Estudi i implementació d'un mètode d'ocultació d'informació en codis QR

Victor Acin-Sanz

Resum—Un QR, o Quick Response codi, és un codi de dos dimensions que permet emmagatzemar informació en diferents formats. Durant aquest TFG, intentarem demostrar que és possible ocultar informació dins d'aquests tipus de codis. La principal motivació per intentar demostrar que és possible amagar aquesta informació, és el fet de que hi ha sistemes de votació electrònica que utilitzen QR. En aquests sistemes, la possibilitat d'incloure informació oculta d'un QR representa una vulnerabilitat. Aquest objectiu el portarem a terme desenvolupant en primer lloc un mètode teòric, i en segon lloc, demostrant la validesa d'aquest mètode implementant-lo. Gràcies a la implementació hem pogut demostrar que és possible amagar informació en els codis QR. La implementació s'ha portat a terme en Java per la plataforma Android, mitjançant l'ús de les llibreries Zxing.

Paraules clau—Esteganografía, Ocultació d'informació, Codis QR.

Abstract—A QR, or Quick Response code, is a two-dimensional code capable of storing data using diferent formats. Through this TFG, we will try to prove the fact that is indeed possible to store hidden information in a QR. The main motivation for this project is the fact that some e-voting systems use QR codes, and proving that is possible to store hidden information in this codes, could vulnerate the system's security. In order to accomplish this objective, we will develop a theoretical method capable of hidding this information in a QR code, and we will then implement it to prove that is indeed possible. Thanks to the implementation, we have proven that we were correct, and therefore, have been succesfull in hidding information in a QR. The implementation was done programming for the Android platform, using Java and the Zxing libraries.

Index Terms—Steganography, information concealment, QR codes.

- *E-mail de contacte: victor.acin@e-campus.uab.cat*
- *Menció realitzada: Enginyeria de Tecnologies de la Informació.*
- *Treball tutoritzat per: Jordi Cucurull (Scytl) i Guillermo Navarro (UAB).*
- *Curs 2013/2014*

1 INTRODUCCIÓ

Inicialment aquest projecte va ser proposat al departament del dEIC per l'empresa Scytl. Aquest treball ha estat desenvolupat en col·laboració amb l'equip de recerca d'aquesta empresa.

1.1 Motivació del treball

El motiu principal pel qual l'empresa Scytl esta interessada en investigar si és realment possible amagar informació en els codis QR (Quick Response) és per que actualment hi ha sistemes de votació electrònica que utilitzen aquests codis en el seu procés de votació, com per exemple el sistema WOMBAT [1]. Així doncs, des d'un punt de vista de seguretat, el fet de poder introduir informació oculta en un QR és vist com un punt feble en el sistema de votació.

1.1.1 Sistemes de votació electrònica

A continuació explicarem dos sistemes de votació electrònica, i per què poden ser vulnerats si s'introdueix informació oculta en els QR.

1.1.2 WOMBAT

Aquest sistema de votació va ser dissenyat per estudiants de enginyeria informàtica de la Universitat de Tel Aviv i de IDC Herzliya. El sistema de votació que van desenvolupar es pot descriure en unes poques fases:

1. El votant s'identifica utilitzant algun document que avali la seva identitat.

2. El votant entra en la cabina de votació i realitza la seva elecció de candidats en la màquina de votacions, i aquesta imprimeix el vot. El vot consta de dues parts que es poden separar. En la meitat superior hi ha la selecció de candidats en text pla, i en la inferior, la mateixa selecció xifrada i codificada dins d'un codi QR.

3. (Opcional) Si el votant vol auditar el vot, la màquina mostra al votant el contingut del QR, per que s'asseguri de que coincideix amb el text pla. Un cop fet això el vot passa a ser nul.

4. El votant parteix per la meitat el vot, i introdueix en una de les busties la meitat que conté el vot en blanc, permetent així portar un doble recompte de vots, i per altre banda, un lector de QRs llegeix el QR del votant i afegeix el vot a la urna virtual. El codi QR se li dona al usuari per poder consultar l'estat del seu vot a través d'un portal web. Fent això, els votants poden cerciorar-se de que el seu vot ha estat pujat a la urna virtual on es porta a terme el recompte.

El fet de que el votant s'emporti el QR planteja una vulnerabilitat en el sistema. En cas de que un atacant pogués afegir informació extra en els QR a través de, per exemple, malware, planteja un problema de seguretat important, ja que utilitzant aquesta informació oculta es podria arribar a veure la selecció de candidats del votant. Aquest tipus d'atacs en sistemes de votació s'anomenen atacs de "Vote Exposure", i permeten, entre d'altres, la compra venta de vots, o la extorsió del poble per obligar-los a votar a un candidat en concret.

1.1.3 beVote

El sistema de votació beVote [2] té un funcionament semblant al sistema WOMBAT i va ser utilitzat en les eleccions municipals de Bèlgica del 2012. El funcionament és el següent:

1. El votant s'identifica utilitzant algun document que avali la seva identitat.

2. El votant entra en la cabina de votació on porta a terme la seva elecció. El sistema de votació emet el vot. Aquest vot consta de dos parts, en una, està el codi QR, i en l'altre, el vot en text pla.

3. El votant passa el codi QR pel lector i introdueix el vot sencer en una urna.

En aquest cas la vulneració no és tant flagrant com en el sistema explicat anteriorment, però igualment un atacant podria afegir informació al vot que ajudes a identificar al votant, tals com l'hora exacta a la que s'ha emès el vot i un identificador de la màquina que l'ha imprès.

1.2 Motivació personal

Per altre banda, personalment, també vam trobar molt interessant el projecte, ja que mai havíem treballat amb codis QR, i a més no es un camp on s'hagin portat a terme gaires investigacions.

Una altra de les raons per les quals ens vam sentir inclinats a fer aquest tipus de projecte, és que un treball d'investigació ofereix molta llibertat a l'hora d'escollir com seguir-lo, i això és un dels camps que normalment no es veuen durant el grau en enginyeria informàtica.

Per tant, les principals raons per les que vam escollir fer aquest projecte són la possibilitat d'investigar sobre un tema desconegut (codis QR), el fet de que ningú hagi intentat fer això abans, i la llibertat que acompanyen aquests tipus de treballs.

1.3 Objectius del projecte

En segon lloc, concretarem els objectius del projecte. Començarem indicant que aquest projecte és tant d'investigació com d'implementació.

En total, tenim tres objectius i dos objectius addicionals. Els objectius principals són els següents:

1. Desenvolupar un mètode d'ocultació d'informació en codis QR.

2. Implementar un lector de codis QR que permeti llegir informació oculta.

3. Implementar un generador que permeti generar codis

QR amb informació oculta.

Els objectius secundaris són els següents:

1. Desenvolupar un mètode que permeti detectar si un QR ha estat manipulat per emmagatzemar informació oculta.

2. Implementar una aplicació que permeti detectar si un QR conté informació oculta.

Tots els objectius de desenvolupament tenen una fase preliminar on s'investiga si és possible portar a terme aquest desenvolupament.

Fins aquí els principals objectius del nostre projecte, que es poden resumir en investigar i implementar un mètode d'ocultació d'informació en codis QR.

1.4 Treballs relacionats

Degut a que es un camp bastant nou, encara no s'han portat a terme gaires treballs de recerca en codis QR. Tot i així si que s'han portat a terme estudis en temes de seguretat en codis QR i de esteganografia mitjançant aquests codis.

Per exemple, un equip de Gisbon Research Corporation ha creat un sistema de login per pàgines web que es (en teoria) més segur que els sistemes de login tradicionals, mitjançant criptografia amb codis QR [3]. Aquest sistema permet a un usuari identificar-se en una pàgina web mitjançant una imatge amb un codi QR i una aplicació especifica utilitzada per llegir aquest codi QR. En la URL que inclou el codi, hi ha una part xifra que només l'aplicació de l'usuari pot desxifrar, permetent així a l'usuari identificar-se en la pàgina web.

Tot i que no és gaire comú, també es poden utilitzar codis QR amb continguts xifrats. Actualment no hi ha cap estàndard sobre codis QR que permeti crear codis QR xifrats, però hi ha diverses aplicacions, cadascuna amb el seu sistema de xifratge, que permeten generar codis QR xifrats.

Un cas de esteganografia en codis QR és el proposat per Wen-Yuan Chen i Jing-Wein Wang en el seu article anomenat Nested image steganography scheme using QR-barcode technique [4]. En aquest article s'explica com es pot arribar a ocultar un codi QR que conté informació en una imatge qualsevol, de forma que la informació del codi QR passa desapercebuda als usuaris, mantenint la capacitat correctora pròpia dels codis QR.

Com podem comprovar actualment no s'ha investigat amb gaire aprofundiment la seguretat en els codis QR i per tant és un camp en el que encara s'hi pot aprofundir molt.

1.5 Continguts

El document comença explicant el funcionament dels codis QR en l'apartat 2, on es repassaran conceptes bàsics que són necessaris per comprendre la resta del treball. A continuació, exposarem la metodologia (apartat 3) utilitzada durant el treball, tant per la part d'investigació com per la part de implementació. Tot seguit explicarem el mètode d'ocultació d'informació proposat, (apartat 4, que constarà de quatre parts principals: 4.1, on s'explicaran les idees inicials per portar a terme el projecte, les coses a tenir en compte a l'hora de dissenyar-lo, i els tests i el mètode definitiu, 4.2, on s'explicarà quin format hem decidit seguir per afegir la informació, 4.3, on s'exposaran les diferents fases per generar un codi QR i on injectem nosaltres la informació, 4.4, on explicarem quin es el procés de lectura d'un codi QR i com recuperem la informació.

A partir d'aquí, passarem a fer una explicació de la implementació (5), on justificarem la plataforma i llibreries escollides (5.1), quines classes s'encarreguen de portar a terme la tasca de generar els codis QR, així com les modificacions pertinents realitzades en aquestes classes (5.2), i els seus homòlegs per la part del lector (5.3).

A continuació presentarem els resultats del nostre treball (6), així com una reflexió sobre que aporta.

Per últim presentarem la conclusió (7) de l'article seguit dels agraïments i la bibliografia.

2 DESCRIPCIÓ BÀSICA D'UN CODI QR

Un codi QR (Quick Response code) és un tipus de codi de dos dimensions en forma de matriu de bits que permet emmagatzemar informació i que disposa de mecanismes de correcció d'errors.

En funció de la informació que es requereixi emmagatzemar, els codis augmentaran la seva mida incrementant la seva versió. Cada versió incrementa la mida de la matriu en 4 files i 4 columnes, la versió més petita és la 1, una matriu de 21x21, i la versió més gran és la 40.

El sistema que utilitzen els codis QR per emmagatzemar la informació (ja siguin dades o codis correctors d'errors) es pot subdividir en tres estructures diferents:

- Bit: Són els punts blancs i negres dels codis QR i poden prendre els valors de 0 o 1.

- Codeword: Són paraules de 8 bits.

- Blocs: Els blocs són conjunts de codewords. La mida dels blocs ve determinada per la versió del QR i el nivell de correcció d'errors.

Els codis QR estan formats per diferents elements (veure figura 1). A continuació les enumerarem i farem una petita explicació sobre cadascun d'ells:

1. Marques d'orientació: Les marques d'orientació són unes marques situades en els extrems del codi QR, utilitzades pels lectors per saber en quina direcció han de llegir el codi. Varien de quantitat en funció de la versió del codi.

2. Informació de format: La informació de format és utilitzada pels lectors de codis per saber quin és el

nivell de correcció del codi QR, i per saber quina màscara s'ha aplicat al codi QR.

3. Mètrica: És una pauta utilitzada pels lectors, sempre és igual a tots els codis QR, i ajuda als lectors a identificar els punts blancs i negres de la matriu.

4. Dades: Les dades s'emmagatzemen en codewords de mòduls (cada mòdul és un bit). Les dades estan precedides per uns bits dedicats a especificar el format de les dades, i la quantitat de caràcters que hi ha en el QR. Actualment, els QR accepten diferents formats per les dades, però els més rellevants pel projecte són el format de 8 bits (format tradicional, per cada caràcter s'utilitzen 8 bits) i el format alfanumèric. Aquest format utilitza 11 bits per representar conjunts de dos caràcters, i 6 bits per representar un únic caràcter.

5. Codis correctors d'errors: Aquests codis utilitzen l'algorisme Reed Solomon per proporcionar capacitat correctora als QR. Aquesta capacitat correctora s'expressa en un percentatge i correspon al tant per cent de dades que es podrà recuperar en cas d'error. Actualment es poden escollir 4 nivells de correcció pels codis QR: L (7%), M (15%), Q (25%), H(30%).

6. Informació de la versió: La informació de la versió, situada en els laterals de dos de les marques d'orientació, s'utilitza únicament en codis QR a partir de la versió 7, i conté informació sobre la versió utilitzada.

7. Sense ús: Depenent de la versió del QR i del nivell de correcció d'aquest, l'ultim codeword dels codis d'error, no s'utilitza.

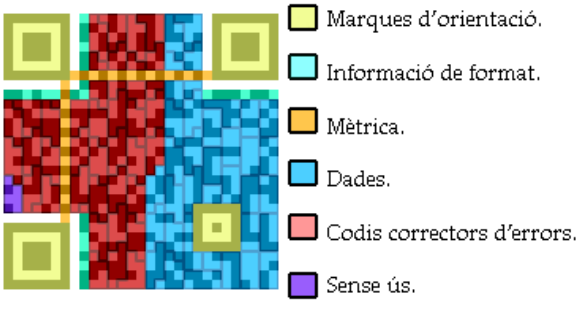

Fig. 1. Dissecció d'un codi QR de versió 3 amb nivell de correcció Q

També trobem important explicar què són els blocs en els QR i com s'escriuen en els QR.

Un bloc en un codi QR és un conjunt de codewords de dades i codis correctors d'errors. Depenent del codi QR (la seva versió i el seu nivell de correcció d'errors) disposarà de més o menys blocs, i de més o menys dades i codis correctors d'errors per bloc. La mida d'aquests blocs i la seva quantitat estan predeterminades en l'estàndard oficial dels codis QR [4].

Els blocs però no s'escriuen en ordre en el codi QR, en realitat s'escriuen alternant uns blocs amb els altres. És a dir, suposem que tenim dos blocs, B1 i B2, el bloc 1 conté els codewords del C1 al C13 i el bloc 2 conté els codewords del C14 al C26. Així doncs, les dades s'escriurien en el QR seguint aquest ordre: D1, D14, D2, D15, D3, D16... A la figura 2 podem veure un codi QR de dos blocs on es detalla l'ordre d'escriptura, aquesta forma de intercalar els blocs s'anomena interleaving. Com també es pot apreciar en aquesta mateixa figura, l'ordre d'escriptura general també segueix un estil peculiar. Es comença des de l'extrem dret inferior del QR cap a la part superior, un cop s'arriba a una marca d'orientació, o al final del QR, es comença a baixar:

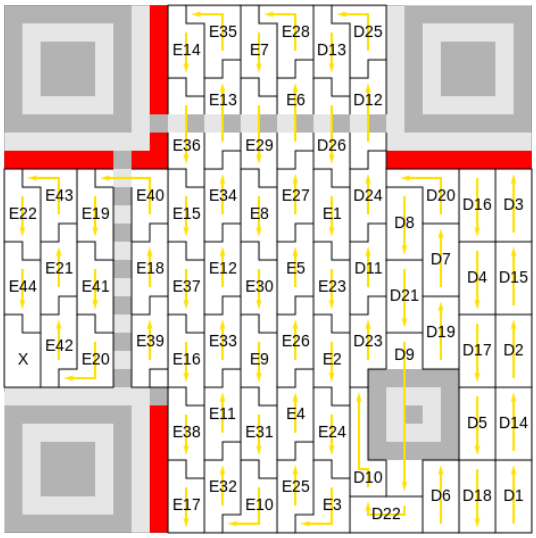

Fig. 2. Imatge on es veu l'interleaving d'un codi QR de versió 3 [5].

Ara passarem a explicar què és la capacitat correctora d'un QR. La capacitat correctora d'un QR, és la quantitat d'errors que pot arribar a corregir. Com hem comentat anteriorment, els codis QR tenen quatre nivells de correcció: L, M, Q, i H. Cadascun permet corregir una quantitat major d'errors que l'anterior, sent L el més baix i H el més alt. Això es tradueix en un increment en la quantitat de codewords dedicats a la correcció d'errors. La capacitat correctora d'un codi QR es pot expressar com el total de codewords dedicats a la correcció, entre dos, truncant el resultat, que és la quantitat d'errors que pot corregir.

Per últim, farem una breu explicació sobre una part dels QR anomenada mascara. Depenent de les dades que hagi de guardar un QR, i de la seva codificació, es pot donar el cas de que en alguna zona de la matriu del QR, en lloc d'obtenir una distribució uniforme de ceros i uns, tinguem conjunts bits amb el mateix valor. Aquests conjunts apareixen en els codis QR com taques blanques o negres, les quals poden arribar a provocar errors de lectura durant el procés de descodificació del codi QR. Les mascares són un mecanisme que s'utilitza precisament per això. En realitat, aquestes màscares són *arrays* de bits ordenats de tal forma que al fer una XOR amb la matriu,

els conjunts de ceros o uns desapareixen, i obtenim la distribució de punts que permet al lector portar a terme una lectura més acurada.

3 METODOLOGIA SEGUIDA EN EL PROJECTE

Podem dividir la metodologia utilitzada en dos parts, la metodologia utilitzada per la part d'investigació i desenvolupament del mètode d'ocultació, i la metodologia utilitzada pel desenvolupament de les aplicacions.

Començarem explicant la metodologia per la part de desenvolupament.

A l'hora de portar a terme els objectius d'investigació i desenvolupament del projecte, es va seguir un mètode de tres fases:

1. Estudi i cerca preliminar d'informació sobre el tema en particular, fins a adquirir els coneixements necessaris per portar a terme la fase de desenvolupament. Reforçar aquests coneixements amb proves empíriques.

2. Desenvolupament teòric de les diferents tècniques a implementar posteriorment, basant-nos en la informació adquirida en la fase 1 i contrastant el resultat amb aquesta informació per assegurar-nos de que no hi ha errors.

3. Finalment, portar a terme una pressa de decisió sobre la tècnica a implementar.

Fins aquí la metodologia utilitzada per la part d'investigació i desenvolupament. Ara passarem a explicar la metodologia per la implementació:

1. En primer lloc, portem a terme un anàlisi de requisits sobre l'aplicació a dissenyar.

2. Disseny dels diagrames UML amb les classes a utilitzar per l'aplicació.

3. Implementació de l'aplicació.

4. Fase de testeig de l'aplicació, per corroborar el bon funcionament d'aquesta.

A l'hora de portar a terme les implementacions es segueix un mètode de desenvolupament en espiral, portant a terme tantes iteracions com sigui necessari, afegint funcionalitats o arreglant errors en cada iteració.

Aquestes dues metodologies són les que es van definir al principi del projecte, s'han utilitzat durant tota la seva duració, i no ha existit la necessitat de canviar-les per unes altres o modificar-les.

4 MÈTODE D'OCULTACIÓ D'INFORMACIÓ

4.1 Explicació del mètode

Durant aquesta secció explicarem les teories inicials pel mètode, les consideracions tingudes a l'hora de desenvolupar-lo i finalment el mètode definitiu.

4.1.1 Idees preliminars

A continuació explicarem quines són les idees preliminars en la que es basa el mètode d'ocultació final.

En un primer lloc, vam partir de la assumpció de que era possible amagar informació en un codi QR. D'aquí, en van sortir dos idees principals (les menys rellevants van ser descartades d'entrada).

La primera idea consisteix en aprofitar la capacitat de correcció que tenen els codis QR per sobreescriure alguns dels bits de les dades o dels errors, de forma que un cop aplicada aquesta correcció el codi QR es pogués llegir correctament, però que un usuari que coneixes l'existència d'aquesta informació la pogués recuperar sense problemes.

La segona idea, consistia en generar una cadena al final del codi QR de forma que un cop aplicat l'algorisme de Reed Solomon, la informació secreta fossin els bits d'error.

La segona idea va ser descartada per la forma en la que es generen els codis de Reed Solomon, ja que no és possible obtenir totes les combinacions de bits com a codis correctors d'errors, i per tant no era possible generar sempre el contingut desitjat.

Un cop teníem clar quina idea volíem implementar (la més viable), vam passar a les consideracions, és a dir, que hem de tenir en compte a l'hora de dissenyar el mètode.

4.1.2 Consideracions

En aquest apartat exposarem les qüestions a considerar a l'hora de desenvolupar el mètode d'ocultació d'informació, tals com problemes en la correcció d'errors o la utilització de màscares.

El primer aspecte que ens vam plantejar és l'aplicació de les màscares. La màscara s'aplica al final de la generació del codi QR, és a dir, quan les dades i els codis d'errors ja estan generats i posats en la matriu. És important tenir això en compte depenent de quan tenim intenció de modificar els bits de dades o d'errors. Si els modifiquem després de que s'apliqui la màscara, quan realitzem la lectura del QR, haurem de extreure els bits abans de que es re-calculi la matriu original. Si en sobreescrivim els bits abans de d'aplicar la màscara, haurem de llegir-los en la matriu original.

El segon aspecte és com es realitza la correcció dels errors. Segons el estàndard del codi QR, totes les dades i els codis correctors d'errors s'inclouen en una serie de blocs. La mida dels blocs (quantitat de codewords de dades i codewords

d'errors) ve determinat per la versió del QR i pel nivell de correcció d'errors.

A partir d'aquí existeixen dos possibilitats, pot ser que els codis correctors d'errors es generin per cada bloc i per tant només puguin corregir els errors d'aquell bloc, o que es generin per totes les dades i després es distribueixin entre tots els blocs i per tant els codis correctors del segon bloc puguin corregir errors en el primer bloc.

Per últim, una altra de les coses que hem de tenir en compte en cas de que escollim escriure la informació abans d'aplicar la màscara, és si sobreescrivim únicament un bloc, o si escrivim en blocs consecutius.

Com s'ha comentat abans, les dades es distribueixen en diferents blocs en funció de la versió i del nivell de correcció d'errors. Per tant, quan estem amagant la informació, podem escollir sobreescriure únicament dades o codis correctors d'errors d'un únic bloc (per exemple, bloc 1) o sobreescriure dades i/o errors en diferents blocs.

Aquestes són les consideracions generals a l'hora de crear el mètode d'ocultació de la informació.

4.1.3 Tests

En aquest apartat explicarem quines proves vam fer per acabar de perfilar el mètode a implementar, així com una explicació en profunditat d'aquest mètode i de les decisions preses.

En primer lloc, explicarem la decisió de posar la informació abans de que s'apliqui la màscara. Ens va semblar que era lo més lògic, ja que si al afegir aquesta informació extra ja estàvem suprimint part o tota la capacitat correctora del codi QR, en cas de que la informació contingues els conjunts de punts d'un mateix color mencionats anteriorment, es podria donar el cas de que el QR no es pogués llegir.

Una de les coses que vam contemplar va ser la possibilitat de que la capacitat correctora de l'estàndard no es traduís a les implementacions amb un 100% de fiabilitat.

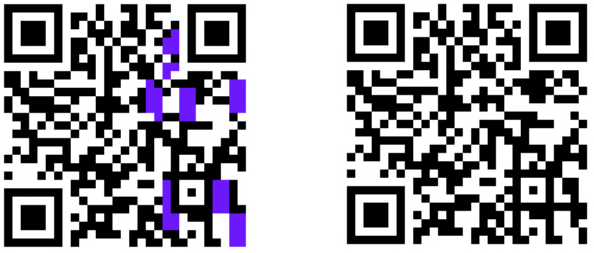

Fig. 3. Exemple de dos tests sobre un QR de versió 3 amb capacitat correctora L amb 7 errors. El QR de la esquerra conté erasures, i el de la dreta, errors.

Per assegurar-nos de que els nostres codis serien llegibles en la majoria dels casos vam portar a terme tests amb codis QR modificats. La capacitat correctora dels codis QR pot corregir dos tipus d'errors: errors que sap on estan (erasures), i errors que desconeix (errors). Els testos intenten demostrar que en qualsevol dels dos casos la capacitat correctora era la correcte, així doncs tenim dos tipus de test diferents, un on s'intentaven corregir errors coneguts, i un altre on s'intentava corregir errors desconeguts (veure figura 3.)

També vam posar a proba la capacitat correctora dels codis provocant errors tant a la part de dades com a la part dels codis correctors d'errors, provocant errors dels dos tipus, coneguts i desconeguts (veure figura 4). S'han portat a terme diferents tests d'aquest tipus, comprovant errors parcials, o canviant només un bit en un codeword. Aproximadament s'han fet 50 proves diferents cadascuna corroborada amb dos lectors per telèfons mòbils i amb un lector de codis de barres.

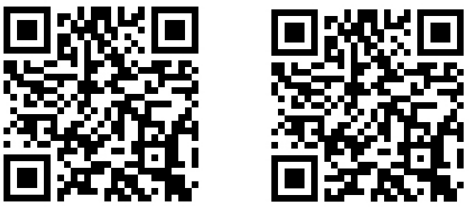

Fig. 4. A l'esquerra podem veure un QR de versió 3 amb 7 errors (llegible) tant a la part de correcció com a la de dades, i a la dreta el mateix codi però amb 8 errors (no llegible).

Tots els resultats aconseguits han estat els esperats, verificant així que la capacitat correctora és sempre la indicada pels diferents casos.

Un altre de les consideracions a verificar ja que vam considerar que era molt important pel projecte, és si els codis d'errors es generaven únicament per un bloc o si es generaven per tot el codi QR. Vam portar a terme diferents tests per comprovar això eliminant únicament informació d'un sol bloc, tant a la part de dades com a la d'errors, i els resultats van corroborar la idea de que els codis correctors d'errors es generaven per cada bloc. Més endavant, durant la implementació, l'ús de llibreries externes ens va permetre verificar aquests resultats, ja que en elles també es generaven els codis correctors de errors en base als blocs de dades.

Un cop portades a terme totes aquestes proves vam prosseguir a desenvolupar el mètode d'ocultació.

4.1.4 Mètode d'ocultació d'informació definitiu

Partint d'aquestes bases vistes en l'apartat anterior, vam decidir que el nostre mètode d'ocultació consistiria en el següent:

Utilitzant la capacitat correctora de Reed Solomon, afegirem informació oculta en un codi QR sobreescrivint els blocs de dades aprofitant al màxim la capacitat correctora d'aquests. És a dir, primer sobreescriurem el primer codeword del primer bloc, després el primer codeword del segon bloc, i així consecutivament, en el mateix ordre en el que s'escriuen en el QR. Substituint les dades d'aquesta forma ens assegurem

d'aprofitar al màxim la capacitat correctora del codi QR ja que la correcció es fa a nivell de bloc. Per tant, si sobreescrivim únicament un bloc (de per exemple un total de 2 blocs) estaríem limitats a la meitat de la capacitat correctora del codi QR.

La injecció de la informació es portarà a terme abans de que s'apliqui la màscara al codi QR, per evitar que es generin agrupacions de bits amb els mateixos colors, la qual cosa, combinada amb el fet de que hem sacrificat la capacitat correctora del QR per afegir informació, pot provocar que el QR no sigui llegible.

Sempre deixarem la possibilitat de corregir un codeword de dades per seguretat. Com hem explicat abans, la capacitat correctora dels codis QR és el total de codewords dedicats a la correcció d'errors entre dos, truncant el resultat, per tant, per evitar que un error en la implementació faci que el codi QR no sigui llegible en algun cas concret, hem decidit que la quantitat total de informació que permetem afegir en el codi QR sigui la següent:

$$N = \lfloor (\frac{ErrorCorrectionCodewords}{2}) - 1 \rfloor$$

On ErrorCorrectionCodewords és el total de codewords dedicats a la correcció d'errors entre tots els blocs, i N la quantitat de codewords d'informació que podrem afegir.

4.2 Format de la informació

En primer lloc, hem diferenciat el format de la informació en funció de la codificació utilitzada.

Intentant optimitzar al màxim l'espai disponible per afegir informació, vam prendre la decisió de no incloure informació sobre el format o la mida de la informació, únicament la adició d'una marca que ens permeti identificar el final d'una cadena.

Els dos formats a utilitzar són el de 8 bits (per la facilitat de traducció, ja que utilitza la mateixa codificació que ASCII) i el format anomenat alfanumèric. Tot i que el format alfanumèric només admet majúscules, números, i alguns símbols, gràcies a la seva representació, permet incloure més informació que el format de 8 bits.

Els caràcters a utilitzar per indicar el final de la informació oculta, són, tant pel format de 8 bits com per l'alfanumèric, els dos punts (:). En el cas de la codificació de 8 bits, el final de la frase el marquen dos dobles punts seguits (::), mentre que en el format alfanumèric s'utilitzen tres dobles punts seguits (:::).

La raó per la qual es va decidir per aquesta nomenclatura, és per que la correcció per part dels codis correctors d'errors es realitza a nivell de codeword, i no de bit. Per tant, sacrificar un codeword sencer era el mateix que sacrificar una cadena de bits inferior.

En el cas del format alfanumèric, és per que durant la lectura, s'assumeix que l'usuari sempre introdueix una quantitat parella de caràcters. Per això és necessari utilitzar tres caràcters per la forma en la que es descodifica. Posem per cas que només utilitzem dos dobles punts (::) i que l'usuari crea una cadena d'informació secreta amb una quantitat parella de caràcters. A l'hora de llegir aquests caràcters, trobaríem els 2 dobles punts seguits, però si l'usuari introdueix una quantitat imparella de caràcters, amb l'ultim caràcter del usuari llegiríem un doble punt, i en la següent lectura, intentaríem llegir 11 bits, mentre que el caràcter s'ha escrit utilitzant-ne només 6, i per tant no llegiríem el final de la cadena. Si n'escrivim tres, això ja no ocorre.

Per tant, el format utilitzat per 8 bits són els dos dobles punts, ja que al llegir els caràcters d'un en un no tenen problemes, i en el cas del format alfanumèric són tres dobles punts.

4.3 Fases de la generació d'un codi QR i injecció de la informació oculta.

Ara passarem a explicar quines són les fases de la generació d'un QR per definir exactament quan afegim la nostre informació addicional.

Fases de generació d'un QR:

1. Entrada de dades: L'usuari entra les dades que vol posar en el QR en l'aplicació.

2. Selecció codificació: Es busca una codificació adequada per l'usuari, o, depenent de l'aplicació, es codifica directament en mode 8 bits.

3. Selecció del nivell de correcció d'errors: Normalment l'usuari especifica un nivell de correcció en concret. En cas contrari, se sol utilitzar un nivell de correcció d'errors Q.

4. Generació de la capçalera: Es crea la capçalera del QR. Aquesta capçalera varia de mida en funció de la versió i la codificació, i s'utilitza per indicar quants caràcters hi ha en el QR i la codificació del QR.

5. Conversió a bits: Conversió de les dades en bits tenint en compte el format utilitzat.

6. Calcul de la versió: Tenint en compte la mida de les dades més la capçalera, juntament amb el nivell de correcció, es fa un calcul de la versió a utilitzar.

7. Creació dels blocs: S'organitzen les dades en forma de blocs en funció de la versió desitjada.

8. Calcul dels codewords de correcció d'errors: A partir dels blocs de les dades, es realitza el calcul dels codewords destinats a la correcció d'errors i s'afegeixen als blocs de dades pertinents.

9. Interleaving: Es realitza la reorganització dels blocs alternant-los entre ells com s'ha mencionat anteriorment.

10. Màscara: S'avalua si es necessari aplicar una màscara, en cas de que sigui així, s'aplica a la matriu.

11. Generació de la informació addicional: Es generen els bits a incorporar en la secció d'informació del format i, si es necessari, de la informació de la versió.

A continuació, es crearia una matriu amb tota la informació resultant d'aquest procés i es convertiria a imatge.

Respectant aquestes fases de generació d'un codi QR i el mètode mencionat anteriorment, la injecció de la informació oculta s'ha de portar a terme entre les fases 9 i 10. Just quan s'ha creat el interleaving entre els diferents blocs de dades, aprofitem per escriure sobre aquests blocs mateixos la nostra informació, en el format mencionat en el primer apartat depenent del tipus de codificació escollida.

4.4 Fases de la lectura d'un codi QR i recuperació d'informació oculta en el codi QR

Durant aquesta secció, explicarem quines són les fases de lectura d'un codi QR, partint de la base de que acabem d'obtindre els bits de la imatge.

1. Obtenció dels bits: El lector de codis QR ha llegit un codi i ha transformat la imatge en una cadena de bits.

2. Identificació dels blocs: El lector identifica els blocs de dades.

3. Correcció: Es porta a terme la correcció de les dades mitjançant els codis correctors d'errors.

4. Separació dels blocs: Es desfà el procés d'interleaving separant els codewords de cada bloc i ajuntant-los en l'ordre pertinent en una cadena de bits.

5. Descodificació: En base al format del codi QR, es porta a terme la descodificació dels bits per convertir-los a un format llegible.

A partir d'aquí, es transmetria la informació a la interfície i es mostraria a l'usuari.

pel nostre cas, la obtenció dels bits d'informació oculta es portaria a terme entre els passos 1 i 3, ja que el pas 3 modificaria els bits que que volem recuperar.

Un cop obtingut el total de les dades, haurem de buscar la informació oculta en el seu interior. Per comoditat, primer buscaríem la informació traduint-la al format de 8 bits, ja que és el més senzill. En cas de no trobar el símbol que indica el final de la informació oculta, procedim a buscar la informació utilitzant el format alfanumèric.

En cas de que cap dels dos resultats indiqui que disposem d'informació oculta, notifiquem al usuari de que no s'ha trobat res. En cas contrari, mostrem la cadena d'informació al usuari.

5 IMPLEMENTACIÓ

Durant aquest apartat explicarem com hem portat a terme la implementació, concretant la plataforma, les llibreries, i les principals classes del nostre codi.

5.1 Plataforma i llibreries escollides

En primer lloc, la plataforma. La implementació del projecte s'ha portat a terme utilitzant el llenguatge de programació Java, i les aplicacions s'han desenvolupat pel sistema operatiu mòbil Android. Aquesta implementació havia de servir per corroborar la part teòrica del projecte, i donat que Android és un sistema molt estable per desenvolupar aplicacions, i proporciona moltes facilitats a l'hora d'accedir a perifèrics (com per exemple la càmera), i de depurar aplicacions des de el propi telefon, es va decidir que era el millor candidat per portar a terme la implementació.

En segon lloc, les llibreries escollides per portar a terme la implementació en Java són les llibreries Zxing (Zebra Crossing Barcode Scanner Library [6]). Aquesta llibreria ens va semblar la més adequada per que després de mirar diverses llibreries del mateix estil, era la que tenia l'estil de programació més entenedor, i la millor estructura de classes, cosa que vam considerar molt important, donat que era el primer cop que fèiem una aplicació per Android.

5.2 Classes del generador i modificacions.

A continuació explicarem quines classes composen el codi de la nostre aplicació. Començarem amb les classes creades per nosaltres, i a continuació descriurem les classes mes rellevants de les llibreries que hem utilitzat.

- *MainActivity*: És la classe que s'utilitza com a interfície d'usuari tant en el generador com en el lector. En el cas del generador, inclou totes les comprovacions de camps per evitar que l'usuari introdueixi valors que no son vàlids.

- *Generator*: Aquesta classe compleix 3 funcions. En primer lloc, és la encarregada de rebre els valors de la classe interfície i transmetre'ls a les llibreries per que els transformin en una matriu de bits. En segon lloc, és l'encarregat de transformar aquests bits en una imatge, i per últim, s'encarrega de guardar la imatge en la memòria del telefon.

Ara passarem a explicar les principals classes de la llibreria Zxing per generar codis QR. Farem menció de les classes més importants i de les que haguem modificat.

- *Encoder*: Aquesta és la classe que s'encarrega de realitzar la codificació del codi QR. És qui porta a terme les principals fases de generació del QR mencionades anteriorment (4.3).

La resta de classes que s'utilitzen en el generador son classes de suport, utilitzades per contenir dades o realitzar funcions

puntuals, les modificacions portades a terme en aquestes classes són majoritariament per que transmetin la informació des de les nostres classes cap a la classe encoder. i per tant no considerem important afegir-les en aquest document.

Les principals modificacions portades a terme en les llibreries han estat en la classe Encoder.

És en la classe Encoder, on afegim realment la informació oculta. Hem hagut de generar funcions addicionals dins d'aquesta classe per adaptar-la a les nostres necessitats. La resta de classes del generador s'han modificat per que puguin transmetre la informació al Encoder, o per realitzar alguna funcionalitat secundaria.

5.3 Classes del lector i modificacions

Com hem comentat anteriorment, la principal classe creada pel lector de codis QR és la classe MainActivity. Aquesta classe simplement conté una crida a classe CaptureActivity, que ja s'encarrega de pràcticament tot.

Classes de la llibreria Zxing:

- *CaptureActivity*: Aquesta classe actua a la vegada d'interfície per l'usuari i de classe principal. S'encarrega de mostrar al usuari les pantalles per fotografiar els codis QR i també de mostrar els resultats al usuari.
- *DecodedBitStreamParser*: En aquesta classe és on es porta a terme la descodificació de la cadena de bits. Rep com a paràmetres una cadena de bits i retorna una cadena de caràcters.

Les principals modificacions realitzades en aquestes classes han estat en la classe *DecodedBitStreamParser*. En aquesta classe és on transformem la nostra cadena de bits en una cadena d'informació llegible. La resta de classes han estat modificades per transmetre la informació que hem afegit al usuari.

6 RESULTATS

El principal mètode d'avaluació utilitzat per verificar les bases teòriques del nostre treball ha estat la implementació. Gràcies a la implementació hem pogut comprovar si realment les suposicions de les que partíem en un principi eren certes.

Després de portar a terme la implementació, vam poder verificar que les assumpcions que vam fer al principi del treball són certes. És possible amagar i llegir aquesta informació en un codi QR. A continuació mostrem dos codis QR, el contingut dels quals és el mateix, excepte que el segon codi QR conté informació oculta (veure figures 5 i 6.).

La aplicació final desenvolupada és capaç d'amagar informació en la majoria de codis QR. No hem portat a terme proves en les 160 combinacions de codis, però funciona en les versions més utilitzades (1-20) en qualsevol dels quatre nivells de correcció d'errors, utilitzant la codificació bàsica de 8 bits.

Cal dir que aquests resultats eren els esperats, donat que la implementació estava reforçada per una gran part d'investigació teòrica.

Fig. 5. Un QR normal amb el contingut: Codi QR mostrant els resultats del treball.

Fig. 6. Un QR amb informació oculta. El contingut visible és: Codi QR mostrant els resultats del treball. El contingut ocult és: Aquest és un exemple d'informació oculta.

7 CONCLUSIONS

Com s'ha comentat anteriorment en els resultats, gràcies a la implementació hem verificat que és possible amagar informació en un codi QR mitjançant la tècnica descrita en aquest mateix document. Un fet important a tenir en compte és que la informació que incorporem nosaltres al QR no disposa de capacitat correctora d'errors, i per tant, qualsevol problema en la lectura, com per exemple una mala posició del lector, pot portar a la obtenció d'informació equívoca.

En l'apartat d'implementació s'explica que actualment la aplicació desenvolupada pel treball codifica la informació en format de 8 bits. La codificació en format alfanumèric estava en procés, però degut a alguns contratemps en la planificació no s'ha pogut dur a terme. En cas de que es seguis amb el desenvolupament del projecte, un dels primers punts és habilitar

la codificació en aquest format, ja que l'increment de quantitat d'informació que podem ocultar amb aquest format, respecte al de 8 bits, és substancial.

Com es pot veure pels resultats del treball, hem pogut portar a terme els objectius generals del projecte. Els objectius addicionals, en canvi, ja es van planificar tenint en compte que només es portarien a terme en cas de que sobrés temps després de realitzar el projecte. Degut a que el projecte s'ha completat en el temps esperat, els objectius addicionals no s'han portat a terme.

Una de les principals línies d'investigació per les quals es podria continuar aquest treball és el desenvolupament de tècniques de contra mesura. És a dir, investigar un mètode que permeti detectar si un codi QR conté informació oculta d'algun tipus. Una de les ideés per portar a terme aquest detector consisteix en comprar la matriu del QR generat, amb una matriu que sapiguem que és correcte. Suposem que llegim un QR que conté la informació "Això és un exemple", però no estem segurs de que no contingui informació addicional, així doncs el que podem fer és generar nosaltres un QR amb el mateix nivell de correcció d'errors i amb el mateix contingut, i a continuació, fer una XOR entre els bits dels dos QR. Si el resultat és 0, sabrem que els dos QR són iguals i que per tant el primer no conté informació oculta, si el resultat és 1, sabrem que el primer QR pot contenir informació oculta. Un dels principals problemes d'aquest mètode són els falsos positius. Si hi ha un error de lectura, o si el QR no esta en bon estat, detectaríem la existència de informació oculta quan en realitat no n'hi ha.

AGRAÏMENTS

Vui donar les gràcies al supervisor del treball en la universitat, Guillermo Navarro, i a l'equip d'investigació i desenvolupament de Scytl format per en Jordi Cucurull, l'Alex Escala i la Sandra Guasch, per la seva ajuda durant la realització d'aquest TFG.

BIBLIOGRAFIA

[1] *WOMBAT voting system project (2012). Wombat voting system, how it works (1ª edició) [Online]. Disponible: http://www.wombat-voting.com/how-to-vote*

[2] *beVote project (2011) Belgium voting system (1ª edició) [Online]. Disponible: http://www.e-voting.cc/en/the-new-electronic-voting-system-in-belgium-in-use-in-2012-archive-1211/*

[3] *SQRL project (2013). Secure Quick Rreliable Login (1ª edició) [Online]. Disponible: https://www.grc.com/sqrl/sqrl.htm*

[4] Wen-Yuan Chen, Jing-Wein Wang (2009). Nested image steganography scheme using QR-barcode technique. (1ª edició) [Online]. Disponible: http://2d-code.co.uk/images/pdf/qr-code-steganography.pdf

[5] *ISO/IEC 18004 (2000). Information technology — Automatic identification and data capture techniques — Bar code symbology — QR Code (1ª edició) [Online]. Disponible: http://raidenii.net/files/datasheets/misc/qr_code.pdf*

[6] *Wikipedia contributtors (2012) . QR VER3 Codeword Ordering [Online]. Disponible : http://en.wikipedia.org/wiki/File:QR_Ver3_Codeword_Ordering.svg*

[7] *Zxing project (2014). Zebra Crossing Barcode Scanner library (1ª edició) [Online]. Disponible: https://github.com/zxing/zxing*