# Presentació

En aquesta pràctica implementarem una simplificació de l'algorisme de xifrat de flux Crypto1, que es troba en dispositius RFID de la marca MIFARE i realitzarem un atac contra textos xifrats que s'han creat reutilitzant una mateixa clau.

# Objectius

Els objectius d'aquesta pràctica són:

1. Estudiar amb detall com funciona un xifrador de flux concret: l'algorisme Crypto1 que incorporen algunes targetes RFID.
2. Analitzar les conseqüències de reutilitzar una clau en un xifrador de flux.

# Descripció de la Pràctica

Els xifradors en flux són utilitzats per al xifrat d'informació en temps real ja que la seva simplicitat d'implementació els confereix una velocitat molt elevada. Més enllà d'aquest àmbit, la simplicitat dels xifradors en flux, en especial aquells basats en LFSRs, els fa atractius en àmbits que requereixen sistemes computacionalment poc costosos. Un d'aquests àmbits és el camp dels RFIDs (identificadors per ràdio freqüència, en anglès, Radio Frequency IDentifiers). El camp d'aplicació dels RFIDs és molt ampli, des dels més simples, que serveixen per etiquetar productes i seran els substituts naturals dels omnipresents codis de barres, fins a les targetes de pagament o autenticació (com ara les de les estacions d'esquí o el servei de bicicletes de Barcelona, Bicing) passant per la identificació electrònica dels passaports.

Una de les problemàtiques que s'ha donat en els últims anys en el sector dels RFIDs és que els fabricants han desenvolupat els sistemes de seguretat d'aquests dispositius sense aplicar el principi de Kerckhoffs. Així, tant els algorismes de xifrat com els generadors pseudoaleatoris que els componen han estat desenvolupats sota secret industrial i, a priori, no se'n coneixen els detalls. Això fa que la seguretat d'aquests sistemes no hagi pogut ser avaluada per experts independents i per tant, les companyies, s'exposen a que els sistemes desenvolupats puguin ser vulnerables.

Aquest és el cas d'un dels dispositius RFID de la marca MIFARE, concretament el model MIFARE Classic, comercialitzats per la companyia NXP Semiconductors i que en l'actualitat encara s'utilit-

zen en molts sistemes en producció. Aquest dispositiu incorpora un xifrador en flux, conegut com a Crypto1, que va ser desenvolupat per la pròpia companyia sense que aquesta proporcionés cap detall del seu funcionament[1]. Així, quan posteriorment es va conèixer el detall i funcionament sistema de xifrat que empra el dispositiu, el seu anàlisi va mostrar que el disseny no era suficientment segur.

En aquesta pràctica implementarem una versió reduïda de l'algorisme de xifrat Crypto1 que incorporen aquest tipus de targes RFID. El funcionament de la versió simplificada que implementarem es detalla en la següent figura:

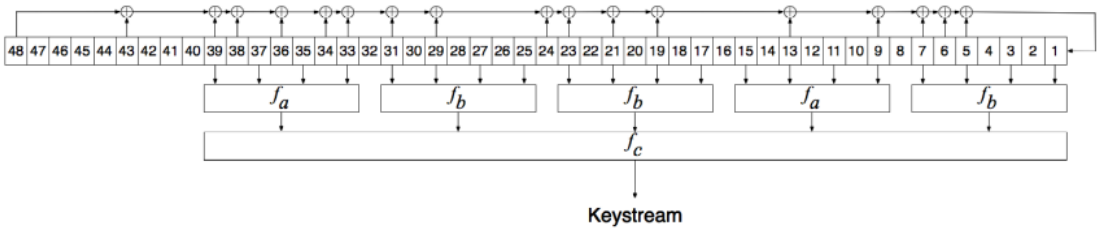


Figura 1: Esquema algorisme Crypto1.

Com es pot veure, el generador pseudoaleatori està format per un LFSR de 48 cel·les que té com a polinomi de connexions,

$$x^{48}+x^{43}+x^{39}+x^{38}+x^{36}+x^{34}+x^{33}+x^{31}+x^{29}+x^{24}+x^{23}+x^{21}+x^{19}+x^{13}+x^{9}+x^{7}+x^{6}+x^{5}+1$$

i un seguit de funcions no lineals descrites segons les següents equacions:

$$f_a(x_0,x_1,x_2,x_3) = ((x_0 \vee x_1) \oplus (x_0 \wedge x_3)) \oplus (x_2 \wedge ((x_0 \oplus x_1) \vee x_3))$$

$$f_b(x_0,x_1,x_2,x_3) = ((x_0 \wedge x_1) \vee x_2) \oplus ((x_0 \oplus x_1) \wedge (x_2 \vee x_3))$$

$$f_c(x_0,x_1,x_2,x_3,x_4) = (x_0 \vee ((x_1 \vee x_4) \wedge (x_3 \oplus x_4))) \oplus ((x_0 \oplus (x_1 \wedge x_3)) \wedge ((x_2 \oplus x_3) \vee (x_1 \wedge x_4)))$$

on els símbols $\vee$, $\wedge$ i $\oplus$ corresponen als operadors lògics OR, AND i XOR respectivament i els valors es prenen d'esquerra a dreta, és a dir, per a la funció $f_a$ tenim que $x_0$ és la posició 39, $x_1$ és la 37, etc.

[1] Si bé el cas d'aquesta companyia és conegut perquè s'ha aconseguit obtenir l'esquema de xifrat que es mantenia en secret i trencar la seva seguretat, és molt possible que sistemes de xifrat desenvolupats en secret per altres companyies puguin tenir també problemes de seguretat.

Per a desenvolupar el generador cal que implementeu les següents funcions que us permetran validar la correcció de la vostra implementació amb el joc de proves que us proporcionem. Per a desenvolupar el generador podeu fer servir la següent funció del paquet matemàtic SAGE: `sage.crypto.lfsr.lfsr_sequence(key, fill, n)`.

# 1 Primera part (7 punts)

1. Funció que implementa un LFSR. (**0.5 punts**)

   La funció prendrà com a variables d'entrada tres valors: `polinomi`, `estat_inicial` i `bits_de_sortida`; i retornarà la seqüència de sortida.

   - A la variable `polinomi` s'escriurà el polinomi de connexions de l'LFSR;
   - La variable `estat_inicial` contindrà l'estat inicial de l'LFSR;
   - La variable `bits_de_sortida` contindrà el nombre de bits de la seqüència de sortida.
   - La funció retornarà la seqüència de sortida.

   Exemple:

   - `polinomi`: `[1,1,0,0]` (equival a $x^4 + x^3 + 1$ on el coeficient corresponent al terme independent no s'especifica en el vector)
   - `estat_inicial`: `[1,0,0,0]`
   - `bits_de_sortida`: 20
   - `sortida`: `[1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1]`

2. Funcions que implementen els filtres no lineals. (**1.5 punts**)

   S'implementaran per separat cada una de les tres funcions de filtre no lineals ($f_a, f_b, f_c$) que prendran com a variable d'entrada un valor i en retornaran la sortida.

   - A la variable entrada s'hi indicaran els bits d'entrada (4 per a les funcions $f_a, f_b$ i 5 per a la funció $f_c$).
   - Cada funció retornarà el bit de sortida corresponent.

3. Funció que implementa la totalitat del filtre no lineal. (**1 punt**)

   Aquesta funció integrarà el conjunt de les funcions no lineals implementades anteriorment. Aquesta funció prendrà com a variable entrada un valor i retornarà el valor del bit de sortida.

   - La variable entrada contindrà 48 bits (que correspondran a l'estat de l'LFSR)

- La funció retornarà el bit de sortida.

4. Funció que implementa la totalitat del generador pseudoaleatori. (**2 punts**)

   Aquesta funció implementarà tot el generador tal i com es descriu en el gràfic mostrat en la Figura 1. Així, la funció prendrà com a variables d'entrada dos valors: `clau` i `bits_de_sortida`; i retornarà la seqüència de sortida.

   - La variable `clau` contindrà els 48 bits de l'estat inicial de l'LFSR del sistema.
   - La variable `bits_de_sortida` contindrà el nombre total de bits de sortida del generador.
   - La funció retornarà la seqüència de sortida.

5. Funció que xifra i desxifra informació utilitzant el generador pseudoaleatori. (**2 punts**)

   Aquesta funció implementarà un algorisme de xifrat en flux on el generador pseudoaleatori serà la implementació de la funció del punt 4. La funció prendrà com a variables d'entrada tres valors: `clau`, `mode` i `missatge`; i retornarà la seqüència de sortida.

   - La variable `clau` contindrà els 48 bits de l'estat inicial de l'LFSR del sistema.
   - La variable `mode` contindrà el caràcter "`e`" (encrypt) si es vol xifrar o bé el caràcter "`d`" (decipher) si es vol desxifrar.
   - La variable `missatge` contindrà o bé una cadena de caràcters amb el text a xifrar (mode "`e`") o bé una cadena de bits amb el text a desxifrar (mode "`d`").
   - La variable seqüència de sortida contindrà o bé una cadena de caràcters amb el text desxifrat (mode "`d`") o bé una cadena de bits amb el text xifrat (mode "`e`").

   Exemple de xifrat:

   - `clau`: `[0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`
   - `mode`: "`e`"
   - `missatge`: "`CRIPTO`"
   - `sortida`: "`000110100100011000001100000000000101110000011101`"

   Exemple de desxifrat:

   - `clau`: `[0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`
   - `mode`: "`d`"
   - `missatge`: "`000110100100011000001100000000000101110000011101`"
   - `sortida`: "`CRIPTO`"

# 2 Segona part (3 punts)

Un dels problemes que poden sorgir a l'hora de fer servir criptosistemes de flux aparentment segurs es produeix al reutilitzar la mateixa clau per xifrar textos diferents reiteradament. En aquesta segona part de la pràctica, veurem com podem atacar el criptosistema implementat en aquesta pràctica si es fa servir la mateixa clau més d'una vegada.

El problema que tenim al xifrar dos textos amb la mateixa clau utilitzant una `xor` és que

$$c_0 = m_0 \oplus k'$$
$$c_1 = m_1 \oplus k'$$
$$c_0 \oplus c_1 = m_0 \oplus k' \oplus m_1 \oplus k' = m_0 \oplus m_1$$

És a dir, fer una `xor` entre dos textos xifrats ens dóna com a resultat el valor de la `xor` dels dos textos en clar (ja que les claus es cancel·len entre elles). Vegem ara com podem aprofitar aquest fet per descobrir els textos en clar.

Per a aquest exercici, sabem que els textos en clar són cadenes de caràcters codificades en ASCII i, a més, els caràcters utilitzats són només lletres en majúscula o bé espais. Amb aquesta informació, podem analitzar quina codificació tenen els caràcters en ASCII i què passa quan fem una `xor` entre dos d'aquests caràcters:

| $char_1$ | | $char_2$ | | $char_1 \oplus char_2$ |
|---|---|---|---|---|
| A | 0x41 | A | 0x41 | 0x00 |
| A | 0x41 | B | 0x42 | 0x03 |
| A | 0x41 | C | 0x43 | 0x02 |
| ... | ... | ... | ... | ... |
| A | 0x41 | Y | 0x59 | 0x18 |
| A | 0x41 | Z | 0x5A | 0x1b |
| ... | ... | ... | ... | ... |
| Z | 0x5A | Z | 0x5A | 0x00 |
| A | 0x41 | (espai) | 0x20 | 0x61 |
| ... | ... | ... | ... | ... |
| Z | 0x5A | (espai) | 0x20 | 0x7A |
| (espai) | 0x20 | (espai) | 0x20 | 0x00 |

És clar, doncs, que observant el resultat de la `xor` entre dos caràcters podem detectar si un d'ells és un espai o no. Haurem d'anar amb compte al tractar els casos on el resultat de la `xor` sigui

`0x00`, ja que aquests simplement ens indicaran que els dos caràcters són iguals (independentment de que siguin lletres majúscules o espais).

Després, podem aprofitar aquest fet per localitzar els espais de tots els textos. Una vegada tenim els espais de cada text identificats, podrem deduir amb facilitat quin és el caràcter que hi ha en qualsevol text per cadascuna de les posicions on hem trobat un espai en algun text. De la mateixa manera, també podrem anar descobrint els valors de la clau.

Arribats a aquest punt, potser ens quedarà alguna part de la clau de la qual en desconeixerem el valor (aquelles posicions on no hi ha cap espai en cap dels textos). Tenint en compte que tindrem tots els textos desxifrats gairebé completament, ens serà fàcil deduir-ne el contingut restant.

Per tal d'implementar el nostre atac, desenvoluparem les següents funcions:

1. Funció que detecta espais en textos xifrats amb la mateixa clau utilitzant un xifrador en flux. (**1 punt**)

   Aquesta funció detectarà en quines posicions hi ha espais en blanc en un conjunt de textos xifrats amb un xifrador de flux amb la mateixa clau. La funció rebrà com a entrada una llista amb tots els textos xifrats (representants com a cadenes hexadecimals) i retornarà una altra llista amb els espais en blanc detectats en cadascun dels textos.

   - La variable `ciphertexts` contindrà una llista de cadenes hexadecimals amb els textos xifrats.
   - La funció retornarà una llista amb els espais detectats per cada text xifrat. Per cada espai (que queda representat per dues posicions hexadecimals), la funció retornarà la posició del primer caràcter hexadecimal.

   Exemple:

   - `ciphertexts`: `["0517119A", "01131596", "0D7519FE"]`
   - `sortida`: `[[], [], [2, 6]]` (és a dir, no s'han detectat espais en els dos primers textos, i el tercer text té dos espais, que comencen a les posicions 2 i 6 (`0x75` i `0xFE`).

2. Funció que recupera una clau a partir de textos xifrats amb la mateixa clau. (**1 punt**)

   A partir d'un conjunt de textos xifrats amb un xifrador de flux fent servir una mateixa clau, aquesta funció recuperarà la seqüència de xifrat utilitzada. Per fer-ho, farà servir la funció anterior per detectar espais en els textos xifrats, i recuperarà la clau per totes les posicions on hi ha un espai en algun dels textos xifrats. Les posicions de la clau que no s'hagin pogut inferir s'indicaran amb dos interrogants ("??").

- La variable `ciphertexts` contindrà una llista de cadenes hexadecimals amb els textos xifrats.
- La funció retornarà una cadena de caràcters hexadecimals amb la clau inferida a partir dels textos xifrats.

Exemple:

- `ciphertexts: ["0517119A", "01131596", "0D7519FE"]`
- `sortida: "??55??DE"`

3. Funció que recupera el text en clar d'un missatge a partir d'un conjunt de missatges xifrats amb la mateixa clau. (**1 punt**)

   Utilitzant les dues funcions anteriors, aquesta funció recuperarà el text en clar de l'últim missatge de la llista de textos xifrats rebuts.

   - La variable `ciphertexts` contindrà una llista de cadenes hexadecimals amb els textos xifrats.
   - La funció retornarà el text en clar corresponent a l'últim missatge de la llista ciphertexts.

   Feu servir aquesta funció per desxifrar el següent missatge:

```
target_ciphertext = "
      36445ABB21254A7DB645DAAE2BC199DFDB961BD00ED72388B3F771FF6
      B0405C28594F650C27D583809911886BEF31CD99821C5B959806E5544
      2517CBF57268AE64238C1318F3DE5BB84BB6014FEC2C1C456D5843EFC
      2A0BF086B6C9F"
```

   tenint en compte que ha estat xifrat amb la mateixa clau que els missatges que trobareu a l'esquelet de la pràctica.

   Nota: no hi ha funcions de test per a aquesta última funció, cal que desxifreu el missatge `target_ciphertext` i que inclogueu el resultat en el mateix worksheet de sage.

## Criteris d'avaluació

La puntuació de cada exercici es troba detallada a l'enunciat.

## Format i data de lliurament

La data màxima de lliurament de la pràctica és el **20/11/2017** (a les 24 hores).

Juntament amb l'enunciat de la pràctica trobareu l'esquelet de la mateixa en format SAGE notebook worksheet (extensió .sws). Aquest mateix fitxer és el que heu de lliurar un cop hi codifiqueu totes les funcions. En aquest esquelet també hi trobareu inclosos els jocs de proves dels diferents apartats. Tal com s'esmenta en el fitxer sws, no es pot modificar cap part del fitxer corresponent al joc de proves. El lliurament de la pràctica constarà de d'un únic fitxer SAGE notebook (extensió .sws) o bé sage cloud (extensió .sagews) on hagueu inclòs la vostra implementació.