lmit} dadesFitx[i] * 256^{i-index}$$

  El fet de que treballi amb 256 és que busquem els valors per a 8 bit, sabent que 2^8=256.
- Mètode - confabulaFitxer().En aquest cas obrim les dues matrius a on hem guardat les dades dels fitxer que volem barrejar i les anem llegit per columnes i files. Per a cada punt que hem de tractar inicialitzem un comptador, si en algun dels colors que el composa hi ha variació, l'incrementem. Aleshores voldrà dir que aquest punt conté informació i que l'hem de modificar. Ho fem fent una i logica amb el darrer bit de cada component de color i un 254, afegint després de forma aleatòria un 1 o un 0. Si el punt no te cap variació, no fem res. Les dades ja confabulades és guarden a la matriu de dades del primer fitxer per estalviar espai.

- Mètode - creaFitxerModificat(). Comencem llençant una excepció perquè treballarem amb fitxers. Creem un fitxer amb el nom desitjat, que el crearem de la combinació de les dades passades com a paràmetres. Comencem escrivint la capçalera, es dir, passem les dades de la matriu que la suporta al dispositiu físic. Després fem el mateix amb les dades. Tindrem en compte que l'amplada en nombre de punts ha d'ésser múltiple de quatre i que si no ho és afegirem zeros de padding fins que ho acompleixi. Finalment és tanca el fitxer.
- Mètode - estaFitxerCreat(String nomFitxe): boolean. Instancia un fitxer amb el nom desitjat i comprova que és pugui llegir retornat cert si ha pogut, fals altrament.

7.6. Classe MarcaBMP

7.6.1. UML de la classe

MarcaBMP
- String nomFitxer,nomSegonFitxer,marca,nom; - BufferedReader entrada; - byte []dadesFitxer= new byte [1500]; - byte []dadesSegonFitxer= new byte [1500]; - byte []word= new byte[4]; - byte [][]dadesBMP= new byte[1500][4500]; - byte [][]dadesSegonBMP= new byte[1500][4500]; - byte CRLF[]={0x0d,0x0a}; - int []V0=new int[7]; - int []V1=new int[7]; - String s=""; - int offset,longitutBitmap,ampladaHoritzontal,ampladaVertical; - int offset1,longitutBitmap1; - int ampladaHoritzontal1,ampladaVertical1; - int offset2,longitutBitmap2; - int ampladaHoritzontal2,ampladaVertical2; - int longitutTrama=7,quadre=10; - byte profunditat=1; - int quadreExtraccio; - BaseDades baseDades; - boolean ferAnalisi;
+ MarcaBMP(String nomfitxer,String marc,String dada_nom,boolean bol) + MarcaBMP(String nomfitxer,String nomsegonfitxer,boolean bol) + extreuMarca():String + colocaMarca():String - llegeixFitxer(String nomfitx,byte []dadesfitx,byte [][] dadesbmp): boolean - llegeixWord(int index,int limit,byte []dadesFitx): long - codificaFitxer() - creaFitxerModificat() - llegeixMarca():String - estaFitxerCreat(String nomFitxe): boolean

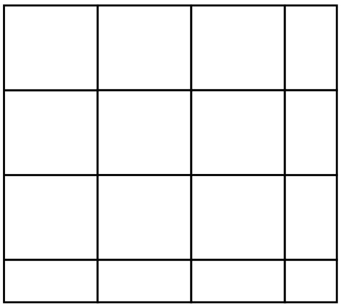
7.6.2. Funcionament

Es basa en dividir la imatge per quadres de 10x10 punts, a on inserirem les marques d'aigua que vindran definides per 7 bits utilitzant un codi dual de Hamming. S'utilitzen aquest codis perquè porten informació redundant que ens permetrà evitar certs atacs. Els codis permesos son els següents

0	0000000
1	0011011
2	0101101
3	0110110
4	1001110
5	1010101

6	1100011
7	1111000

La manera de dividir la imatge serà,



Tindrem quadres complets de 10x10 però en funció de la mida de la imatge si és o no múltiple de 10 els quadres seran de la mida estàndard o no. En els que no ho siguin també s'incorporarà la marca de la forma habitual.
Dins de cada quadre col·locarem la marca M={m1,m2,m3,m4,m5,m6,m7}de la manera següent,

m1	m2	m3	m4	m5	m6	m7	m1	m2	m3
m1	m2	m3	m4	m5	m6	m7	m1	m2	m3
m1	m2	m3	m4	m5	m6	m7	m1	m2	m3
m1	m2	m3	m4	m5	m6	m7	m1	m2	m3
m1	m2	m3	m4	m5	m6	m7	m1	m2	m3
m1	m2	m3	m4	m5	m6	m7	m1	m2	m3
m1	m2	m3	m4	m5	m6	m7	m1	m2	m3
m1	m2	m3	m4	m5	m6	m7	m1	m2	m3
m1	m2	m3	m4	m5	m6	m7	m1	m2	m3
m1	m2	m3	m4	m5	m6	m7	m1	m2	m3

Per afegir la marca, el que farem és mirar si el valor que volem inserir es 0 o 1. En cas de que sigui zero no farem res, en l'altre cas modificarem a tots els byte que defineixen un punt el bit de menys pes.
Encara que teòricament la imatge original i la modificada tindrien de tenir la mateixa mida, s'ha incorporat un control que permet de buscar la marca en quadres que siguin de diferent mida per solucionar fins on és pugui els atacs d'escalat d'imatge.
La descodificació és farà a l'inrevés utilitzant dos vectors fent un recompte per majories. És dir, tindrem un vector V0={v00, v01, v02, v03, v04, v05, v06, v07} i un vector V1={v10, v11, v12, v13, v14, v15, v16, v17}, aleshores farem un recorregut pels quadres controlant cadascun dels punts. Comprovarem els valors RGB del punt de la imatge original i de la modificada i per cada canvi que hi hagi en els valor incrementarem un comptador. Si han variat mes de 1 bit direm que és un 1, si ho han fet zero o un, direm que és un zero. En funció de la posició dins la marca incrementarem la mateixa posició dins del vector dels zeros, si és un zero, o en la dels 1 si és un 1. Finalment per a recollir el valor de la marca el que és fa es recórrer ambdós vectors i el que per a una mateixa posició tingui un número mes alt, serà zero si és el vector del zeros o a l'inrevés. Per a recuperar la identitat del confabulat, serà buscar dintre de les marques possibles la que mes s'assembli a l'obtinguda en la descodificació i utilitzar esta per fer la cerca a la BD.
Per acabar fem servir sobrecarrega en els constructors dedicant un per a la codificació i l'altre per la descodificació.

7.6.3. Mètodes

- Mètode + extreuMarca():String. El que primer farem és analitzar el primer fitxer amb l'ajuda de la classe AnalisiBMP i guardem les dades que ens interessin. Desprès fem el mateix amb el segon fitxer i comparem per si tenen el mateix format i el mateix fitxer original. Si tot l'anterior s'ha acomplert, fem la crida al mètode .cercaNomComprador

per a buscar el nom del comprador a la BD fent servir la parella try –catch. Si tot ha anat be donarem com ha resposta fet i sinó donarem l'error que correspongui retornant una cadena.

- Mètode + colocaMarca():String. Ara volem amagar la marca i el primer que cal fer és comprovar que la marca és una de les acceptades. Degut al funcionament de la interfície gràfica, aquest pas és podria obviar però ha quedat per seguretat. Com que tenim que accedir a un fitxer fem servir la parella try – catch com sempre. Ara abans de començar a procedir el que fem és comprovar a la BD que aquesta marca no estigui ja associada a aquest fitxer, perquè no permetem que és pugui repetir. Si ho està dona l'error. Aleshores el que és fa es comprovar que no existeixi cap fitxer amb aquest nom dins del mateix directori. Cal recordar que la BD emmagatzema el fitxer amb adreça en valor absolut. Fem l'anàlisi del fitxer si toca i aleshores fem la crida al mètode .codificaFitxer. Acabat això intentem crear el fitxer modificat, com sempre amb la parella try – catch i amb una parella similar intentem incorporar la informació dins la BD. Per a cada possible error en tots els passos anteriors, és retorna una cadena indicant-ho i si s'ha pogut de manera correcta també ho indica.
- Mètode - llegeixFitxer(String nomfitx,byte []dadesfitx,byte [][] dadesbmp): boolean. Com que tornem a treballar amb fitxers, fem servir com sempre el conjunt try –catch i aleshores fem una lectura de la capçalera del primer fitxer colocant-la en una matriu ja definida al començament de la classe per a aquestes funcions. Llegim un màxim de 54 dades i portem un control per saber si s'ha llegit alguna cosa o en cas contrari retornar l'error. Recull les dades que ens interessa i comença a omplir la matriu que se li ha passat com a paràmetre i a on deixarem les dades. Així doncs, el que és fa és d'un sol fitxer crear dues matrius, la primera conté la capçalera i la segona les dades. La lectura de les columnes és fa sabent que el valor de l'amplada ve donat en punts i nosaltres treballem en bytes, per tant tindrem que multiplicar aquest valor per tres, que son els colors que conformen un punt. Per coses del format BMP, l'amplada ha d'ésser multiple de 4 i si no és així, el que fa és omplir-ho de zeros que és el que s'anomena padding. En aquest mètode, aquestes dades s'obvien i no s'emmagatzemen. La de les files és llegeix amb el valor màxim que ve determinat per l'amplada vertical. Com sempre el que és fa finalment és tancar el fitxer i retornem cert si tot ha anat be, fals altrament.
- Mètode - llegeixWord(int index,int limit,byte []dadesFitx): long. Amb aquest mètode aconseguim alliberar de tasques repetitives als mètodes que l'utilitzen. És de tipus privat perquè no ens interessa que és pugui accedir des d'altres classes. El que fa és donat un punt inicial i quant valors volem llegir concreta la següent equació, havent passat com a paràmetre la matriu que ens interessa llegir,

$$long = \sum_{i=index}^{i=index+lmit} dadesFitx[i] * 256^{i-index}$$

El fet de que treballi amb 256 és que busquem els valors per a 8 bit, sabent que 2^8=256.

- Mètode - codificaFitxer(). En aquest mètode el que fem és introduir la marca dins del fitxer, tractem la matriu per files i per columnes, essent aquestes ultimes les més complicades perquè porten el control de quadre. Per a fer-ho dins de cada seguiment de línia, dividim per quadres i en cada posició del quadre incorporem el valor de la marca que li pertoca. Si és un 1 modifiquem els tres byte RGB i sinó ho és no fem res. El programa a nivell de codi permet col·locar la marca en el bit que ens interessi però per defecte ho fem en el de menys pes. No és pot accedir a través de la interfície a aquesta funció. Com que en un quadre i cap mes d'una marca, el que fem és començar de nou quan calgui fins que completem una línia de quadre. Això és fa fins a completar tots els quadres d'una línia i després la resta de línies de la matriu.
- Mètode - creaFitxerModificat(). Comencem llençant una excepció perquè treballarem amb fitxers. Creem un fitxer amb el nom desitjat, que el crearem de la combinació de les dades passades com a paràmetres. Comencem escrivint la capçalera, es dir, passem les dades de la matriu que la suporta al dispositiu físic. Després fem el mateix amb les dades. Tindrem en compte que l'amplada en nombre de punts ha d'ésser múltiple de quatre i que si no ho és afegirem zeros de padding fins que ho acompleixi. Finalment és tanca el fitxer.

- Mètode - llegeixMarca():String D'una manera similar a la del mètode de codificació, fem l'extracció. Primer fem un càlcul per a trobar el quadre del que farem l'extracció en funció de la mida original i de la mida del fitxer codificat. Això és fa pels possibles escalats de la imatge codificada. Aleshores, és fa la comparació punt a punt de la imatge. Si dels components de punt trobem que dos o tots tres estan modificats, en la posició que estem treballant del vector d'uns incrementem una unitat, si és menor ho fem en el vector dels zeros. Com que novament els quadres son més grans que les marques en quant a posicions tindrem de inicialitzar el comptador de posició de marca per a poder seguir llegint el quadre i finalment tota la línia. Ara el que cal fer és llegint el vector de uns o de zeros determinar per a cada posició qui te el número mes gran de coincidències i aconseguir la marca llegida. Trobada aquesta el que farem és comparar-la amb les vuit possibles marques i la que mes a prop és trobi numèricament donar-la com a resultat. Sinó és trobés cap donaríem un error
- Mètode - estaFitxerCreat(String nomFitxe): boolean. Instancia un fitxer amb el nom desitjat i comprova que és pugui llegir retornat cert si ha pogut, fals altrament.

7.7. Classe Tfc

7.7.1. UML de la classe

Tfc
- Frame finestra; - Button codifica,descodifica,confabula,ok; - Button disco1,disco2,disco3,disco4,disco5; - Label valor1,valor2,valor3,valor4,valor5,valor6; - Panel panel1,panel2,panel3,panel4,panel5,panel6,panel7,panel8,panel20; - TextField text1,text2,text3,text4,text5,text6; - Choice choice2; - String marca="1010101",onSom; - String nomFitxer="BaseDeDades.txt"; - FileDialog fitxer1,fitxer2; - PopupMenu alertes; - MenuBar barraMenu; - Menu esborraBD,ajuda,configura; - MenuItem autor,si,no; - CheckboxMenuItem analisi; - String comprador[]={"0000000","0011011","0101101","0110110","1001110","1010101", "1100011","1111000"};
+ Tfc() - creaFitxer() - creaFinestra() + actionPerformed(ActionEvent e) + getNomFitxer():String - menuBar() + getComprador(int valor): String + getAnalisi():boolean + windowClosing(WindowEvent e) + windowActivated(WindowEvent e) + windowClosed(WindowEvent e) + windowDeactivated(WindowEvent e) + windowDeiconified(WindowEvent e) + windowIconified(WindowEvent e) + windowOpened(WindowEvent e)

7.7.2. Funcionament

La classe Tfc és la que s'encarrega de gestionar la interfície gràfica i també d'incorporar en aquest cas el main de l'aplicació. El que fa és crear les finestres i els controls que determinaran el funcionament correcta dels sistema, a mes de ser la dipositaria de la gran majoria de dades

per al correcta funcionament degut a que és la que porta la relació amb l'usuari. Les finestres de treball compliran la disposició de la pagina següent

L'única variació serà en la zona central a on variaran els paràmetres a introduir en funció del que s'estigui fent.

<table>
<tr><td colspan="4">Nom finestra</td></tr>
<tr><td colspan="4">Barra de menú</td></tr>
<tr><td>Botó 1</td><td>Botó 2</td><td colspan="2">Botó 3</td></tr>
<tr><td>Etiquetes</td><td colspan="2">Dades a introduir</td><td>Botó de cerca</td></tr>
<tr><td colspan="4">Botó d'actuació</td></tr>
</table>

7.7.3. Mètodes

- Mètode - creaFitxer(). Comprova que la BD estigui creada, sinó hi és la crea
- Mètode - creaFinestra(). Crea la finestra amb mides, barra de menú i control d'events. Per a fer-ho és crea una frame amb nom, mida i posició determinada i se l'incorpora el control. Posteriorment és fa una crida al mètode menuBar(), que serà l'encarregat de la gestió de la barra de menús. Després el que és fa és crear tots els pannells necessaris per el funcionament de les diferents combinacions de finestres. Tots tenen el format de GridLayout, és dir que la disposició dels elements restarà com si fos una taula. Mes endavant és creen els botons i se'ls afegeixen els listeners per al seu control. Seguim creant coses com serien les etiquetes i els TextField per al final afegir tot al pannell que li pertoca per aconseguir la seva funció. Inicialitzem la variable onSom per indicar que la primera finestra que apareixerà serà la de codificar i donem els últims retocs a la finestra, fent-la que no és pugui ampliar, ajustant els marges al contingut i fent-la visible.
- Mètode + actionPerformed(ActionEvent e) El que fa és portar el control de quin botó s'ha premut. Podem definir tres funcions per als botons, d'execució quan el botó del qual parlem és el de Ok, de visualització quan és tracta dels botons de codificar, descodificar o confabular i de cerca quan el que volem és agafar el valor de l'adreça del fitxer. El que primer comprovem és que estiguem parlant del botó Ok. En funció de la variable onSom farem una o altra cosa mostrant al textField6 el resultat de les operacions i actualitzant els textField necessaris. També és fa aquí la comprovació que no falti cap dada per a l'operació desitjada. Si parlem d'un botó de visualització modifiquem la finestra, mentre que si parlem d'un boto de cerca fem una instanciació a un FileDialog per a recollir la dada. Un altre tipus de botó del que no s'ha parlat fins ara és que és pot trobar a la barra de menús. Aqui trobarem els botons d'autor del TFC i de Si que el que generaran son un altre tipus de finestres que son els popUp. El de Si seria un segon pas en la confirmació de l'esborrat de la BD per assegurar que no és un error.
- Mètode + getNomFitxer():String El que retorna és el nom del fitxer de la base de dades perquè sigui comú a totes les classe
- Mètode - menuBar() Te un funcionament molt similar al cas de la creació de la finestra però en aquest cas els elements dels quals estem parlant son menu, menuItem i menuBar
- Mètode + getComprador(int valor): String Retorna la cadena associada a un valor numèric.
- Mètode + getAnalisi():boolean. Comprova que dins del menú de configuració, l'opció Fer anàlisi estigui o no activada, per a que en la resta de classes llancin la crida a la classe AnalisiBMP

- Mètode + windowClosing(WindowEvent e). Controla l'event de la finestra quan s'està tancant. En aquest cas tanquem l'aplicació.
- Mètode + windowActivated(WindowEvent e). Controla l'event de la finestra quan és activada. Aquí és posa per a poder implementar la classe WindowListener
- Mètode + windowClosed(WindowEvent e). Controla l'event de la finestra quan està tancada. Aquí és posa per a poder implementar la classe WindowListener
- Mètode + windowDeactivated(WindowEvent e). Controla l'event de la finestra quan és desactivada. Aquí és posa per a poder implementar la classe WindowListener
- Mètode + windowDeiconified(WindowEvent e). Controla l'event de la finestra quan no està minimitzada. Aquí és posa per a poder implementar la classe WindowListener
- Mètode + windowIconified(WindowEvent e). Controla l'event de la finestra quan està minimitzada. Aquí és posa per a poder implementar la classe WindowListener
- Mètode + windowOpened(WindowEvent e). Controla l'event de la finestra quan està oberta. Aquí és posa per a poder implementar la classe WindowListener

8. Atac suportats

Aquí farem una descripció de les proves fetes a fi i efecte de comprovar el bon rendiment del sistema.

8.1. Atac de mosaic.

Agafem la imatge següent,

Es tracta d'una carta d'ajust senzilla i que serà sobre la qual farem les proves inicials. La informació que el programa ens dona sobre ella és la següent,

- Nom del fitxer: cartaAjust.bmp
- Identificador: BM
- Longitud del fitxer: 14894 bytes
- Offset fins a les dades del bitmap: 54 bytes
- Longitud de la capçalera bitmap: 40 bytes
- Amplada horitzontal del bitmap: 70 punts
- Amplada vertical del bitmap: 70 punts
- Numero de plans del bitmap: 1
- Numero de bits per punt: 24
- Especificació de la compressió: 0
- Mida del bitmap: 14840 bytes
- Resolució horitzontal: 0 punts per metre
- Resolució vertical: 0 punts per metre
- Nombre de colors utilitzats en el bitmap: 0
- Nombre de colors importants utilitzats en el bitmap: 0

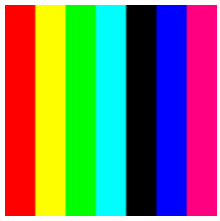
Per tant fem quatre números tindrem 49 quadres a llegir, es dir que la imatge és de 7x7 quadres o be de 70x70 punts.

Ara agafem els dos quadres següents,

El primer serà quadrat de NxN punts amb inici al zero. Això suposa que podem llegir la marca perquè podem sincronitzar la lectura, però no és així perquè el sistema està fet per resistir atacs d'escalat i no de mosaic. Una manera d'aconseguir l'obtenció de la marca seria retallar l'original amb la mateixa mida tant vertical com horitzontal i comparar-la. Aleshores segur que és podria extreure la marca.
Amb la segona imatge seria impossible perquè no ens podríem sincronitzar amb l'inici, tret que per casualitat el punt de tall coincidís amb un inici de quadre.
Si el atac de mosaic redueix la mida de la imatge a un valor inferior a la del quadre, tan sols podríem recuperar-la si aquesta mida és igual o superior a 7x7 punts i l'inici coincideix amb un inici de quadre. Novament tindríem que tallar la imatge per a poder comparar-les.

8.2. Atac d'escalat de imatge

Ara escalem la imatge augmentant-la en un 50%, quedant així



Donant com a resultat el nom del propietari de forma correcta. Fem el mateix reduint la imatge fins al 90% sempre dona el resultat independent de la marca. Ara agafem una imatge com lena.bmp

i obtenim que ampliant o reduint és indetectable. Una imatge com peppers.bmp, suporta la reducció al 90% però no l'ampliació

Per tant podem suposar que depèn molt de l'algorisme d'ampliació o reducció la forma en que és modifica la imatge. De fet la generació del codi s'ha basat en la primera imatge perquè és la que ha permès identificar de millor manera el sistema d'escalat, però que poder sigui molt diferent en funció del tipus d'imatge a modificar.

8.3. Atac de distorsió

Ja veiem el que succeirà només mirant com quedaria la imatge per a 10º de distorsió,

Es impossible que puguem trobar el sincronisme de la marca i per tant extreure- la. Si amb la mateixa imatge la tornem a endreçar la marca ha desaparegut.

8.4. Atac de compressió

Fent el mateix podem comprovar que la imatge que sembla una carta d'ajust suporta el canvi de format de BMP a JPG del 95% i tornar a passar a BMP sense problemes. Altres valors fan indetectable la marca. Depenent de la imatge arriben fins al 90% de compressió però mai mes enllà. Lena.bmp no suporta cap tipus de compressió

8.5. Atac de confabulació

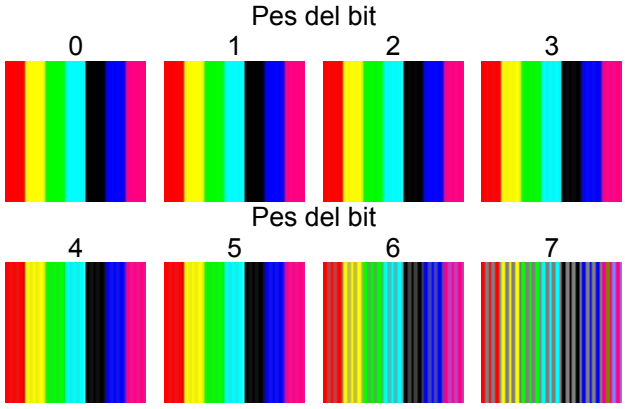
La utilització de codis de Hamming limitats a 8 possibles valors ha permès que cap dels atacs de confabulació, hagi reeixit perquè s'ha pogut extreure sempre un dels noms dels responsables de l'atac.

8.6. Taula resum

Atac	**Resultat**
Mosaic amb referència zero	Recuperable si retallem la imatge original amb les mides del tros per a poder comparar
Mosaic amb referència no zero	Si coincideix l'inici de la imatge amb inici de quadre si. En la resta de casos no pot recuperar.
Escalat	Depèn del tipus de imatge. Imatges senzilles gairebé totes suporten el 90% d'augment o disminució, en alguns casos fins al 50 % d'augment. En imatges complexes suporta la reducció en alguns casos al 90%, no l'augment
Distorsió	No el suporta en cap prova feta
Compressió	El mateix que en l'escalat, depèn de si la imatge és o no senzilla. Quan el suporta arriba al 90% de compressió i en algun cas concret fins al 80%
Confabulació	En tots els casos provats ha detectat el nom d'un dels compradors

8.7. Relació robustesa - distorsió respecte de l'atac

Codificarem la marca 0011011 dintre de la mateixa imatge modificant el pes del bit a on s'incorpora



Com és pot comprovar i degut a la manera de marcar que és fa de manera gairebé lineal en funció de l'amplitud de la imatge, a mida que anem modificant la posició, comencen a aparèixer una sèrie de ratlles que denoten a on és troba la marca i degut a la manera de marcar, fins i tot el valor. Per descomptat si opten per la d'introduir la informació al bit de mes pes, segur que resisteix molts millor els atacs que el que s'ha explicat en els altres apartats, però la distorsió és totalment inacceptable. D'altre banda, la marca no és gens visible, però la resistència als atacs és molt limitada.
També hem de tenir en compte que tot el que hem fet ha estan en el domini espacial i no en el de la transformada, que segur que ens hagués permès suportar millor els atacs.

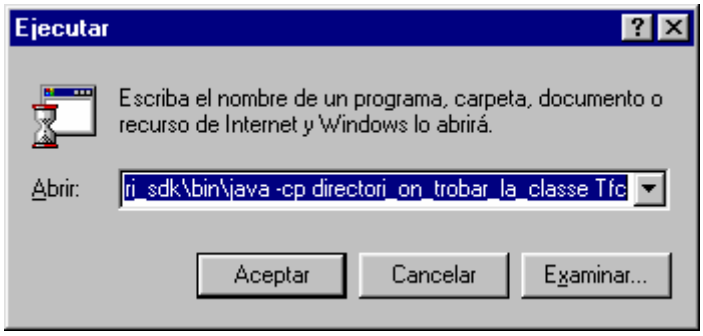
9. Manual d'usuari

9.1. Com executar.

Ho podem fer de dues maneres, o be ho fem obrim una pantalla de MS-DOS i escrivim el següent,

Directori_sdk\bin>java -cp directori_on_trobar_la_classe Tfc

O be entrant a l'opció d'executar de Windows i fer el mateix



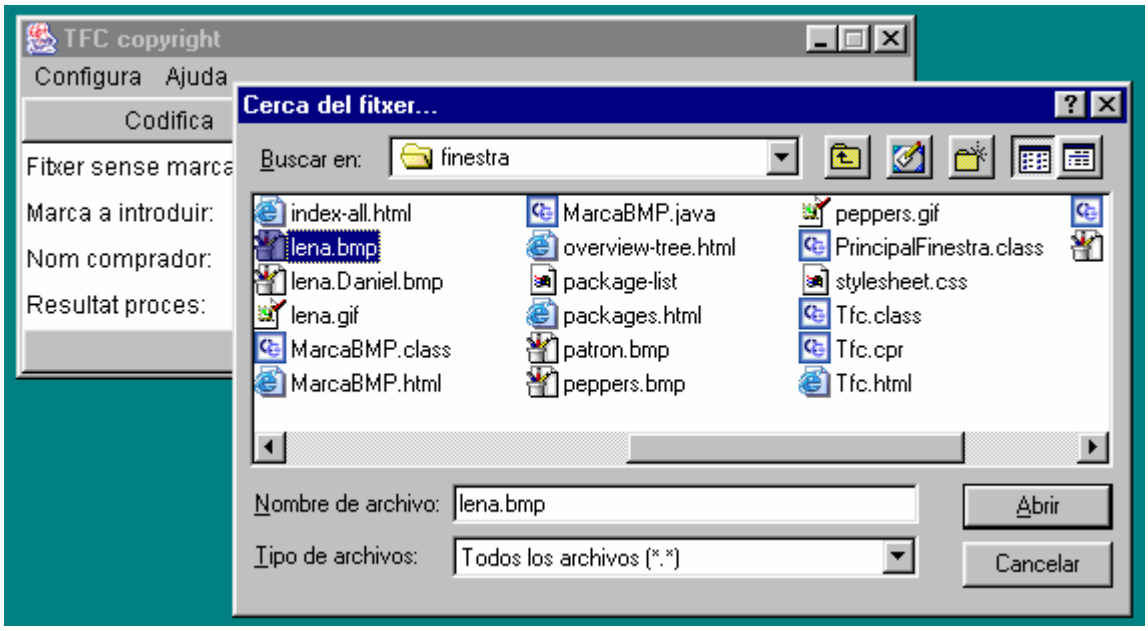
9.2. Codificació d'un fitxer.

Un cop executat el programa la pantalla que se'ns mostrarà serà la següent,

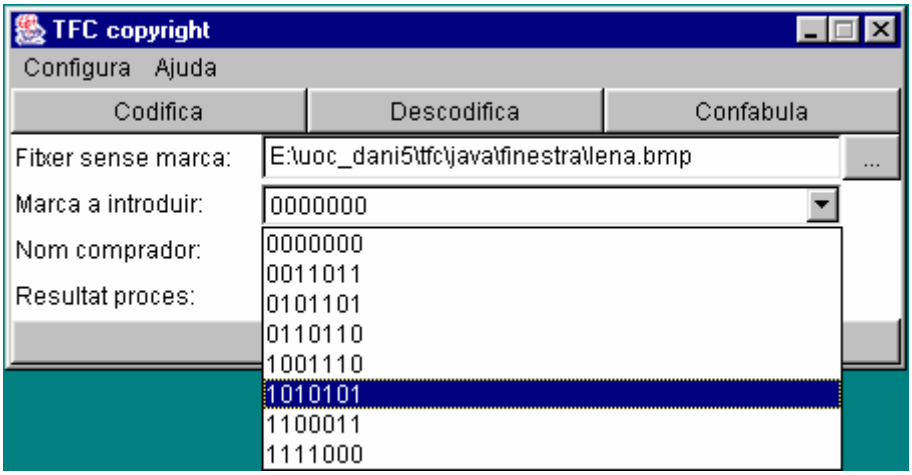
Aquesta és la pantalla de codificació. Per a codificar cal un fitxer, una marca i un nom de comprador per a poder barrejar-ho i si és pot fer, les dades son emmagatzemades per a posteriors processos.

Per fer la codificació d'un fitxer, tenim que omplir els següents camps,

- Camp Fitxer sense marca. Aquí tindrem que introduir l'adreça del fitxer a codificar o be de forma manual o be amb el botó lateral que permet accedir al menú habitual de cerca de fitxer i aleshores seleccionar-lo, com és pot veure en el dibuix següent,



- Camp Marca a introduir. El següent pas serà escollir la marca obrint el desplegable que acompanya a l'etiqueta marca a introduir,



- Camp Nom comprador. Omplint el nom de comprador com a últim pas podem prémer el botó Ok per a començar a processar les dades.

- Camp Resultat procés. En tot moment en la casella resultat del procés apareixen missatges d'informació dient que s'està fent i si s'ha aconseguit o no l'acció.

9.3. Descodificació d'un fitxer.

Per accedir a la pantalla de descodificació només cal prémer el botó que ho indica i apareixerà la pantalla següent,

Un fitxer que creiem marcat, per a poder processar-lo necessiten l'ajuda del fitxer original i la base de dades que és la que porta el registre físic de les dades. Aleshores conegut això, podem omplir els camps necessaris per aconseguir-ho,

- Camp Fitxer sense marca. Aquí tindrem que introduir l'adreça del fitxer a codificar o be de forma manual o be amb el botó lateral que permet accedir al menú habitual de cerca de fitxer com s'ha explicat en l'apartat anterior.
- Camp Fitxer amb marca 1. És procedeix a omplir els camps com s'ha explicat en altres apartats o be manual o per selecció directa. Ara ja podem prémer el botó Ok per a fer el processament.
- Camp Resultat procés. En tot moment en la casella resultat del procés apareixen missatges d'informació dient que s'està fent i si s'ha aconseguit o no l'acció.

9.4. Confabulació d'un fitxer.

Com sempre per a accedir a aquesta pantalla cal prémer el botó confabula i apareix el següent,

Per a confabular un fitxer calen dos fitxers marcats i aleshores el sistema s'encarrega de generar un tercer amb la barreja dels dos.

Aleshores cal anar omplint camps de la següent manera,

- Camp Fitxer amb marca 1. És procedeix a omplir els camps com s'ha explicat en altres apartats o be manual o per selecció directa.
- Camp Fitxer amb marca 2. S'omple com sempre i ara ja podem prémer el botó Ok per a fer el processament.
- Camp Resultat procés. Com sempre en la casella resultat del procés apareixen missatges d'informació dient que s'està fent i si s'ha aconseguit o no l'acció.

9.5. Barra de menús.

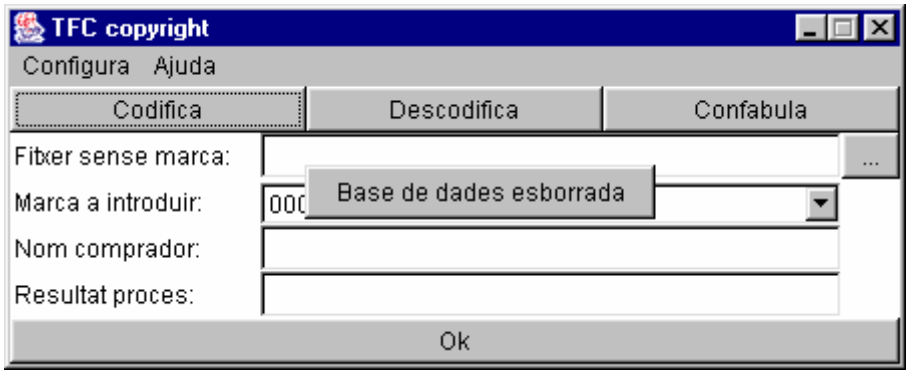
És la zona que queda definida pel nom de la finestra i els botons de selecció de pantalla. Podem fer dues coses seleccionar, configura i ajuda

9.5.1. Menú Configura

La pantalla següent ens mostra el que apareix,

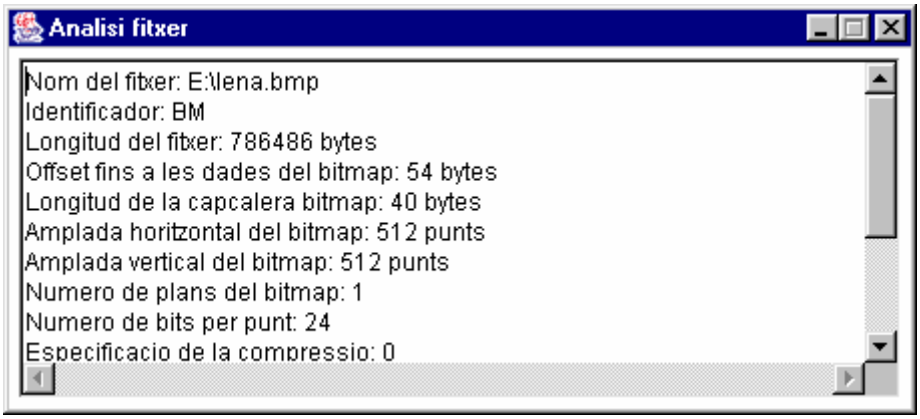
En primer lloc podem esborrar la BD. Com que es un procés molt important, ens hem d'assegurar que no és pot fer de manera directa. Una manera de fer-ho seria com és veu a la pantalla següent,

Per a poder esborrar de forma voluntària hem de moure el ratolí i seleccionar una possibilitat que per moviment directa no s'hi accedeix. Si finalment ho fem apareixerà un missatge de confirmació com és el següent,



La segona opció es Fer anàlisi.

Per defecte esta marcada sempre quan arranca el programa i es pot anular. El que decidim és que quan es processa un fitxer BMP és faci el seu anàlisi o no, si ho fem sortirà una finestra com la següent,



9.5.2. Menú Ajuda

L'únic que fa és mostrar el nom de l'autor del projecte. Quan és selecciona novament apareix un menú com el següent,

Prement el botó apareix el següent,

9.6. Automatismes.

Com que la finalitat del TFC és l'estudi de les marques d'aigua si fem una codificació de manera correcta, automàticament s'ompliran els camps que necessaris per a la descodificació i així anar mes ràpida la comprovació de la marca del fitxer. I omple la primera casella per a poder confabular si és vol.
En el cas de que treballem amb confabula podem després anar a descodificar i hi haurà alguns camps omplerts.

9.7. Missatges d'error.

9.7.1. Missatges en la codificació

- Marca no admesa. Marca no admesa en el procés
- Marca de fitxer associada a un altre comprador. La marca ja esta en la base de dades associada a un altre comprador
- Error a l'arxiu de base de dades. S'han succeït errors en el procés de lectura del fitxer de la base de dades
- Nom de fitxer ja associat a una altre marca. Aquest fitxer ja esta creat en el directori en el que s'està treballant.
- Error en la creació arxiu modificat. S'han succeït errors en el procés de lectura o escriptura del fitxer codificat
- Procés realitzat. S'ha pogut realitzar el procés
- Procés no realitzat. No s'ha pogut realitzar el procés
- Error al escriure dades al fitxer de base de dades. S'han succeït errors en el procés de escriptura del fitxer de la base de dades
- Error en l'accés del arxiu. No s'ha pogut fer l'anàlisi del fitxer.

9.7.2. Missatges en la descodificació

- Error a l'arxiu de base de dades. S'han succeït errors en el procés de lectura del fitxer de la base de dades
- Error en l'accés del arxiu. No s'ha pogut fer l'anàlisi del fitxer.
- Fitxers amb fitxer original diferent. No és pot fer la comparació perquè parlem de fitxers diferents.
- Propietari desconegut. Marca trobada: xxxxxxxx. Quan és fa una descodificació i no és troba el propietari a la base de dades, retorna la marca llegida

9.7.3. Missatges en la confabulació

- Nom de fitxer ja associat a una altre marca. Aquest fitxer ja esta creat en el directori en el que s'està treballant.
- Problema amb la creació de l'arxiu. S'han succeït errors en el procés de lectura o escriptura del fitxer confabulat
- Fitxers amb fitxer original diferent. No és pot fer la comparació perquè parlem de fitxers diferents.
- Error en l'accés del arxiu. No s'ha pogut fer l'anàlisi del fitxer.
- Fet. S'ha pogut realitzar el procés

9.7.4. Missatges generals

- Treballant... Apareix quan és fan processos que necessiten temps
- Falten dades Cal omplir tots els camps per a realitzar el procés

10. Conclusions

Aconseguir que una imatge quedi marcada és una tasca senzilla i a l'abast de qualsevol persona amb coneixements de programació. Que aquesta marca suporti el tracta habitual que qualsevol persona realitza de manera usual amb la imatge, compressió, canvi de mida o de format és una altra cosa. Aquestes modificacions poden ésser titulades com atacs, però en la majoria de cops no ho son perquè son manipulacions habituals que esborren o anul·len la marca. Si amb aquestes manipulacions aconseguim esborrar la marca que no podrà fer algú amb els coneixements i temps necessari.
L'increment d'usuaris i consumidors d'informació digital fa preveure que aquest camp serà un dels quals se'ls hi dedicaran mes recursos perquè tota la indústria de mitjans audiovisuals hi esta al darrera. El problema com sempre és que la legislació en els camps tècnics esta molt endarrerida i quan actua sobre un determinat punt, aquest esta en la seva vellesa i d'altres estan ja emergint com a noves propostes.
Malgrat tot el pitjor per a aquestes tècniques és que els usuaris professionals que els podrien fer servir mes extensament, no s'hi volen abocar perquè de moment totes les marques d'aigua modifiquen lleugerament els originals i molts d'ells prefereixen que les seves obres quedin tal i com és van crear.

11. Glossari

A – B

BBP	Abreviatura que significa bits per punt
Benchmarks	Conjunt de tests que permeten determinar una sèrie de prestacions del sistema a provar.
BMP	Format d'imatge definit per IBM i Microsoft, es pot dir d'altra manera mapa de bits o en anglès bitmap

C- D

Clau.	Permeten d'encriptar i desencriptar les dades de manera que encara que la informació pugui ésser llegida, no podrà interpretar-se sense la clau.
Codi Hamming	Codis binaris redundants que presenten entre ells equidistància numèrica.
Compressió	Reducció de la mida d'un fitxer per a poder fer mes eficaç la seva transmissió o emmagatzemat. Permet el pas invers sense pèrdues d'informació significativa d'informació però que pot afectar negativament a un fitxer marcat.
Confabulació.	Acte que dos o mes persones fan en conjunt per a actuar de forma negativa en contra d'altres. En aquest cas és quan dos atacants utilitzen les seves imatges marcades de forma individual per a generar una tercera independentment del propietari original
Constructor	És el responsable que una classe és concreti per a poder treballar amb ella.
Copyright	Paraula anglesa que es tradueix com a drets d'autor
Distorsió	Modificació en la imatge que permet eliminar la marca en segons quins atacs. Llindar que determina quan una imatge és o no perceptible.
DNS	Servei de dominis. Tradueix les adreces per text a valors numèrics comprensibles pels servidors

E – F

Escalat	Modificació de la imatge en la seva mida. Podem fer-la o be mes gran o mes petita.
Esteganografia	Sistema de enviar dades amagades entre dos punts sense que ningú sàpiga d'antuvi que hi son incorporades
FileDialog	Classe que genera una finestra per al tractament de fitxers de forma estàndard
Fractal	Sistema de codificació de la informació.
Frame	Cadascun dels plans estàtics que defineixen una imatge de vídeo

G - H

Gantt	Mètode de planificació de tasques

I - J

Imperceptibilitat.	Totes les modificacions que comporta la introducció de la marca d'aigua han d'estar per sota d'un llindar que faci que sigui perceptible
Interfície	Manera com és relaciona el programa amb el usuari humà
JPEG	Format de compressió de imatges

K - L

Lena.bmp	Imatge clàssica en el tractament de imatge

M - N

Marca d'aigua	Sistema que només és pot examinar sota certes condicions que assegura alguna particularitat d'allò al que esta incorporat.
Mètode	Forma d'accedir a una classe.
Mosaic	Atac a una imatge que consisteix en dividir-la en trossos mes petits i aleshores col·locar-les juntes perquè semblin la imatge sencera
MPEG	Format de compressió de vídeo

O - P

Peppers.bmp	Imatge clàssica en el tractament de imatge
Pert	Mètode de planificació de tasques
PopUp	Tipus d'interfície que mostra les finestres com elements emergents dins la pantalla

Q - R

Redundància.	Per assegurar la robustesa la marca d'aigua és redundant i està distribuïda, permetent que només amb una part de les dades originals, aquesta pugui ésser recuperada.
Resistència	En marques d'aigua és un sinònim de robustesa
RGB	Format en el que s'emmagatzema la informació de un punt. Es donen valors a cadascun del components de color (RGB o vermell, verd i blau) que defineixen un punt
Robustesa	Resistència de la marca d'aigua a desaparèixer davant de qualsevol atac.
Roy	Mètode de planificació de tasques

S - T

Signatura	Clau incorporada que per a un tercer no significa res, però per al signant assegura que allò és o ha estat propietat seva.
SNR	Signal to noise ratio o el que és el mateix, relació senyal envers soroll
TDC	Transformada discreta del cosinus. Mètode matemàtic per al tractament de senyals. Aplicable a les marques d'aigua si pensem en estes com a senyals que és modulen a sobre d'unes altres (la imatge).
TDF	Transformada discreta de Fourier. Mètode matemàtic per al tractament de senyals. Aplicable a les marques d'aigua si pensem en estes com a senyals que és modulen a sobre d'unes altres (la imatge).

U - V

UML	Sistema de representació unificada dels sistemes orientats a objectes.

W – X

Wavelet	Transformada wavelet. Mètode matemàtic per al tractament de senyals. Aplicable a les marques d'aigua si pensem en estes com a senyals que és modulen a sobre d'unes altres (la imatge).
Webmaster	Persona responsable de una Web

Y - Z

---

12. Bibliografia

Aquests son els materials sobre els que s'ha treballat,

B. Schneier, Applied Cryptography. (2nd Edition). New York: Wiley, 1996.

Alfred J. Menezes, Paul C. van Oorschot i Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1999. http://cacr.math.uwaterloo.ca/hac/

Oppliger, R. Security technologies for the Word Wide Web. Artech House. 2000
Katzenbeisser, S. And Petitcolas, F.A.P. Information Hiding: Techniques for Steganography and Digital Watermarking. Artech House. 2000

Carlisle Adams, Steve Lloyd. Understanding PKI: Concepts, Standards, and Deployment Considerations. (2nd Edition). Addison Wesley Professional. 2002.

I tota la informació trobada a Internet relacionada amb el tema.

De visita obligada per a tots aquells que els interessi el tema

http://www.petitcolas.net/

Sobre marques d'aigua

http://www.petitcolas.net/fabien/watermarking/index.html

Sobre Benchmarks,

http://www.petitcolas.net/fabien/watermarking/stirmark/

Sobre esteganografia

http://members.lycos.nl/elhocicon/esteganografia/esteganografia.html

Sobre legislació en diferents països

http://rechten.uvt.nl/koops/cryptolaw/index.htm

Sobre codi Hamming

http://web.usna.navy.mil/~wdj/codes.html

Imatges per a fer proves i considerades estàndard en el tractament de imatges

Lena. Courtesy of the Signal and Image Processing Institute at the University of Southern California.

http://www.petitcolas.net/fabien/watermarking/image_database/lena.jpg

Peppers. Courtesy of the Signal and Image Processing Institute at the University of Southern California

http://www.petitcolas.net/fabien/watermarking/image_database/peppers.jpg

Pocket Watch on a Gold Chain. Copyright image courtesy of Kevin Odhner

(jko@home.com)http://www.petitcolas.net/fabien/watermarking/image_database/watch.jpg

Protecció del copyright

13. Annex 1: El format BMP

Daniel Mulero Roig
ETIS

Antoni Martínez Ballesté

18/06/04

13.1. Introducció

Aquest document descriu els fitxer d'imatges del tipus bitmap (BMP) de Microsoft Windows i de IBM OS/2. La descripció d'aquest tipus de fitxer és farà a nivell de byte.

13.2. El format de fitxer bitmap

Els següents apartats contenen informació detallada dels continguts dels fitxers BMP.

13.2.1. Generalitats

El format BMP va ésser creat per Microsoft i IBM en funció del hardware suportat per ambdós companyies a on anava dirigit, el PC compatible de IBM. Això comporta que tots els valors emmagatzemats en un fitxer BMP, ho son en el format de Intel, també anomenat Little Endian.
Els fitxers BMP son la forma que Windows emmagatzema les imatges de mapa de bits. Així la imatges esta empaquetada a nivell de bit, però les línies ho estan a nivell de doble paraula (4 byte) i per tant si numèricament no s'hi arriba, s'omple de bytes amb contingut zero fins que ho aconseguim. (Padding).
El format BMP s'ha concretat en quatre implementacions, dos sota Windows(vella i nova) i dos sota OS/2.

13.2.2. Contingut d'un fitxer BMP

La següent taula mostra el contingut d'un fitxer BMP

Offset	**Camp**	**Mida**	**Contingut**
0000h	Identificador	2 byte	Els dos caràcters identifiquen el BMP. Els següents valors son possibles, ▪ BM Windows 3.1x, 9x, NT,... ▪ BA Matriu BMP OS/2 ▪ CI Icona de color OS/2 ▪ CP Punter de color OS/2 ▪ IC Icona OS/2 ▪ PT Punter OS/2
0002h	Mida del fitxer	4 byte	Mida completa del fitxer en bytes
0006h	Reservat	4 byte	Reservat per usos futurs
000Ah	Offset de les dades del bitmap	4 byte	Offset des del començament del fitxer fins a l'inici de les dades del bitmap.
000Eh	Mida de la capçalera del BMP.	4 byte	Longitud de la capçalera del BMP on descrivim els colors, la compressió,.. Els següents valors son possibles, ▪ 28h Windows 3.1x, 9x, NT,... ▪ 0Ch OS/2 1.x ▪ F0h OS/2 2.x
0012h	Ample	4 byte	Mida horitzontal del BMP en punts
0016h	Alçada	4 byte	Mida vertical del BMP en punts
001Ah	Plans	2 byte	Número de plans del BMP
001Ch	Bits per punt	2 byte	Número de bits utilitzat per guardar la informació de la paleta de colors. Els valors possibles son, ▪ 1 Monocrom ▪ 4 16 colors ▪ 8 256 colors ▪ 16 16 bit (high color) ▪ 24 24 bit (true color) ▪ 32 32 bit (true color)
001Eh	Compressió	4 byte	Especificacions de la compressió. Els valors possibles son, ▪ 0 Cap (BI_RGB) ▪ 1 RLE 8-bit/punt (BI_RLE8) ▪ 2 RLE 4-bit/punt (BI_RLE4) ▪ 3 Bitfields (BI_BITFIELDS)
0022h	Mida de les dades del BMP	4 byte	Mida de les dades del BMP en bytes. El número de bytes s'arrodoneix a múltiple de 4

0026h	Resolució horitzontal	4 byte	Resolució horitzontal expressada en punt per metre
002Ah	Resolució vertical	4 byte	Resolució vertical expressada en punt per metre
002Eh	Colors	4 byte	Número de colors usats pel BMP. Si és 8 bit son 256 o 100h
0032h	Colors importants	4 byte	Número de colors importants. Aquest serà igual al número de colors quan tots els colors siguin importants.
0036h	Paleta	N*4 byte	Especificacions de la paleta. Per cada entrada de la paleta és fan servir 4 byte, un pel vermell, un pel verd i un pel blau. L'altre és zero per acomplir que el número sigui múltiple de 4
0436h	Dades BMP	x byte	Depenent de les especificacions de la compressió, aquest camp conté totes les dades del BMP els quals representen els índex de la paleta de color

13.3. Els camps en detall

13.3.1. El camp alçada

Aquest canvi identifica l'alçada del BMP en punts. En altres paraules descriu el nombre de línies del BMP. Si aquest camp es negatiu indicant un top - down DIB, el camp compressió deu ser o bé BI_RGB o BI_BITFIELDS. El top - down DIB no pot estar comprimit.

13.3.2. El camp bits per punt. (BPP)

Aquest camp defineix el número de bits per punt i el número màxim de colors en el BMP.

- Quan aquest camp val 1

El BMP es monocrom i la paleta només te dues entrades. Cada bit del BMP representa un punt, si el bit es 0 el punt es mostra en el color de la primera entrada de la paleta, si es 1 amb la segona entrada.

- Quan aquest camp val 4

El BMP té un màxim de 16 colors i la paleta conté fins a 16 colors. Cada punt del BMP està representat per un índex de 4 bits dintre de la paleta, per tant amb un byte definim dos punts. El primer punt conté el color en la entrada de la segona paleta i el segon punt conté el color en la setzena entrada de la paleta.

- Quan aquest camp val 8

El BMP té un màxim de 256 colors i la paleta conté fins a 256 entrades, en aquest cas cada byte de BMP representa un punt.

- Quan aquest camp val 16

El BMP té un màxim de 2^{16} colors. Si el camp compressió del BMP té el valor BI_RGB, el camp de la paleta no té cap entrada, cada paraula en el BMP representa un punt, la intensitat relativa del RGB està representat per 5 bits per cada component de color, el blau es el menys significatiu seguit pel verd i el vermell. El mes significatiu no s'utilitza. Si el camp compressió del BMP val BI_BITFIELDS la paleta conté 3 màscares de color de doble paraula que especifiquen el RGB per cada punt. Cada paraula en el BMP representa un sol punt.

Especificació pel Windows NT

Quan el camp compressió val BI_BITFIELDS els bits posats a 1 en cada màscara de doble paraula tenen que ser continus i no deurien sobreposar-se als bits d'un altre màscara. No tots el bits del punt tenen que estar utilitzats.

Especificació pel Windows 95

Quan el camp compressió val BI_BITFIELDS, W95 només suporta les següents màscares de color de 16 BPP: una 5-5-5 imatge de 16 bits on la màscara blava es 0x001F, la màscara verda es 0x03E0 i la màscara vermella es 0x7C00; i una 5-6-5 imatge de 16 bits on la màscara blava es 0x001F, la màscara verda es 0x07E0 i la màscara vermella es 0xF800.

- Quan el camp es igual a 24

El BMP té un màxim de 2^{24} colors, i la paleta no conté entrades. Cada 3 bytes que defineixen un punt en el BMP representa la intensitat relativa de blau, verd, i vermell, respectivament, per a cada punt.

- Quan el camp es igual a 32

El BMP te un màxim de 2^{32} colors. Si el camp compressió del BMP val BI_RGB, la paleta de colors no conté cap entrada. Cada doble paraula en el BMP representa la intensitat relativa del blau, verd i vermell, respectivament, per a cada punt. El byte alt de cada doble paraula no s'utilitza.

Si el camp compressió de un BMP val BI_BITFIELDS, la paleta contindrà tres màscares de mida doble paraula que especifica les components vermell, verd i blau corresponents per cada punt. Cada doble paraula en el BMP representa un punt senzill.

Especificació pel Windows NT

Quan el camp compressió val BI_BITFIELDS els bits posats a 1 en cada màscara de doble paraula tenen que ser continus i no deurien sobreposar-se als bits d'un altre màscara. No tots el bits del punt tenen que estar utilitzats.

Especificació pel Windows 95

Quan el camp compressió val BI_BITFIELDS W95 només suporta la següent màscara de color de 32 BPP: la màscara blava es 0x000000FF, la màscara verda es 0x0000FF00 i la màscara vermella es 0x00FF0000.

13.3.3. El camp compressió

Aquest camp especifica la manera que el BMP emmagatzema les dades dintre del fitxer. La informació juntament amb el camp BPP identifica l'algoritme de compressió de la següent manera:

Valor	Significat
BI_RGB	Format no comprimit
BI_RLE4	És un format RLE (Run-length encoding) per BMP amb 4 bits per punt. El format de compressió consisteix en un format de dos bytes que és fa servir un byte com a comptador seguit per dos índex de color de mida doble paraula.
BI_RLE8	És un format RLE per BMP amb 8 bits per punt. El format de compressió consisteix en un format de dos bytes que és fa servir un byte com a comptador seguit per dos índex de color de mida byte.
BI_BITFIELDS	Especifica que el BMP no està comprimit i que la taula de colors consisteix en tres màscares de color de doble paraula que especifiquen el verd, vermell i blau per a cada punt. Això és vàlid quan es fan servir 16 o 32 BPP BMP

Quan el camp compressió es BI_RLE8, el BMP està comprimit posant un codi RLE per un BMP de 8 bits. Aquest format pot ser comprimit en mode codificat o en mode absolut, tots dos modes poden ocórrer en el mateix BMP.

Mode codificat.

Està format per 2 bytes, el primer byte especifica el número de punts consecutius que seran dibuixats utilitzant el índex de color contingut en el segon byte. A mes a mes el primer byte pot ser zero per indicar un ESC (escapament) que marca un final de línia, un final de BMP o una Delta. La interpretació del ESC depèn del valor del segon byte de la parella que pot ser un dels següents

0 fi de línia
1 fi de BMP
2 DELTA en els dos bytes que segueixen al ESC contenen valors sense signe indicant el desplaçament horitzontal i vertical del proper punt des de la posició actual.

Mode absolut.

El primer byte es zero i el segon byte està comprés entre 03H i FFH. El segon byte representa el número de bytes que segueixen, cadascun dels quals conté l'índex de color d'un punt sol. Quan el segon bytes es 2 o menys, el ESC té el mateix significat que el mode codificat. En el mode absolut, cada línia té que està arrodonida en número de bytes a una paraula.

L'exemple següent mostra els valors hexadecimals de un BMP 8 bit comprimits

03 04 05 06 00 03 45 56 67 00 02 78 00 02 05 01 02 78 00 00 09 1E 00 01

Expandim el BMP tenint en compte que el valor de dos dígits representa l'índex de color per a cada punt.

04 04 04
06 06 06 06 06
45 56 67
78 78
movem posició actual 5 a la dreta i 1 avall
78 78
fi de línia
1E 1E 1E 1E 1E 1E 1E 1E 1E
fi del BMP RLE

Quan el camp de compressió es BI_RL4 el BMP és comprimit utilitzant un format de codificació de 4 bits que també utilitzen modes codificat i absolut.

Mode codificat

El primer byte del parell conté el número de punts que seran dibuixats usant el índex de color del segon byte. El segon byte conté dos índex de color, els 4 bits de la part alta es un índex i el 4 bits de la part baixa es un altre.
El primer dels punts es dibuixa utilitzant el color especificat pels 4 bits de major ordre, el segon es dibuixa utilitzant els 4 bits de menor ordre, el tercer es dibuixa utilitzant el color dels 4 bits de major ordre fins que el punts especificats pel primer bytes hagin estat dibuixats.

Mode absolut

El primer byte es zero, el segon byte conté número de colors que el segueixen i els següents bytes contenen l'índex de color tenint en conte els nibbles, es a dir un índex de color per cada punt. El mode absolut cada línia té que està arrodonida en número de bytes a una paraula. El final de línia, el final de BMP i Delta es fa com en el cas anterior.

Un exemple serà,

03 04 05 06 00 06 45 56 67 00 04 78 00 02 05 01 04 78 00 00 09 1E 00 01

0 4 0
0 6 0 6 0
4 5 5 6 6 7
7 8 7 8
movem posició actual 5 a la dreta i 1 avall
7 8 7 8
fi de línia
1 E 1 E 1 E 1 E 1
fi del BMP RLE

13.3.4. Camps de colors

El camp de colors especifica els números d'índex dels colors en una taula de colors que actualment utilitza el BMP. Si aquest valor es zero, el BMP utilitza el màxim número de colors corresponent al camp BPP pel mode de compressió especificat pel camp compressió.

Si el camps de colors no es zero i el camp BBP es menor que 16, el camp de colors especifica el número de colors actuals que pot ser accedit pel driver del dispositiu o pel motor gràfic.

Si el camp BPP es mes gran o igual que 16, el camp colors especifica la mida de la taula de colors utilitzada per fer òptimes les característiques de les paletes de color de windows.

Si BPP es igual a 16 o 32, la paleta de color òptima comença immediatament seguida d'una màscara de 3 dobles paraules.

Si el BMP es un BMP empaquetat (o sigui un BMP en el qual la matriu de dades segueix immediatament a la matriu capçalera i que està referenciada per només un sol punter) el camp color, té que ser o bé zero o la mida actual de la taula de color.

13.3.5. Camps de colors importants

Aquest camp especifica el número de colors, el índex que son considerats importants per mostrar el BMP, si aquest valor es zero, tots el colors son importants.

Protecció del copyright

14. Annex 2: Codi del TFC

Daniel Mulero Roig
ETIS

Antoni Martínez Ballesté

18/06/04

14.1. Codi AnalisiBMP.java

```
/**
 *  Aquesta classe analitza el fitxer BMP desitjat passat com a
parametre I/O.
 * @author  Daniel Mulero Roig
 *          Treball de Fi de Carrera,
 *          Universitat Oberta de Catalunya (UOC)
 * @version juny 2004
 *
 */

// packages a incorporar

import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import java.util.Random;
import java.lang.*;

public class AnalisiBMP implements WindowListener {

// Zona de variables
      private String nomFitxer;
      private BufferedReader entrada;
      private PrintWriter sortida;
      private byte []capcalera= new byte [1500];
      private byte []word= new byte[4];
      private String s="";
      private Frame finestraAnalisi;
      private Panel panel1;
      private TextArea dades;
      private byte CRLF[]={0x0d,0x0a};
      private boolean retorn=false,ferAnalisi;

/**
 *  Constructor amb dos parametres que inicialitzen
 *  les propietats de AnalisiBMP
 *
 *  @param String nomFitx    nom del fitxer a tractar.
 *  @param boolean bol       autoritza a fer analisi.
 *
 **/

      public AnalisiBMP(String nomFitx,boolean bol){
            nomFitxer=nomFitx;
            ferAnalisi=bol;
      }
/**
 *  Metode que llegeix el fitxer i mostra dades a pantalla.
 *
 *  @return  cert, si ha pogut fer-ho, fals altrament
 *
 **/
       public boolean llegeixFitxer(){
            int numdades,t=0,colors,i,aux;
            long wordllegit;
            String s;

            try{
```

```
// Instanciem un fluxe de dades procedent d'un fitxer amb nomFitxer
                  FileInputStream f = new FileInputStream (nomFitxer);
// Fem crida al mètode .read per llegir la capçalera
                  numdades=f.read(capcalera);
// Existeixen dades?
                  if(numdades>=0){
// Comprova que sigui fitxer *.bmp, que sigui 24 bit color i sense
compressio
                        if   (((nomFitxer.substring(nomFitxer.length()-
3).toLowerCase()).equals("bmp"))&&

      (llegeixWord(28,2)==24)&&(llegeixWord(30,4)==0)){
// Hem de fer analisi?
                              if(ferAnalisi){
//    Generem    interficie    grafica    fent    crida    al    metode
creafinestraAnalisi()
                                    creafinestraAnalisi();
// Anem afegint dades a la variable de tipus textArea de la finestra
creada
// fem crida a llegeix 4 dades consecutives a la capçalera amb
llegeixWord
// portem control de final de linea amb CRLF
                                    dades.append(("Nom    del    fitxer:
"+nomFitxer).concat(new String(CRLF)));
                                    dades.append(("Identificador: "+new
String(capcalera,0,2)).concat(new String(CRLF)));
                                    dades.append(("Longitud del fitxer:
"+llegeixWord(2,4)+" bytes").concat(new String(CRLF)));
                                    dades.append(("Offset  fins  a  les
dades    del    bitmap:    "+llegeixWord(10,4)+"    bytes").concat(new
String(CRLF)));
                                    dades.append(("Longitud    de    la
capcalera     bitmap:     "+llegeixWord(14,4)+"     bytes").concat(new
String(CRLF)));
                                    dades.append(("Amplada  horitzontal
del bitmap: "+llegeixWord(18,4)+" punts").concat(new String(CRLF)));
                                    dades.append(("Amplada vertical del
bitmap: "+llegeixWord(22,4)+" punts").concat(new String(CRLF)));
                                    dades.append(("Numero de plans del
bitmap: "+llegeixWord(26,2)).concat(new String(CRLF)));
                                    dades.append(("Numero  de  bits  per
punt: "+llegeixWord(28,2)).concat(new String(CRLF)));
                                    dades.append(("Especificacio de la
compressio: "+llegeixWord(30,4)).concat(new String(CRLF)));
                                    dades.append(("Mida   del   bitmap:
"+llegeixWord(34,4)+" bytes").concat(new String(CRLF)));
                                    dades.append(("Resolucio
horitzontal:   "+llegeixWord(38,4)+"   punts   per   metre").concat(new
String(CRLF)));
                                    dades.append(("Resolucio  vertical:
"+llegeixWord(42,4)+" punts per metre").concat(new String(CRLF)));
// Cerca si hi han colors importants, sino és zero
                                    colors=(int)llegeixWord(46,4);
                                    dades.append(("Nombre   de   colors
utilitzats en el bitmap: "+colors).concat(new String(CRLF)));
                                    dades.append(("Nombre   de   colors
importants  utilitzats  en  el  bitmap:  "+llegeixWord(50,4)).concat(new
String(CRLF)));
// Hi han colors importants?
                                    if (colors!=0){
```

```
// Mostraria la seva existencia fent una taula mostran valors numerats
per color
// i mostrant valors per al vermell, blau i verd
                                        dades.append(("\n\tTAULA   DE
COLORS\n").concat(new String(CRLF)));

      dades.append(("\tCOLOR\tBLAU\tVERT\tVERMELL").concat(new
String(CRLF)));
                                        for (i=0;i<colors;i++){
                                              dades.append("\t"+i);
                                              aux=capcalera[54+i*4];
                                              if (aux<0){
                                                    aux+=256;
                                              }

      dades.append("\t\t"+aux);
                                              aux=capcalera[55+i*4];
                                              if (aux<0){
                                                    aux+=256;
                                              }

      dades.append("\t\t"+aux);
                                              aux=capcalera[56+i*4];
                                              if (aux<0){
                                                    aux+=256;
                                              }

      dades.append(("\t\t"+aux).concat(new String(CRLF)));
                                        }
                                  }
// Posiciona el cursor al començament de la finestra
                                  dades.setCaretPosition(0);
// Hem trobat tot be i hem creat la taula
                                  retorn=true;
                              }
                              else{
// Hem trobat tot be
                                  retorn=true;
                              }
                        }
                        else{
// Hem trobat errors en el proces
                              retorn=false;
                        }
                  }
// Tanquem el fluxe de dades procedent del fitxer
                  f.close();
            }
// Hem detectat errors a l'accedir al fitxer?
            catch (IOException e){
// Hem trobat errors en el proces
                  retorn=false;
            }
// Retornem valor d'estat
            return retorn;
      }
/**
 *     Metode que llegeix 4 bytes de la capçalera del fitxer
 *
 *  @param int          index valor a on començar a llegir
 *  @param int          limit valor en número de byte a llegir
```

```
 *  @return  long valor numeric dels 4 byte
 *
**/

      private long llegeixWord(int index,int limit){
            long valor=0;
            int i=1,aux=256,aux2;

            aux2=capcalera[index];
            if(aux2<0){
                  aux2+=256;
            }
            valor=aux2;
            for(i=1;i<limit;i++){
                  aux2=capcalera[index+i];
                  if(aux2<0){
                        aux2+=256;
                  }
                  valor+=(aux2*aux);
// Es per fer la potencia de 256
                  aux*=256;
            }
            return valor;
      }
/**
 *     Metode que genera la pantalla per visualitzar les dades del
fitxer
 *
**/

      private void creafinestraAnalisi()
      {
            int valorAfegit=0,x=150,y=150;

// Es fa un valor aleatori perque no aparegui la pantalla sempre
// en la mateixa posició
            valorAfegit=(new Random()).nextInt(100);
            s="Analisi fitxer ";
// Generem nova finestra amb el nom donat a s
            finestraAnalisi=new Frame(s);
// Donem posició i mides de la finestra

      finestraAnalisi.setBounds(x+valorAfegit,y+valorAfegit,500,150);
// Afegim control d'events per la finestra
            finestraAnalisi.addWindowListener(this);
// Fem que no és pugui modificar la mida de la finestra
            finestraAnalisi.setResizable(false);
// Creem un nou panel que afegirem a la finestra
            panel1=new Panel();
// Controlem l'aspecte que tindrà
            panel1.setLayout(new FlowLayout());
// Donem posició i mides del panell
            panel1.setBounds(x+valorAfegit,y+valorAfegit,500,150);
// Afegim un element del tipus TextArea
            dades=new TextArea();
// Afegim l'element anterior al panell
            panel1.add(dades);
// Afegim el panell a la finestra
            finestraAnalisi.add(panel1);
// Fem que tot l'antrior és visualitzi
            finestraAnalisi.setVisible(true);
```

```
// Ajustem la finestra a la mida del que hi ha dintre
            finestraAnalisi.pack();
      }
/**
 *     Mètode que controla l'event de la finestra quan s'està tancant
 *
**/

            public void windowClosing(WindowEvent e)
            {
                  finestraAnalisi.setVisible(false);
            }
/**
 *     Mètode que controla l'event de la finestra quan és activada
 *
**/
            public void windowActivated(WindowEvent e){}
/**
 *     Mètode que controla l'event de la finestra quan està tancada
 *
**/
            public void windowClosed(WindowEvent e){}
/**
 *     Mètode que controla l'event de la finestra quan és desactivada
 *
**/
            public void windowDeactivated(WindowEvent e){}
/**
 *     Mètode  que  controla  l'event  de  la  finestra  quan  no  està
minimitzada
 *
**/
            public void windowDeiconified(WindowEvent e){}
/**
 *     Mètode  que  controla  l'event  de  la  finestra  quan  està
minimitzada
 *
**/
            public void windowIconified(WindowEvent e){}
/**
 *     Mètode que controla l'event de la finestra quan està oberta
 *
**/
            public void windowOpened(WindowEvent e){}
}
```

14.2. Codi BaseDades.java

```
/**
 * Aquesta classe gestiona la base de dades
 * @author  Daniel Mulero Roig
 *          Treball de Fi de Carrera,
 *          Universitat Oberta de Catalunya (UOC)
 * @version juny 2004
 *
 **/

// packages a incorporar

import java.io.*;
```

```
import java.net.*;
import java.awt.*;
import java.util.StringTokenizer;
import java.lang.*;

public class BaseDades{
      private String nomFitxer;
      private RandomAccessFile fitxerDades,fitxerDadesAuxiliar;
      private byte []dadesFitxer= new byte [1500];
      private byte CRLF[]={0x0d,0x0a};
      private String s="";

/**
 *  Constructor amb un parametre que inicialitza
 *  les propietats de BaseDades
 *
 *  @param String nomFitx    nom del fitxer a tractar.
 *
 **/

      public BaseDades(String nomFitx){
            nomFitxer=nomFitx;
      }
/**
 *     Metode crea el fitxer
 *
**/
      public void creaFitxer()throws IOException{

            fitxerDades=new RandomAccessFile(nomFitxer,"rw");
            fitxerDades.close();
      }
/**
 *     Metode que comprova que el fitxer estigui creat
 *
 *  @return  boolean   cert, si està creat, fals altrament
 *
**/
      public boolean estaFitxerCreat(){

            File nomFitx= new File(nomFitxer);
// retorna si ha pogut llegir el fitxer instanciat
            return nomFitx.canRead();
      }
/**
 *    Metode que escriu una dada al fitxer, ordenan-les primer per nom
fitxer
 *    i després per valor de la marca
 *
 *  @param        String            dada
 *  @return       boolean           cert, si ha pogut, fals altrament
 *
**/

      public boolean escriuDada(String dada) throws IOException{
            long i,auxPunter=0;;
            int j=0,k;
            StringTokenizer token,token2;

            fitxerDades=new RandomAccessFile(nomFitxer,"rw");
            s="~".concat(nomFitxer);
```

```
// Creem un fitxer auxilar "~nomFitxer" a on escriure les dades
// amb el nou valor i ja ordenades
           fitxerDadesAuxiliar=new RandomAccessFile(s,"rw");
// Posicionem el punter al valor zero
           fitxerDades.seek(0);
// Anem llegint mentre existeixin
           for(i=0;i<fitxerDades.length();i++){
// Llegim una linia
                  s=fitxerDades.readLine();
                  if((s!=null)&&(!s.equals(""))){
// Si no és buida o nul·la comencem a treballar amb ella
                        token=new StringTokenizer(s);
                        token2=new StringTokenizer(dada);

      j=token.nextToken().compareTo(token2.nextToken());
// Es mes petita la dada del fitxer que la nova?
                        if(j<0){
// Afegim control de final de linia
                              s=s.concat(new String(CRLF));
// Passem dada al fitxer auxiliar perquè no ens interesa
                              fitxerDadesAuxiliar.writeBytes(s);
                        }
// Es igual la dada del fitxer que la nova?
                        if(j==0){
// Busquem com a màxim les vuit possibles marques
                              for(k=0;k<8;k++){

      j=token.nextToken().compareTo(token2.nextToken());
// La marc nova és mes gran?
                                    if (j>0){
// Trenquem rutina, ja tenim posició on guardar la nova dada
                                          break;
                                    }
// Afegim control de final de linia
                                    s=s.concat(new String(CRLF));
// Passem dada al fitxer auxiliar perquè no ens interesa
                                    fitxerDadesAuxiliar.writeBytes(s);
// Comprovem si hi han mes elements al darrera amb el mateix nom de
fitxer
                                    token=new StringTokenizer(s);
                                    token2=new StringTokenizer(dada);

      j=token.nextToken().compareTo(token2.nextToken());

      j=token.nextToken().compareTo(token2.nextToken());
                                    if(j>0){
// Trenquem rutina, ja tenim posició on guardar la nova dada
                                          break;
                                    }
// Trenquem rutina, ja tenim posició on guardar la nova dada
                                    break;
                              }
                        }
// La marc nova és mes petita?
                        if(j>0){
// Afegim control de final de linia
                              s=s.concat(new String(CRLF));
// Trenquem rutina, ja tenim posició on guardar la nova dada
                              break;
                        }
                  }
```

```
                  else{
// Hem arribat al final del fitxer
// afegim control de final de linia
                        s=dada.concat(new String(CRLF));
// Passem dada al fitxer auxiliar perquè no ens interesa
                        fitxerDadesAuxiliar.writeBytes(s);
// Trenquem rutina, ja tenim posició on guardar la nova dada
                        break;
                  }
            }
            if(j>0){
// Hem arribat a la posició desitjada
// passem dada al fitxer auxiliar que ens interesa afegir
                  fitxerDadesAuxiliar.writeBytes(dada);
// Passem dada al fitxer auxiliar perquè no ens interesa
                  fitxerDadesAuxiliar.writeBytes(s);
                  s=fitxerDades.readLine();
// Fins que acabem, llegim i afegim dades
                  for(;s!=null;){
                        s=s.concat(new String(CRLF));
                        fitxerDadesAuxiliar.writeBytes(s);
                        s=fitxerDades.readLine();
                  }
            }
// Es el final del fitxer i encara no l'hem introduit?
            if(j==0){
                  fitxerDadesAuxiliar.writeBytes(dada);
            }
// Tanquem fitxer
            fitxerDades.close();
            fitxerDadesAuxiliar.close();
            File auxBorrar= new File(nomFitxer);
// Eliminem fitxer original
            auxBorrar.delete();
            File auxBorrar2= new File("~".concat(nomFitxer));
//Renombrem fitxer auxiliar;
            auxBorrar2.renameTo(auxBorrar);
            return true;
      }
/**
 *    Metode que cerca una dada al fitxer i retorna si existeix o no
 *
 *  @param        String           dada1 és el nom del fitxer
 *  @param        String           dada2 és el valor de la marca
 *  @return       boolean          cert,   si   l'ha   trobat,   fals
altrament
 *
**/
      public  boolean  cercaDada(String  dada1,String  dada2)  throws
IOException{
            long i;
            boolean retorn=false;
// Obrim fitxer
            fitxerDades=new RandomAccessFile(nomFitxer,"rw");
// Posicionem el punter de cerca a l'inici
            fitxerDades.seek(0);
// Llegim una linia per entrar a l'iteracio de cerca
            s=fitxerDades.readLine();
// Es nul o buit?
            if ((s!=null)&&(!s.equals(""))){
// Posicionem el punter de cerca a l'inici
```

```
                  fitxerDades.seek(0);
// Mentre hi hagin dades...
                  for(i=0;i<fitxerDades.length();i++){
                        s=fitxerDades.readLine();
                        if((s!=null)&&(!s.equals(""))){
                              StringTokenizer                  token=new
StringTokenizer(s);
// Compara linea llegida amb la pasada com a paràmetre

      if((token.nextToken()).equals(dada1)&&(token.nextToken()).equals
(dada2)){
// Trobada!
                                    retorn=true;
                                    break;
                              }
                        }
                        else{
// Fitxer acabat o linia buida
                              retorn=false;
                              break;
                        }
                  }
            }
// Tanquem fitxer
            fitxerDades.close();
            return retorn;
      }
/**
 *    Metode que cerca una dada al fitxer i retorna nom del comprador
 *
 *  @param        String            dada1 és el nom del fitxer
 *  @param        String            dada2 és el valor de la marca
 *  @return       String            nom del comprador o buit
 *
**/
      public   String   cercaNomComprador(String   dada1,String   dada2)
throws IOException{
            long i;
            String retorn="";

// Obrim fitxer
            fitxerDades=new RandomAccessFile(nomFitxer,"rw");
// Posicionem el punter de cerca a l'inici
            fitxerDades.seek(0);
// Llegim una linia per entrar a l'iteracio de cerca
            s=fitxerDades.readLine();
// Es nul o buit?
            if ((s!=null)&&(!s.equals(""))){
// Posicionem el punter de cerca a l'inici
                  fitxerDades.seek(0);
// Mentre hi hagin dades...
                  for(i=0;i<fitxerDades.length();i++){
                        s=fitxerDades.readLine();
                        if((s!=null)&&(!s.equals(""))){
                              StringTokenizer                  token=new
StringTokenizer(s);
// Compara linea llegida amb la pasada com a paràmetre

      if((token.nextToken()).equals(dada1)&&(token.nextToken()).equals
(dada2)){
// Trobada!
```

```
                                    retorn=token.nextToken();
                                    break;
                              }
                        }
                        else{
// Fitxer acabat o linia buida
                              retorn=" .".concat(dada2);
                        }
                  }
            }
// Tanquem fitxer
            fitxerDades.close();
            return retorn;
      }
/**
 *    Metode que esborra la base de dades
 *
**/
      public void esborraBD(){
            new File(nomFitxer).delete();
      }
}
```

14.3. Codi Confabula.java

```
/**
 * Aquesta classe confabula els fitxers BMP desitjats passats com a
parametre I/O.
 * @author  Daniel Mulero Roig
 *          Treball de Fi de Carrera,
 *          Universitat Oberta de Catalunya (UOC)
 * @version juny 2004
 *
 **/

// packages a incorporar

import java.io.*;
import java.net.*;
import java.util.Random;
import java.lang.*;


public class Confabula{
      private String nomFitxer,nomSegonFitxer,marca,nom;
      private BufferedReader entrada;
      private byte []dadesFitxer= new byte [1500];
      private byte []dadesSegonFitxer= new byte [1500];
      private byte []word= new byte[4];
      private byte [][]dadesBMP= new byte[1500][4500];
      private byte [][]dadesSegonBMP= new byte[1500][4500];
      private int []V0=new int[7];
      private int []V1=new int[7];
      private String s="";
      private                                                    int
offset,longitutBitmap,ampladaHoritzontal,ampladaVertical;
      private int offset1,longitutBitmap1;
      private int ampladaHoritzontal1,ampladaVertical1;
      private int offset2,longitutBitmap2;
      private int ampladaHoritzontal2,ampladaVertical2;
```

```
      private int longitutTrama=7,quadre=10;
      private byte profunditat=1;
      private int quadreExtraccio;
      private boolean ferAnalisi;

/**
 *  Constructor amb tres parametres que inicialitza
 *  les propietats de Confabula
 *
 *  @param String nomfitxer        nom del fitxer a tractar.
 *  @param String nomsegonfitxer   nom de l'altre fitxer a tractar.
 *  @param boolean bol                  autoritzacio     per     fer
l'analisi del fitxer
 *
 **/

      public  Confabula(String  nomfitxer,String  nomsegonfitxer,boolean
bol){
            nomFitxer=nomfitxer;
            nomSegonFitxer=nomsegonfitxer;
            ferAnalisi=bol;
      }
/**
 *    Metode que genera una nova marca amb la barreja d'altres
 *
 *  @return  String retorna cadena amb resultat de les operacions
 *
**/
      public String generaNovaMarca(){
// Genera cadena amb el nom del fitxer a crear
            s=nomFitxer.substring(0,nomFitxer.indexOf(".")+1);
            s=(s.concat("conf."))
            .concat(nomFitxer.substring(nomFitxer.indexOf(".")+1
            ,nomFitxer.lastIndexOf(".")+1));

      s=(s.concat(nomSegonFitxer.substring(nomSegonFitxer.indexOf(".")
+1
            ,nomSegonFitxer.lastIndexOf(".")+1))).concat("bmp");
// Ja existeix?
            if(estaFitxerCreat(s)){
// No permetem sobreescriptura
                  return "Nom de fitxer ja associat a una altre marca";
            }
// Fitxer no existeix
            else{
// Fem la instanciació del primer fitxer
                  AnalisiBMP                              analisiBMP=new
AnalisiBMP(nomFitxer,ferAnalisi);
// S'ha de fer anàlisi?
                  if(analisiBMP.llegeixFitxer()){
// Llegeix primer fitxer, si pot.

      if(this.llegeixFitxer(nomFitxer,dadesFitxer,dadesBMP)){
// Guarda valors per a poder compara després
                              offset1=offset;
                              longitutBitmap1=longitutBitmap;
                              ampladaHoritzontal1=ampladaHoritzontal;
                              ampladaVertical1=ampladaVertical;
// Fem la instanciació del segon fitxer
                              AnalisiBMP            analisiSegonBMP=new
AnalisiBMP(nomSegonFitxer,ferAnalisi);
```

```
// S'ha de fer anàlisi?
                              if(analisiSegonBMP.llegeixFitxer()){
// Llegeix segon fitxer, si pot.

      if(this.llegeixFitxer(nomSegonFitxer,dadesSegonFitxer,dadesSegon
BMP)){
// Guarda valors per a poder compara després
                                        offset2=offset;

      longitutBitmap2=longitutBitmap;

      ampladaHoritzontal2=ampladaHoritzontal;

      ampladaVertical2=ampladaVertical;
// Coincideixen capçaleres?
                                        if
((ampladaVertical2==ampladaVertical1)&&

      (ampladaHoritzontal2==ampladaHoritzontal1)&&

      (longitutBitmap1==longitutBitmap2)&&

      (nomFitxer.substring(0,nomFitxer.indexOf("."))).equals(nomSegonF
itxer.substring(0,nomSegonFitxer.indexOf("."))){
// Generem confabulació
                                              this.confabulaFitxer();
// Intentem crear el fitxer modificat
                                              try{


      this.creaFitxerModificat();
                                              }
                                              catch (IOException e){
// No hem pogut crear el fitxer modificat
                                                    return   "Problema
amb la creació de l'arxiu";
                                              }
                                        }
// Els fitxer no pertanyen al mateix original
                                        else{
                                             return "Fitxers amb
fitxer original diferent";
                                        }
                                  }
// No hem pogut llegir l'arxiu
                                  else{
                                        return "Error en l'acces del
arxiu";
                                  }
                            }
// No hem pogut llegir l'arxiu
                            else{
                                  return   "Error   en   l'acces   del
arxiu";
                            }
                      }
// No hem pogut llegir l'arxiu
                      else{
                            return "Error en l'acces del arxiu";
                      }
                }
```

```
// No hem pogut llegir l'arxiu
                  else{
                        return "Error en l'acces del arxiu";
                  }
// S'ha pogut fer
                  return "Fet";
            }
      }
/**
 *    Metode que llegeix el fitxer i guarda la capçalera i les dades
per separat
 *
 *  @param   String nomFitx   nom del fitxer a llegi
 *  @param   byte []dadesfitx       dades on guardar la capçalera
 *  @param   byte [][] dadesbmp dades on guardar valor del pixels en
matriu x,y
 *  @return  boolean               cert, si ha pogut, fals altrament
 *
**/
      private      boolean      llegeixFitxer(String      nomfitx,byte
[]dadesfitx,byte [][] dadesbmp){
            int numdades,t=0,colors,i,j,aux;
            byte aux_b;
            long wordLlegit;
            boolean noError=false;

            try{
// Obre fitxer desitjat
                  FileInputStream f = new FileInputStream (nomfitx);
// Llegeix longitut màxima de capçalera per a 24 bit
                  numdades=f.read(dadesfitx,0,240);
// Hem llegit?
                  if(numdades>=0){
// Guarda variables desitjades
                        offset=(int)llegeixWord(10,4,dadesfitx);

      longitutBitmap=(int)llegeixWord(34,4,dadesfitx);

      ampladaHoritzontal=(int)llegeixWord(18,4,dadesfitx);

      ampladaVertical=(int)llegeixWord(22,4,dadesfitx);
                  }
// Llegeix per columnes cada pixel que te 24 bit
                  for(i=0;i<ampladaVertical;i++){
// Llegeix per files cada pixel que te 24 bit
                        for(j=0;j<(ampladaHoritzontal*3);j++){
                              dadesbmp[i][j]=(byte)f.read();
                        }
// Control del padding per si hem llegit un valor no modul 4
// Multipliquem per 3 (pixel Red, Green and Blue) i dividim per 4
                        aux=(ampladaHoritzontal*3)>>2;
// Mirem si és 1,2 ó 3
                        aux=ampladaHoritzontal*3-aux*4;
                        if(aux>0){
                              aux=4-aux;
// Llegim sense guardar en matriu
                              for (j=0;j<aux;j++){
                                    f.read();
                              }
                        }
                  }
```

```
// No hem trobat error
                  noError=true;
// Tanquem fitxer
                  f.close();
            }
            catch (IOException e){
// Hem trobat error
                  noError=false;
            }
            return noError;
      }
/**
 *     Metode que llegeix 4 bytes de la capçalera del fitxer guardada
a memoria
 *
 *  @param int          index             valor a on començar a llegir
 *  @param int          limit             valor  en  número  de  byte  a
llegir
 *  @param byte   []dadesFitx matriu on son les dades a tractar
 *  @return  long valor numeric dels 4 byte
 *
**/

      private long llegeixWord(int index,int limit,byte []dadesFitx){
            long valor=0;
            int i=1,aux=256,aux2;

            aux2=dadesFitx[index];
            if(aux2<0){
                  aux2+=256;
            }
            valor=aux2;
            for(i=1;i<limit;i++){
                  aux2=dadesFitx[index+i];
                  if(aux2<0){
                        aux2+=256;
                  }
                  valor+=(aux2*aux);
// Es per fer la potencia de 256
                  aux*=256;
            }
            return valor;
      }
/**
 *     Metode que barreja tots dos fitxers
 *
**/

      private void confabulaFitxer(){
            int i,j,k,uns=0,zeros=0,contador;
            byte aux,aux2;

            Random r=new Random();
// Llegeix per columnes cada pixel que te 24 bit
            for(i=0;i<ampladaVertical;i++){
// Llegeix per files cada pixel que te 24 bit
                  for(j=0;j<(ampladaHoritzontal*3);j+=3){
                        contador=0;
// Comprova que al menys un Byte deks 3 que composa un pixel
// és diferent a cada fitxer
                        if(dadesBMP[i][j]!=dadesSegonBMP[i][j]){
```

```
                              contador++;
                        }
                        if(dadesBMP[i][j+1]!=dadesSegonBMP[i][j+1]){
                              contador++;
                        }
                        if(dadesBMP[i][j+2]!=dadesSegonBMP[i][j+2]){
                              contador++;
                        }
// Hi ha variació?
                        if (contador>0){
// Varia el darrer bit de forma aleatoria per a cada pixel i per a
cada color
// elimina el darre bit
                              dadesBMP[i][j]&=0xfe;
// Generem bit aleatori 1 ó 0
                              k=r.nextInt(2);
// Afegim el que sigui al final del byte i el guardem a la matriu de
dades del
// primer fitxer
                              dadesBMP[i][j]+=k;
                              dadesBMP[i][j+1]&=0xfe;
                              k=r.nextInt(2);
                              dadesBMP[i][j+1]+=k;
                              dadesBMP[i][j+2]&=0xfe;
                              k=r.nextInt(2);
                              dadesBMP[i][j+2]+=k;
                        }
                  }
            }
      }
/**
 *     Metode que crea el fitxer modificat
 *
**/

      private void creaFitxerModificat()throws IOException{
            int i,j,aux;
            String s="";

// Genera cadena amb el nom del fitxer a crear
            i=nomFitxer.indexOf(".");
            s=nomFitxer.substring(0,i+1);
            j=nomFitxer.lastIndexOf(".");
            s=(s.concat("conf.")).concat(nomFitxer.substring(i+1,j+1));
            j=nomSegonFitxer.lastIndexOf(".");

      s=(s.concat(nomSegonFitxer.substring(i+1,j+1))).concat("bmp");
            RandomAccessFile fitxer=new RandomAccessFile(s,"rw");
// Escribim primer la capçalera
            for(i=0;i<offset;i++){
                  fitxer.writeByte(dadesFitxer[i]);
            }
// Escriu per columnes cada pixel que te 24 bit
            for(i=0;i<ampladaVertical;i++){
// Escriu per files cada pixel que te 24 bit
                  for(j=0;j<ampladaHoritzontal*3;j++){
                        aux=dadesBMP[i][j];
                        fitxer.writeByte(dadesBMP[i][j]);
                  }
// Control del padding per si hem llegit un valor no modul 4
// Multipliquem per 3 (pixel Red, Green and Blue) i dividim per 4
```

```
                  aux=(ampladaHoritzontal*3)>>2;
                  aux=ampladaHoritzontal*3-aux*4;
// Mirem si és 1,2 ó 3
                  if(aux>0){
                        aux=4-aux;
// Escribim el padding
                        for (j=0;j<aux;j++){
                              fitxer.writeByte(0);
                        }
                  }
            }
// Tanquem fitxer
            fitxer.close();
      }
/**
 *     Metode que comprova que el fitxer estigui creat
 *
 *  @param        String nomFitxe nom del fitxer a controlar
 *
 *  @return       boolean     cert, si està creat, fals altrament
 *
**/

      private boolean estaFitxerCreat(String nomFitxe){

            File nomFitx= new File(nomFitxe);
// retorna si ha pogut llegir el fitxer instanciat
            return nomFitx.canRead();
      }
}
```

14.4. Codi MarcaBMP.java

```
/**
 * Aquesta classe marca el fitxer BMP desitjat passat com a parametre
I/O
 *  o be extreu la marca
 * @author  Daniel Mulero Roig
 *          Treball de Fi de Carrera,
 *          Universitat Oberta de Catalunya (UOC)
 * @version juny 2004
 *
 **/

// packages a incorporar

import java.io.*;
import java.net.*;
import java.lang.*;

public class MarcaBMP{
      private String nomFitxer,nomSegonFitxer,marca,nom;
      private BufferedReader entrada;
      private byte []dadesFitxer= new byte [1500];
      private byte []dadesSegonFitxer= new byte [1500];
      private byte []word= new byte[4];
      private byte [][]dadesBMP= new byte[1500][4500];
      private byte [][]dadesSegonBMP= new byte[1500][4500];
      private byte CRLF[]={0x0d,0x0a};
      private int []V0=new int[7];
```

```
      private int []V1=new int[7];
      private    String s="";
      private                                                    int
offset,longitutBitmap,ampladaHoritzontal,ampladaVertical;
      private int offset1,longitutBitmap1;
      private int ampladaHoritzontal1,ampladaVertical1;
      private int offset2,longitutBitmap2;
      private int ampladaHoritzontal2,ampladaVertical2;
      private int longitutTrama=7,quadre=10;
      private byte profunditat=1;
      private int quadreExtraccio;
      private BaseDades baseDades;
      private boolean ferAnalisi;

/**
 *  Constructor amb quatre parametres que inicialitza
 *  les propietats de MarcaBMPper a codificar
 *
 *  @param String nomfitxer        nom del fitxer a tractar.
 *  @param String marc                  marca a grabar.
 *  @param String dada_nom              nom  a  posar  a  la  base  de
dades
 *  @param boolean bol                  autoritzacio      per      fer
l'analisi del fitxer
 *
 **/

      public      MarcaBMP(String      nomfitxer,String      marc,String
dada_nom,boolean bol){
            nomFitxer=nomfitxer;
            marca=marc ;
            nom=dada_nom;
            ferAnalisi=bol;
      }
/**
 *  Constructor amb tres parametres que inicialitza
 *  les propietats de MarcaBMP per a descodificar
 *
 *  @param String nomfitxer        nom del fitxer a tractar.
 *  @param String nomsegonfitxer   nom del segon fitxer a tractar.
 *  @param boolean bol                  autoritzacio      per      fer
l'analisi del fitxer
 *
 **/

      public  MarcaBMP(String  nomfitxer,String  nomsegonfitxer,boolean
bol){
            nomFitxer=nomfitxer;
            nomSegonFitxer=nomsegonfitxer;
            ferAnalisi=bol;
      }
/**
 *  Metode que llegeix la marca.
 *
 *  @return  String retorna cadena amb informacio
 *
 **/

      public String extreuMarca(){
            AnalisiBMP analisiBMP=new AnalisiBMP(nomFitxer,ferAnalisi);
// Hem de fer analisi?
```

```
            if (analisiBMP.llegeixFitxer()){
// Llegim dades del primer fitxer
                  this.llegeixFitxer(nomFitxer,dadesFitxer,dadesBMP);
// Guardem valors de la capçalera del primer fitxer
                  ampladaVertical1=ampladaVertical;
                  ampladaHoritzontal1=ampladaHoritzontal;
                  longitutBitmap1=longitutBitmap;
                  AnalisiBMP                        analisiSegonBMP=new
AnalisiBMP(nomSegonFitxer,ferAnalisi);
// Hem de fer analisi?
                  if(analisiSegonBMP.llegeixFitxer()){
// Llegim dades del segon fitxer

      this.llegeixFitxer(nomSegonFitxer,dadesSegonFitxer,dadesSegonBMP
);
// Guardem valors de la capçalera del segon fitxer
                        ampladaVertical2=ampladaVertical;
                        ampladaHoritzontal2=ampladaHoritzontal;
                        longitutBitmap2=longitutBitmap;
                        if
((nomFitxer.substring(0,nomFitxer.indexOf("."))).equals(nomSegonFitxer
.substring(0,nomSegonFitxer.indexOf("")))){
                              baseDades=new              BaseDades((new
Tfc()).getNomFitxer());
                              try{
// Busquem el valor del nom del comprador, passant la marca i el nom
del fitxer
                                    return
baseDades.cercaNomComprador(nomFitxer,this.llegeixMarca());
                              }
                              catch (IOException e){
// No hem pogut fer-ho
                              return  "Error  a  l'arxiu  de  base  de
dades.";
                              }
                        }
                        else{
                            return   "Fitxers   amb   fitxer   original
diferent";
                        }
                  }
                  else{
// No hem pogut llegir l'arxiu
                        return "Error en l'acces del arxiu";
                  }
            }
            else{
// No hem pogut llegir l'arxiu
                  return "Error en l'acces del arxiu";
            }
      }
/**
 *  Metode que amaga la marca.
 *
 *  @return  String retorna cadena amb informacio
 *
 **/

      public String colocaMarca(){
            boolean haPogut,haTrobat;
            String missatge;
```

```
            int i;
// Comprova que la marca és una de les acceptades
            for (i=0;i<8;i++){
                  if((new Tfc()).getComprador(i).equals(marca)){
                        break;
                  }
            }
//  Esta admesa?
            if (i==8){
                  return "Marca no admesa";
            }
            try{
                  baseDades=new BaseDades((new Tfc()).getNomFitxer());
// Comprova que no estigui asociada
                  haTrobat=baseDades.cercaDada(nomFitxer,marca);
// Ho ha tobat?
                  if (haTrobat){
                        return "Marca de fitxer associada a un altre
comprador";
                  }
            }
            catch (IOException e){
                  missatge="Error a l'arxiu de base de dades.";
            }
            s=nomFitxer.substring(0,(nomFitxer.length()-
3)).concat(s.concat(nom)).concat(".bmp");
// Existeix fitxer a codificar amb el mateix nom?
            if(estaFitxerCreat(s)){
                  return "Nom de fitxer ja associat a una altre marca";
            }
            else{
                  AnalisiBMP                              analisiBMP=new
AnalisiBMP(nomFitxer,ferAnalisi);
// Hem de fer analisi?
                  if(analisiBMP.llegeixFitxer()){
// Llegim dades del fitxer

      this.llegeixFitxer(nomFitxer,dadesFitxer,dadesBMP);
// Codificquem el fitxer
                        this.codificaFitxer();
                        try{
// Creem fitxer modificat
                              this.creaFitxerModificat();
                        }
                        catch (IOException e){
// No hem pogut fer-ho
                              missatge="Error   en   la   creació   arxiu
modificat";
                        }
                        try{
// Genera linia a guardar amb el format adequat
                              s=nomFitxer.concat(" ");
                              s=s.concat(marca);
                              s=s.concat(" ");
                              s=s.concat(nom);
                              s=s.concat(new String(CRLF));
                              baseDades=new              BaseDades((new
Tfc()).getNomFitxer());
// Guardem la dada
                              haPogut=baseDades.escriuDada(s);
// Ha pogut?
```

```
                              if (haPogut){
                                    missatge="Proces realitzat";
                              }
                              else{
                                    missatge="Proces no realitzat";
                              }
                        }
                        catch (IOException e){
// No hem pogut llegir l'arxiu
                              missatge="Error  a  l'escriure  dades  al
fitxer de base de dades";
                        }
                  }
                  else{
// No hem pogut llegir l'arxiu
                        missatge="Error en l'acces del arxiu";
                  }
                  return missatge;
            }
      }
/**
 *    Metode  que  llegeix  el  fitxer  i  guarda  la  capçalera  i  les  dades
per separat
 *
 *  @param   String nomFitx   nom del fitxer a llegi
 *  @param   byte []dadesfitx      dades on guardar la capçalera
 *  @param   byte [][] dadesbmp dades on guardar valor del pixels en
matriu x,y
 *  @return  boolean              cert, si ha pogut, fals altrament
 *
**/

      private      boolean      llegeixFitxer(String      nomfitx,byte
[]dadesfitx,byte [][] dadesbmp){
            int numdades,t=0,colors,i,j,aux;
            byte aux_b;
            long wordLlegit;
            boolean noError=false;

            try{
// Obre fitxer desitjat
                  FileInputStream f = new FileInputStream (nomfitx);
// Llegeix longitut màxima de capçalera per a 24 bit
                  numdades=f.read(dadesfitx,0,240);
// Hem llegit?
                  if(numdades>=0){
// Guarda variables desitjades
                        offset=(int)llegeixWord(10,4,dadesfitx);

      longitutBitmap=(int)llegeixWord(34,4,dadesfitx);

      ampladaHoritzontal=(int)llegeixWord(18,4,dadesfitx);

      ampladaVertical=(int)llegeixWord(22,4,dadesfitx);
                  }
// Llegeix per columnes cada pixel que te 24 bit
                  for(i=0;i<ampladaVertical;i++){
// Llegeix per files cada pixel que te 24 bit
                        for(j=0;j<(ampladaHoritzontal*3);j++){
                              dadesbmp[i][j]=(byte)f.read();
                        }
```

```
// Control del padding per si hem llegit un valor no modul 4
// Multipliquem per 3 (pixel Red, Green and Blue) i dividim per 4
                        aux=(ampladaHoritzontal*3)>>2;
// Mirem si és 1,2 ó 3
                        aux=ampladaHoritzontal*3-aux*4;
                        if(aux>0){
                              aux=4-aux;
// Llegim sense guardar en matriu
                              for (j=0;j<aux;j++){
                                    f.read();
                              }
                        }
                  }
// No hem trobat error
                  noError=true;
// Tanquem fitxer
                  f.close();
            }
            catch (IOException e){
// Hem trobat error
                  noError=false;
            }
            return noError;
      }
/**
 *     Metode que llegeix 4 bytes de la capçalera del fitxer guardada
a memoria
 *
 *  @param int          index             valor a on començar a llegir
 *  @param int          limit             valor  en  número  de  byte  a
llegir
 *  @param byte   []dadesFitx matriu on son les dades a tractar
 *  @return  long valor numeric dels 4 byte
 *
**/

      private long llegeixWord(int index,int limit,byte []dadesFitx){
            long valor=0;
            int i=1,aux=256,aux2;

            aux2=dadesFitx[index];
            if(aux2<0){
                  aux2+=256;
            }
            valor=aux2;
            for(i=1;i<limit;i++){
                  aux2=dadesFitx[index+i];
                  if(aux2<0){
                        aux2+=256;
                  }
                  valor+=(aux2*aux);
// Es per fer la potencia de 256
                  aux*=256;
            }
            return valor;
      }
/**
 *     Metode que codifica el fitxer
 *
**/
```

```
      private void codificaFitxer(){
            int i,j,k,contadorTrama,contadorQuadres=0;
            byte aux;

// Llegim files
            for(k=0;k<ampladaVertical;k++){
// Llegim columnes
                  for (i=0;i<ampladaHoritzontal;i+=quadre){
                        contadorTrama=0;
// Codifiquem cada linia per quadres
                        for (j=0;j<quadre;j++){
// Es un 1?
                              if (marca.charAt(contadorTrama)=='1'){
// Ho fem per a cada pixel (3 byte)
                                    aux=dadesBMP[k][(i+j)*3];
// Profunditat és quin bit és vol amagar la marca
// modificable per codi i no per programa
                                    aux&=(255-profunditat);
                                    if (aux==dadesBMP[k][(i+j)*3]){

      dadesBMP[k][(i+j)*3]+=profunditat;
                                    }
                                    else{
                                          dadesBMP[k][(i+j)*3]-
=profunditat;
                                    }
                                    aux=dadesBMP[k][(i+j)*3+1];
                                    aux&=(255-profunditat);
                                    if (aux==dadesBMP[k][(i+j)*3+1]){

      dadesBMP[k][(i+j)*3+1]+=profunditat;
                                    }
                                    else{
                                          dadesBMP[k][(i+j)*3+1]-
=profunditat;
                                    }
                                    aux=dadesBMP[k][(i+j)*3+2];
                                    aux&=(255-profunditat);
                                    if (aux==dadesBMP[k][(i+j)*3+2]){

      dadesBMP[k][(i+j)*3+2]+=profunditat;
                                    }
                                    else{
                                          dadesBMP[k][(i+j)*3+2]-
=profunditat;
                                    }
                              }
// Fem control per reinicialitzar la marca dins d'un quadre
// marca te 7 bit i el quadre treballa amb 10 pixels x 10 pixels
                              if (++contadorTrama==7){
                                    contadorTrama=0;
                              }
                        }
                  }
// Hem completat el quadre?
                  if (++contadorQuadres==10){
                        contadorQuadres=0;
                  }
            }
      }
/**
```

```
 *     Metode que crea el fitxer modificat
 *
**/

      private void creaFitxerModificat()throws IOException{
            int i,j,aux;
            String s="";

// Genera cadena amb el nom del fitxer a crear
            s=nomFitxer.substring(0,(nomFitxer.length()-
3)).concat(s.concat(nom)).concat(".bmp");
            RandomAccessFile fitxer=new RandomAccessFile(s,"rw");
// Escribim primer la capçalera
            for(i=0;i<offset;i++){
                  fitxer.writeByte(dadesFitxer[i]);
            }
// Escriu per columnes cada pixel que te 24 bit
            for(i=0;i<ampladaVertical;i++){
// Escriu per files cada pixel que te 24 bit
                  for(j=0;j<ampladaHoritzontal*3;j++){
                        aux=dadesBMP[i][j];
                        fitxer.writeByte(dadesBMP[i][j]);
                  }
// Control del padding per si hem llegit un valor no modul 4
// Multipliquem per 3 (pixel Red, Green and Blue) i dividim per 4
                  aux=(ampladaHoritzontal*3)>>2;
                  aux=ampladaHoritzontal*3-aux*4;
// Mirem si és 1,2 ó 3
                  if(aux>0){
                        aux=4-aux;
// Escribim el padding
                        for (j=0;j<aux;j++){
                              fitxer.writeByte(0);
                        }
                  }
            }
// Tanquem fitxer
            fitxer.close();
      }
/**
 *     Metode que llegeix la marca
 *
 *  @return  String cadena que conte la marca
 *
**/

      private String llegeixMarca(){
            int
i,j,k,contadorTrama,contadorQuadres=0,aux_int,mesAjustat;
            byte aux,aux2,auxContar;
            String s;

// Quadre d'extracció variable pels escaltats de les imatges

      quadreExtraccio=quadre*ampladaHoritzontal2/ampladaHoritzontal1;
// Llegim files
            for(k=0;k<ampladaVertical2;k++){
// Llegim columnes en funcio dels quadres
                  for (i=0;i<ampladaHoritzontal2;i+=quadreExtraccio){
                        contadorTrama=0;
// Llegim quadres
```

```
                        for (j=0;j<quadreExtraccio;j++){
// Llegim cada byte al dos fitxers l'origina i el modificata per cada
pixel
                              auxContar=0;
                              aux_int=i*quadre/quadreExtraccio;
                              aux=dadesBMP[k][(aux_int+j)*3];
                              aux&=(profunditat);
                              aux2=dadesSegonBMP[k][(i+j)*3];
                              aux2&=(profunditat);
// Portem control de les modificacions trobades
                              if(aux!=aux2){
                                    auxContar++;
                              }
                              aux_int=i*quadre/quadreExtraccio;
                              aux=dadesBMP[k][(aux_int+j)*3+1];
                              aux&=(profunditat);
                              aux2=dadesSegonBMP[k][(i+j)*3+1];
                              aux2&=(profunditat);
                              if(aux!=aux2){
                                    auxContar++;
                              }
                              aux_int=i*quadre/quadreExtraccio;
                              aux=dadesBMP[k][(aux_int+j)*3+2];
                              aux&=(profunditat);
                              aux2=dadesSegonBMP[k][(i+j)*3+2];
                              aux2&=(profunditat);
                              if(aux!=aux2){
                                    auxContar++;
                              }
// Si modificacions trobade per pixel son mes grans que 1 és un 1
binari
                              if (auxContar>1){
                                    V1[contadorTrama]++;
                              }
                              else{
                                    V0[contadorTrama]++;
                              }
// Fem control per reinicialitzar la marca dins d'un quadre
// marca te 7 bit i el quadre treballa amb 10 pixels x 10 pixels
                              if (++contadorTrama==7){
                                    contadorTrama=0;
                              }
                        }
                  }
// Hem completat el quadre?
                  if (++contadorQuadres==10){
                        contadorQuadres=0;
                  }
            }
            s="";
// Comprobem qui és mes gran per generar el String que contindra la
marca
            for(i=0;i<longitutTrama;i++){
                  if(V1[i]>V0[i]){
                        s=s.concat("1");
                  }
                  else{
                        s=s.concat("0");
                  }
            }
            mesAjustat=10000;
```

```
            k=0;
// Busquem quin valor trobat s'ajusta mes als vuit possibles
            for(i=0;i<8;i++){
                  j=0;
                  for(aux_int=0;aux_int<7;aux_int++){
                        if(s.charAt(aux_int)!=(new
Tfc()).getComprador(i).charAt(aux_int)){
                              j++;
                        }
                  }
// El que mes s'ajusta és el que mes s'apropa a zero
                  if(j<mesAjustat){
                        mesAjustat=j;
                        k=i;
                  }
            }
// Amb el valor més ajustat busquem la marca asociada per retornar-la
            s=(new Tfc()).getComprador(k);
            return s;
      }
/**
 *     Metode que comprova que el fitxer estigui creat
 *
 *  @param        String nomFitxe nom del fitxer a controlar
 *
 *  @return       boolean     cert, si està creat, fals altrament
 *
**/

      private boolean estaFitxerCreat(String nomFitxe){

            File nomFitx= new File(nomFitxe);
// retorna si ha pogut llegir el fitxer instanciat
            return nomFitx.canRead();
      }
}
```

14.5. Codi Tfc.java

```
/**
 * Aquesta classe es aa principal del TFC
 * @author  Daniel Mulero Roig
 *          Treball de Fi de Carrera,
 *          Universitat Oberta de Catalunya (UOC)
 * @version juny 2004
 *
 **/

// packages a incorporar

import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.lang.*;

public class Tfc implements ActionListener, WindowListener
{
      private Frame finestra;
      private Button codifica,descodifica,confabula,ok;
      private Button disco1,disco2,disco3,disco4,disco5;
```

```
      private Label valor1,valor2,valor3,valor4,valor5,valor6;
      private                                                    Panel
panel1,panel2,panel3,panel4,panel5,panel6,panel7,panel8,panel20;
      private TextField text1,text2,text3,text4,text5,text6;
      private Choice choice2;
      private String marca="1010101",onSom;
      private String nomFitxer="BaseDeDades.txt";
      private FileDialog fitxer1,fitxer2;
      private PopupMenu alertes;
      private MenuBar barraMenu;
      private Menu esborraBD,ajuda,configura;
      private MenuItem autor,si,no;
      private CheckboxMenuItem analisi;
      private                                                 String
comprador[]={"0000000","0011011","0101101","0110110",

      "1001110","1010101","1100011","1111000"};
/**
 *  Constructor buit de la classe Tfc
 *
 **/


      public Tfc(){}

/**
 *     Metode que crea el fitxer sino existeix
 *
**/

      private void creaFitxer()
      {
            BaseDades esto=new BaseDades(nomFitxer);
            if (!esto.estaFitxerCreat()){
                  try{
                        esto.creaFitxer();
                  }
                  catch (IOException e){
// Posició de finestra on retorna l'error
                        text6.setText("Problema amb l'arxiu");
                  }
            }
      }
/**
 *     Metode que crea la finestra principal
 *
**/

      private void creaFinestra()
      {
            int i;

// Crea finestra amb mides, barra de menu   i control d'events de
finestra
            finestra=new Frame("TFC copyright");
            finestra.setBounds(150,150,300,100);
            finestra.addWindowListener(this);
            menuBar();
// Crea panels amb  les seves mides  per a colocar a la finestra
// amb un layout del tipus grid
            panel1=new Panel();
```

```
            panel1.setLayout(new GridLayout(1,3));
            panel1.setBounds(150,150,300,50);
            panel2=new Panel();
            panel2.setLayout(new GridLayout(4,1));
            panel2.setBounds(150,150,300,50);
            panel3=new Panel();
            panel3.setLayout(new GridLayout(3,1));
            panel3.setBounds(150,150,300,50);
            panel4=new Panel();
            panel4.setLayout(new GridLayout(3,1));
            panel4.setBounds(150,150,300,50);
            panel5=new Panel();
            panel5.setLayout(new GridLayout(1,1));
            panel5.setBounds(150,150,300,50);
            panel6=new Panel();
            panel6.setLayout(new GridLayout(4,1));
            panel6.setBounds(150,150,50,300);
            panel7=new Panel();
            panel7.setLayout(new GridLayout(3,1));
            panel7.setBounds(150,150,50,300);
            panel8=new Panel();
            panel8.setLayout(new GridLayout(3,1));
            panel8.setBounds(150,150,50,300);
            panel20=new Panel();
            panel20.setLayout(new GridLayout(4,1));
            panel2.setBounds(150,150,300,50);
// Creem els butons i el seu control d'events
            codifica=new Button("Codifica");
            codifica.addActionListener(this);
            descodifica=new Button("Descodifica");
            descodifica.addActionListener(this);
            confabula=new Button("Confabula");
            confabula.addActionListener(this);
            ok=new Button("Ok");
            ok.addActionListener(this);
            disco1=new Button(" ... ");
            disco1.addActionListener(this);
            disco2=new Button(" ... ");
            disco2.addActionListener(this);
            disco3=new Button(".. ..");
            disco3.addActionListener(this);
            disco4=new Button(" ... ");
            disco4.addActionListener(this);
            disco5=new Button(".. ..");
            disco5.addActionListener(this);
// Creem la variable per saber a quina pantalla hi som
            onSom="Codifica";
// Creem les etiquetes
            valor1=new Label("Fitxer sense marca: ");
            valor2=new Label("Marca a introduir: ");
            valor3=new Label("Fitxer amb marca 1: ");
            valor4=new Label("Nom comprador: ");
            valor5=new Label("Fitxer amb marca 2: ");
            valor6=new Label("Resultat proces: ");
// Afegim a la finestra els panels necessaris per a la primera
pantalla
            finestra.add("North",panel1);
            finestra.add("West",panel20);
            finestra.add("Center",panel2);
            finestra.add("South",panel5);
            finestra.add("East",panel6);
```

```
// Afegim a la finestra control de popups
            alertes=new PopupMenu();
            finestra.add(alertes);
// Afegim als panels els butons
            panel1.add(codifica);
            panel1.add(descodifica);
            panel1.add(confabula);
            panel5.add(ok);
            panel6.add(disco1);
// Creem les variables de tipus choice  i TextField
            text1=new TextField(38);
            choice2=new Choice();
            for(i=0;i<8;i++){
                  choice2.add(comprador[i]);
            }
            text3=new TextField(38);
            text4=new TextField(38);
            text5=new TextField(38);
            text6=new TextField(38);
// Afegim als panels les variables choice i TextField
            panel20.add(valor1);
            panel2.add(text1);
            panel20.add(valor2);
            panel2.add(choice2);
            panel20.add(valor4);
            panel2.add(text4);
            panel20.add(valor6);
            panel2.add(text6);
// Finestra amb mida no ampliable
            finestra.setResizable(false);
// Finestra visible
            finestra.setVisible(true);
// Ajustem mida al que hi ha dins de la finesta
            finestra.pack();
      }
/**
 *     Metode que gestiona les accions a la finestra principal
 *
**/

            public void actionPerformed(ActionEvent e)
            {
                  String s;
                  int i,j     ;

// Llegim que ha fet la crida
                  s=e.getActionCommand();
                  text6.setText("");
// Es el boto Ok?
                  if(s.equals("Ok")){
                        text6.setText("Treballant...");
// Volem confabular?
                        if(onSom.equals("Confabula")){
// Tenim totes les dades?
                              if
((text3.getText().equals(""))||(text5.getText().equals(""))){
                                    text6.setText("Falten dades");
                              }
                              else{
// Fem crida a confabular i refrequem els camps TextField necessaris
```

```
                                    Confabula             confabula=new
Confabula(text3.getText(),text5.getText(),analisi.getState());
                                    i=text3.getText().indexOf(".");
                                    s=text3.getText().substring(0,i+1);
                                    j=text3.getText().lastIndexOf(".");

      s=(s.concat("conf.")).concat(text3.getText().substring(i+1,j+1))
;
                                    j=text5.getText().lastIndexOf(".");

      text6.setText(confabula.generaNovaMarca());
                                }
                          }
                          else{
// Volem codificar?
                                if(onSom.equals("Codifica")){
                                    boolean
bl=!text1.getText().equals("");
// Tenim totes les dades?
                                    if
((text1.getText().equals(""))||(text4.getText().equals(""))){
                                        text6.setText("Falten
dades");
                                    }
                                    else{
// Fem crida a codificar i refrequem els camps TextField necessaris
                                        MarcaBMP          marcaBMP=new
MarcaBMP(text1.getText(),choice2.getSelectedItem(),text4.getText(),ana
lisi.getState());

      text6.setText(marcaBMP.colocaMarca());
                                        s="";

      s=text1.getText().substring(0,(text1.getText().length()-
3)).concat(s.concat(text4.getText())).concat(".bmp");
                                        text3.setText(s);
                                    }
                                }
                                else{
// Descodifiquem!
// Tenim totes les dades?
                                    if
((text1.getText().equals(""))||(text3.getText().equals(""))){
                                        text6.setText("Falten
dades");
                                    }
                                    else{
// Fem crida a descodificar i refrequem els camps TextField necessaris
                                        MarcaBMP          marcaBMP=new
MarcaBMP(text1.getText(),text3.getText(),analisi.getState());
                                        s=marcaBMP.extreuMarca();
                                        i=s.indexOf(".");

                                        if(i>0){

      s=s.substring(i+1,s.length());
                                            s="Propietari
desconegut. Marca trobada: "+s;
                                        }
                                        text6.setText(s);
                                    }
```

```
                              }
                         }
                    }
                    else{
// Boto apretat per confabular?
                         if(s.equals("Confabula")){
// Actualitzem variable on som i cambien panels de la finestra
                              onSom="Confabula";
                              finestra.remove(panel2);
                              finestra.remove(panel3);
                              finestra.remove(panel6);
                              finestra.remove(panel7);
                              panel20.removeAll();
                              panel20.setLayout(new GridLayout(3,1));
                              panel20.add(valor3);
                              panel4.add(text3);
                              panel20.add(valor5);
                              panel4.add(text5);
                              panel20.add(valor6);
                              panel4.add(text6);
                              panel8.add(disco4);
                              panel8.add(disco5);
                              finestra.add("West",panel20);
                              finestra.add("Center",panel4);
                              finestra.add("East",panel8);
                              finestra.pack();
                         }
                         else{
// Boto apretat per codificar?
                              if(s.equals("Codifica")){
// Actualitzem variable on som i cambien panels de la finestra
                                   onSom="Codifica";
                                   finestra.remove(panel3);
                                   finestra.remove(panel4);
                                   finestra.remove(panel7);
                                   finestra.remove(panel8);
                                   panel20.removeAll();
                                   panel20.setLayout(new
GridLayout(4,1));
                                   panel20.add(valor1);
                                   panel2.add(text1);
                                   panel20.add(valor2);
                                   panel2.add(choice2);
                                   panel20.add(valor4);
                                   panel2.add(text4);
                                   panel20.add(valor6);
                                   panel2.add(text6);
                                   panel6.add(disco1);
                                   finestra.add("West",panel20);
                                   finestra.add("Center",panel2);
                                   finestra.add("East",panel6);
                                   finestra.pack();
                              }
                              else{
// Boto apretat per descodificar?
                                   if(s.equals("Descodifica")){
// Actualitzem variable on som i cambien panels de la finestra
                                        onSom="Descodifica";
                                        finestra.remove(panel2);
                                        finestra.remove(panel4);
                                        finestra.remove(panel6);
```

```
                                        finestra.remove(panel8);
                                        panel20.removeAll();
                                        panel20.setLayout(new
GridLayout(3,1));
                                        panel20.add(valor1);
                                        panel3.add(text1);
                                        panel20.add(valor3);
                                        panel3.add(text3);
                                        panel20.add(valor6);
                                        panel3.add(text6);
                                        panel7.add(disco2);
                                        panel7.add(disco3);
                                        finestra.add("West",panel20);

      finestra.add("Center",panel3);
                                        finestra.add("East",panel7);
                                        finestra.pack();
                                    }
                                    else{
// Boto apretat cerca primer fitxer?
                                        if(s.equals(" ... ")){
// Creem el fileDialog
                                              fitxer1=new
FileDialog(finestra,"Cerca del fitxer...");

      fitxer1.setVisible(true);
// Actualitzem els TextField en funció de onSom amb el valor escollit
si s'escau

      if(onSom.equals("Codifica")){

      if(fitxer1.getFile()!=null){

      text1.setText(fitxer1.getDirectory().concat(fitxer1.getFile()));
                                                    }
                                              }
                                              else{

      if(onSom.equals("Descodifica")){

      if(fitxer1.getFile()!=null){

      text1.setText(fitxer1.getDirectory().concat(fitxer1.getFile()));
                                                          }
                                                    }
                                                    else{

      if(fitxer1.getFile()!=null){

      text3.setText(fitxer1.getDirectory().concat(fitxer1.getFile()));
                                                          }
                                                    }
                                              }
                                        }
                                        else{
// Boto apretat cerca segon fitxer?
                                              if(s.equals(".. ..")){
// Creem el fileDialog
                                                    fitxer1=new
FileDialog(finestra,"Cerca del fitxer...");
```

```
      fitxer1.setVisible(true);
// Actualitzem els TextField en funció de onSom amb el valor escollit
si s'escau

      if(onSom.equals("Descodifica")){

      if(fitxer1.getFile()!=null){

      text3.setText(fitxer1.getDirectory().concat(fitxer1.getFile()));
                                                                  }
                                                            }
                                                            else{

      if(fitxer1.getFile()!=null){

      text5.setText(fitxer1.getDirectory().concat(fitxer1.getFile()));
                                                                  }
                                                            }
                                                      }
                                                      else{
// MenuBar Autor TFC?

      if(s.equals("Autor TFC")){
// Gestionem el popUp

      alertes.removeAll();

      alertes.add("Autor del TFC: Daniel Mulero Roig");

      alertes.show(finestra,150,80);
                                                            }
                                                            else{
// MenuBar esborrat BD igual a Si

      if(s.equals("Si")){
// Esborrem la BD
                                                                        new
BaseDades(nomFitxer).esborraBD();
// Creem BD buida

      this.creaFitxer();
// Gestionem el popUp

      alertes.removeAll();

      alertes.add("Base de dades esborrada");

      alertes.show(finestra,150,80);
                                                                        }
                                                            }
                                                      }
                                                }
                                          }
                                    }
                              }
                        }
            }
/**
 *     Metode que retorna el nom del fitxer
 *
```

```
**/

      public String getNomFitxer()
      {
                  return nomFitxer;
      }
/**
 *     Metode que gestiona i crea le menuBar
 *
**/

      private void menuBar()
      {
// Creem barra de menu i afegim desplegables
            barraMenu=new MenuBar();
            configura=new Menu("Configura");
            esborraBD=new Menu("Esborra BD");
// Afegim control d'events
            esborraBD.addActionListener(this);
// Creem submenus
            si=new MenuItem("Si");
            no=new MenuItem("No");
// Afegim tot alla on toca
            esborraBD.add(no);
            esborraBD.add(si);
            configura.add(esborraBD);
            configura.addSeparator();
// Fem que fer analisi sigui del tipus checkbox
            analisi=new CheckboxMenuItem("Fer analisi");
// Afegim control d'events
            analisi.addActionListener(this);
            analisi.setState(true);
            configura.add(analisi);
            barraMenu.add(configura);
//Creem desplegable ajuda
            ajuda=new Menu("Ajuda");
            autor=new MenuItem("Autor TFC");
// Afegim control d'events
            autor.addActionListener(this);
            ajuda.add(autor);
            barraMenu.add(ajuda);
// Afegim el menu a la finestra
            finestra.setMenuBar(barraMenu);
      }
/**
 *     Metode que retorna el nom del comprador
 *
**/

      public String getComprador(int valor){
            return comprador[valor];
      }
/**
 *     Metode que retorna l'estat del checkbox fer analisi?
 *
**/

      public boolean getAnalisi(){
            return analisi.getState();
      }
/**
```

```
 *     Mètode que controla l'event de la finestra quan s'està tancant
 *
**/

            public void windowClosing(WindowEvent e)
            {
                  System.exit(0);
            }
/**
 *     Mètode que controla l'event de la finestra quan és activada
 *
**/

            public void windowActivated(WindowEvent e){}
/**
 *     Mètode que controla l'event de la finestra quan està tancada
 *
**/

            public void windowClosed(WindowEvent e){}
/**
 *     Mètode que controla l'event de la finestra quan és desactivada
 *
**/

            public void windowDeactivated(WindowEvent e){}
/**
 *     Mètode  que  controla  l'event  de  la  finestra  quan  no  està
minimitzada
 *
**/

            public void windowDeiconified(WindowEvent e){}
/**
 *     Mètode  que  controla  l'event  de  la  finestra  quan  està
minimitzada
 *
**/

            public void windowIconified(WindowEvent e){}
/**
 *     Mètode que controla l'event de la finestra quan està oberta
 *
**/

            public void windowOpened(WindowEvent e){}
/**
 *    Metode main que fa les crides inicial
 *
**/

            public static void main(String[] args){

            Tfc that=new Tfc();
// Comprova que existeixi la base de dades i sino la crea
            that.creaFitxer();
// Crea la finestra principal
            that.creaFinestra();

      }
}
```