A Biztonságos elektronikus kereskedelem alapjai c. tárgy házi feladata

MIFARE kártyák feltörése

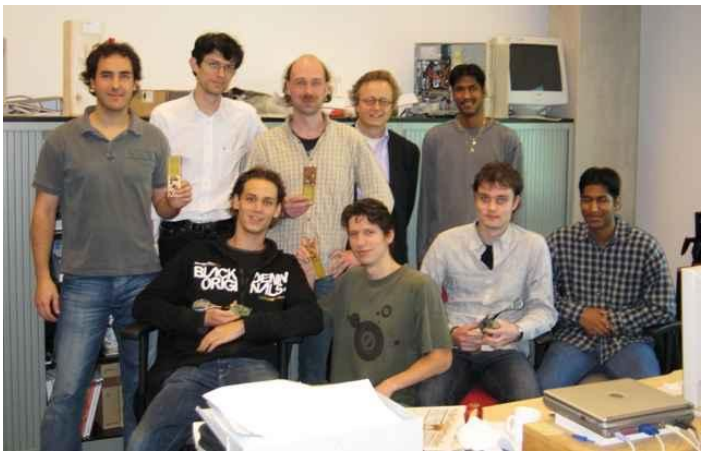
Készítette

Sárdi György

d24b9w

Bevezetés

A Holland kormány 2008 elején egy nyilatkozatban hívta fel a figyelmet a világszerte széles körben használt MiFare Classic RFID kártyában felfedezett biztonsági résekre.
A biztonsági réseket egy német kutató – Henryk Plötz – és egy doktorandusz hallgató – Karsten Nohl – közösen fedezték fel világon elsőként, majd publikálták 2007 decemberében a Németországban megrendezett 24-dik Chaos Communications Congress (24C3) alkalmával.
A 24C3 egy négynapos hacker konferencia, amit a híres német Chaos Computer Club (CCC) rendez meg évente. Ezen a konferencián Nohl egy áttekintő bemutatót tartott az RFID technológia biztonsági sebezhetőségéről és bemutatott egy filmet a MiFare chip Crypto-1 titkosításának feltöréséről.
A kormány fentebb említett nyilatkozatában ekkor még úgy nyilatkozott, hogy a sikeres feltörés ellenére a MiFare kártyák még várhatóan további egy-két éven belül biztonságosnak számítanak, ugyanis feltörésükhöz nagy szakértelemre és drága, egyedi és speciális hardver eszközökre van szükség.
Ugyanakkor Nohl által nem sokkal később kiadott "Cryptanalysis of Crypto-1" című cikkében, illetve az Isztambulban tartott EuroCrypt kriptográfiai konferencián már egy olyan új módszert mutatott be, amiben másodpercek alatt visszafejtette egy MiFare Classic RFID ajtónyitó kártya titkosító kulcsát, majd ezek után sikeresen klónozta a kártyát, és ehhez mindössze csak egy átlagos PC-re volt szüksége. Ennek tudatában a Holland kormány egy új határozatában már az összes – nem privát felhasználású – Classic kártya cseréjét el rendelte.
A chip hackeléséhez Nohl és Plötz reverse-engineering módszereket alkalmazott. Egy mikroszkóp és egy fényképezőgép segítségével aprólékos munkával, sok próbálkozás árán föltérképezték és visszafejtették a MiFare chip egyes rétegeit, illetve egy olvasó készülék segítségével analizálták a chip dinamikus működését. Céljuk az volt, hogy bemutassák; egy ilyen támadáshoz a – vélekedésekkel szemben – szerényebb anyagi erőforrások is elegendőek.
Néhány hónappal később – a Nohl-Pöltz által végrehajtott törés során, a Crypto-1-ről megszerzett ismereteket felhasználva – a holland Radboud Egyetemen működő Digital Security csoport is felfedezett és közreadott egy újabb biztonsági rést a Mifare Classic RFID chipekben. Ezt a hibát kihasználva már nem csak klónozni tudták a kártyát, hanem adataikon változtatásokat is végre tudtak hajtani. A csoport azonban nem állt meg a teszteléssel és feltöréssel. Így újabb és újabb hiányosságokra derítettek fényt. Ezeket a biztonsági réseket a csoportban részt vevő végzős diákok elsősorban diplomamunkáikban ismertették.



1. ábra: A Digital Security csoport

A fenti törések következtében minden olyan rendszer, ami a MiFare Classic kártya nulladik szektorában tárolt titkos kulcsra támaszkodott, a továbbiakban már nem minősült biztonságosnak. A „hackerek" azt a következtetést vonták le egymástól függetlenül mindkét csapatban, hogy a „security by obscurity" alapelv ebben az esetben is csődöt mondott. Hiszen a fizikai felépítésnek, elvi működésnek és algoritmusoknak titokban tartása nem mentette meg a rendszert a töréstől. A Kerckhoff elvet javasolták, mint követendő példát: ez mind a terveknek, felépítésnek, mind az algoritmusok nyilvánosságra hozatalát jelenti, és egyedül a kulcsok által nyújtott védelemre támaszkodik. Ezáltal lehetővé válik a nyilvános közösség által folyamatosan végrehajtott tesztelése, hibakeresésre és feltörési kísérletekre.

RFID rövid leírás

Az RFID, Radio-Frequency IDentification azonosítási eljárás során fizikai vagy vizuális kontaktus nélküli táv-azonosítás történik. Ehhez az azonosítandó személyt vagy tárgyat fel kell szerelni egy ún. transzponderrel azaz jeladóval, amivel már adott távolságon belül tudunk kommunikálni. Ezt a jeladót az RFID eszközökkel kapcsolatos szabványokban Proximitly Coupling Device-nak nevezik.
Ez az azonosítási módszer korántsem számít újdonságnak. Hiszen használták már beazonosítás céljából akár a hadseregben, akár a civil életben egyaránt. Ugyanakkor napjainkban népszerűsége ugrásszerűen megnövekedett köszönhetően a modern elektronikai miniatürizálási technológiáknak. Lehetővé vált ugyanis a kisméretű, olcsón előállítható transzponderek nagytömegben történő előállítása.
Az így készülő eszközöket „electronic tag"-nak nevezik, ami magyarra fordítva elektronikus cédulát vagy címkét jelent. Az „electronic tag"-ok családjába tartoznak az érintés nélküli intelligens RFID kártyák is, amiket más néven smart kártyáknak is nevezünk. Ezek a „smart card"-ok gyakorlatilag hagyományos bankkártyák integrált RFID tag-gel, továbbá az intelligenciát biztosító és az aktív működést lehetővé tévő számítási kapacitással. A számítási kapacitás alatt – mai számítógépes mércével mérve – szerény teljesítményt értünk, ami a kártyák típusától függően általában az egyszerű alapműveletek végrehajtásától, a szimmetrikus- esetleg a hash protokollok ismeretén keresztül, egészen az aszimmetrikus kriptográfiai protokollok futtatásáig terjedhet. A smart kártyák intelligens működéséhez szükség van beépített memóriára is. A kártyákba ültetett memória mérete szintén a kártyák tudásszintjének illetve az alkalmazási helyük függvénye. Általában néhány kilobyte a terjedelmük. A tudás, illetve felszereltség szint az adott kártya árát nagymértékben befolyásolja.

A kártyák fontos jellemzője még az energia-ellátás típusa is. Jellemzően ez passzív működést jelent, melynek során a kártya csak akkor éled föl, ha az a leolvasó készülékhez bizonyos távolságnál közelebb kerül. Ilyenkor a leolvasó által keltett elektromos térben indukálódik a működésükhöz szükséges energia. Ugyanakkor van saját, aktív energiaforrással felszerelt kártya is.
Az energia-ellátás típusa befolyással van a kártya kommunikációs képességeire: az érzékelési távolságra és az adatátvitel sebességére is. A saját energiaforrással ellátott kártyáknak – értelemszerűen – nagyobb az érzékelési távolságuk, és nagyobb sebességet tudnak elérni. Ugyanakkor a működési távolságot és az adatrátát az alkalmazott adatátviteli frekvencia tartomány és az átküldendő adatblokkok mérete is befolyásolja. A frekvenciatartományok szabványosítva vannak: LF azaz alacsony-, HF azaz magas- és UHF azaz ultra magas frekvenciájú tartományok szerint. A kommunikációs távolság így jellemzően csak maximálisan néhány méternek adódik, az átviteli sebesség pedig néhány kilobit és megabit között van másodpercenként.
Az RFID kártya működéséhez szükség van egy olvasó berendezésre is. Az olvasó lehet önmagában autonóm módon működő vagy számítógéphez csatlakoztatott. Amikor számítógéphez van csatlakoztatva, akkor gyakorlatilag csak egy kommunikációs interfész a PC és a tag között. Ez az olvasó berendezés biztosítja a kommunikációhoz és a passzív kártyák tápellátásához szükséges elektromágneses teret. Meghatározott időpontokban 2.28μs hosszan (ISO-14443A szabványú protokoll esetén) megszakítja a teret. A 0-ás és 1-es bitek adatátvitelét a megszakítások gyakoriságával modulálja a módosított Miller kódolási eljárás alapján. Egy bit átviteléhez így 9.44μs-ra van szükség. A megszakítások miatt a passzív kártyák kondenzátorokat tartalmaznak, hogy kisimíthassák a tápellátásban keletkező ingadozásokat. A másik irányba az RFID kártyák – az olvasó erőterébe kerülve – ellenállásuk megváltoztatásával küldenek vissza adatokat. Manchester kódolási eljárást használnak a bitek megkülönböztetéséhez. A vivő frekvencia 847.5KHz, ami egész számú osztója a 13.56MHz-es kommunikációs frekvenciának.
Az RFID kártyák az alacsony árszint miatt általában nem tartalmaznak feltörési próbálkozások elleni védelmet sem. Az szabványok csak az adatátviteli réteget definiálják és nem határozzák meg a kártyák szükséges védelmi szintjét: sem a hitelesítés, az integritás, az engedélyezés sem a rendelkezésre állást.
A kis méret, az elfogadható ár és a beépített intelligencia következménye, hogy a smart-card-ok felhasználása egyre nagyobb volumenű az utóbbi időkben. Kvázi univerzális azonosítóvá válhatnak; a legtöbb területen átveszik a hagyományos papíralapú jegyek, vonalkódos, mágnes csíkos, ujjlenyomat-olvasós, PIN kódos és más egyéb azonosító technikák szerepét.

Hol használnak napjainkban érintésnélküli RFID kártyákat?
Ezeknek a kártyáknak számos nagyléptékű alkalmazási területük van világszerte. Használják őket többek között beléptető rendszerekben, a gépjárműiparban, a hadiiparban, elektronikus személyi azonosítóként (igazolványként) vagy elektronikus fizetési eszközként. Széles körben elterjedt a városi tömegközlekedésben is, ahol a hagyományos bérletet helyettesíti. Olyan helyeken, mint például London, ahol metrókban, buszokon, a Docklands Light Railway vonalakon, villamosokon és a helyi érdekű vasúti közlekedésben terjedt el. További példa a tömegközlekedésre a hollandiai OV-chipkaart bérlet, vagy Malajziában használt Touch ‘n Go, illetve a hong kongi Octopus Card bérletek. Mindezek a kártyák egyébként MiFare Classic típusúak.

	Bérlet	Jegy	Token
Bangkok Blue Line	Mifare 1		
Delhi Metro			Ultralight
Nanjing Metro			Ultralight
Singapore Circle Line	Mifare 1		
Taipei	Mifare 1		Ultralight
Stasbourg	Mifare 1		
Hollandia	Mifare 4K	Ultralight	
Oslo		Ultralight	
Dánia	Mifare 4K		
Hong Kong Octopus	Mifare 4K		
London	Olyster (Mifare)		
Malajzia	Olyster (Mifare)		

2 ábra: MiFare kártyák típusai

A visszafejtéshez használt eszközök

1. Ghost

A Ghost-nak nevezték el a feltöréshez használt egyik eszközt. Ez tulajdonképpen egy RFID tag, ami a hagyományos RFID kártyához hasonlóan tud kommunikálni az olvasóval. A különbség csak annyi, hogy programozható, és soros protokollal egy számítógéphez köthető. Ennek köszönhetően lehallgathatja a kártyák és az olvasó közti kommunikációt, illetve megszemélyesíthet egy adott kártyát.

2. ProxMark

Egy olyan általános eszköz, ami az ISO14443-A szabvány alapján tud kommunikálni az RFID kártyákkal és olvasókkal. A Ghosthoz hasonló funkcionalitással rendelkezik.

3. OpenPCD

Nyílt hardverű és forráskódú olvasó; teljes betekintést nyújt a kártya és az olvasó közötti üzenetváltásokba. Többek között egy Philips (NXP) MiFare tag és egy ARM mikrokontroller van rajta; emiatt tökéletes a kontroll az időzítések felett.

3. ábra: Az OpenPCD

4. MiFare kártyák

Gyártójuk a NXP Semiconductors, régi nevén Philips Semiconductors. A MiFare az egyik legelterjedtebb RFID rendszer, a világ összes RFID azonosítóinak több mint 70%-ka MiFare technológián alapul. Nem egy

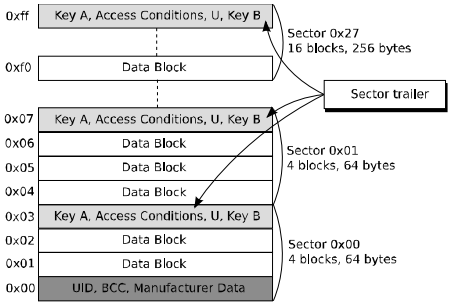
termék, hanem egy egész termékcsalád, különböző tudásszintű elemekkel. A család tagjai: MiFare Ultralight, Classic, DESFire és SmartMX.

A feltörésekben a MiFare Classic volt a célpont. A Classicból is több eltérő felszereltségű kártyát létezik: a mini, az 1K, a 4K és a Fudan változat.

	Szektorok		Blokkok		Memória		slack space	
	64b	256b	összesen	szabad	összesen	szabad	min	max
mini	5	0	20	14	320	224	5	35
1K	16	0	64	47	1024	752	7	112
4K	32	8	256	215	4096	3440	40	280
Fudan	64	0	256	191	4096	3056	64	448

4. ábra: MiFare Classic változatok

A Mifare Classic kártya egy EEPROM-mal és biztonságos RF kommunikációval ellátott memóriakártya. A memória tartamán alapműveleteket végezhetünk: olvasás, írás, növelés, csökkentés. A memória szektorokba szervezett. Minden szektor tovább van bontva 16 byte-os blokkokra. Minden egyes szektor utolsó blokkja két titkos kulcsot és az adott szektorra vonatkozó elérési feltételeket tárolja. Ezt a blokkot trailernek nevezik.



5. ábra: A Trailer felépítése

Támadások típusai

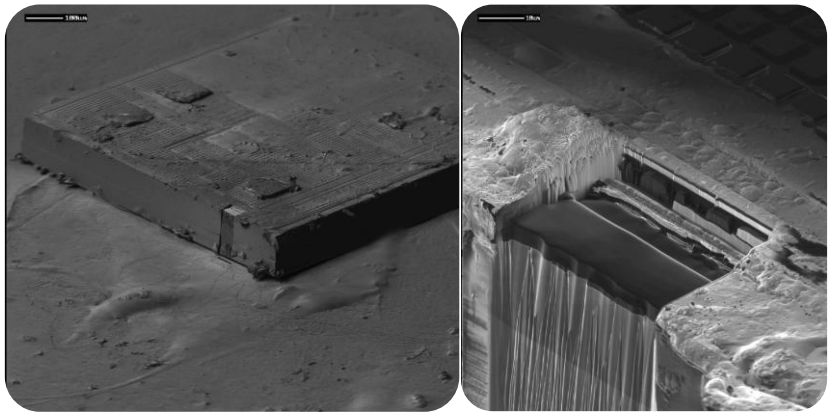
1. Lehallgatás: A kártyaolvasó vagy a kártya által küldött adatok megismerésére, majd megfelelő eljárásokkal az adatok visszafejtése.
2. Relay attack: Az érintésnélküli interfész miatt lehetségesek az ilyen típusú támadások. Megoldás lehet erre a támadásra a relay time megfelelő megszorítása a distance bounding kriptográfiai protokollt felhasználva. Ld. [Han05] és [Kas06].
3. Visszajátszás: A támadó egy korábban rögzített kommunikációs üzenetet játszik vissza. A rögztésre lehetőség van a kommunikáció miatt. Ismert törési lehetőség UID kódot elküldő RFID kártyák esetében. Ld. [Kas06]
4. Eldobható RFID kártyák esetében problémát jelenthet az a tervezési hiba, amikor maguknak az ún. lock biteknek letiltását nem ellenőrzi a rendszer. A lock bitekkel lehet letiltani az egyes érzékeny

információt tartalmazó memóriaterületek írását. Így a rendszer képtelen lezárni ezeket a fontos területeket, ahova így megtévesztő információt lehet visszaírni. Ld. [SvdS07]

5. Klónozás: Ha az RFID kártya nincs védve titkosítással és ismert a kártya utasításkészlete, a funkcionalitása és a memória terület felosztása. Ekkor egyszerű az ilyen kártya lemásolása és a másolattal való helyettesítése. Problémát akkor okoz, amennyiben a rendszer nem figyel a duplikált kártyák jelenlétére. Ld. [Ver08]
6. Man in the Middle: Egy eredeti kártya és olvasó, valamint egy lehallgatáshoz használt OpenPCD olvasó és Ghost kártya szükséges hozzá. A Ghost felveszi a kapcsolatot az eredeti olvasóval és továbbítja a kéréseket a számítógép felé. A számítógép elküldi ezeket az OpenPCD-vel az eredeti kártyának, ami visszaküldi válaszát a számítógépnek. Erre a PC Ghost-on keresztül válaszol az eredeti olvasónak. Időzítési hiba léphet föl.
7. Reverse engineering módszerekkel visszafejthető tamper ellen nem védett kártya hardver felépítése, működése, utasításkészlete stb. Ld. [Nohl08]

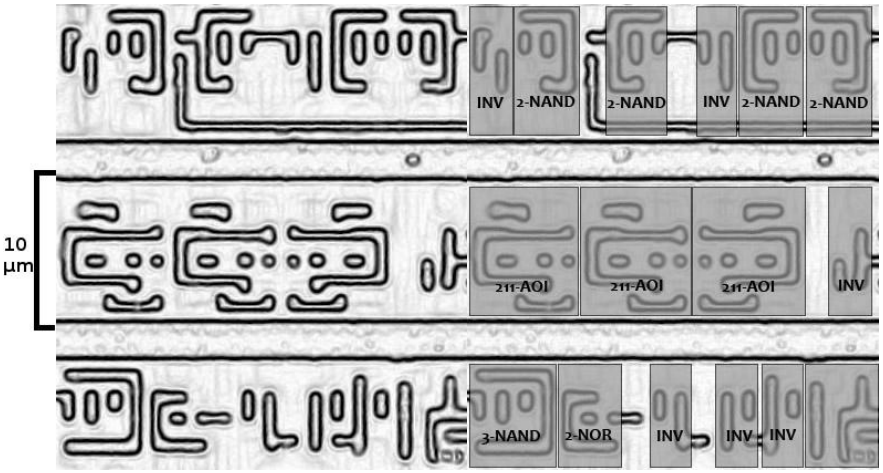
A kiindulás: Nohl, Starbug és Plötz reverse engineering munkája

A MiFare Classic kártya chipje elég kicsi méretű, csupán 1mm^2 a területe. Ennek egynegyedét a memória, egynegyedét a rádiófrekvenciás kommunikáció vezérlője tölti ki. A maradék tartalmazza a kriptográfiáért felelős részt. A kriptográfiai megvalósítása kb. 400 NAND kapu ekvivalensből épül fel. Ez nagyon kicsinek számít még a nagymértékben optimalizált implementációk között is. Ezért is, olyan gyorsak a MiFare kriptográfiai algoritmusai; minden órajel-ciklusban új kulcsfolyam bit generálódik.
A kártya fizikai visszafejtéséhez először is hozzá kellett férni a beültetett chiphez. Emiatt acetonban oldották fel a műanyag részeket, ami kb. fél órát igényelt. Ezután polírozással mechanikai úton távolították el a fennmaradó rétegeket. Azzal a nehézséggel is meg kellett küzdeniük, hogyan tartsák a tag-ot vízszintesen, hogy mindenütt egyenletesen lehessen leszedni a rétegeket.
A chip 6 rétegből épül fel. A legalsó tartalmazza a tranzisztorokat. Egy hagyományos 500 szoros nagyítású optikai mikroszkóp segítségével felvételeket készítettek a lemart területekről. Több képkészlet segítségével automatikusan generálták a képeket ún. kép tilling (mozaik illesztés) technikával.



6. ábra: Logikai áramkörök elektronmikroszkóp alatt

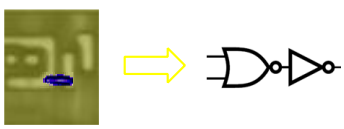
A tranzisztorok kapukba vannak csoportosítva; mindegyik egy-egy logikai műveletet hajt végre, mint pl. AND, XOR vagy flip-flop. A chip néhány ezer ilyen logikai kapuból áll, de ezek közül kb. csak 70 különböző típusú van. A különböző fajtákból egy táblázatot készítettek és a mintaillesztést úgy implementálták, hogy egy adott típus alapján az összes vele egyező fajtát megtalálja az algoritmus. A mintaillesztés ún. normalizált keresztkorrelációs eljáráson alapult, amit a MATLAB image processing könyvtárából kiindulva implementáltak. Azért döntöttek így, mert ez a módszer kevésbé érzékeny az egyes felvételek között előforduló – polírozásból adódó – világosság és színárnyalat különbségekre. A polírozást követően manuálisan minden egyes beazonosított kapuhoz megadták a hozzá tartozó funkciót.



7. ábra: (a) 2. rétegbeli kép éldetektálás után (b) automatikus mintaillesztés után

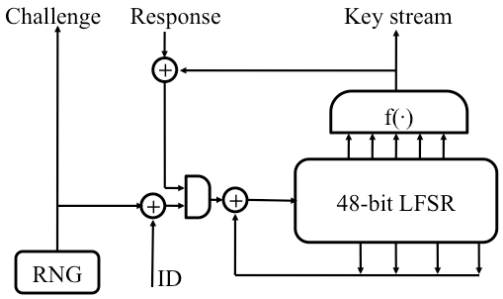
Az illesztési eljárás könnyen felfedhette volna a chip összes logikai kapuját, azonban ezek közül csak a titkosításért felelős részt vizsgálták meg részletesebben. Ehhez a vizsgálathoz elméleti ismereteikre támaszkodtak; Tudták, hogy a kulcsfolyam titkosító legalább egy 48 bites regisztert és néhány XOR kaput kell, hogy tartalmazzon. Így könnyedén be tudták azonosítani ezt a részt a chip egyik sarkában, mellette egy véletlenszám-generátornak tűnő áramkörrel. (Ugyanis nem volt bemenete csak kimenete.) Ezek után ezeken a részeken visszafejtették az áramkörök huzalozását is. Ez jelentékeny kézi munkát és nagy idő befektetést igényelt, valamint a hibázás lehetősége is magas volt. A hibákat redundáns ellenőrzésekkel és különböző statisztikai módszerek alkalmazásával tudták kiszűrni. (Például feltételezték, hogy a titkosító egyenletes blokk eloszlású az f szűrő függvényben.)

Az ily módon visszafejtett chipben nem találtak semmilyen tervezési vagy implementálási félrevezetést, vagy akár visszafejtést-gátló technológiákat. Sőt! A chip felépítése olyan szépen strukturált volt, hogy abból könnyedén lehetett működéséről következtetéseket levonni.
Az így megkapott különböző építőelemek visszafejtését bizonyos statisztikai tulajdonságok meglétével ellenőrizték. Az eredmény az lett, hogy a logikai kapukról felállított „térkép”, és a köztük lévő huzalozások



8. ábra: Reverse- engineering

visszaállítása után majdnem elegendő információval rendelkeztek a kriptoalgoritmus teljes visszafejtéséhez is. Ehhez azonban a vezérlésért felelős részt is vissza kellett volna fejteni, így nem ismerhették a pontos ütemezéseket és a titkosító egyes bemeneteinek működését. Továbbá a hardver teljes ismerete sem lett volna elég a tökéletes visszafejtéshez, ugyanis a memória tartalmakat nem tudták volna kiolvasni. Ezért az algoritmust a kártya és a hozzá tartozó olvasó közti kommunikáció visszafejtésével határozták meg inkább.



9. ábra: Crypto-1 titkosító és inicializálása

Támadás

Chip reverse engineering és Időzítés vezérlése az OPenPCD-vel, és véletlenszám generálás vezérlése.

Milyen gyengeségeket azonosítottak?

1. Strukturális gyengeség
   A fizikai reverse engineering során nyújtott segítséget az áramkörök azonosításához.
2. Támadés nyers erővel
   A 48 bites kulcshossz túl kevés, ehhez járul a rendszer gyenge kriptografikus struktúrája. A támadás során a rendszer két kihívás – válasz cserét rögzít egy valódi kártya és az olvasó között, és ehhez minden lehetséges kombinációban keresi a kulcsot. A titkosító alacsony komplexitása miatt egy Xilinx Virex-5 LX50 FPGA eszközzel képesek voltak szimulálni párhuzamosan 6 kártya –olvasó egységet. 64 darab FPGA-ból épített fürt segítségével a 2^{48} lehetőség végignézése kevesebb, mint 50 percig tart.

10. ábra: Brute-force támadás

3. Hiányzik a nem-linearitás a visszacsatolási körben, ami miatt nincs forward secrecy.
4. Output bit előre rögzített bit alhalmazból van származtatva. Emiatt nem lesz optimális a lavinaeffektus.
5. Véletlenszám generátor gyengeségei
   A véletlenszám a bekapcsolástól eltelt időtől függ, ezért bármikor beállítható. Valamint csak 16 bitet, ezért minden 0.6mp után újraindul.
   Az olvasó 32 bites RNG-je is ilyen felépítésű, ezért az is teljesen kontroll alatt tartható.
6. Újraindítás után minden alapértékre, azaz egy ismert kiinduló értékre áll be.

Digital Security csoport munkája

A feltörést Nohl és társai munkája során megszerzett és közzétett információkból kiindulóan, csupán a kártya – olvasó kommunikáció analizálásával végezték.

Kártya hitelesítési folyamata

Egy blokkművelet elvégzéséhez az olvasó eszköznek először hitelesíttetni kell magát a blokkot tartalmazó szektorba. Ehhez a két kulcs közül az elérési feltételek szerint kell egyet választani.

	Sender	Hex	Abstract
01	Reader	26	req type A
02	Tag	04 00	answer req
03	Reader	93 20	select
04	Tag	c2 a8 2d f4 b3	uid,bcc
05	Reader	93 70 c2 a8 2d f4 b3 ba a3	select(uid)
06	Tag	08 b6 dd	mifare 1k
07	Reader	60 30 76 4a	auth(block 30)
08	Tag	42 97 c0 a4	n_T
09	Reader	7d db 9b 83 67 eb 5d 83	$n_R \oplus ks_1, a_R \oplus ks_2$
10	Tag	8b d4 10 08	

11. ábra: Inicializálás

Amint egy RF kártya belép egy olvasó hatómezejébe, feléled és belekezd az ún. ütközéselhárító protokollba saját UID-jének küldésével. Ezt észlelve az olvasó berendezés az ISO 14443A szabványban foglaltak szerint kiválasztja a kártyát, majd elküld neki egy adott szektora vonatkozó hitelesítés kérelmet. Erre a kártya kódolatlanul visszaküld egy ún. kihívó n_T nonce véletlen elemet, amire válaszul az olvasó átküldi saját n_R nonce-ját, együtt a kártya kihívására adott a_R válasszal. A tag saját a_T elküldésével fejezi be a hitelesítést. n_R –től fogva minden kommunikáció titkosított, azaz n_R, a_R, a_T össze van XOR-va a k_{s1}, k_{s2}, k_{s3} kulcsfolyamokkal.

Lépés	Küldő	Hex érték	Jelentés
01	Olvasó	26	req type A
02	Tag	04 00	answer req
03	Olvasó	93 20	select
04	Tag	c2 a8 2d f4 b3	uid,bcc
05	Olvasó	93 70 c2 a8 2d f4 b3 ba a3	select(uid)
06	Tag	08 b6 dd	mifare 1k
07	Olvasó	60 30 76 4a	auth(block 30)
08	Tag	42 97 c0 a4	nT
09	Olvasó	7d db 9b 83 67 eb 5d 83	nR . ks1,aR . ks2
10	Tag	8b d4 10 08	aT . ks3

12. ábra: Kommunikáció

A kártya pszeudó véletlen generátora determinisztikus. Emiatt egy előállított nonce csak a tag feléledése és a kommunikáció kezdete között eltelt időtől függ. Mivel a feltöréshez használt speciális olvasók igény szerint vezérelhetők – ezért az időzítések –, és ezáltal a véletlenszám-generálás vezérelhető. A Ghost olvasót tag üzemmódban működtetve egyedi nonce-kat és uid-ket lehet generálni.
Az n_T és uid értékek rögzítésével és ismétlődő hitelesíttetésekkel kiderítették, hogy az olvasó minden újraindítást követően ugyanazokat a nonce szekvenciákat generálja. Véletlenszám-generátorának állapota tehát csak a meghívásainak számától függ. A tag véletlenszám generátora egy 16 bites, $x^{16}+x^{14}+x^{13}+x^{11}+1$ generátor-polinomú LFSR. Mivel a nonce hossza 32 bit, és a LFSR-nek 16 bitnyi állapota van, ezért az n_T első fele meghatározza második felét. Ezért egy adott 32 bites értékről eldönthető, hogy megfelelő nonce vagy sem. (Azaz $n_k \oplus n_{k+2} \oplus n_{k+3} \oplus n_{k+5} \oplus n_{k+16} = 0$ minden $k \in \{0, 1, \ldots, 15\}$ -ra.) Ez azért fontos, mert a Ghost olvasó bármilyen – nem a csak megfelelő – nonce-ok küldésére alkalmas.
Különböző uid és tag nonce-okkal folytatott próbálgatások során kiderült, ha $n_T \oplus$uid konstans marad, akkor a titkosított szöveg sem változik. Ebből arra lehet következtetni, hogy a kulcsfolyam is konstans és az a_T és a_R n_T-től függ. Ezekből a próbálkozásból származó $a_R \oplus k_{s2}$–ők össze XOR-ozásával megkaphatók $a_R \oplus a'_R$-k.
Összefoglalva, a hitelesítés menete a következő. Miután a tag elküldte n_T –t a tag és az olvasó is K megosztott kulccsal, uid-vel és n_T-vel inicializálja a titkosítót. Az olvasó ezután megválasztja saját n_R-jét és ks_1 kulcsfolyam első részével titkosítva elküldi. Ezután updateli a titkosító állapotát n_R-rel. Az olvasó $suc^2(n_T)$ kódolt átküldésével hitelesít. Ezen a ponton a tag is képes updatelni a titkosítója állapotát ezekhez hasonlóan. A kulcsfolyam többi része ks_3, ks_4… megállapításra kerül és ettől fogva minden kommunikáció kódolt. A tag $suc^3(n_T) \oplus ks3$ elküldésével fejezi be a hitelesítést. Ezután az olvasó hitelesíteni tudja a tag-ot,

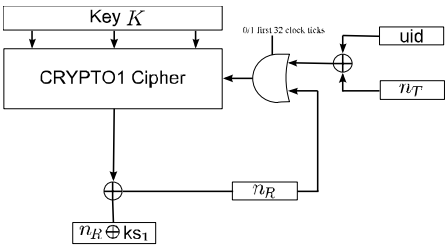
Ismert nyílt szövegű támadás

A hitelesítési protokoll leírásából látszik, hogy a kulcsfolyam részei kitalálhatóak. Miután megfigyeltük n_T-t és $suc^2(n_T) \oplus k_{s2}$-t, k_{s2} visszaállítható $suc^2(n_T)$ kiszámításával és XOR-sával. Sőt! Ha a hitelesítési protokoll végén kártya nem válaszol, akkor a legtöbb olvasó idő túllépés miatt „halt” vagy „read” parancsot küld ki. Miután mindkét parancs byte kódja ismert a „halt” illetve „read”$\oplus k_{s3}$ kódolt üzenetekből k_{s3} számolható. Amennyiben egy támadó rendelkezik az olvasó és a kártya felett, úgy még a fenti két parancs nélkül is ki tudja számítani k_{s2} k_{s3} –at a lehallgatott $suc^2(n_T) \oplus k_{s2}$ and $suc^3(n_T) \oplus k_{s3}$ üzenetváltásokból.

CRYPTO1 Cipher

A Plötz és társai fedték fel először a MiFare által használt CRYPTO1 titkosítót. Ennek a lelke egy 48-bites linear feedback shift register (LFSR), aminek a $g(x) = x^{48} + x^{43} + x^{39} + x^{38} + x^{36} + x^{34} + x^{33} + x^{31} + x^{29} + x^{24} + x^{23} + x^{21} + x^{19} + x^{13} + x^9 + x^7 + x^6 + x^5 + 1$ a generátor polinomja. A 20 rögzített állapotbitből a kulcsfolyam minden egyes órajelre ki van számítva. Az LFSR tartalmaz 18 elágazást, amik lineárisan kombinálva vannak, hogy minden órajelre kitöltsék ez első regiszter bitet. Az update függvény nem tartalmaz semmilyen nem-linearitást, ami mostani ismereteink szerint súlyos tervezési hiba.

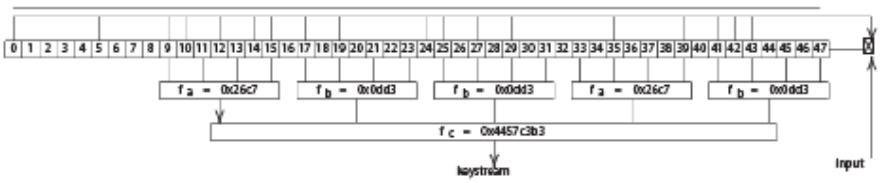
A regiszter minden órajellel balra shiftelődik eggyel, miközben a balról „kieső” bit el van hagyva. A feedback bit a g(x) polinom alapján számítódik ki. Van még egy input bit is, ami össze van XOR-va a feedback bittel és léptetéskor az LFSR jobb oldalán lép be. Ha az LFSR állapota egy adott k időpillanatban $r_k r_{k+1} \ldots r_{k+47}$, akkor k+1-ik pillanatban $r_{k+48} = r_k \oplus r_{k+5} \oplus r_{k+9} \oplus r_{k+10} \oplus r_{k+12} \oplus r_{k+14} \oplus r_{k+15} \oplus r_{k+17} \oplus r_{k+19} \oplus r_{k+24} \oplus r_{k+27} \oplus r_{k+29} \oplus r_{k+35} \oplus r_{k+39} \oplus rk_{+41} \oplus r_{k+42} \oplus r_{k+43} \oplus i$ lesz, ahol i az input bit. Az input bit csak az inicializáláshoz van felhasználva.
Titkosításkor a LFSR bizonyos bitjei – egy kezdetben ismeretlen – ún. f szűrő függvényen mennek keresztül. Az NXP-nek (Philipsnek) létezik egy Hitag2 nevű tag-ja is. Kripto protokolljának részletes leírása az fellelhető az Interneten. Kiindulási alap az volt, hogy a MiFare Crypto1 titkosítása hasonló a Hitag2-jéhöz. Ebből Garcia és csapata következtetéseket tudott levonni az f függvényről és a titkosító inicializálásáról.
Az LFSR inicializálása a hitelesítés fázisában történik meg. A korábbiak alapján tudjuk; ha $n_T \oplus$uid konstans marad, akkor az olvasó titkosított nonce-ja szintén konstans marad. Ebből az következik, hogy legelőször $n_T \oplus$uid kerül be LFSR-be. Ezen kívül a feedback bit variálásával sikerült a $n_T \oplus$uid-t és a K kulcsot oly módon módosítani, hogy titkosított szöveg a hitelesítés után is konstans maradjon. Ebből két dolog derült ki. Egyrészt az, hogy az LFSR-nek a K titkos kulcs a kezdőállapota. Másrészt pedig, hogy a Hitag2–ben szereplő kulcsfolyam visszacsatolás nincs implementálva a CRYPTO1-ben. Ez nagyban leegyszerűsítette a protokoll visszafejtését.
A hitelesítés következő lépéseként az olvasó nonce n_R is bekerül az LFSR-be. Ezzel véget is ér az inicializálás és az LFSR input bitjének máshol nincs szerepe. A következő ábra mutatja az inicializálás menetét az olvasó és a kártya szempontjából. A különbség csak annyi, hogy az olvasó legenerálja n_R-t és kiszámítja, majd elküldi $n_R \oplus ks_1$-t, a kártya veszi $n_R \oplus ks_1$-t és ebből számolja n_R-t :



13. ábra: Crypto1

A fentiekből kiderül, hogy a K, az uid, és az n_T tag nonce megfelelő választásával tökéletesen meg tudjuk határozni LFSR állapotát – pontosan azelőtt –, mielőtt az olvasó nonce-ja be lenne adva. Gyakorlatban ez annyit jelent, ha meg akarjuk figyelni LFSR állapotát egy α állapottól kezdődően, akkor a K kulcsot 0-ra állítjuk, hagyjuk, hogy a Ghost válasszon egy 0-nak megfelelő uid-t, és kiszámítjuk milyen n_T-t kellene küldenie, hogy ezt az α állapot elérjük. Ugyanakkor felmerül az a probléma, hogy az n_T csak 32 bit, az α pedig 48 bit hosszú. Azonban sok olvasó elfogadja, ha a hiányzó bitek 0-k.
LFSR állapotának megfelelő beállításával az f függvényre vonatkozóan nyerhetünk ki információt. Megfelelően megválasztva α-t, megfigyelhetjük $(n_R,0) \oplus f(\alpha)$-t, hiszen ezt küldi el az olvasó. Mivel minden

indulás után azonos nonce értéket generál az olvasó, ezért még ha nem is ismerjük (n_R,0)-t, azt konstansnak tekinthetjük. A következő lépésben ki kellett deríteni, hogy az LFSR melyik bitjei vannak bevezetve az f be. Ehhez egy véletlen α-t kell választani és megfigyelni (n_R,0)⊕f(α)-t. Majd meg kell változtatni egy bitet is ismét megfigyelni (n_R,0)⊕f(α_2)-t. Ha f(α) <> f(α_2), akkor az i-dik bit az f-nek bemenete. Visszafele irányban ez nem működik; több bit kell egyenlőség esetén ahhoz, hogy biztonsággal el tudjuk dönteni az adott bitről, hogy nem bemenete f-nek. A Hitag2-vel való hasonlóság miatt azt is feltételezték, hogy 5 darab első rétegbeli áramkör van, egyenként négy bemenettel. (9, 11, 13, 15 számúak a leg-baloldali és 41, 43, 45, 47 a leg-jobboldali áramkörökhöz.) Valamint, hogy első rétegbeli kimeneteket egy második rétegbeli áramkör kapja meg, ami ezekből állítja elő a kulcsfolyamot. Így vissza lehet fejteni az f szűrő függvényt.



14. ábra: LFSR és F függvény

LFSR állapot-visszaállítás

Az előzőek szerint a kártya által küldött nonce-val közvetlenül manipulálhatjuk az LFSR belső állapotát. Így – egy adott kulcsfolyam egy szegmensével – visszafejthető az LFSR állapota. Ehhez először egy ún. rainbow (gyorsító) táblát kell építeni az (lfsr, ks) párakból, ahol lfsr végigfut 0x000WWWWWWWW formájú állapotokon, míg ks az ezekhez az állapotokhoz tartozó kulcsfolyamok első 64 bitje. Ez a számítás 2^{36} sorból álló táblázatot generál. Ez a számítás egy asztali PC-n végezve kb. 6 órán keresztül fut, és a kiszámított adatok kb. 1 terabyte-nyi memórát foglalnak le.
Egy adott kártyaolvasó megtámadásához minden egyes 0xXXX formájú 12 bites számhoz azonos uid-vel kell lefuttatni a hitelesítést. Beállítjuk kártya a challenge nonce-át n_T = 0x0000XXX0-ra. Erre az olvasó válasza: n_R⊕ks_1, $suc^2(n_T)$⊕ks_2, amire nem reagálunk. Ekkor – a korábbiak szerint – a legtöbb olvasó halt⊕ks_3-t küld, és vissza tudjuk fejteni ks_2, ks_3-at. Pontosan egy 0xXXX érték lesz, ami visszaadja az 0xYYYYYYYY000Y formájú LFSR állapotot n_T = 0x0000XXX0-t beadása után.
Miközben n_R olvasó nonce beadódik, az LFSR-ben a nullák balra mozognak 000YZZZZZZZZ formájú állapotot adva. Miután minden LFSR állapot ilyen formában szerepel a rainbow táblánkban, vissza tudjuk állítani ks_2, ks_3 keresésével. Vissza tudjuk állítani a LFSR állapotát abban az esetben is, amikor az olvasó nem küld vissza „halt" választ. Ilyenkor minden egyes 0xXXX-höz csak a ks_2-öt keressük a táblában. Mivel összesen 2^{48-12} bejegyzés van, és ks_2 32 bit hosszú, ezért átlagban 2^4 megfelelést kapunk. Miután 2^{12} lehetséges 0xXXX-et veszünk figyelembe, ezért ebből összesen 2^{16} lehetséges LFSR állapotot kapunk. Minden ilyen LFSR állapot ad számunkra egy lehetséges kulcsot, amik közül – a korábban már említett

részleges bejelentkeztetési eljárással – ki lehet választani a megfelelőt. A gyakorlatban 5-35 darab részleges bejelentkezési menet lehetséges másodpercenként. Így a szükséges 4096 darab részleges bejelentkezés generálása csupán 2–14 percig tart.

LFSR Rollback

Egy bármilyen $r_k r_{k+1} \dots r_{k+47}$ állapot alapján az (1XXX) egyenletből ki lehet számítani LFSR azt megelőző $r_{k-1} r_k \dots r_{k+46}$ állapotát a következőek szerint. Tegyük fel, hogy a korábbiaknak megfelelően megszereztük az LFSR-nek éppen azt az állapotát, amikor az olvasó nonce be lett adva, és le tudtuk hallgatni a titkosított olvasó nonce-ot.

Az f szűrő függvény bemenete nem tartalmazza az LFSR leg-baloldali bitjét. Ha jobbra shifteljük egyszer az LFSR-t, miközben a bemenetére egy r értékű bitet adunk, aminek értéke nincs meghatározva. Majd kiszámítjuk f-et és megkapunk egy bitet a kulcsfolyamból, ami $n_{R,31}$ titkosításánál használva volt. A titkosított n_R-rel vissza lehet állítani $n_{R,31}$-et. Az LFSR feedback-jét feldolgozva be lehet állítani r megfelelő értékét, tehát LFSR $n_{R,31}$ küldése előtti állapotba kerül. Ezt az eljárást 31-szer megismételve megkapjuk LFSR az olvasó nonce beküldése előtti teljes állapotát. Mivel a kártya nonce és uid nyílt szövegként van átküldve, visszaállítjuk LFSR $n_T \oplus$uid előtti állapotát is megkapjuk, ami maga a titkos kulcs!

Szűrő függvény páratlan bemenetei

Az f szűrőbe az LFSR-nek csak páratlan sorszámú kimenetei vannak bevezetve. Ez az egyenletesség kihasználható. Adott kulcsfolyam egy részével előállíthatjuk LFSR azon releváns bitjeit, ami a kulcsfolyam páros bitjeit adják meg, és azokat, amik a kulcsfolyam páratlan bitjeit adják meg. A visszacsatolást is két részre vágva, majd a két részt összefűzve visszaállíthatjuk LFSR azon állapotait, amik egy adott kulcsfolyamot állítanak elő. Ezt f egyfajta invertálását jelenti.

Legyen $b_0 b_1 \dots b_{n-1}$ n darab egymást követő kulcsfolyam bit. A gyakorlatban 32 vagy 64 darabnyi. Vissza szeretnénk állítani az LFSR minden olyan állapotát, amik előállítják ezt a kulcsfolyamot. Azaz minden olyan $\overline{r} = r_0 r_1 \dots r_{46+n}$ szekvenciát keresünk, hogy $r_k \oplus r_{k+5} \oplus r_{k+9} \oplus r_{k+10} \oplus r_{k+12} \oplus r_{k+14} \oplus r_{k+15} \oplus r_{k+17} \oplus r_{k+19} \oplus r_{k+24} \oplus r_{k+25} \oplus r_{k+27} \oplus r_{k+29} \oplus r_{k+35} \oplus r_{k+39} \oplus r_{k+41} \oplus r_{k+42} \oplus r_{k+43} \oplus r_{k+48} = 0$, minden $k \in \{0, \dots, n-2\}$-ra, úgy, hogy $f(r_k \dots r_{k+47}) = b_k$, minden $k \in \{0, \dots, n-1\}$-re.

Az első feltétel azt jelenti, hogy LFSR $\overline{r}$-et generálja egymás után. A második pedig azt, hogy a szükséges kulcsfolyamot generálja. Mivel f LFSR 20 páratlan bitjétől függ csak, ezért a támadáshoz két darab egyenként kb. 2^{19} elemből álló rainbow táblázatot generáltak az LFSR szekvenciák páros és páratlan számú bitjenek, amelyekből a kulcsfolyam páros és a páratlan bitjei állítódnak elő. A kulcsfolyam első b_0 bitjét tekintve minden $s_0 s_1 \dots s_{19}$ 20 bites szekvenciát generálunk úgy, hogy $f(s_0, s_1, \dots, s_{19}) = b_0$. f struktúrája garantálja, hogy pontosan 2^{19} ilyen szekvencia van. A keresett LFSR $\overline{r}$ szekvenciáknak kell legyen egy ilyen szekvenciája mint $r_0, r_1, \dots, r_{19}$. A rainbow tábla minden bejegyzésére a következőket csináljuk. Úgy tekintjük a bejegyzést, mintha az LFSR 9, 11, . . . , 47 bitje lenne. Majd két pozícióval balra shifteljük az LFSR-t. A s_{20} feedback másodiknak van beforgatva. Azonban nincs további információnk a feedback bitről, mivel nem tudunk semmit sem az LFST páros számú sem az alacsony számozású páratlan bitjeiről. Leellenőrizhetjük ugyanakkor, s_{20} bit két értéke közül melyik egyezik a kulcsfolyammal, azaz melyik elégíti ki az $f(s_1, s_2, \dots, s_{20}) = b_2$-t. Ha s_{20}-nak csak egy értéke egyezik, kiegészítjük bejegyzésünket a táblában s_{20}-al. Ha mindkettő megegyezik, megkettőzzük a bejegyzést kiterjesztve egyszer 0-val, egyszer

1-gyel. Ha nincs egyezés, kitöröljük a bejegyzést. A tábla mérete nem változik, mert átlagban az alkalmak ¼-ében duplikálunk, ¼-ében törlünk és felében kiterjesztjük a bejegyzéseket. Megismételjük ezt az eljárást a kulcsfolyam $b_4, b_6, \ldots, b_{n-1}$ bitjeire is. Ez alapján kb. 2^{19} $s_0s_1 \ldots s_{19+n/2}$ bejegyzést tartalmazó táblát kapunk, úgy hogy $f(s_i, s_{i+1}, \ldots, s_{i+19}) = b_{2i}$ minden $i \in \{0, 1, \ldots, n/2\}$. Következésképpen az LFSR $\overline{r}$ keresett szekvenciái meg kell hogy feleljenek a tábla bejegyzésének $r_9, r_{11}, \ldots, r_{47+n}$-formában. Hasonlóan kapunk egy kb. 2^{19} bejegyzésből álló $t_0t_1 \ldots t_{19+n/2}$ táblát $f(t_i, t_{i+1}, \ldots, t_{i+19}) = b_{2i+1}$ tulajdonsággal, ahol minden $i \in \{0, 1, \ldots, n/2\}$. Mindkét tábla 4 darab kiterjesztése után minden bejegyzés 24 hosszú már lehet próbálkozni minden egyes $s_0s_1 \ldots s_{23}$ az első és $t_0t_1 \ldots t_{23}$ a második táblából, hogy előállítják-e a korrekt kulcsfolyamot. Ezzel lecsökken a nyers erős támadáshoz szükséges esetek száma 2^{48}-ról 2^{38} darabra. Tovább lehet csökkenteni a keresés komplexitását a következő módszerrel. Tekintsünk egy $\overline{s}$ = s0s1 . . . s19+n/2 bejegyzést az első és egy $\overline{t}$ = t0t1 . . . t19+n/2 bejegyzést a második táblából. Ahhoz, hogy $\overline{r}$ = s0t0s1 . . . t19+n/2 –t előállítsa LFSR, szükséges és elégséges, hogy minden 49 egymást követő bit elégítse ki az LFSR egyenletet (2), vagyis a 49-dik bit feedback bitnek kell lennie, ami az előző 48 bit alapján generálódott. Tehát, a $\overline{s}$ 25 egymást követő minden egyes $s_is_{i+1} \ldots s_{i+24}$ alszekvenciájára számoljuk ki $b_i^{1,\bar{s}} = s_k \oplus s_{i+5} \oplus s_{i+6} \oplus s_{i+7} \oplus s_{i+12} \oplus s_{i+21} \oplus s_{i+24}$ hozzájárulásukat az LFSR relációhoz, és $\overline{t}$- 25 egymást követő minden egyes $t_it_{i+1} \ldots t_{i+24}$ alszekvenciájára számoljuk ki $b_i^{2,\bar{t}} = t_{i+2} \oplus t_{i+4} \oplus t_{i+7} \oplus t_{i+8} \oplus t_{i+9} \oplus t_{i+12} \oplus t_{i+13} \oplus t_{i+14} \oplus t_{i+17} \oplus t_{i+19} \oplus t_{i+20} \oplus t_{i+21}$ hozzájárulását az LFSR relációhoz.

Ha $s_0t_0s_1 \ldots t_{n/2}$-t generálja LFSR, akkor $b_i^{1,\bar{s}} = b_i^{2,\bar{t}}$ minden $i \in \{0, \ldots, n/2 - 5\}$.

Szimmetrikusan; $\overline{s}$ 24 egymást követő minden alszekvenciájára, és $\overline{t}$-nek az azokhoz tartozó 25 egymást követő bitjeihez kiszámoljuk $\tilde{b}_i^{1,\bar{s}} = s_{i+2} \oplus s_{i+4} \oplus s_{i+7} \oplus s_{i+8} \oplus s_{i+9} \oplus s_{i+12} \oplus s_{i+13} \oplus s_{i+14} \oplus s_{i+17} \oplus s_{i+19} \oplus s_{i+20} \oplus s_{i+21}$ –t, és $\tilde{b}_i^{2,\bar{s}} = t_i \oplus t_{i+5} \oplus t_{i+6} \oplus t_{i+7} \oplus t_{i+12} \oplus t_{i+21} \oplus t_{i+24}$–t. Itt szintén, ha LSFR generálja $s_0t_0s_1 \ldots t_{n/2}$ –t is, akkor $\tilde{b}_i^{1,\bar{s}} = \tilde{b}_i^{2,\bar{t}}$ minden $i \in \{0, \ldots, n/2 - 5\}$–re.

A keresett LFSR állapotszekvenciák hatékony meghatározása miatt rendezni kell az első táblát az újonnan kiszámolt bitek $b_0^{1,\bar{s}} \ldots b_{\frac{n}{2-5}}^{1,\bar{s}}$ $\tilde{b}_0^{1,\bar{s}} \ldots \tilde{b}_{\frac{n}{2-5}}^{1,\bar{s}}$ és a második táblát $b_0^{2,\bar{t}} \ldots b_{\frac{n}{2-5}}^{2,\bar{t}}$ $\tilde{b}_0^{2,\bar{t}} \ldots \tilde{b}_{\frac{n}{2-5}}^{2,\bar{t}}$ alapján.

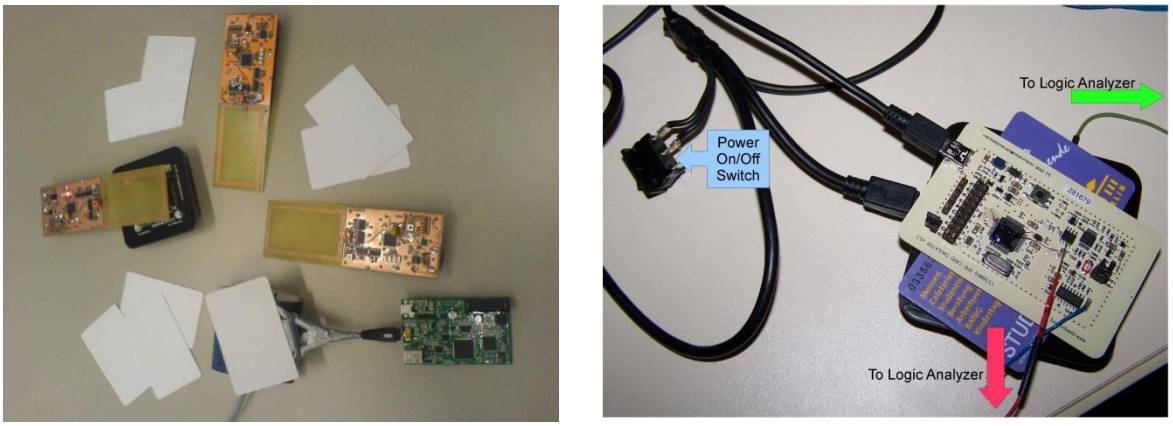
Miután az LFSR akkor és csak akkor generálja $s_0t_0s_1 \ldots t_{n/2}$–t, ha $b^{1,\bar{s}}\tilde{b}^{1,\bar{s}} = b^{2,\bar{t}}\tilde{b}^{2,\bar{t}}$, és a konstrukció miatt generálja a szükséges kulcsfolyamot, nem kell tovább keresni. A számítási idő lecsökken n ciklusra a két egyenként kb. 2^{19} méretű táblában, valamint ezen táblák két rendezésére.

Paritásbit képzés

Az átviteli protokoll paritásbitet generál és küld minden 8 bithez. A paritást nem a nyílt szöveg alapján számolja ki, hanem titkosítva van, mégpedig a következő bithez használt kulcsfolyam bittel. Ily módon a nyílt szöveg minden egyes byte-jából egy bitnyi információt kiszivárogtat. Ez a brute-force támadások során lényegesen lecsökkentheti a keresési teret.

Többszektoros azonosítás

Sok rendszer több szektorra azonosítja magát. A második szektortól kezdődően a protokoll kicsit más. Mivel ilyenkor már van egy kulcs, az új hitelesítési parancs titkosítva van ezzel. Ebben a szakaszban az új szektor K titkos kulcsa van az LFSR-be betöltve. A különbség csak annyi, hogy a kártya n_T nonce K-val titkosítva van miközben be van adva az LFSR-be.
Ettől a ponttól kezdődően a protokoll többi része már teljesen úgy folytatódik, mint az egy szektoros esetben. A kártya klónozásához mindazon információra szükség van, amit az olvasó kiolvas a kártyából, és ez általában több szektort érint. Ehhez először egy
olyan egyedüli, teljes szakaszt kell kihallgatni, ami többszektoros hitelesítést tartalmaz. Miután a korábbiak alapján visszaállítottuk a kulcsot az első szektorhoz a másodikkal próbálkozunk. Mint említettük ilyenkor már titkosítva van az egész hitelesítési kérés, de éppen azzal a kulccsal, amit éppen fel lett derítve!



15. ábra: Feltöréshez használt eszközök

Referenciák

1. Garcia F. D., de Koning Gans, R. Muijrers, P. van Rossum, R. Verdult, R. Wichers Schreur, és B. Jacobs: Dismantling MIFARE Classic (2008)
2. de Koning Gans, G. Hoepman, Garcia, F.D.: A practical attack on the MIFARE Classic. (2008)
3. Nohl, K., Evans, D., Starbug, Plötz, H.: Reverse-engineering a cryptographic RFID tag. (2008)
4. Nohl, K., Plötz, H.: Mifare, little security, despite obscurity. (2007)
5. Teepe, W., Nohl, K.: Making the best of MIFARE Classic (2008)