Màster Interuniversitari en Seguretat de les TIC (MISTIC)

TFM – CRIPTOGRÀFIA

“Aplicació per a dispositiu mòbil de xifrat d'imatges “

Estudiant: Joan Sergi Olona i Soto

Director: Jordi Herrera Joancomartí

Universitat: Universitat Oberta de Catalunya

Data realització: 14/06/2013

ÍNDEX

Resum 3

- Català 3

- Anglès 3

1- Introducció 4

1.1 - Antecedents 5

1.2 - Estat de l'art. 7

1.3 – Objectius del projecte 9

1.4 - Temporització i metodologia 10

2 – Fase de disseny / Fonaments teòrics 12

2.1 – Funcionament general de l'aplicació. 12

2.2 – Càmera. 13

2.3 - Algorisme d'encriptació de les imatges i gestió de contrasenyes 13

2.4 - Casos i formes d'ús 16

3 – Fase d'implementació 17

3.1 – Càmera. 17

3.2 - Algorisme d'encriptació de les imatges 23

3.3 - Gestió de contrasenyes 25

4 – Fase de prova 27

4.1 – Problemes apareguts 30

4.2 – Resolució dels problemes apareguts 31

5 – Conclusions 32

5.1 – Possibilitats de millora 33

6 – Webgrafia 34

7 – Annexes 36

7.1 – Manual d'usuari 36

RESUM DEL PROJECTE

Català

El següent projecte té per finalitat el disseny i la implementació d´una aplicació en llenguatge Android per a dispositius mòbils que , de forma autònoma, realitzi fotos i les encripti. Per funcionament autònom entenem que no utilitzi el propi software de fotografia dels dispositius, és a dir , que instal·lant l'aplicació no requereixi cap software preexistent al dispositiu.

Cada cop que es realitza una fotografia amb l'aplicació , aquesta s'encripta i queda emmagatzemada al dispositiu en format JPG . En el supòsit que algú sense autorització volgués obrir aquesta fotografia encriptada, el que es mostraria seria una imatge pixelada.

La gestió de les claus, per encriptar i desencriptar les fotografies, es fa de tal forma que no s'emmagatzemen al dispositiu per tal de donar un plus de seguretat a l'aplicació.

Anglès

This project aims to design and implement an application for Android mobile devices that, independently, take and encrypt photos. For autonomous operation we understand that the software itself takes the photos without using any existing software on the device. By installing the application will be not necessary any other software existing on the device.

Every time we take a picture with the application, it is encrypted and stored on the device as JPG. In case that someone wanted to open this encrypted photograph without permission, will be shown a pixelated image.

The management of keys to encrypt and decrypt the photos is made in such a way that will be not stored any key on the device in order to provide extra security to the application.

1- INTRODUCCIÓ

La UOC és una universitat en línia, reconeguda internacionalment, amb una comunitat de més de 60.000 estudiants.

És una universitat catalana transversal amb presència internacional, coneixedora de la diversitat del seu entorn i compromesa amb la capacitat de l'educació i la cultura per a dur a terme el canvi social que, alhora, reforça la cooperació i l'intercanvi dins de la seva comunitat universitària i amb altres universitats, institucions, la comunitat empresarial i la societat civil, forjant aliances internacionals per permetre l'ús compartit de recursos i de l'aprenentatge.

Comença la seva activitat el curs 1995-1996, amb 200 estudiants de les titulacions homologades de Psicopedagogia i Empresarials que cursaven els estudis en català. En aquest temps ha crescut i, actualment, més de 200.000 persones formen part de la comunitat universitària.

Al llarg d'aquests anys, la Universitat ha anat augmentant i diversificant l'oferta formativa, incloent-hi estudis en espanyol i anglès i, millorat el seu model educatiu, prenent l'estudiant com a centre de l'activitat d'aprenentatge. S'ha expandit progressivament al territori català i espanyol amb una extensa xarxa de seus i punts i ha fet el salt a l'àmbit internacional, consolidant una seu a Mèxic com a eix d'expansió per al mercat llatinoamericà. També ha engegat dos centres de recerca especialitzats en societat de la informació i el coneixement i en *e-learning,* de manera que s'ha convertit en un referent internacional en aquests camps.

El "Màster de Seguretat de les tecnologies de la informació i les comunicacions" (MISTIC) proporciona una formació oficial d'alt nivell i rigor acadèmic dirigida als professionals que es vulguin especialitzar en l'àmbit de la seguretat informàtica

La demanda d'enginyers, informàtics o de telecomunicacions, específicament preparats per a treballar en el camp de la seguretat de les tecnologies de la informació i de les comunicacions és, com més s´avança, més gran. D'una banda, les transaccions electròniques cada vegada són més habituals i la legislació que hi ha entorn seu és més exigent. De l'altra, les empreses són més conscients dels riscos de seguretat i, la voluntat d'invertir en sistemes de protecció, ha augmentat.
Per a donar resposta a aquesta realitat, la Universitat Oberta de Catalunya, la Universitat Autònoma de Barcelona i la Universitat Rovira i Virgili ofereixen conjuntament el màster interuniversitari de Seguretat de les tecnologies de la informació i les comunicacions (MISTIC), en el qual també participa la Universitat de les Illes Balears.
El màster ofereix unes competències generals referents a les vulnerabilitats de seguretat dels sistemes informàtics, la manera com es poden protegir aquests sistemes i aspectes de la legislació nacional i internacional relacionada amb l'àmbit de la seguretat (lleis de protecció de dades, lleis de comerç electrònic, lleis de signatura electrònica, etc.).

L'estudiant adquireix coneixements teòrics i pràctics, per la qual cosa -després de completar el programa- és capaç de dissenyar estratègies que puguin garantir la seguretat dels recursos informàtics, implantar polítiques que salvaguardin els actius empresarials i desenvolupar i desplegar solucions de seguretat en entorns reals. I tot això basant-se en els estàndards i aspectes ètico-legals que regeixen la seguretat informàtica

Aquesta memòria recull el treball de final de màster dels estudis "Màster de Seguretat de les tecnologies de la informació i les comunicacions" (MISTIC) impartits per la Universitat Oberta de Catalunya (UOC).

1.1 - Antecedents

Android[7] és un sistema operatiu basat en Linux dissenyat, principalment, per a dispositius mòbils amb pantalla tàctil, com telèfons intel·ligents i TabletPC's. Desenvolupat inicialment per Android Inc, empresa que Google recolzava financerament i que més tard, al 2005 , va comprar .

Android es va donar a conèixer en 2007 amb la fundació de l'Open Handset Alliance: un consorci de companyies de maquinari, programari i telecomunicacions dedicades a la promoció oberta d'estàndards per a dispositius mòbils. El primer telèfon amb Android va ser venut a l'octubre de 2008.

Android té una gran comunitat de desenvolupadors que escriuen aplicacions ("*apps*") que estenen les funcions dels dispositius. Hi ha, actualment, unes 300.000 aplicacions disponibles per a l'Android. Els desenvolupadors escriuen principalment en llenguatge Java.

A febrer de 2013 domina el mercat dels telèfons intel·ligents amb una quota de mercat del 53%.

Una de les pedres angulars, sobre les quals es basa el model de seguretat d'Android, és que un usuari no pot llegir o escriure en els arxius d'altres aplicacions. Per a aquesta finalitat, android utilitza el model de permisos de Linux que assigna a cada aplicació el seu propi identificador d'usuari. D'aquesta manera s'evita, en teoria, que les dades de la nostra aplicació siguin accessibles des d´aplicacions de tercers.

Aquest model funciona perfectament sempre i quan no hi hagi un "superusuari" o “root” amb accés a tot el sistema de fitxers . Per defecte, la majoria dels telèfons Android del mercat no el tenen aquest usuari , amb algunes excepcions , encara que s'han desenvolupat mètodes per obtenir-ne l´ accés (a tots els telèfons). S'ha convertit en una tasca tan fàcil que, en molts telèfons, només cal instal·lar una aplicació i fer clic a un botó.

Un altre risc de seguretat conegut és que, diverses aplicacions, com el client de correu electrònic i el navegador, emmagatzemen les contrasenyes dels usuaris sense cap tipus d'encriptació. Ho fan perquè estan confiant en la seguretat, per defecte, d'Android que garanteix que no hi hagi altres aplicacions capaces de llegir o escriure en aquests arxius. No obstant això, com hem vist abans, l'accés “ root “ en la majoria dels dispositius actuals és tan fàcil que aquesta mesura de seguretat és clarament insuficient.

Els algorismes criptogràfics són seqüències de processos, o regles, que s'utilitzen per xifrar i desxifrar missatges en un sistema criptogràfic. Són processos que protegeixen les dades assegurant-se que, persones no desitjades, no hi puguin accedir. Aquests algorismes tenen una àmplia varietat d'usos així, un dels més coneguts i aplicats, és el que permet garantir transaccions financeres segures i autenticades.

La majoria dels algorismes de criptografia impliquen l'ús d´una codificació que permet comunicar-se a dues parts, alhora que evita que, tercers no autoritzats, interceptin aquestes comunicacions. L'operació de xifrat transforma text clar, llegible per humans, en alguna cosa il·legible coneguda com a text xifrat. Així, el xifrat de dades és desxifrat, “traduït”, per tal de fer-lo comprensible al receptor. Tant el xifrat com el desxifrat operen amb algorismes.

Hi ha molts tipus diferents d'algorismes criptogràfics tot i que, la majoria d'ells, encaixen en una d´aquestes dues classificacions: simètrics o asimètrics. Alguns sistemes, però, utilitzen un híbrid d'ambdues classificacions.

Els algorismes simètrics, també coneguts com a algoritmes de clau simètrica o de clau compartida, treballen mitjançant l'ús d'una clau que només coneixen les dues parts autoritzades. Si bé, aquests poden ser implementats en forma de xifrats de bloc o xifrats de flux, la mateixa clau s'utilitza tant per xifrar com per a desxifrar el missatge. El Data Encryption Standard (DES) i Advanced Encryption Standard (AES) són els exemples més populars d'algorismes de criptografia simètrica.

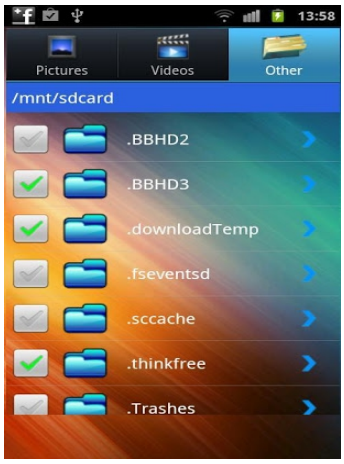
1.2 - *Estat de l'art.*

Partint de les premisses de l'enunciat del projecte i, dels fets exposats en els paràgrafs anteriors, un cop realitzat un anàlisi de les aplicacions existents al mercat per a encriptar imatges amb dispositius Android, he constatat la mínima existència d'aquestes. Essent, la majoria, encarades a l´ocultació d´arxius o, a la protecció d'aquests, mitjançant contrasenyes.

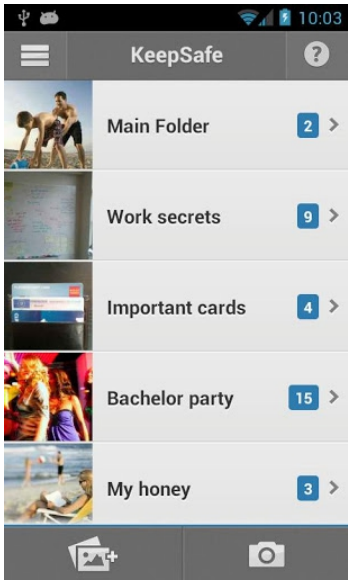
D´altra banda, no he trobat cap aplicació que acompleixi amb els requisits de partida del projecte en relació a, que la pròpia aplicació, faci les fotos. Totes treballen a partir d'arxius creats per altres aplicacions, en el nostre cas, per la pròpia càmera del dispositiu Android.
Pel que fa a la gestió de les contrasenyes no existeix cap aplicació que no guardi les contrasenyes al dispositiu.
Com a exemple il·lustratiu de les aplicacions que he trobat (totes a Google Play[10]), amb les característiques esmentades, presento les següents tres aplicacions:

- File Encrypter : Utilitzant l'estàndard de xifrat AES, xifra arxius fent-los impossibles de llegir o, fer servir, sense la contrasenya correcta. La contrasenya s'emmagatzema al dispositiu. Els arxius són creats per altres aplicacions.

-Super Vault : encripta arxius amb l'algorisme AES i, a més, canvia l'extensió de l'arxiu per ocultar-lo/s a simple vista. La contrasenya d'encriptació s'emmagatzema al dispositiu. L'aplicació no crea cap dels arxius a encriptar.



- Keep Safe : oculta imatges en directoris ocults. L'accés es fa mitjançant una contrasenya que no està encriptada i, a més, s'emmagatzema al dispositiu. Les imatges són creades per la aplicació de la càmera del dispositiu.



1.3 Objectius del projecte

Els objectius a assolir amb aquest projecte són els següents:

- Pel que fa a l'aplicació:

+ Crear una aplicació autònoma dins el dispositiu mòbil: Implementació d'una aplicació sense dependències , instal·lable en qualsevol dispositiu androide amb dispositiu de càmera.

+ Nivell de seguretat alt, tant en el moment d'encriptar les imatges, com en el de gestió de claus: Èmfasi en la seguretat. És prioritzarà tant l'algorisme d'encriptat de les imatges, com la seguretat alhora d'emmagatzemar i recuperar les contrasenyes d'encriptació.

+ Eficiència en quant a recursos consumits: Consum mínim de recursos del dispositiu i una velocitat de funcionament dins d'uns paràmetres raonables.

- Respecte als coneixements a adquirir:

+ Obtenir desimboltura a l'hora de programar aplicacions en Android.

+ Ampliació de coneixements en criptografia.

+ Conèixer els procediments i les fases del disseny i la implementació d'una aplicació per a un dispositiu mòbil

2.*4 – Temporització i metodologia*

Partint de l'anàlisi previ mostrat a l'apartat 3.1 i dels objectius marcats a l'apartat 3.2, estructuro el projecte en les següents fases:

- 1a FASE - Recollida d'informació:
  + Sobre la programació en android.
  + Algorismes d'encriptació.
  + Mètodes de gestió de claus.

- 2na FASE - Disseny:
  + De l'aplicació base (càmera).
  + Algorisme d'encriptació en android.
  + Sistema de gestió de claus.
  + Elaboració de l'index de la memòria del projecte i, complementació dels apartats de la memòria referents a les dues primeres fases del projecte.

- 3ra FASE – Implementació en android:
  + De l'aplicació base (càmera).
  + Algorisme d'encriptació.
  + Sistema de gestió de claus.
  + Complementació dels apartats de la memòria referents a la tercera fase del projecte.

- 4ta FASE – Prova de l'aplicació:
  + Provar l'aplicació a la recerca d´errors de disseny i/o funcionalitat.
  + Complementació dels apartats de la memòria referents a la quarta fase del projecte.

- 5na FASE – Memòria final:
  + Elaboració de la versió final de la memòria que reculli tot el procediment seguit per a elaborar l'aplicació i, una documentació de la mateixa.
  + Elaboració d'una presentació virtual de l'aplicació.

A partir del comentat en el paràgraf anterior, estableixo la següent temporització:

- 1a FASE i 2na FASE
  Inici: 22/03/2013
  Finalització: 22/04/2013

- 3ra FASE
  Inici: 23/04/2013
  Finalització: 31/05/2013

- 4ta FASE i 5ena FASE
  Inici: 01/06/2013
  Finalització: 14/06/2013

Les dues primeres fases segueixen una metodologia eminentment de recerca teòrica, és a dir , es tracta d'una recerca exhaustiva d'informació relativa al projecte i dun disseny teòric de l'aplicació.

La tercera fase , és de caràcter totalment pràctic . Un cop realitzat el disseny , mitjançant l'entorn de treball "Eclipse " (www.**eclipse**.org/) amb un plugin per al llenguatge android´, prèviament instal-lat (ADT plugin[6]), és duu a terme l'implementació de l'aplicació.

Destaco la metodologia seguida, en el moment d'elaborar la memòria del projecte, per tal que sigui un document viu que em permeti acomplir amb tots els objectius marcats i seguir una metodologia, de treball, estructurada i coherent. L'actualitzo al final de cada fase amb la feina feta fins a aquell punt.

Finalment, les dues ultimes fases , la quarta i la cinquena, són fases de conclusió , és a dir , es duen a terme les últimes proves, amb l'aplicació ja finalitzada, per tal de comprovar el seu correcte funcionament i s'elabora, la versió final, de la memòria del projecte.

2 - FASE DE DISSENY / FONAMENTS TEÒRICS

2.1 - Funcionament general de l'aplicació.

S´ha intentat simplificar el funcionament al màxim. Quan s'executi l'aplicació, apareixerà una pantalla que ens demanarà que introduïm la contrasenya per a encriptar i desencriptar. Des d´aquí, accedirem a un menú on podrem escollir entre, fer una foto o, desencriptar una imatge.

La funcionalitat de la càmera consistirà en capturar imatges i emmagatzemar-les, encriptades, en una carpeta a la memòria externa del dispositiu.

Si escollim desencriptar una imatge, s'agafarà la imatge de la carpeta creada per la càmera, es tractarà i s'emmagatzemarà a la carpeta .

Les imatges es podran visualitzar des de la mateixa galeria d'imatges del dispositiu. En cas que la imatge estigui encriptada , observarem un símbol de “imatge no disponible “.

A més a més, s'implementarà una funcionalitat que permetrà convertir, una imatge encriptada, en una imatge pixelada, mostrant-se també a la galeria d´imatges del dispositiu.

2.2 – Càmera

Android inclou suport per a diverses càmeres i funcions, disponibles en els dispositius, cosa que permet capturar fotos i vídeos a les seves aplicacions.

Utilitzaré la càmera hardware[1] ,del dispositiu mòbil, per a crear una interfície d'usuari que ens permeti previsualitzar el que es vol fotografiar i, a través d'un botó a la mateixa pantalla, disparar l'obturador de la càmera.

S'utilitza la càmera hardware per donar independència a l'aplicació, és a dir , un cop instal·lada l'aplicació no necessitarem cap altra aplicació del dispositiu per al seu correcte funcionament.

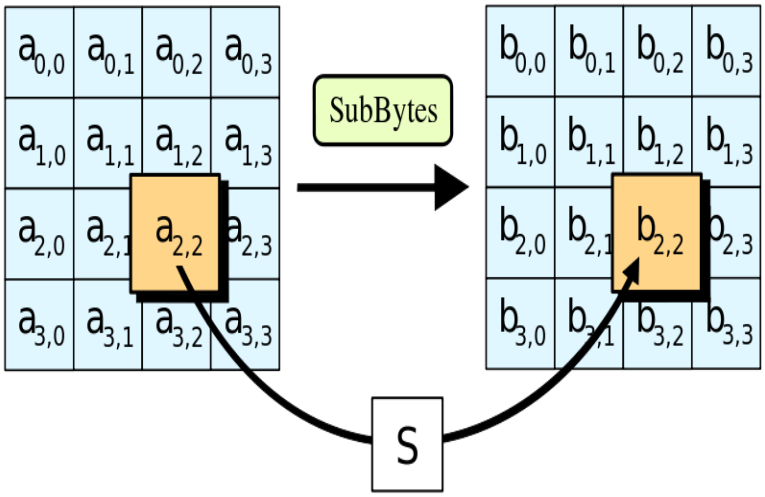
2.3 - Algorisme d'encriptació de les imatges i gestió de contrasenyes

L'aplicació utilitzarà l'algorisme AES[4][9] per a la encriptació de les imatges.
L'AES és el millor esquema de xifrat en bloc que es coneix actualment. Va ser escollit al 2001, després d'una competició oberta públicament a tothom, en la que hi van participar experts criptògrafs d´arreu del món i va durar, gairebé, 5 anys.

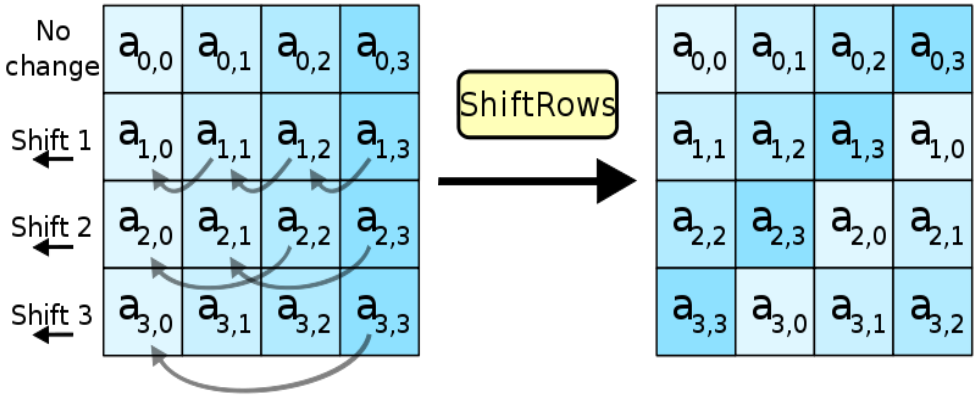
L'Advanced Encryption Standard (AES), també conegut com Rijndael, és un esquema de xifrat per blocs. És un dels algorismes més populars usats en criptografia simètrica.

L'algorisme AES encripta blocs de 128, 192, o 256 bits, usant claus de 128, 192 o 256 bits. El procés consisteix en una sèrie de quatre transformacions matemàtiques, les quals es repeteixen 10, 12 o 14 vegades, depenent de la longitud del bloc i de la longitud de la clau. Tots els cicles, excepte l'últim, són similars i consisteixen en les següents transformacions:

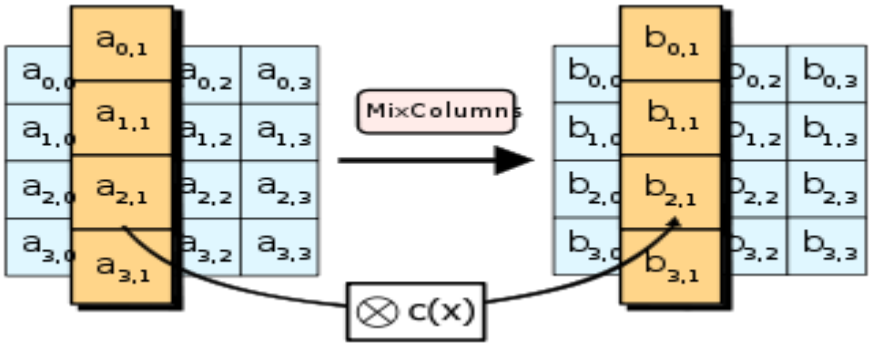
- Transformació ByteSub (Substitució de bytes).



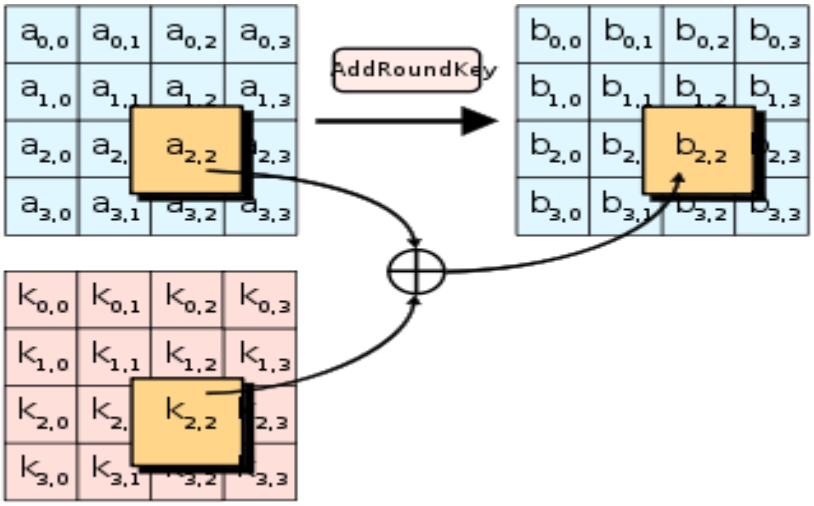
- Transformació ShiftRow (Desplaçament de files).



- Transformació MixColumns (Multiplicació de columnes).



- Transformació AddRoundKey (S'aplica una or-exclusiva entre els bits del text i la clau).



En l'aplicació utilitzaré un xifrat de 256 bits. De tota manera, l'aplicació implementarà un mètode per a que, en el cas que no fos possible utilitzar el xifrat de 256 bits , l'utilitzés de 192 bits i, si tampoc fos possible, l'utilitzaria de 128 bits.

Per a suplir la manca de seguretat en l'emmagatzenament de les contrasenyes al dispositiu mòbil, treballaré amb una variable que emmagatzemarà la contrasenya d'encriptació, mentre l'aplicació estigui funcionant. És a dir, l'aplicació sol·licitarà a l'usuari la contrasenya d'encriptació al accedir a l'aplicació. Aquesta es transmetrà a la classe corresponent i, al sortir de l'aplicació ,el valor d'aquesta variable serà eliminat.

No he previst cap altre tipus de sistema de seguretat , com ara un " login " d'accés a l'aplicació , ja que aquesta està pensada per a un entorn de treball monousuari.

En el cas que a l'usuari l'hi robessin el dispositiu , si aquest s'ha assegurat de sortir de l'aplicació, no hi hauria el perill d'un accés a la contrasenya d'encriptació i desencriptació i per tant , a les imatges encriptades emmagatzemades al dispositiu.

2.4 - Casos i formes d'ús

L'aplicació està pensada per a un entorn professional. Dos o més professionals podran dur a terme un intercanvi d'imatges de manera segura en temps real.
Es pressuposa que els professionals són d'un àmbit industrial i que, per tant, les fotografies preses seran de coses petites o properes.

El procediment d'accés i us de l'aplicació serà el següent:

1- L'usuari accedirà a l'aplicació i se l'hi mostrara una pantalla on haurà d'introduir la contrasenya d'encriptació/desencriptació. Aquesta contrasenya podrà ser diferent cada cop, només caldrà tenir en compte que haurà de ser la mateixa en el procés d´encriptar i desencripatr la mateixa imatge.

2 – Visualitzarà un menú amb dos camps: "Foto" i "Desencriptar".

+ Foto --> Des d´ aquest apartat es realitzaran, s'encriptaran i s'emmagatzemaran, a la targeta de memòria del dispositiu, les imatges .

+ Desencriptar --> A través d'aquest menú, es podran desencriptar les imatges.

3 – Es treballarà cada cop amb una imatge

4 – Un cop finalitzada la sessió de treball, l'usuari tancarà l'aplicació per tal d'eliminar la contrasenya d'encriptació/desencriptació.

3 – FASE D'IMPLEMENTACIÓ

3.1 – Càmera.

He decidit implementar una càmera autònoma[1], de la pròpia aplicació de càmera dels dispositius, per tal de poder encriptar la imatge directament quan és capturada, sense guardar-la prèviament a la targeta de memòria del dispositiu. Amb aquesta característica afegeixo un plus de seguretat a l´aplicació , doncs, si abans d'encriptar la imatge la guardés a la targeta de memòria, encara que un cop encriptada elimines la imatge en clar, aquesta podria ser recuperada de la targeta, a no ser que és fes un esborrat de seguretat consistent en reescriure els sectors de la targeta on s'emmagatzemava la imatge. Per altra banda, he considerat més eficient a nivell de funcionament, implementar una càmera personalitzada, que un mètode de esborrat de seguretat.

El procés que he seguit per a implementar la càmera ha estat el següent;

- Crear una classe “ CameraActivity ” que s'encarregarà de :

+ Comprovar si existeix la càmera i sol·licitar l'accés a la càmera.

+ Connectar els “listners “ amb els controls per iniciar la previsualització d´imatges i la captura. Els controls , en aquest cas, seran botons que respondran quan l'usuari faci click.

+ Cridar els mètodes de encriptació.

+ Fer les fotografies i emmagatzemar el resultat.

+ Alliberar la càmera. És a dir , quan l'aplicació deixi d'utilitzar la càmera ,alliberar-la per a que la puguin utilitzar altres aplicacions.

- Crear una Class de vista prèvia (preview): Aquesta class estén SurfaceView i implementa la interfície SurfaceHolder. Generarà les vistes prèvies de les imatges en temps real.

- Construir un Layout: Crear un disseny de vista que incorpora la vista prèvia i els controls de la càmera, com ara el botó per fer una captura.

Seguidament comentaré les parts més importants de codi de la classe “ CameraActivity”.

Amb el següent mètode comprovo l'existència de la càmera i sol-licito accés. En cas que no existeixi o no pugui accedir-hi, em retornarà “ null “

```
public static Camera getCameraInstance(){
    Camera c = null;
    try {
        c = Camera.open();
    }
    catch (Exception e){

    }
    return c;
}
```

Per recuperar una imatge, utilitzo el mètode Camera.takePicture (). Aquest mètode pren tres paràmetres que reben dades de la càmera. Per rebre les dades en format JPEG, s'ha d'implementar una interfície Camera.PictureCallback per rebre les dades de la imatge i gravar-la en un arxiu. Incideixo en el fet que tal com s'han rebut les dades capturades per la càmera s'han enviat al mètode "*encryptBytes()*" per encriptar-les i posteriorment emmagatzemar-les. El codi mostra la implementació de Camera.PictureCallback per guardar una imatge rebuda de la càmera, encriptar-la i guardar-la en un arxiu anomenat "abc.jpg".

```
 final PictureCallback mPicture = new PictureCallback() {

      @Override
        public void onPictureTaken(byte[] data, Camera camera) {

            try {

                  String k=CryptoAES.getKEY();
                  byte[] increpT = CryptoAES.encryptBytes(k, data);
                  byte[] d=increpT;

                  File mediaStorage = new
File(Environment.getExternalStoragePublicDirectory(
                        Environment.DIRECTORY_PICTURES), "Camera1");
                  if(!mediaStorage.exists())
                {
                    if(mediaStorage.mkdir())
                      {
                       //directory is created;
                      }

                }
                  FileOutputStream fts = new FileOutputStream(new
File(mediaStorage.getPath() + File.separator +"abc.jpg"));
                  fts.flush();
                  fts.write(increpT);
                fts.close();
                sendBroadcast(new Intent(Intent.ACTION_MEDIA_MOUNTED,
Uri.parse("file://"+ Environment.getExternalStorageDirectory())));


                  ByteArrayOutputStream bos = new ByteArrayOutputStream();
                  BitmapEncoder.encodeToBitmap(d,bos);
                  bos.toByteArray();

                   File mediaStorage1 = new
```

```
File(Environment.getExternalStoragePublicDirectory(
                            Environment.DIRECTORY_PICTURES), "BitMap");

                    if(!mediaStorage1.exists())
                     {
                         if(mediaStorage1.mkdir())
                           {
                            //directory is created;
                           }

                     }
                     FileOutputStream ft = new FileOutputStream(new
File(mediaStorage1.getPath() + File.separator +"ab.jpg"));
                    bos.writeTo(ft);
                    bos.flush();
                    ft.flush();

                    bos.close();
                    ft.close();
                   sendBroadcast(new Intent(Intent.ACTION_MEDIA_MOUNTED,
Uri.parse("file://"+ Environment.getExternalStorageDirectory())));

                  mCamera.startPreview();

             } catch (FileNotFoundException e) {
                 Log.d(TAG, "File not found: " + e.getMessage());
             } catch (IOException e) {
                 Log.d(TAG, "Error accessing file: " + e.getMessage());
             } catch (Exception e) {
                         // TODO Auto-generated catch block
                         e.printStackTrace();
                    }
           }
         };
```

Per cridar aquest mètode ,utilitzo el botó “ Captura “ . I el crido de la següent forma:

```
captureButton = (Button) findViewById(id.button_capture);
                    captureButton.setOnClickListener(
                              new View.OnClickListener() {
                                  @Override
                                  public void onClick(View v) {

                                      mCamera.takePicture(null, null,mPicture);


                  }
             }
         );
```

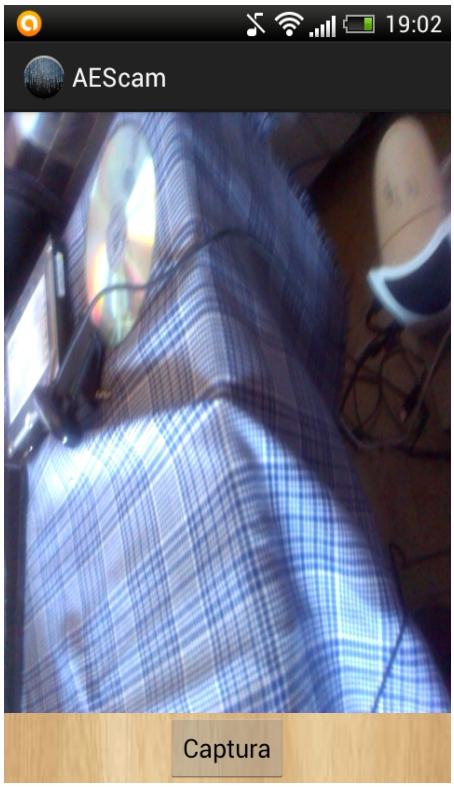
Les imatges encriptades les guardo al directori d'emmagatzematge extern del dispositiu (targeta SD) per estalviar espai al sistema i perquè els usuaris puguin accedir a aquests arxius sense el dispositiu.
Per tal d'obtenir la ubicació estàndard per a guardar imatges, utilitzo el mètode Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES). Aquest directori és compartit (públic), per tant, altres aplicacions podran accedir fàcilment als arxius guardats en aquesta ubicació. Si l'aplicació es desinstal·la , no s'eliminaran els arxius multimèdia guardats en aquesta ubicació. Per evitar interferències o confusions amb fotos i vídeos existents, he creat un subdirectori per als arxius de la meva càmera, aquest s'anomena “ Camera1 “.

La càmera és un recurs compartit amb altres aplicacions del dispositiu, per tant, s'ha de tenir especial cura en alliberar-la quan l'aplicació deixa de utilitzar-la, i tan aviat com s'atura l'aplicació (Activity.onPause ()). Si l'aplicació no allibera correctament la càmera, tots els intents posteriors per accedir a la càmera, inclosos els de la seva pròpia aplicació, produirien un error.

Per alliberar la càmera he utilitzat aquest codi:

```
@Override
protected void onPause() {
    super.onPause();
    releaseCamera();
}

private void releaseCamera(){

       if (mCamera != null){
        mCamera.release();
        mCamera = null;
    }
}
```

El layout implementat ,constarà d'una pantalla que mostrarà la previsualització de la càmera i un botó de captura. Té aquesta aparença:



Finalment , afegeixo al AndroidManifest els permisos necessaris per a accedir a la càmera i poder escriure fitxers a la targeta SD:

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

També afegiré la següent línia per a que “ Google Play” no permeti instal·lar l'aplicació si no existeix una càmera al dispositiu o aquesta no compleix amb els requisits de l'aplicació.

```
<uses-feature android:name="android.hardware.camera" />
```

3.2 - Algorisme d'encriptació de les imatges

M'he decantat per la utilització de l'algorisme d'encriptació AES perquè és el que he considerat més robust. Fins al 2009, només s'ha constatat un atac reeixit contra l'AES i no va ser contra l'algorisme pròpiament dit, sinó contra algunes de les seves implementacions. L'Agència de Seguretat Nacional dels Estats Units (NSA) al 2003, va declarar que era prou segur per al seu ús en la protecció d´informació classificada del govern dels Estats Units

Per implementar la funcionalitat d'encriptació i desencriptació de les imatges, he utilitzat la classe "CryptoAES[8] " que conté els mètodes d'encriptació i desencriptació. Aquests mètodes són cridats ,d´una banda per la classe "CameraActivity" per a fer l'encriptació i de l'altra, per la classe "MainActivity" per a realitzar la desencriptació.

Considero com a més important, la part de codi que genera la contrasenya d´ encriptació i desencriptació que, seguidament, comentaré. La resta de mètodes es poden veure directament al codi font que acompanya aquesta memòria.

Amb la finalitat de generar la clau AES utilitzo el mètode " `getRawKey(byte[] seed)` ". Aquest mètode mitjançant la llavor o "seed " (és a dir, el contingut que em fixat a la variable " KEY " de la classe "CryptoAES" convertit a byte[]) i un numero aleatori generat amb " `SecureRandom.getInstance("SHA1PRNG");`"[3] ,primerament , genera un objecte aleatori a partir dels 8 bytes del "seed " i el nombre aleatori creat amb "SecureRandom ".

La classe " SecureRandom" genera un numero aleatori criptogràficament fort. En el meu cas , aquest numero es genera utilitzant l'algorisme de generació de números pseudoaleatoris " SHA1PRNG ".

Seguidament, amb el mètode " `init(256, sr)`" de la classe " KeyGenerator "[2] s'inicialitza ,juntament amb el numero aleatori , una clau de 256 bits.

Finalment, es genera la clau amb el mètode "`generateKey()`" de la classe "KeyGenerator".

Primer provarà a generar una clau de 256 bits , sinó pot, ho provarà amb 192 bits i si tampoc pot, la generarà de 128 bits.

El codi complet del mètode és:

```
private static byte[] getRawKey(byte[] seed) throws Exception {
    KeyGenerator kgen = KeyGenerator.getInstance("AES"); // , "SC");
    SecureRandom sr = null;
    if (android.os.Build.VERSION.SDK_INT >= JELLY_BEAN_4_2) {
        sr = SecureRandom.getInstance("SHA1PRNG", "Crypto");
    } else {
        sr = SecureRandom.getInstance("SHA1PRNG");
    }
    sr.setSeed(seed);
    try {
        kgen.init(256, sr);

    } catch (Exception e) {

        try {
            kgen.init(192, sr);
        } catch (Exception e1) {

            kgen.init(128, sr);
        }
    }


    SecretKey skey = kgen.generateKey();
    byte[] raw = skey.getEncoded();
    return raw;
}
```

Un dels requisits de l'enunciat del projecte és mostrar la imatge encriptada com un seguit de píxels de colors a la galeria d'imatges del dispositiu. Per a dur a terme aquest objectiu he decidit inserir una capçalera o " header " de BitMap (format gràfic BMP) al principi de l'arxiu que conté la imatge encriptada. M'he decantat per aquesta solució ja que es tracta de la més senzilla d'implementar . Vaig sospesar altres possibilitats ,com ara la esteganografia (ocultar una imatge dins d'una altra) o l´ encriptació ,només, de les dades de la imatge ,deixant en clar les capçaleres.

D'aquest procés se n´ocupa la classe "BitmapEncoder "[5] que és cridada des de " CameraActivity " just després d'encriptar la imatge mitjançant :

```
BitmapEncoder.encodeToBitmap(d,bos);
```

On " d" és la imatge encriptada i "bos " el stream de sortida per emmagatzemar el nou arxiu generat.

Els mètodes de la classe " BitmapEncoder " codificaran l'arxiu encriptat en un mapa de bits (Bitmap) de 24 bits. Generaran una capçalera BMP i després l'escriuran al principi de l'arxiu BMP, com si es tractés de les dades d'una imatge RGB.

3.3 - Gestió de contrasenyes

La contrasenya d'encriptació i desencriptació es guarda a la variable " KEY " de la classe "CryptoAES " en forma de string. Per a poder accedir a la variable des de fora de la classe he implementat el següent " getter " a dins la classe " CryptoActivity ":

```
public static String getKEY() {
return KEY;
}
```

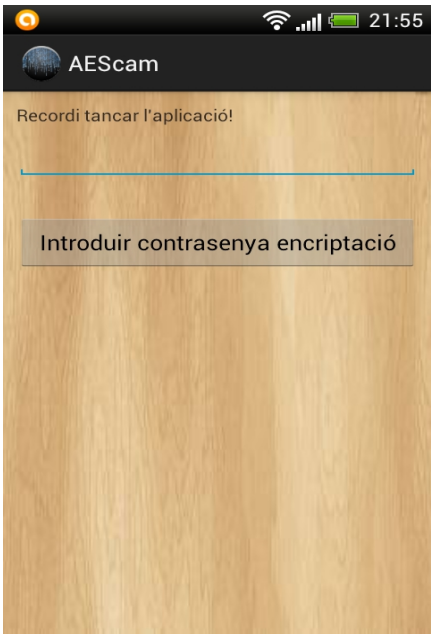
Al accedir a l'aplicació, la primera activitat que s'executa és "KeyActivity", on mostro un layout amb un camp de text on s'introduirà la contrasenya i un botó que cridarà al mètode comentat en el paràgraf anterior i ens enviarà a la següent activitat que és " MainActivity ".

El listener associat al botó és :

```
public OnClickListener btnInListner = new OnClickListener() {

              public void onClick(View v) {

                  CryptoActivity.setKEY(pass.getText().toString());
                   Intent j = new Intent(getApplicationContext(),
MainActivity.class);
                        startActivity(j);
                  try {

                  } catch (Exception e) {
                      e.printStackTrace();
                  }
              }
          };
```

La variable “ KEY “ l'he definit com una variable “static “ , cosa que implica que al tancar l'aplicació s'eliminarà el valor associat a aquesta variable . Per tant, no quedarà emmagatzemat al dispositiu el valor de la contrasenya d'encriptació i desencriptació., donant així per satisfeta una de les premisses de l'enunciat del projecte.

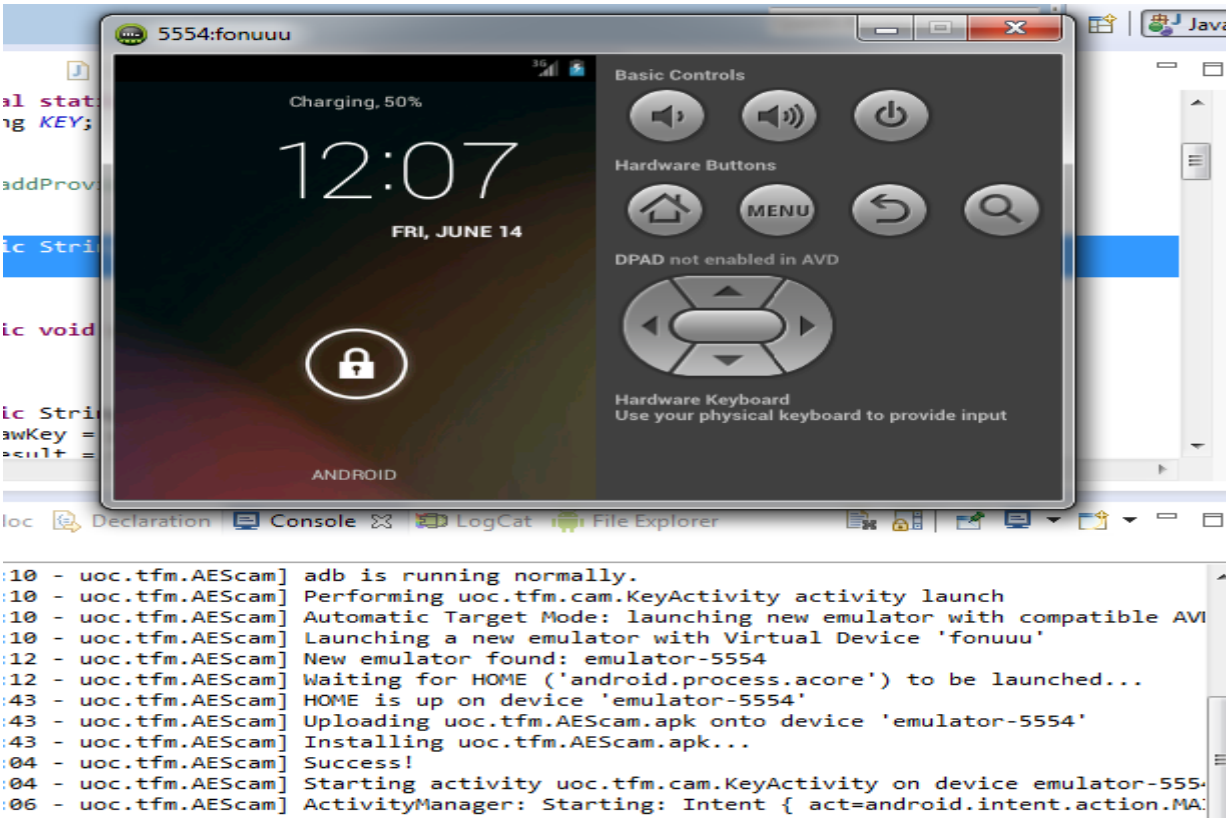
El layout de la activitat “ KeyActivity “ se'ns mostrarà així:

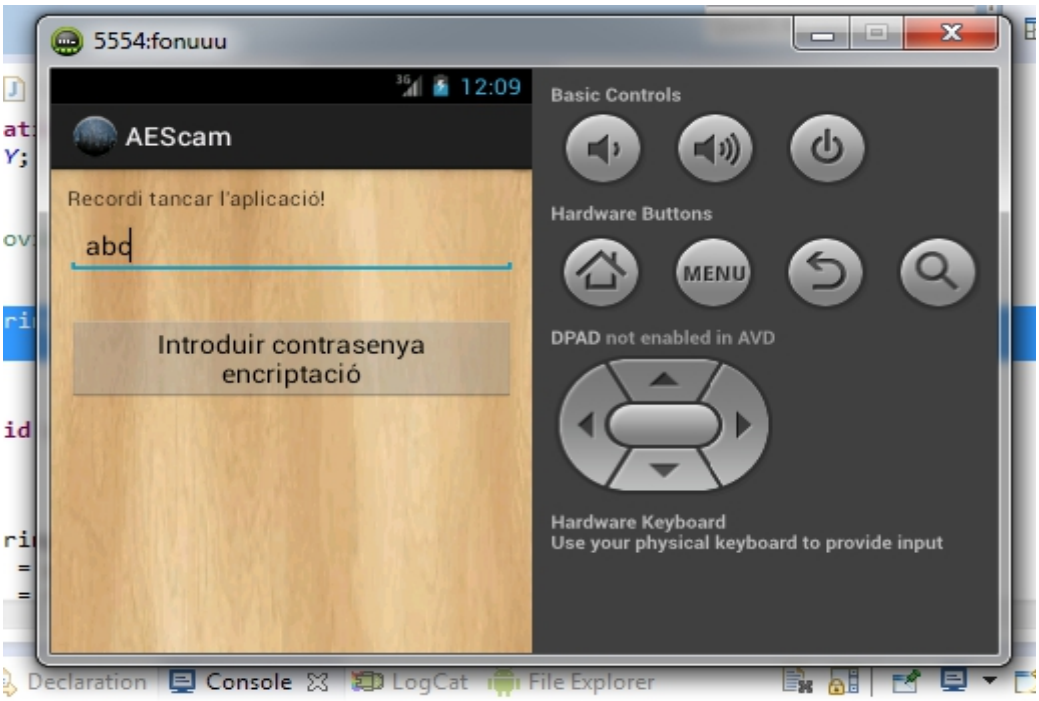


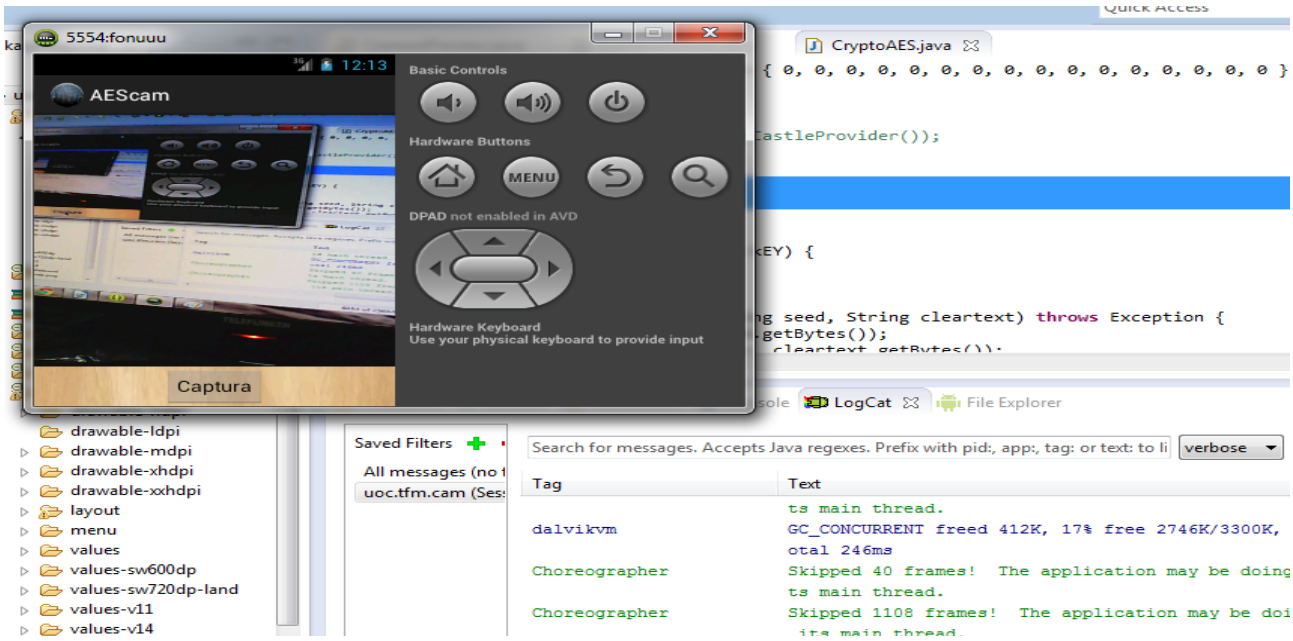
4 – FASE DE PROVA

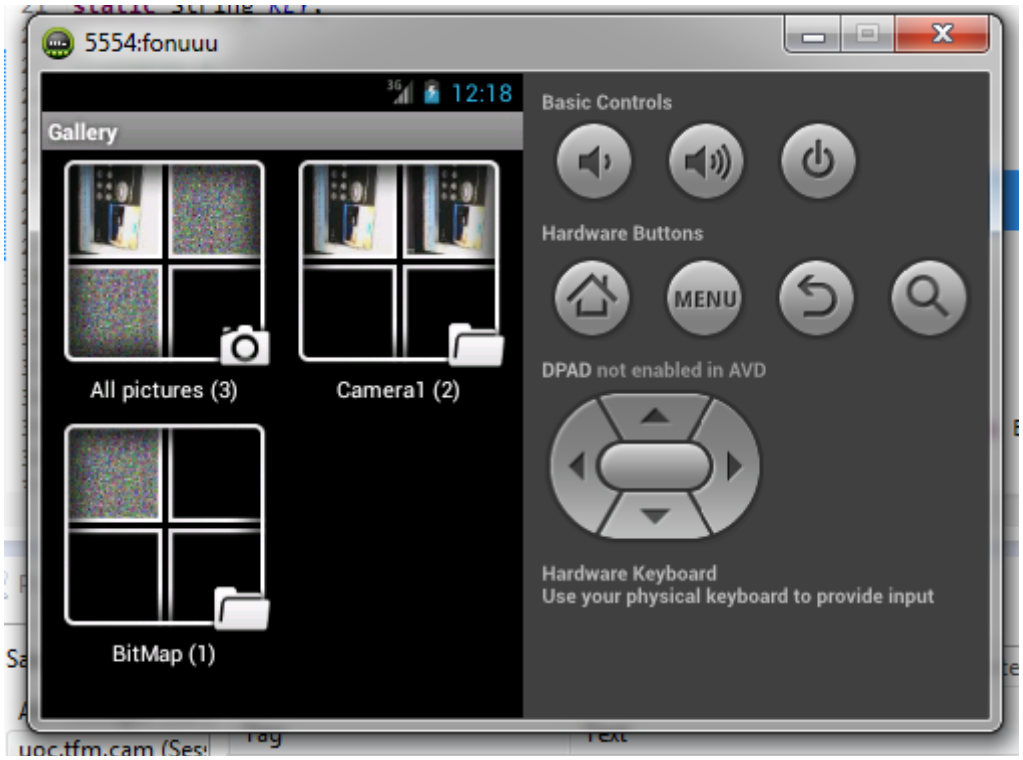
La metodologia que he seguit durant la fase d´ implementació i posteriorment en la de prova, ha consistit en implementar les diferents funcionalitats per separat i un cop funcionaven correctament, unir-les en un únic projecte.

La fase de prova l'he tractat des de dues vessants diferents, per una banda, a mesura que anava implementant funcionalitats les anava provant amb l'emulador d'android AVT i simulant la càmera del dispositiu amb una webcam connectada a l'ordinador, per acabar provant ,finalment, tot el projecte al esmentat emulador. I per altra banda , un cop validat el funcionament de l'aplicació amb l'emulador, he provat l'aplicació sencera en un dispositiu android real, en aquest cas un dispositiu HTC desire X amb la versió del sistema operatiu android 4.0 Ice Cream Sandwich .
Seguidament mostro les captures de pantalla del procés de prova mitjançant l'emulador de android:




5554:fonuuu
12:09
AEScam
Recordi tancar l'aplicació!
abc
Introduir contrasenya encriptació
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input
Declaration
Console
LogCat
File Explorer


5554:fonuuu
12:13
AEScam
Captura
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard
Use your physical keyboard to provide input
CryptoAES.java
LogCat
File Explorer
Saved Filters
Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to li
verbose
Tag
Text
drawable-ldpi
drawable-mdpi
drawable-xhdpi
drawable-xxhdpi
layout
menu
values
values-sw600dp
values-sw720dp-land
values-v11
values-v14



Destaco que en aquesta darrera captura de pantalla on es mostra la galeria d´imatges, dins la carpeta “Camera1”, hi ha la imatge encriptada i la imatge desencriptada. Tot-hi que els thumbnails es vegin iguals (amb la imatge desencriptada) al accedir-hi, un és la pantalla negra de “imatge no disponible “ (de fet és el que veurem que es mostra en un dispositiu real) i l'altra és la imatge que correspon al thumbnail

Per mostrar el funcionament en un dispositiu HTC ,he adjuntat a la memòria un arxiu de vídeo .

Durant la fase de prova en el dispositiu HTC , també he realitzat una prova per assegurar l'efectivitat de la contrasenya d´encriptació i desencriptació. He provat a desencriptar una imatge amb una contrasenya diferent a la d´encriptació, el resultat ha estat negatiu , no he pogut desencriptar la imatge.

També, s'ha provat l'aplicació en dispositius diferents al meu, en aquest cas s'ha provat en un Samsung Galaxy S3 , un Samsung Galaxy S3 mini i un Nexus 7. En el primer dispositiu , l'aplicació no ha funcionat i en els altres dos aparells , ha funcionat correctament. He adjuntat a la memòria un parell de videos que mostren el correcte funcionament en el Samsung Galaxy S3 mini i en el Nexus 7

Destaco, que s'ha observat lentitud a l'hora d'actualitzar el contingut de la galeria d ´imatges en aquests dos aparells, no sent així en el HTC Desire X. Aquesta lentitud, no repercuteix en el moment de crear els arxius al dispositiu. Mitjançant " FileExplorer" de l'entorn " Eclipse" he pogut constatar que els arxius es creaven automàticament al dispositiu en prémer els diferents botons de la aplicació.

4.1 – Problemes apareguts

Deixant de banda els errors associats a la programació, com ara sintaxis errònies amb el llenguatge Java i errors de configuració de l'entorn de treball , em centraré en els problemes més destacats en quan a implementació que m'he trobat.

El primer problema que em va sorgir va ser durant la implementació de la funcionalitat de la càmera i,per extensió, de l´encriptació i desencriptació i la conversió de l'arxiu a mapa de bits. Em trobava que, tot-hi guardar correctament els arxius, això ho podia constatar gràcies al " File Explorer" de l'entorn de treball " eclipse " , les imatges no es mostraven a la galeria d´imatges fins que no reiniciava l´emulador AVT o provocava, de forma manual, el muntatge de la targeta SD.

Un altre problema que em va sorgir va ser a l'hora de generar el mapa de bits . La meva idea era utilitzar la classe "BitmapFactory " per codificar un arxiu BMP a partir de la imatge encriptada. L'únic que vaig aconseguir va ser generar una imatge tota negra.

Finalment, al provar l'aplicació al dispositiu Samsung Galaxy S3 s'ha comprovat que en aquest dispositiu l'aplicació no funciona. Aparentment al clickar sobre el botó “Captura”,l ´aplicació no fa res.

4.2 – Resolució dels problemes apareguts

El primer problema el vaig poder corregir inserint una línia de codi després de tancar cada stream de sortida al guardar els arxius que, fa actualitzar el contingut de la targeta SD.

La línia és aquesta:

```
 sendBroadcast(new Intent(Intent.ACTION_MEDIA_MOUNTED, Uri.parse("file://"+
Environment.getExternalStorageDirectory())));
```

Amb això vaig aconseguir que cada cop que fes una operació sobre un arxiu i aquest es guardés,seguidament és mostres a la galeria d´imatges del dispositiu.

L'altre problema el vaig solucionar fent un canvi de plantejament del problema. Vaig optar per la solució explicada en apartats anteriors i consistent a afegir una capçalera o “ header “ de mapa de bits a l'arxiu encriptat.

Pel que fa al funcionament en un dispositiu Samsung Galaxy S3, no hi he pogut fer res al respecte al no poder experimentar amb el dispositiu. Seria interessant comprovar mitjançant un gestor de fitxers, si els arxius es generen tot-hi no mostrar-se a la galeria d'imatges.

5 – CONCLUSIONS

Avaluant els objectius plantejats al principi del projecte, he arribat a les següents conclusions.

- Respecte a l´aplicació:

    + L'aplicació és totalment autònoma. Només requereix que el dispositiu disposi de càmera.

    + El nivell de seguretat és alt. L'algorisme d'encriptació i desencriptació és robust i la clau utilitzada no s'emmagatzema al dispositiu.

    + L'aplicació consumeix pocs recursos ,oferint una velocitat de treball raonable i sent eficient.

- Respecte als coneixements a adquirir:

    + Posseeixo un nivell alt de desimboltura en l'entorn de treball android i de programació en aquest llenguatge.

    + He aprés com implementar l'algorisme AES per a encriptar i desencriptar arxius.

    + He adquirit una bona base en quant a conèixer els procediments , les fases del disseny i la implementació d'una aplicació per a un dispositiu mòbil

En resum, he finalitzat el projecte amb un grau de satisfacció alt. Per una banda , he creat una aplicació que acompleix amb tots els requisits de l'enunciat i per l'altra, partint de coneixements gairebé inexistents en android i el seu entorn de treball, he adquirit una bona base per a futurs projectes en aquest camp.

5.1 – Possibilitats de millora

Tot-hi que l'index de satisfacció és alt, penso que es podria millorar l'aplicació. Els aspectes que m'agradaria tenir en compte de cara a futures modificacions de l'aplicació serien els següents:

- Convertir l'aplicació en multi-usuari afegint un login d'accés i fent una gestió de les contrasenyes a través d´una base de dades implementada en un servidor web.

- Poder treballar amb diverses imatges alhora,podent escollir ,per exemple, quina o quines imatges es volen desencriptar.

- Crear una galeria d'imatges pròpia de l'aplicació.

6 – WEBGRÀFIA

[1] Media and camera.Android developers. Darrer accés el 21 de maig del 2013. Disponible a:

http://developer.android.com/guide/topics/media/camera.html#custom-camera

[2] Class KeyGenerator. Java Platform. Darrer accés el 13 de juny del 2013. Disponible a:

http://docs.oracle.com/javase/6/docs/api/javax/crypto/KeyGenerator.html#init(int, java.security.SecureRandom)

[3] Class SecureRandom. Java Platform. Darrer accés el 13 de juny del 2013. Disponible a:

http://docs.oracle.com/javase/6/docs/api/java/security/SecureRandom.html#setSeed(long)

[4] Advanced Encryption Standard. Wikipedia. Darrer accés el 18 de març del 2013 Disponible a:

https://ca.wikipedia.org/wiki/Advanced_Encryption_Standard

[5] Java: Encode any byte array, stream or file into a 24-bit bitmap image (BMP). Yet another Web Log. Darrer accés el 13 de juny del 2013. Disponible a:

http://blog.philippheckel.com/2013/03/02/java-encode-any-byte-array-stream-or-file-into-a-24-bit-bitmap-bmp/

[6] Setting Up the ADT Bundle.Android developers. Darrer accés el 23 de abril del 2013 Disponible a:

http://developer.android.com/sdk/installing/bundle.html

[7] Curso de programación android en castellano.Android.es .Darrer accés el 23 de abril del 2013 . Disponible a:

http://www.android.es/curso-de-programacion-android-en-castellano.html#axzz2W9rff7GR

[8] Encryption and decryption of image. StackOverflow. Darrer accés el 31 de maig del 2013 Disponible a :

http://stackoverflow.com/questions/14154096/encryption-and-decryption-of-image

[9]Muñoz, Alberto. Algoritmo criptografico Rijndael. Madrid,2004. Darrer accés el 13 de juny del 2013. Disponible a:

http://www.um.edu.ar/catedras/claroline/backends/download.php?url=L0FsZ29yaXRtb19yaWpuZGFlbC5wZGY%3D&cidReset=true&cidReq=SIIISR

[10]Encriptar imagenes. Google Play. Darrer accés el 13 de juny del 2013. Disponible a:

https://play.google.com/store/search?q=encriptar+imagenes

7 – ANNEXES

7.1 – Manual d'usuari

Instal·lació

Primer de tot , marcar l´opció “Orígenes desconocidos” del nostre dispositiu. Generalment, la trobarem al menú “ajustes” ,dins del submenú “Seguridad”. Amb això podrem instal3lar aplicacions que no provenen de “GooglePlay”. Seguidament,carregarem l'arxiu “uoc.tfm.AEScam.apk” al dispositiu mòbil.

Finalment, amb un explorador qualsevol d'arxius, la seleccionarem i la instal·larem.

Utilització

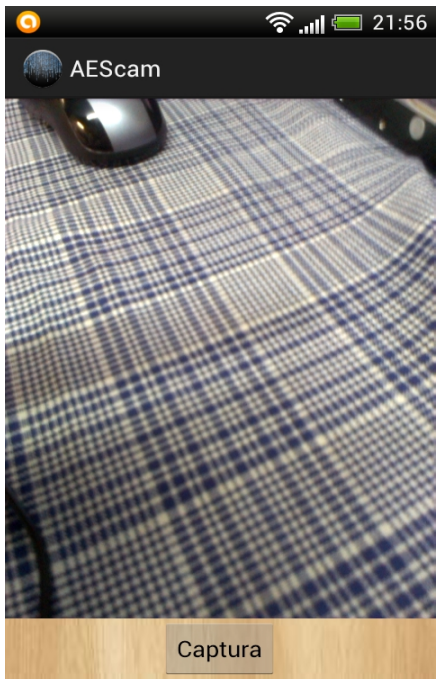
Després de prémer sobre el botó de l´aplicació, ja la podrem començar a utilitzar.

A la pantalla principal se'ns demana que introduïm una contrasenya:

Aquesta contrasenya haurà de ser la mateixa per a cada procés d´ encriptació i desencriptació, podent variar d'un procés a l'altre.

Un cop introduïda, podrem escollir entre desencriptar una imatge o capturar una imatge:

Si escollim capturar una imatge ,procedirem a la previsualització de la mateixa.



Quan tinguem la previsualització que desitgem ,s'haurà de prémer el botó “ Captura “ per dur a terme l´ encriptació de la imatge adquirida per la càmera i el seu posterior emmagatzematge a la carpeta “Camera1” del dispositiu mòbil. A més a més, prémer aquest botó implica que, paral·lelament a l´esmentat en les línies anteriors, es crearà , també, una representació pixelada de la imatge encriptada, emmagatzemant-se aquesta al directori “Bitmap”.

Si escollim l´opció de desencriptar una imatge, és procedirà, en segon terme , a la desencriptació de la imatge encriptada prèviament i emmagatzemada a la carpeta “Camera1” de la galeria d´imatges del dispositiu. Aquesta imatge desencriptada ,també s'emmagatzemarà a l´ esmentada carpeta del dispositiu.

És important recordar que un cop finalitzem l'us de l´ aplicació cal tancar-la a fi d´el-·iminar la contrasenya del dispositiu.

Si observem la galeria d´ imatges del dispositiu, podrem veure que l'aplicació guarda les imatges encriptades i desencriptades a la carpeta “Camera1” i la imatge pixelada a la carpeta “BitMap”

Desinstal·lació

Accedirem al menú “ajustes” del dispositiu , un cop allà entrarem al submenú “aplicaciones” i des d´ allà podrem seleccionar l´ aplicació i eliminar-la.

Destacar que els fitxers creats per l´ aplicació ,siguin en format d´ imatge o criptogràfic , es mantindran a la memòria del dispositiu i, per tant, a la galeria d´imatges, un cop desinstal·lada l´ aplicació.