Cryptanalysis in real life II

研究團隊 王昶凱 邱弈豪 周立平 鄭振牟
演講者 周立平

綱要

- 1. Mifare Classic(Crypto1)與Hitag2的介紹
  - YoYo card in Taipei
- 2. 攻擊手法和結果概述
- 3. Extra: Non-NIST ciphers
- 4. Q&A

Crypto1與Hitag2的介紹

- NXP 公司所發售的產品 (Mifare Series & hitag2)
- Mifare Classic 系列的智慧卡使用 Crypto1，已售5000萬個MIFARE讀卡機，50億個tag，使用範圍相當廣泛(地鐵等)
- Hitag2使用於汽車電子安全鎖

Crypto1與Hitag2的介紹

- Mifare Classic - Crypto1
  - Ex. YoYo 卡
  - Issue： Crypto1 Structure、Mifare Classic protocol
- Hitag2
  - Hitag2 is very similar to Mifare Classic but stronger
- 攻擊手法：
  - 代數, 差分攻擊, or Both

Crypto1與Hitag2的介紹

- Mifare Classic protocol

讀卡機
Reader

卡片
Tag

ISO14443

我要與你通訊

OK!我選擇Mifare與我通訊

Mifare Classic protocol

問一個問題：隨機亂數nT

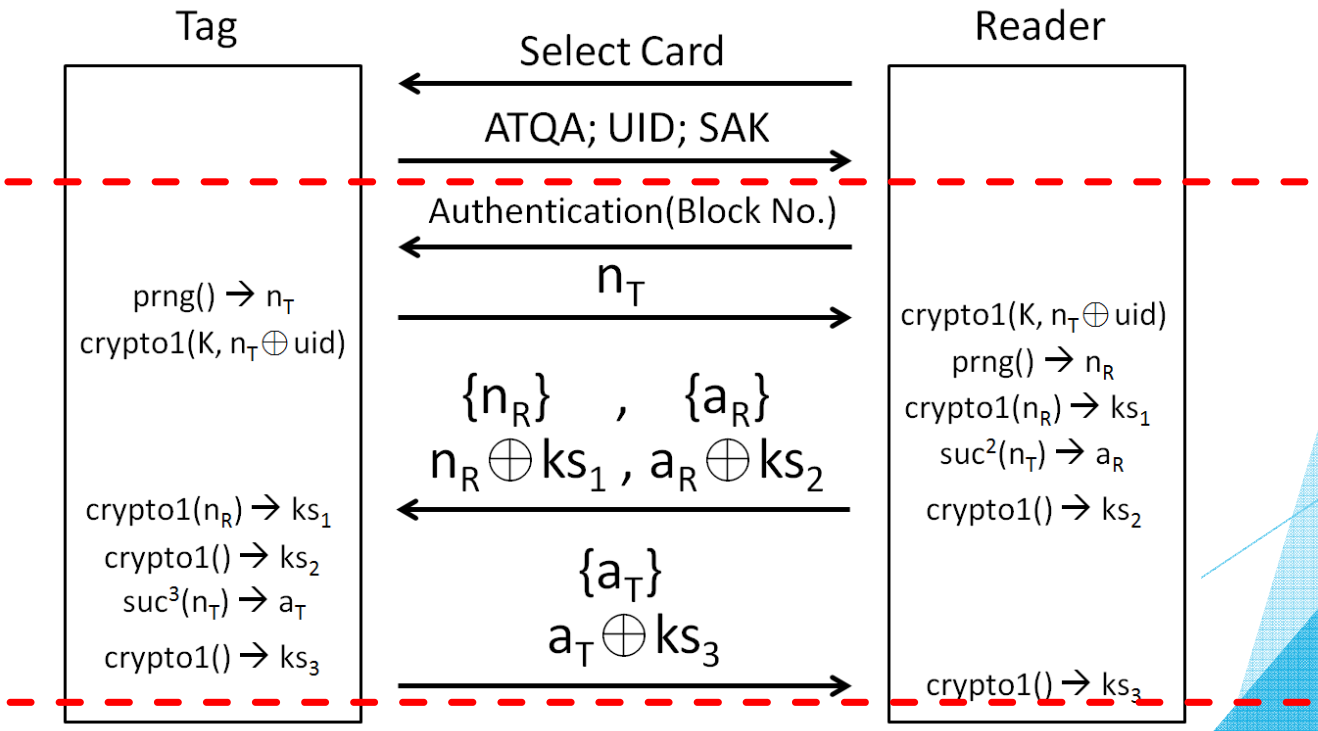
先問你一個問題：隨機亂數{nR}
回答問題：{aR}從nT計算出來

讀卡機回答正確：卡片回答問題{aT} ->從nR計算出來
讀卡機回答錯誤：卡片回答 -> error code (0x5)(0x0)

認證完成，開始進行讀、寫..等動作

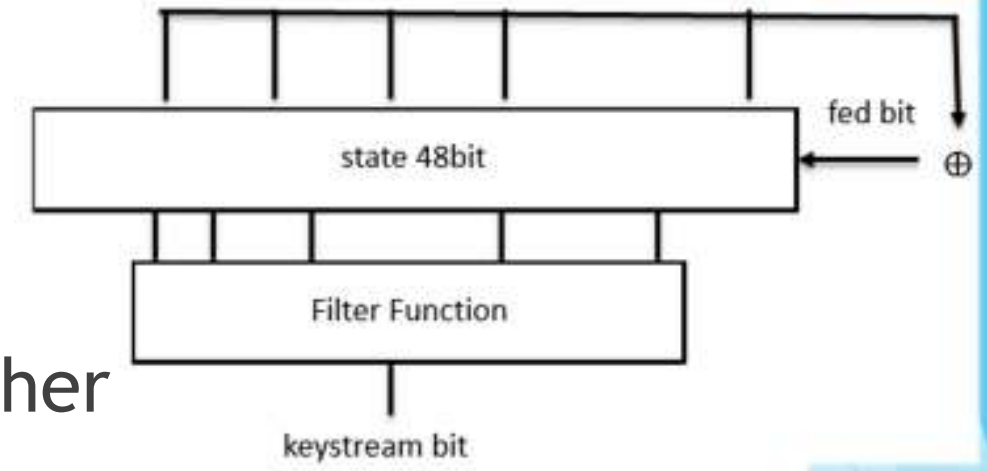
Crypto1與Hitag2的介紹

- MIFARE Classic
  - CHAP(Challenge-Handshake Authentication Protocol)



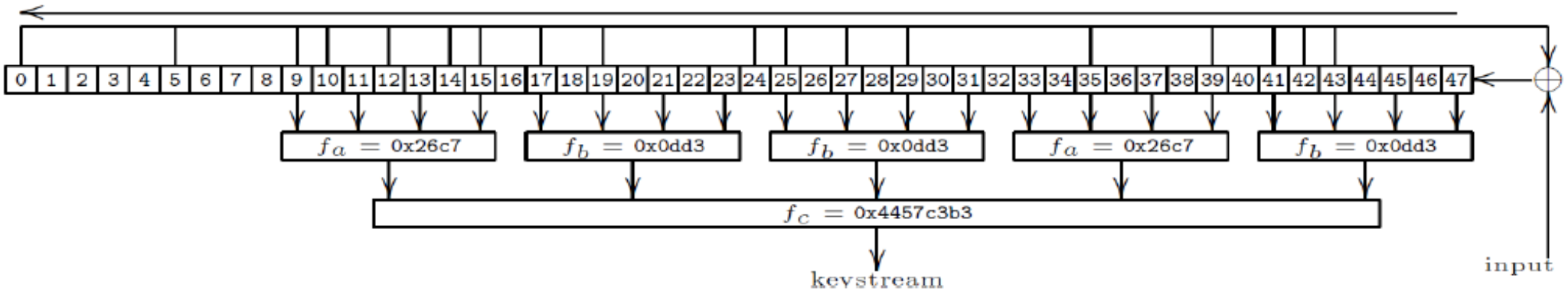
Crypto1與Hitag2的介紹
- Structure

- Structure
  - Crypto1 ->Stream cipher
    - Easy LFSR
  - Hitag2 ->Stream cipher
    - Hard LFSR

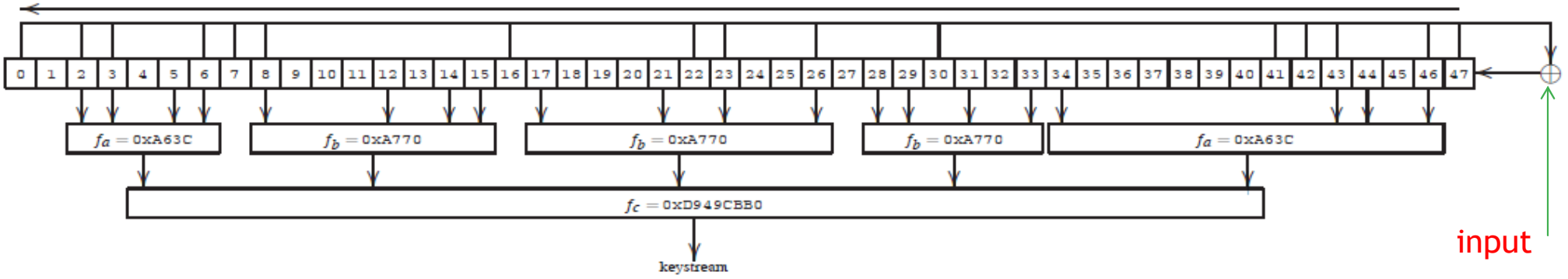


Crypto1與Hitag2的介紹
- Structure

- Mifare Classic - Crypto1



- Mifare Classic – Hitag2



探討Mifare Classic 既有的安全議題

- Brute force
  - Crypto1 ： CPU 50 years / GPU 14 hrs
  - Hitag2 ： N/A
- Non-Brute force
  - PRNG(Pseudo random number generator)設計不良使其安全性降低
  - Parity 設計不良會洩漏部分的明文資訊
  - Filter Function 輸入部分設計不良
  - 利用 keystream 將原始的密鑰還原

281/sec (use every possible key value to authenticate)
2^48/281 = 1001690308578.84 sec = 16694838476.31 min = 278247307.93 hr = 11593637.83 day = 31763.39 year

探討Mifare Classic 既有的安全議題

Attack method	Miffare Classic Crypto1		
	Practical	Computation	特殊條件
Brute force	O	14 小時	有特殊 GPU 運算能力的電腦
奇偶攻擊	O	< 2 秒	需要到真讀卡機設備旁側錄通訊過程
連續認證攻擊	O	約 25 秒	需要先知道卡片裡的其中一把 key
代數差分攻擊	O	2~10 分鐘	僅需用假讀卡機取得通訊資料

探討Mifare Classic 既有的安全議題

New card

Attack method	Miffare Classic Crypto1		
	Practical	Computation	特殊條件
Brute force	永遠有效	14 小時	有特殊 GPU 運算能力的電腦
奇偶攻擊	nT已經不太會重複，幾千個甚至幾萬次才一個重複	< 2 秒	需要到真讀卡機設備旁側錄通訊過程
連續認證攻擊	nT的變化規律已經無法被預測	約 25 秒	需要先知道卡片裡的其中一把 key
代數差分攻擊	對於可變變數(nT、nR)限制較為寬鬆	2~10 分鐘	僅需用假讀卡機取得通訊資料

計算資源龐大$$

風險太高(刑責)

代價太高(前兩者選一達成)

一般電腦在家即可完成

探討Mifare Classic 既有的安全議題
- 代數 ※ 解方程式 ※ SAT solver

- 針對Crypto1 stream cipher attack：

  奇偶攻擊、連續認證攻擊

  優點：有效、快速

  缺點：稍微改變保護機制，攻擊就會失效

- 代數攻擊： (布林代數攻擊)

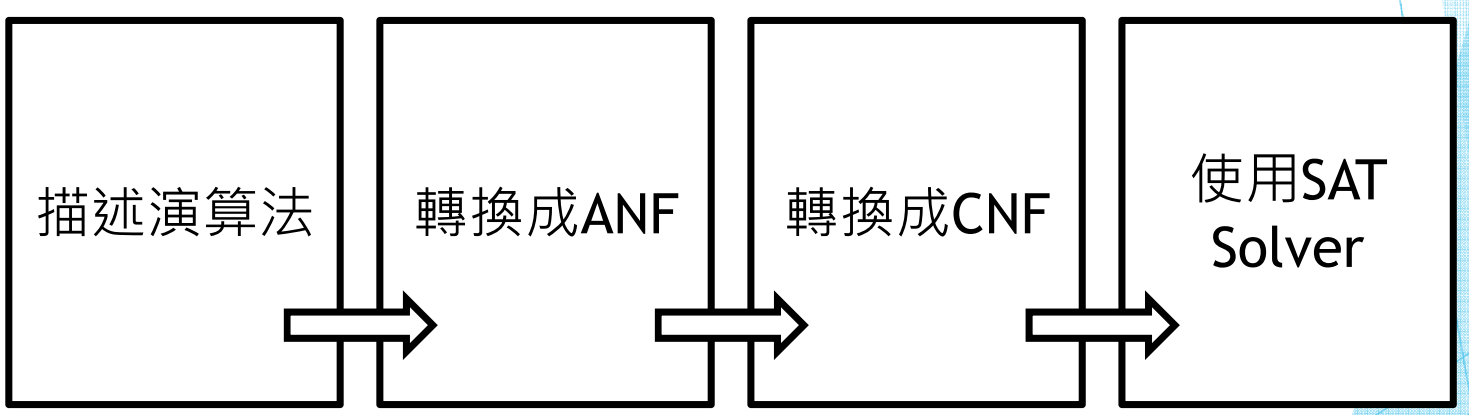
  使用代數CNF表示，再用SAT solver解方程式

  優點：適用一般性的解方程式問題(攻擊其他種類)

  缺點：比針對性攻擊要慢些

代數攻擊

- Mate Soos: Extending SAT Solvers to Cryptographic Problems
- CryptoMiniSat2



SAT問題

- 給定一組布林方程式，求解「有一組變數值，可滿足方程式為真」是否成立
- $(X_1 \vee X_2 \vee X_3) \wedge (X_2 \vee X_3 \vee X_4) \wedge (X_1 \vee \neg X_4) = \text{TRUE}$

- 轉換為CNF：假如某布林方程式為50=(10×-12) 50 = (10 and -12) 要改寫成

-50 10 0	¬50 ∨ 10 = TRUE
-50 -12 0	¬50 ∨ ¬12 = TRUE
50 -10 12 0	50 ∨ ¬10 ∨ 12 = TRUE

X_1	X_2	X_3	X_4
T	T	T	T
T	F	F	T
F	T	T	F
T	F	T	F

探討Mifare Classic 既有的安全議題
- 代數 ※ 解方程式 ※ SAT solver

- Example :

  General (Boolean) algebra = A,B,C,X,Y,Z ….

  (A V B V C)^(X V Y V Z) = True

  - For SAT solver Boolean algebra = 1 , 2 , 3…..

  (1 V 2 V 3) ^ (4 V 5 V 6) = True

  ⇒ 1 2 3 0

  4 5 6 0

  (For SAT solver input format = DIMACS format , XXX.cnf)

探討Mifare Classic 既有的安全議題
-SAT solver – CryptoMiniSat2

Input file :
'DIMACS Format'

SAT solver Tool :
CryptoMiniSat2

Output :
Boolean value of algebraic
每個布林代數的值

```
x-61 63 64 67 68 69 77 83 84 87 91 102 103 104 107 108 109 173 0
x-62 64 65 68 69 70 78 84 85 88 92 103 104 105 108 109 110 174 0
x-63 65 66 69 70 71 79 85 86 89 93 104 105 106 109 110 111 175 0
x-64 66 67 70 71 72 80 86 87 90 94 105 106 107 110 111 112 176 0
c filter number (iii-33) = -1
c start run filter (countks) 1 ====>>> 35 36 38 39 41 45 47 48 50
c start into fa =>> 35 36 38 39
35 -177 0
36 -177 0
38 -177 0
-35 -36 -38 177 0
35 -178 0
38 -178 0
-35 -38 178 0
```

```
root@ubuntu:~/ccc# ls
cmsat-2.9.5
```

```
s SATISFIABLE
v -1 2 3 4 5 -6 7 -8 9 -10 -11 12 -13 -14 -15 16 -17 18 19 20 21
-22 23 -24 -25 -26 27 28 29 -30 -31 32 -33 34 -35 36 37 -38 -39 4
0 41 -42 43 44 45 -46 47 48 -49 50 -51 -52 -53 54 55 -56 -57 58 5
9 60 -61 62 63 -64 65 -66 -67 68 -69 70 -71 72 73 -74 -75 76 -77
78 -79 -80 81 -82 -83 -84 -85 86 87 88 -89 90 -91 -92 -93 94 95 -
96 97 98 99 100 101 102 -103 -104 -105 106 -107 -108 -109 -110 11
1 -112 -113 -114 -115 116 -117 118 119 120 -121 122 123 124 125 -
126 127 128 129 130 -131 -132 -133 134 135 136 -137 -138 -139 140
 -141 -142 143 -144 -145 146 -147 148 149 150 151 152 -153 -154 -
155 -156 157 158 159 160 -161 162 163 -164 165 -166 -167 168 169
170 -171 -172 -173 -174 -175 -176 -177 -178 -179 -180 -181 182 18
3 184 185 186 187 -188 189 -190 -191 192 -193 -194 -195 -196 -197
 -198 -199 -200 -201 -202 -203 -204 -205 -206 -207 -208 -209 -210
 -211 -212 -213 -214 -215 -216 -217 -218 -219 -220 221 -222 -223
224 -225 226 -227 -228 -229 -230 -231 -232 233 -234 -235 -236 -23
7 -238 -239 240 -241 -242 -243 244 245 246 -247 -248 249 -250 -25
1 252 253 -254 -255 -256 -257 258 -259 260 261 -262 263 264 -265
```

代數攻擊分析

- Generate cnf file

CNF 規則：1. 每條式子都需要為True

2. 每條式子預設使用OR運算，若使用XOR運算則式子最前使用X開頭

3. 式子結尾需加上 ‘0’ ，代表結束字元 (DIMACS format)

測試一：

Ex.

1 or 2 = true => {1,0} ,{0,1} , {1,1}

2 = true => {1}

符合以上兩個條件的解 (取交集 ∩)

Solve = {0,1} , {1,1}

= -1 2 ; 1 2

用cryptominisat2解，結果完全正確

```
root@ubuntu:~/TMP# cat test.cnf
1 2 0
2 0
root@ubuntu:~/TMP# time /home/kk/ccc/cmsat-2.9.5/build/cryptominisat --verbosity
=0 --maxsolutions=9994096 test.cnf
s SATISFIABLE
v -1 2 0
s SATISFIABLE
v 1 2 0
s UNSATISFIABLE

real    0m0.004s
user    0m0.004s
sys     0m0.000s
root@ubuntu:~/TMP#
```

代數攻擊分析

- 測試二：

  Ex.

  1 xor 2 = true => {0,1} , {1,0}

  2 = true => {1}

  符合以上兩個條件的解 (取交集 ∩)

  Solve = {0,1}

  = -1 2

  用cryptominisat2解，結果完全正確

```
root@ubuntu:~/TMP# cat test.cnf
x 1 2 0
2 0
root@ubuntu:~/TMP# time /home/kk/ccc/cmsat-2.9.5/build/cryptominisat --verbosity
=0 --maxsolutions=9994096 test.cnf
s SATISFIABLE
v -1 2 0
s UNSATISFIABLE

real    0m0.005s
user    0m0.004s
sys     0m0.000s
root@ubuntu:~/TMP#
```

探討Mifare Classic 既有的安全議題 - Differential 差分

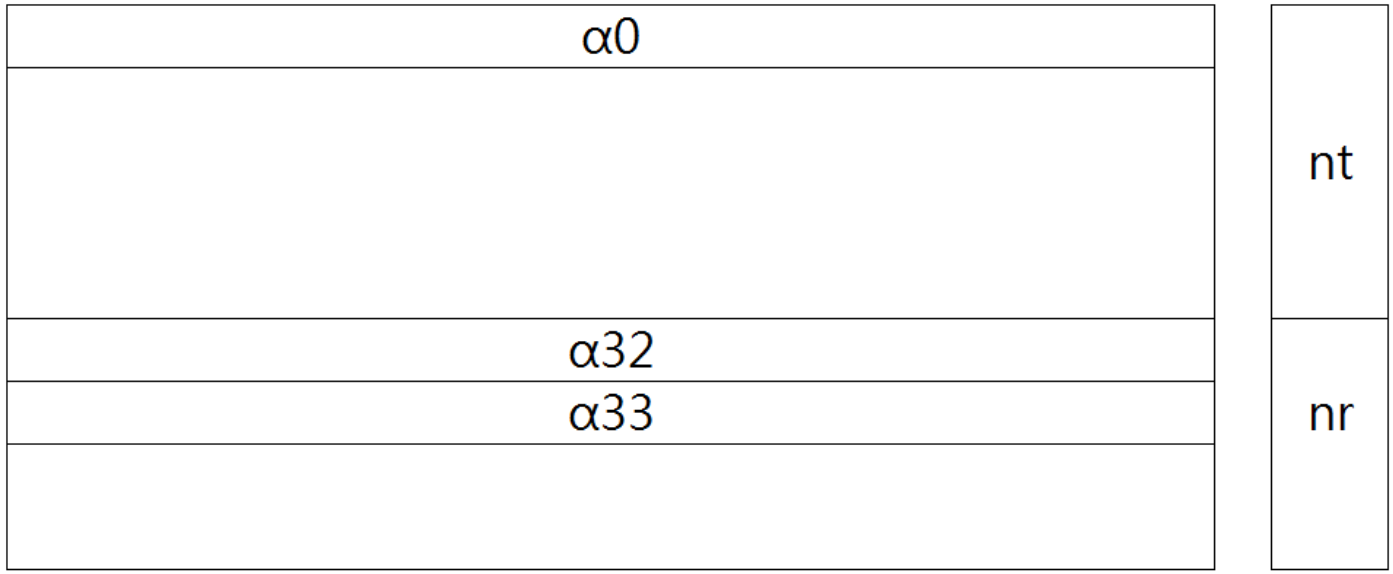
- 差分：兩個數間的差異值,
- 可用xor表達

A	B	差分
0	0	0
0	1	1
1	0	1
1	1	0

- 用途：
  1. 控制state狀態變化
  2. 控制僅有一個 bit 擁有差分值1, 其他 bit 為0

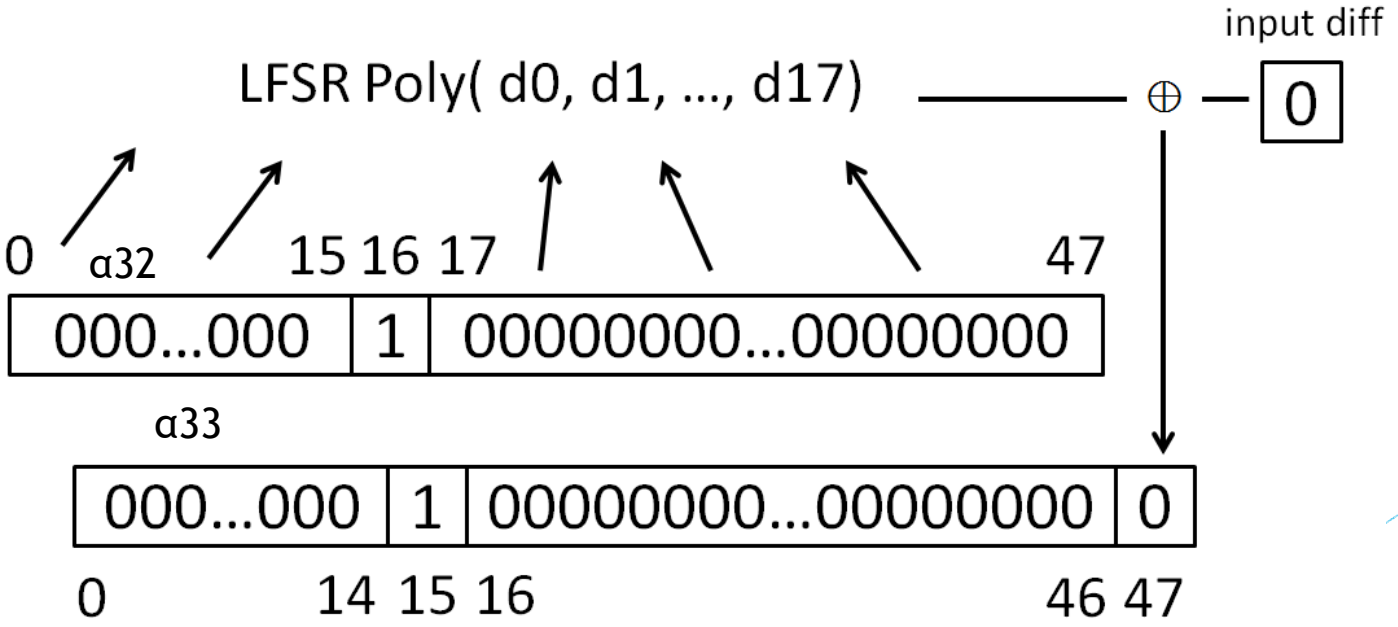
探討Mifare Classic 既有的安全議題 - Differential 差分

- 由n_T差分計算LFSR差分，n_T使得48bits有特別的狀態，再用n_r做出另一個特別的狀態
- We assume 2 keystreams are identical then diff of $\{n_r\}$ = diff of n_r



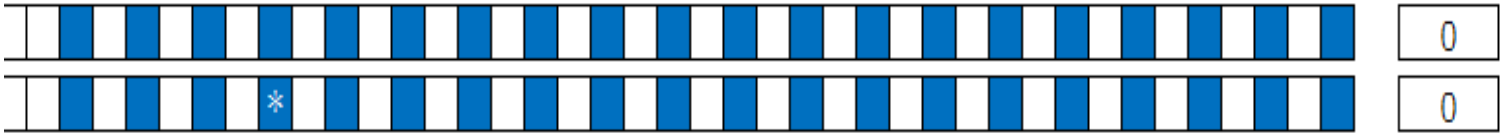
探討Mifare Classic 既有的安全議題 - Differential 差分

- 由LFSR差分計算n_R差分的必要條件



探討Mifare Classic 既有的安全議題 - Differential 差分

- 設ks差分為已知，可由n_R差分求得$\{n_R\}$差分



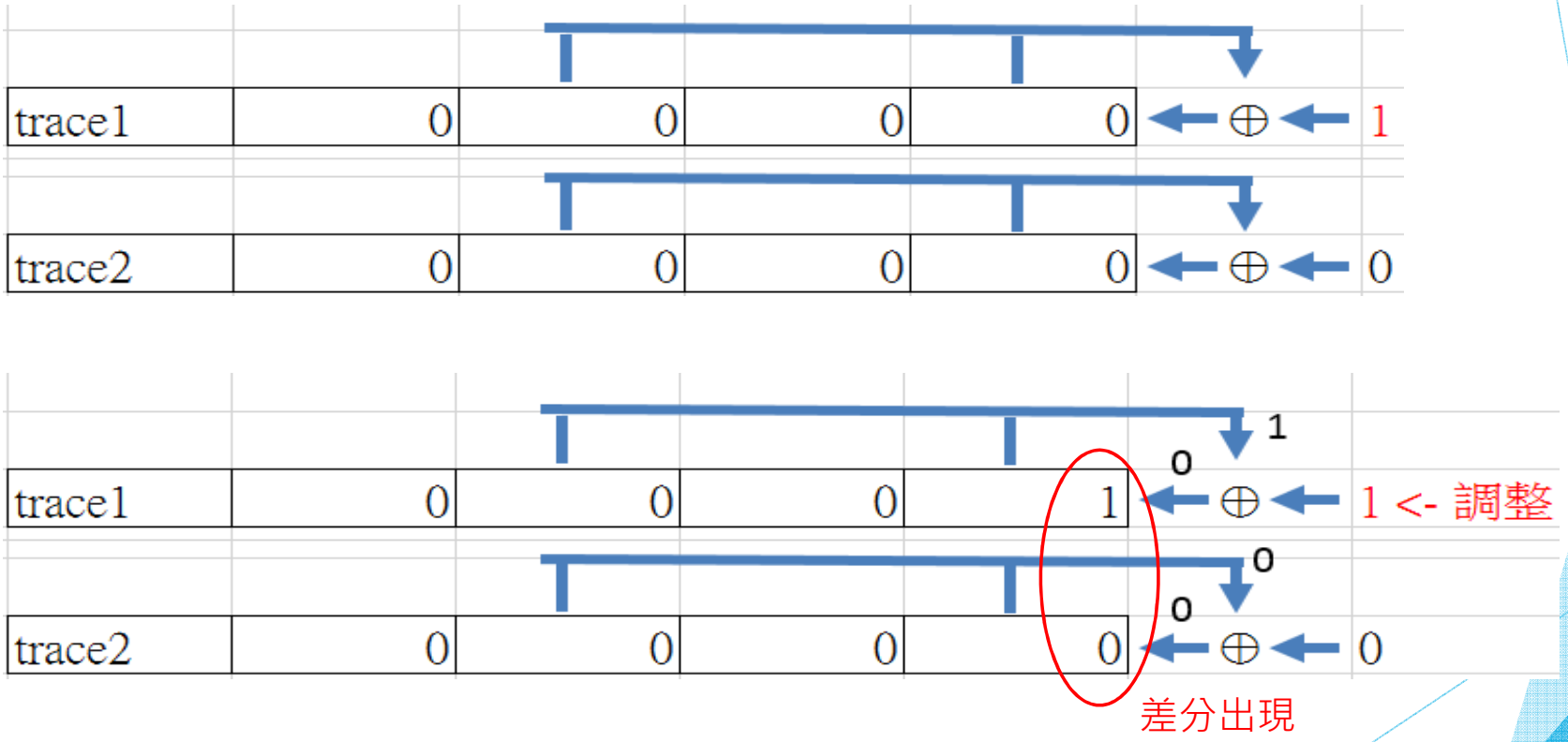
$$fc(\ state\ ) = fc(\ state + differential\ )$$

$$fc(\ state\ ) - fc(\ state + differential\ ) = 0$$

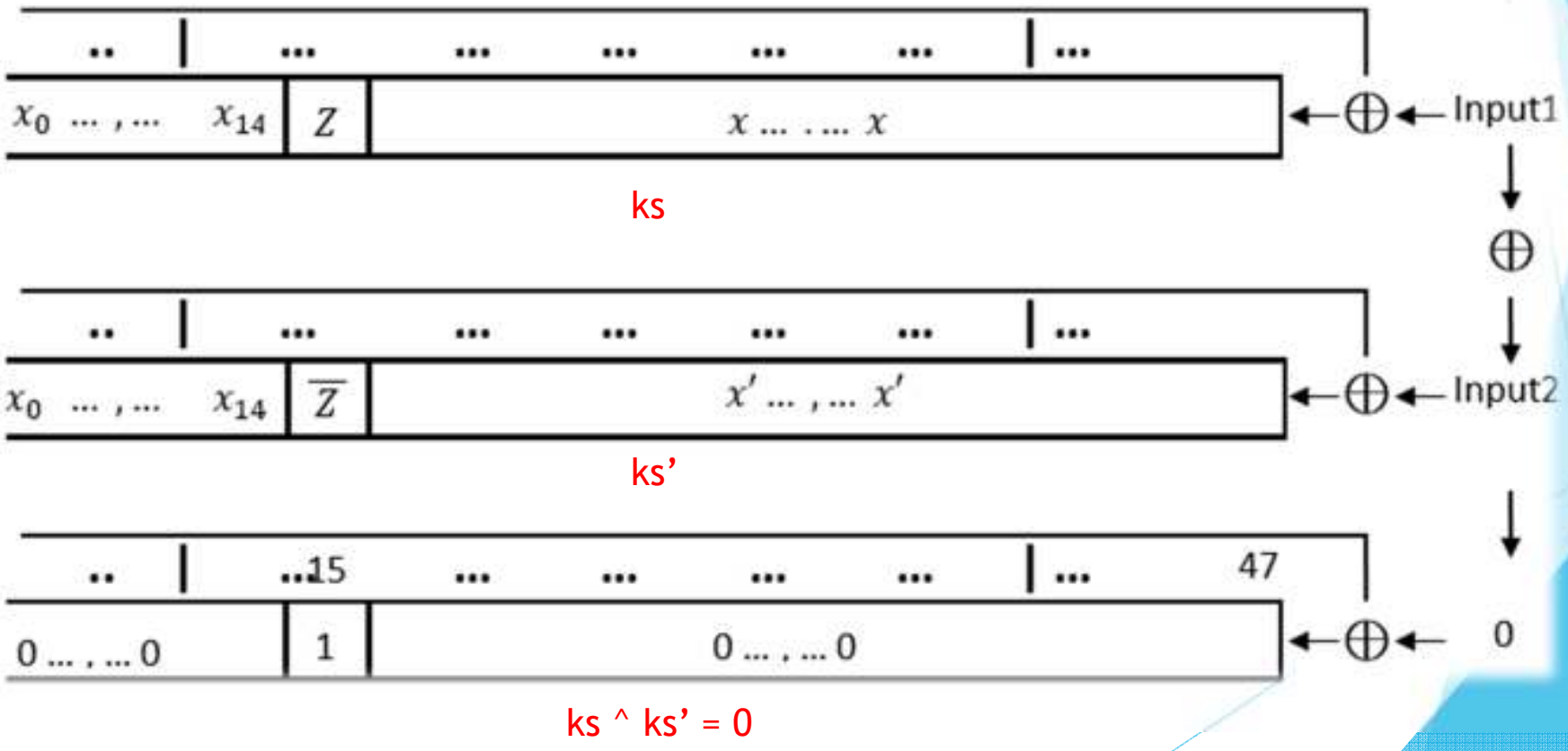
$$\frac{dfc}{dx15} = \frac{dfc}{dfa} \cdot \frac{dfa}{dx15}$$

Calculus : Chain rule
Benefit: one variable is gone

探討Mifare Classic 既有的安全議題
- Differential – special trace



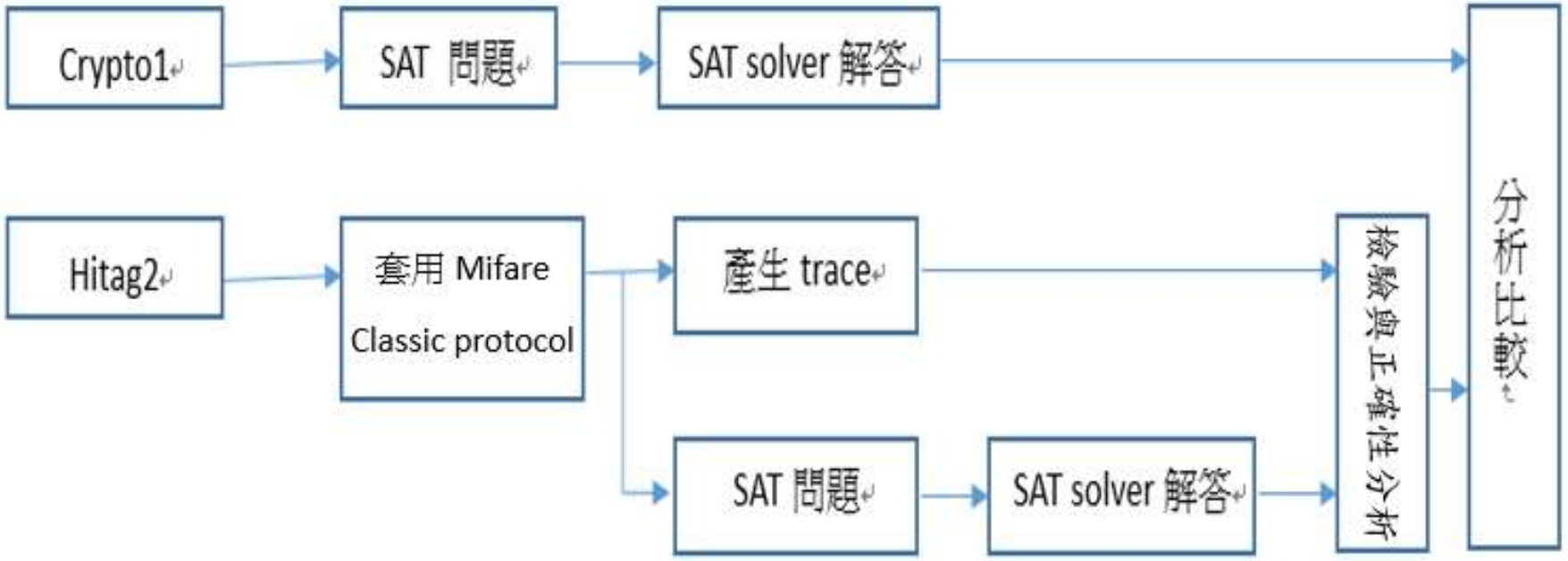
探討Mifare Classic 既有的安全議題
- Differential - special trace



探討Mifare Classic 既有的安全議題

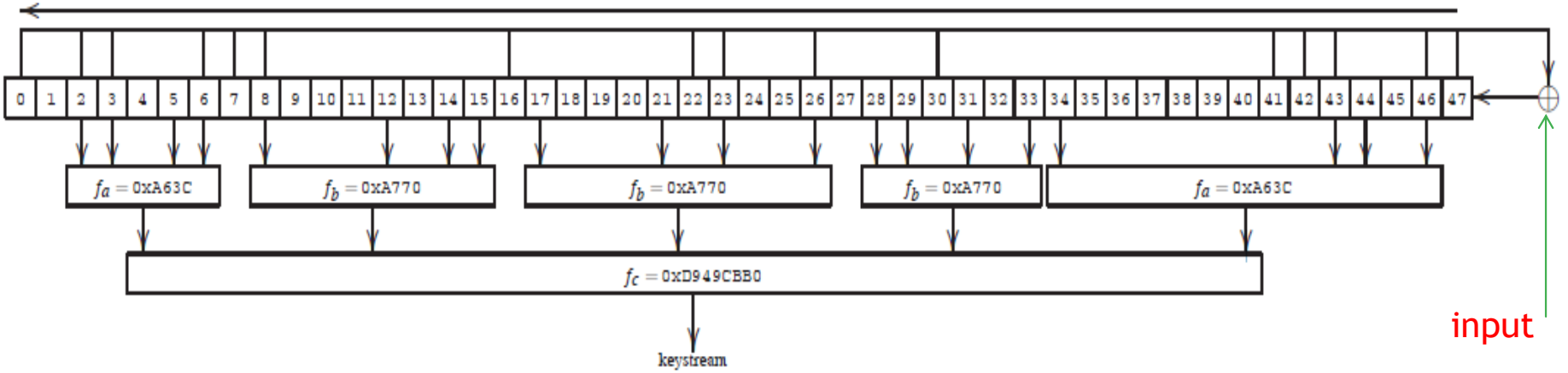
- 代數差分攻擊的影響？
  - 針對Crypto1(Mifare Classic protocol)
- 進一步的想法：
  - 相似的Stream cipher ： Hitag2
  - 套用Mifare Classic protocol
  - 使用代數差分進行攻擊與分析結果

探討Mifare Classic 既有的安全議題



Mifare Classic vs Hitag2

- Generation Mifare Classic – Hitag2 traces
  - Select Hitag2 cipher and its filter function
  - Secret initial Key => input (uid^nT、nR)
  - Output keystream : ks1,ks2,ks3

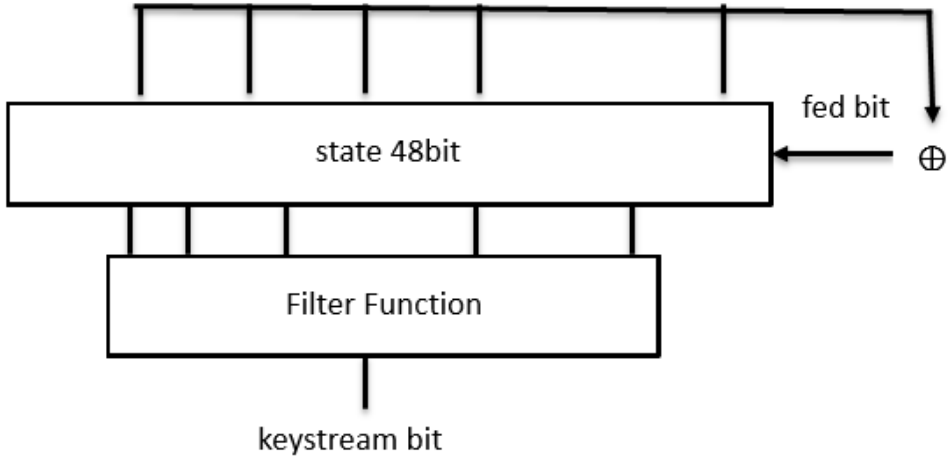


新的安全議題
- 實驗與分析方式

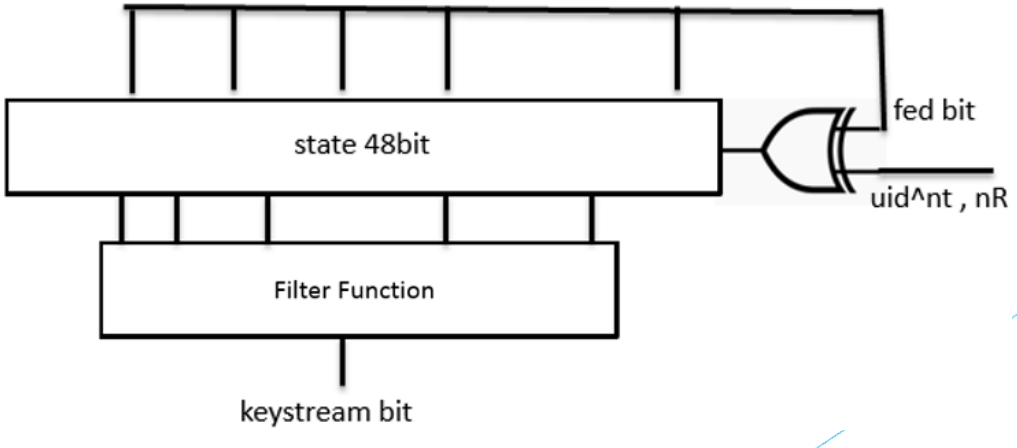
- 1.純代數攻擊方式
  - 64bits continuous keystream
  - 64bits continuous keystream & α33 HELP bits
- 2.代數攻擊+差分攻擊
  - 代數 & 差分方程式
  - 代數 & 差分方程式 & α33 HELP bits

新的安全議題
- 1. 純代數攻擊分析

- Grain of salt
  - ks0, ks1 only



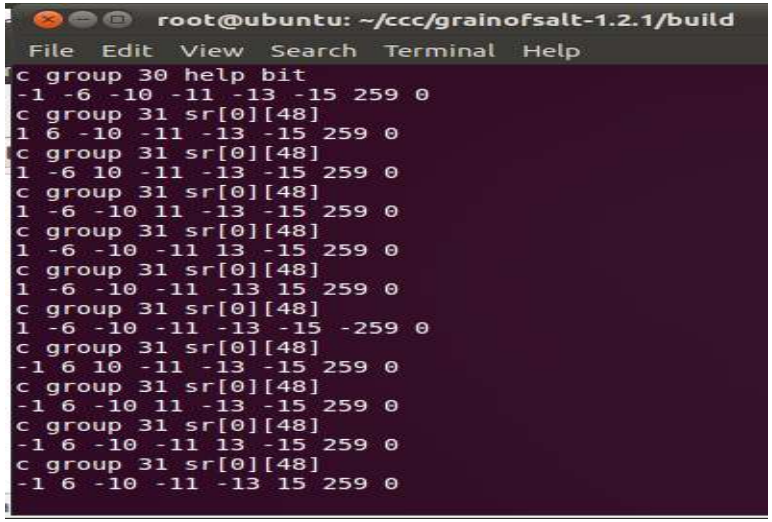
- Mifare Classic protocol
  - ks2,ks3 in practice



新的安全議題

-1. 純代數攻擊分析 - Grain of salt 工具

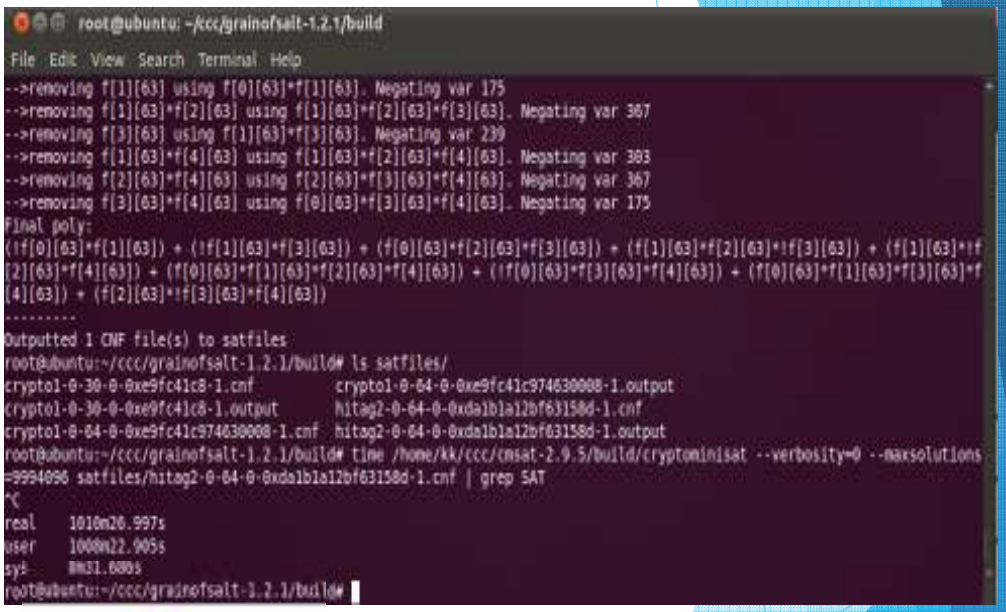
- Grain of salt本身可以產生crypto1與hitag2兩種的cnf file(下圖)
- 使用Grain of salt 產生出來的cnf 檔案內容(右圖)




```
./grainofsalt --num 1 --crypto crypto1 --outputs 64
./grainofsalt --num 1 --crypto hitag2 --outputs 64
```

```
crypto1-0-30-0-0xe9fc41c8-1.cnf          crypto1-0-64-0-0xe9fc41c974630008-1.output
crypto1-0-30-0-0xe9fc41c8-1.output       hitag2-0-64-0-0xda1b1a12bf63158d-1.cnf
crypto1-0-64-0-0xe9fc41c974630008-1.cnf  hitag2-0-64-0-0xda1b1a12bf63158d-1.output
```

純代數攻擊分析 - Grain of salt



Grain of salt	Crypto1	Hitag2
Solver time	2m43.315s	NA

純代數攻擊分析

- Mifare Classic protocol – We generate cnf files

- cnf file rule ：
  * Secret Key algebra = 1 ~ 48
  1. LFSR
  2. Filter Function
  3. HELP bits : uid^nt
  4. HELP bits : {nR}
  5. HELP bits : ks2 , ks3
     (64bits)

```
x-61 63 64 67 68 69 77 83 84 87 91 102 103 104 107 108 109 173 0
x-62 64 65 68 69 70 78 84 85 88 92 103 104 105 108 109 110 174 0
x-63 65 66 69 70 71 79 85 86 89 93 104 105 106 109 110 111 175 0
x-64 66 67 70 71 72 80 86 87 90 94 105 106 107 110 111 112 176 0
c filter number (iii-33) = -1
c start run filter (countks) 1 ====>>> 35 36 38 39 41 45 47 48 50
c start into fa =>> 35 36 38 39
35 -177 0
36 -177 0
38 -177 0
-35 -36 -38 177 0
35 -178 0
38 -178 0
-35 -38 178 0
```

```
c*--    uidnT  HELP-bit 0x7CE41029-----*
x-113 0
x114 0
x115 0
```

```
c*-- ks1 HELP-bit 0x258AFFB5-----*c*-- {nr}
x-221 145 1452 0
x-266 146 1453 0
x-311 147 1454 0
```

純代數攻擊分析 - Mifare Classic protocol

- 實驗為求取較正確值，因此每一項目會做3-10次的數據結果，然後再使用平均值標準差的方式取得一個代表性數據，平均值與表準差的公式如下：

平均值

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \quad ,\forall_N \in [3,10] \,,\forall_i \in [1,10] \,,\forall_{xi} \in [times]$$

標準差

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2} \quad ,\forall_N \in [3,10] \,,\forall_i \in [1,10], \forall_{xi} \in [times]$$

純代數攻擊分析 - Mifare Classic protocol

Mifare Clissic - Crypto1 , 30 trace solve time				
159.3806	=	2m39.380s	using30traces	平均值
159.4339	=	2m39.433s	using30traces	165.5218308
159.4473	=	2m39.447s	using30traces	
159.5892	=	2m39.589s	using30traces	
168.2262	=	2m48.226s	using30traces	
169.1538	=	2m49.153s	using30traces	計算標準差σ
169.4664	=	2m49.466s	using30traces	4.985676242
169.5214	=	2m49.521s	using30traces	
170.3094	=	2m50.309s	using30traces	代表性數據
170.6902	=	2m50.690s	using30traces	168.22623

```
root@ubuntu: ~/project/41.163
File  Edit  View  Search  Terminal  Help
root@ubuntu:~# cd /home/kk/project/41.163
root@ubuntu:~/project/41.163# ll
total 248
drwxr-xr-x 2 root root   4096 2014-04-21 23:12 ./
drwxr-xr-x 7 root root   4096 2014-04-21 23:08 ../
-rwxr--r-- 1 root root    580 2014-04-21 23:12 CMD.sh*
-rwxr--r-- 1 root root 140787 2014-04-20 08:08 hitag2*
-rwxr--r-- 1 root root  95179 2014-04-20 08:08 main.c*
root@ubuntu:~/project/41.163# ./CMD.sh
start time date
continuous_have_input_HELP_uidnt32bit_{nr}32bit_keystream_ks2_ks3_64bit solve 30
......

 Job run ok ......
^C*** INTERRUPTED ***

real    37223m3.256s
user    36729m10.398s
sys     362m22.779s
continuous_have_input_HELP_uidnt32bit_{nr}32bit_keystream_ks2_ks3_64bit solve 50
```

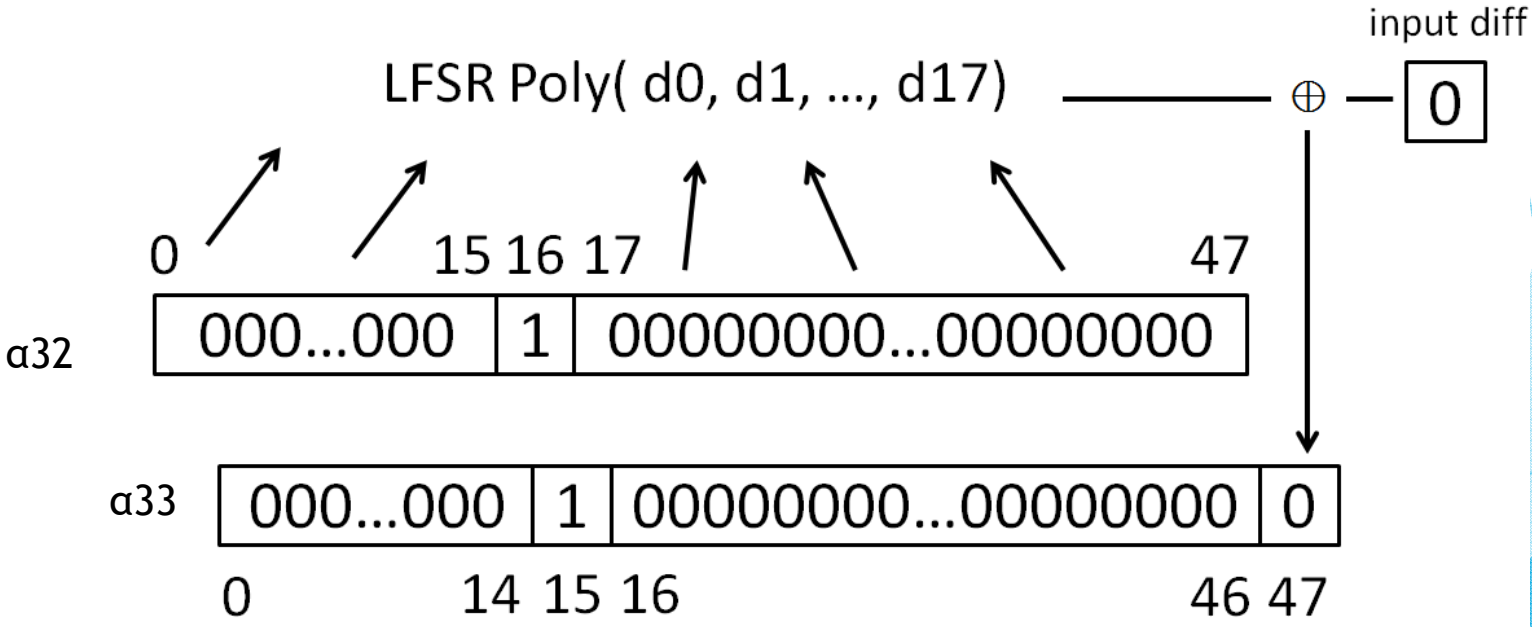
Given ks2,ks3 64bits

My cnf 1	Crypto1	Hitag2
Solver time	2m48.226s	NA

Hitag2：3萬7千多分鐘……unsat

純代數攻擊分析 - Mifare Classic protocol - Give α33 HELP bits

- α32 state vs α33 state



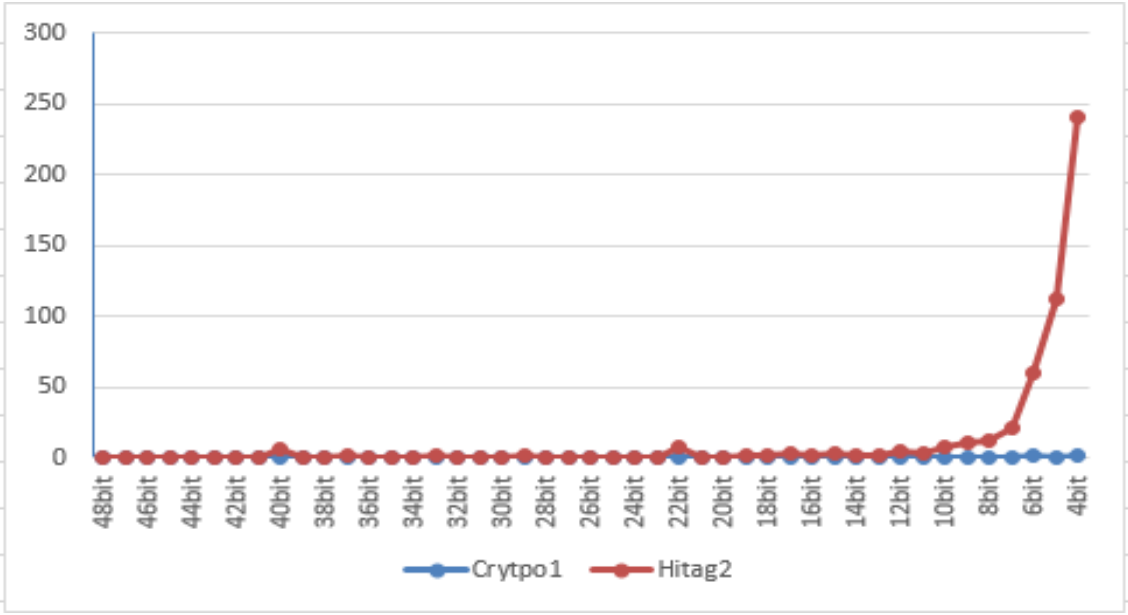
純代數攻擊分析 - Mifare Classic protocol - Give α33 HELP bits

64bit keystream and alpha33 HELP bits (1trace)			
alpha33_HELP bits ↓	Cyrpto1	Hitag2	相差倍率
48bit	0m0.015	0m0.123	4.92
47bit	0m0.015	0m0.017	0.653846
46bit	0m0.021	0m0.059	1.685714
45bit	0m0.042	0m0.022	0.309859
44bit	0m0.061	0m3.203	31.40196
43bit	0m0.069	0m6.206	53.5
42bit	0m12.341	0m59.590	2.897078
41bit	0m0.136	0m0.366	1.619469
40bit	0m0.155	6m6.017	254.8919
39bit	0m0.18	0m1.649	5.478405
38bit	0m8.461	1m1.566	8.201674
37bit	0m5.715	1m8.768	1.970187
36bit	0m0.444	0m19.656	2.652632
35bit	0m0.651	0m1.345	1.23849
34bit	0m0.894	0m13.659	0.916711
33bit	0m1.314	1m8.604	8.494977
32bit	0m1.88	0m2.481	0.79164
31bit	0m2.713	0m1.339	0.296042
30bit	0m6.586	0m13.557	1.235149
29bit	0m5.299	1m8.696	2.116848
28bit	0m6.098	0m2.488	2.447855
27bit	0m8.228	0m19.036	1.388071
26bit	0m46.342	0m54.798	0.709479
25bit	0m32.192	0m4.234	0.78913
24bit	0m9.817	0m19.881	1.214997
23bit	0m11.581	0m20.485	1.061289
22bit	0m14.472	7m41.511	30.7413
21bit	0m34.12	0m39.790	0.699703
20bit	0m10.547	0m48.211	2.742534
19bit →	0m15.421	2m59.954	10.11455
18bit	0m18.797	2m8.816	9.198161
17bit	0m19.464	3m14.303	9.688748
16bit	0m17.435	2m35.995	8.121516
15bit	0m40.218	4m33.893	6.47302
14bit	0m21.203	2m41.361	6.829876
13bit	0m44.793	2m59.337	3.473806
12bit	0m30.419	5m16.190	10.18166
11bit	0m29.083	4m21.904	8.704256
10bit	0m27.333	8m25.080	18.11173
9bit	0m41.032	10m46.935	15.3092
8bit	0m32.712	13m24.324	24.29061
7bit →	1m9.141	22m26.123	19.31812
6bit	1m50.741	60m22.338	32.6292
5bit	1m11.3	112m17398	94.39553
4bit	2m39.106	240m15.779	90.56509
3bit	2m1.862	300m<	NA
2bit	2m9.226	300m<	NA
1bit	3m8.885	300m<	NA
0bit	2m6.861	300m<	NA

- 承上條件與實驗，再給一些α33 的狀態值當成 HELP bits，看看是否有幫助或可加速 for Hitag2

純代數攻擊分析 - Mifare Classic protocol - Give α33 HELP bits

轉化成圖表比較



純代數攻擊分析 - Mifare Classic protocol - Give α33 HELP bits

64bit keystream and alpha33 HELP bits (1trace)			
alpha33_HELP bits ↓	Crypto1	Hitag2	相差倍率
48bit	0m0.015	0m0.123	4.92
47bit	0m0.015	0m0.017	0.653846
46bit	0m0.021	0m0.059	1.685714
45bit	0m0.042	0m0.022	0.309859
44bit	0m0.061	0m3.203	31.40196
43bit	0m0.069	0m6.206	53.5
42bit	0m12.341	0m59.590	2.897078
41bit	0m0.136	0m0.366	1.619469
40bit	0m0.155	6m6.017	254.8919
39bit	0m0.18	0m1.649	5.478405
38bit	0m8.461	1m1.566	8.201674
37bit	0m5.715	1m8.768	1.970187
36bit	0m0.444	0m19.656	2.652632
35bit	0m0.651	0m1.345	1.23849
34bit	0m0.894	0m13.[illegible]	
33bit	0m1.314	1m8.6[illegible]	
32bit	0m1.88	0m2.4[illegible]	
31bit	0m2.713	0m1.3[illegible]	
30bit	0m6.586	0m13.557	1.235149
29bit	0m5.299	1m8.696	2.116848
28bit	0m6.098	0m2.488	2.447855
27bit	0m8.228	0m19.036	1.388071
26bit	0m46.342	0m54.798	0.709479
25bit	0m32.192	0m4.234	0.78913
24bit	0m9.817	0m19.881	1.214997
23bit	0m11.581	0m20.485	1.061289
22bit	0m14.472	7m41.511	30.7413
21bit	0m34.12	0m39.790	0.699703
20bit	0m10.547	0m48.211	2.742534
19bit	0m15.421	2m59.954	10.11455
18bit	0m18.797	2m8.816	9.198161
17bit	0m19.464	3m14.303	9.688748
16bit	0m17.435	2m35.995	8.121516
15bit	0m40.218	4m33.893	6.47302
14bit	0m21.203	2m41.361	6.829876
13bit	0m44.793	2m59.337	3.473806
12bit	0m30.419	5m16.190	10.18166
7bit	1m9.141	22m26.123	19.31812
6bit	1m50.741	60m22.338	32.6292
5bit	1m11.3	112m17398	94.39553
4bit	2m39.106	240m15.779	90.56509
3bit	2m1.862	300m<	NA
2bit	2m9.226	300m<	NA
1bit	3m8.885	300m<	NA
0bit	2m6.861	300m<	NA

64bit keystream and alpha33 HELP bits (15trace)		
↓	Crypto1	Hitag2
平均值	0.60023625	10.88980792
計算標準差σ	0.751345504	37.88828869
代表性數據	0m34.12	10m46.935

My cnf 2	Crypto1	Hitag2
Solver time	0m34.12s	10m46.935s

純代數攻擊分析 - 比較分析

	Crypto1	Hitag2
Grain of salt	2m43.315s	NA
My cnf 1(30 traces)	2m48.226s	NA
My cnf 2(1 trace) Give α33 HELP bits	0m34.12s	10m46.935s

雖然Hitag2有比較好的防禦能力，但是也有機會在容許的時間範圍內被解出來

代數差分攻擊
for new YY card

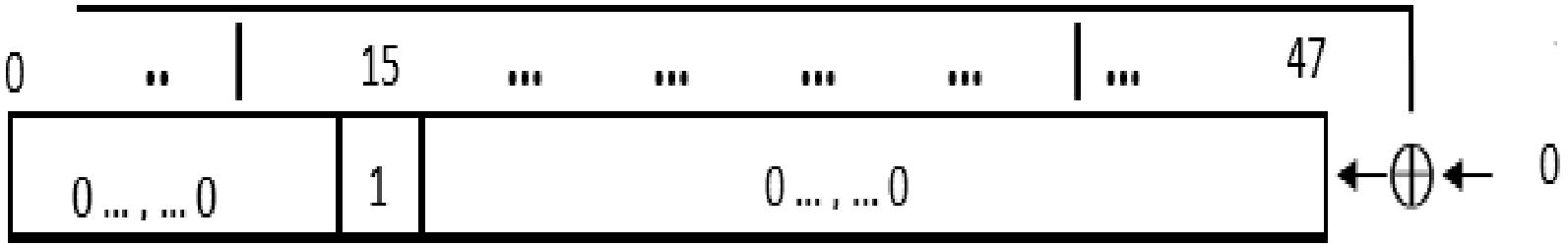
- 1. 首先必須取得特殊的traces
  - Nt are different, so does {Nr}
- 2. 定義差分方程式
- 3. 定義產生CNF的條件

代數差分攻擊 -取得特殊的trace

- 同一張卡uid固定，nT能夠與fed bit作用被放入state，因此nT可以直接影響state的狀態，nR明文也有此功能，但是nR明文無法直接控制，因此讓差分在nT的第一個bit就產生，當nT使用完時兩條trace差分為0x000100000000，之後nR無法精準控制，而state保持fed bit持續為0，因此必須從keystream來挑選，若ks = ks' 則可認定state保持僅一個bit有差分.
- 特殊nT差分值： 0x87441585 Crypto1； 0xe7002260 hitag2

```
nt2=nt1^0x87441585;
nr2=nr1^0x6B108000;
input2 = uid ^ nt2;
input2 = (input2 << 32) | nr2;
```

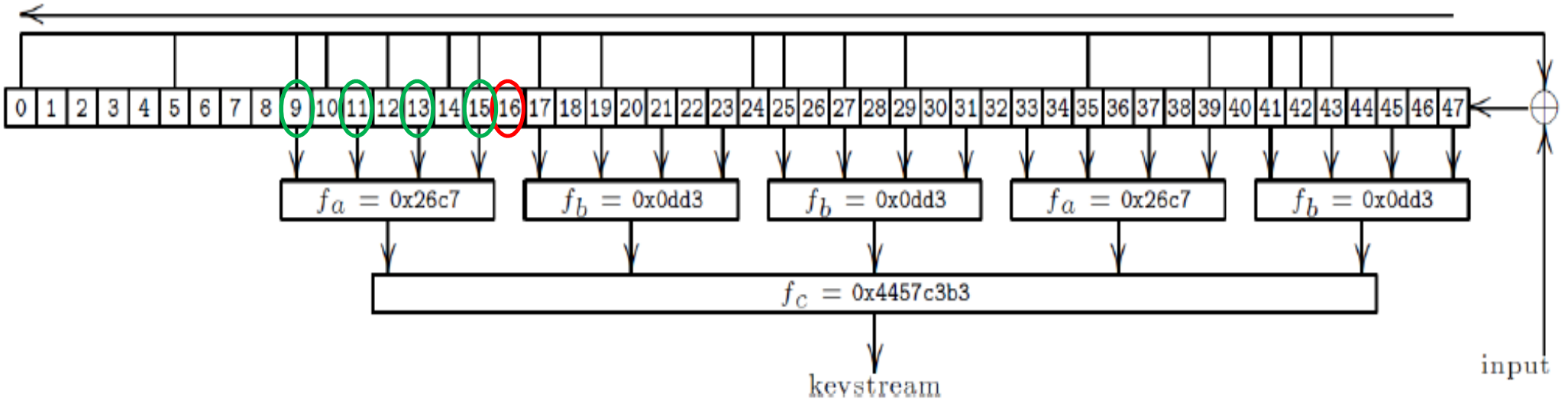
```
nt2=nt1^0xe7002260;
nr2=nr1^0x80e68000;
input2 = uid ^ nt2;
input2 = (input2 << 32) | nr2;
```



代數差分攻擊 - 差分方程式

- 差分方程式的產生
  - 為了保持state 僅1個bit有差分

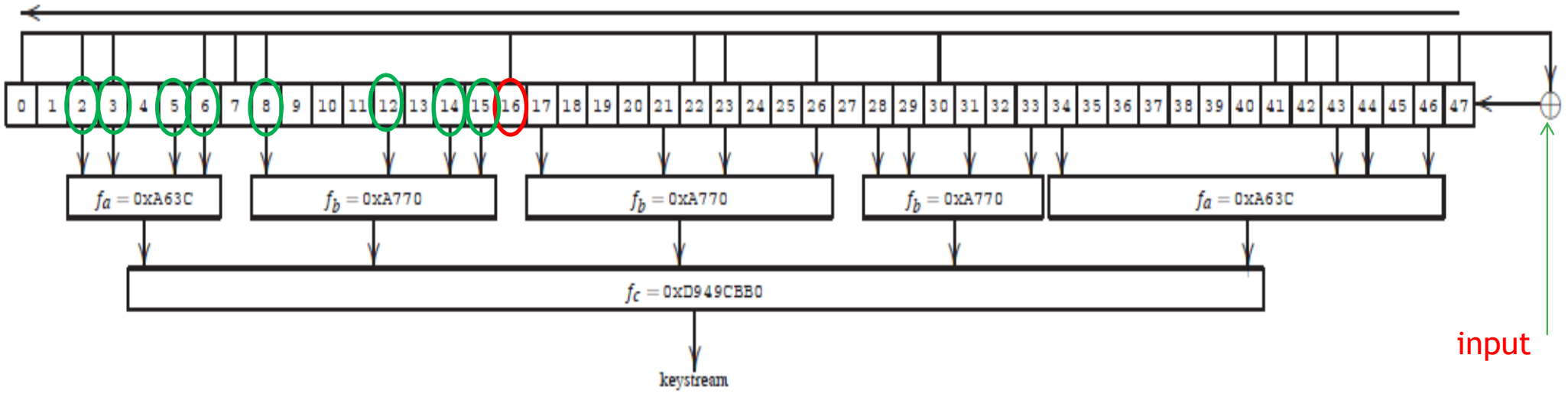
Crypto1



代數差分攻擊 - 差分方程式

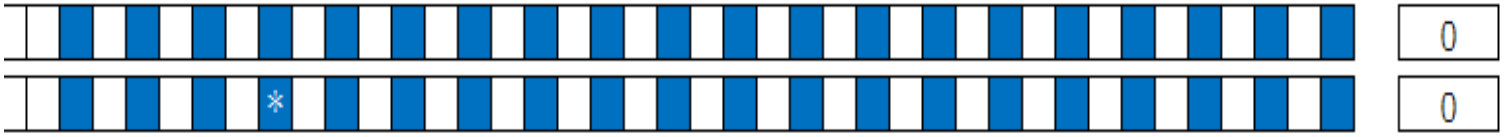
- 差分方程式的產生
  - 為了保持state 僅1個bit有差分

Hitag2



代數差分攻擊 - 差分方程式

- 輔助資訊
  - 設ks差分為已知，可由n_R差分求得{n_R}差分



$$fc(\ state\ ) = fc(\ state + differential\ )$$

$$fc(\ state\ ) - fc(\ state + differential\ ) = 0$$

$$\frac{dfc}{dx15} = \frac{dfc}{dfa} \cdot \frac{dfa}{dx15}$$

Calculus : Chain rule
Benefit: one variable is gone

代數差分攻擊 - 差分方程式

Crypto1 差分方程式：

$$Diff\_1 = fc(fa(9,11,13,df(15)),fb,fb,fa,fb)$$
$$Diff\_2 = fc(fa(9,11,df(13),15),fb,fb,fa,fb)$$
$$Diff\_3 = fc(fa(9,df(11),13,15),fb,fb,fa,fb)$$
$$Diff\_4 = fc(fa(df(9),11,13,15),fb,fb,fa,fb)$$

4 diff rules

Hitag2 差分方程式：

$$Diff\_1 = fc(fa(2,3,5,df(6)),fb,fb,fa,fb)$$
$$Diff\_2 = fc(fa(2,3,df(5),6),fb,fb,fa,fb)$$
$$Diff\_3 = fc(fa(2,df(3),5,6),fb,fb,fa,fb)$$
$$Diff\_4 = fc(fa(df(2),3,5,6),fb,fb,fa,fb)$$
$$Diff\_5 = fc(fa(8,12,14,df(15)),fb,fb,fa,fb)$$
$$Diff\_6 = fc(fa(8,12,df(14),15),fb,fb,fa,fb)$$
$$Diff\_7 = fc(fa(8,df(12),14,15),fb,fb,fa,fb)$$
$$Diff\_8 = fc(fa(df(8),12,14,15),fb,fb,fa,fb)$$

8 diff rules

代數差分攻擊 -產生cnf的條件

- CNF file rules ：

  * Secret Key algebra = 1 ~ 48

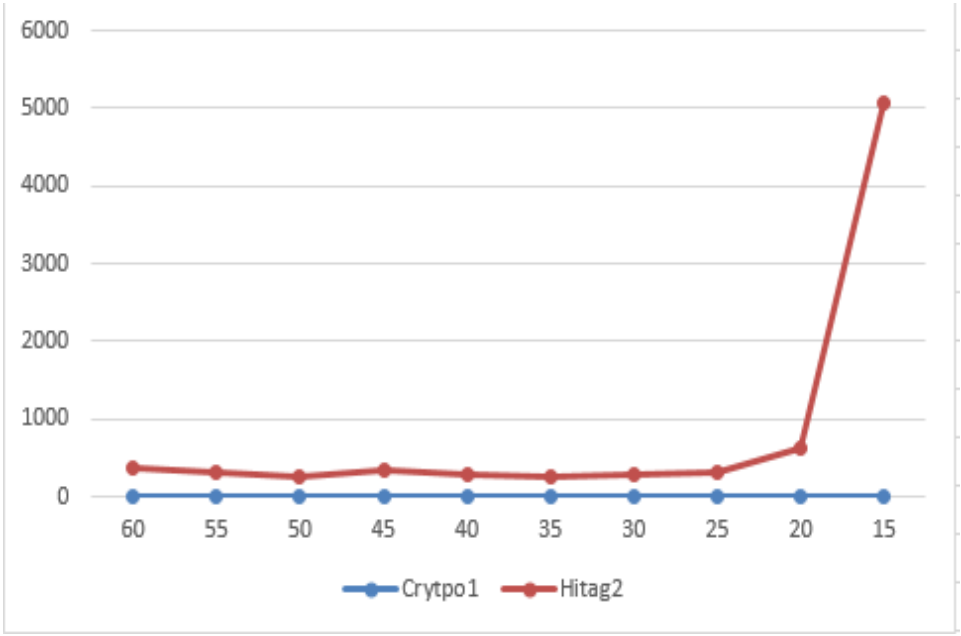
  1. LFSR
  2. Filter Function
  3. HELP bits : uid^nt
  4. HELP bits : {nR}
  5. 差分方程式 ： crypto1*4條、Hitag2*8條

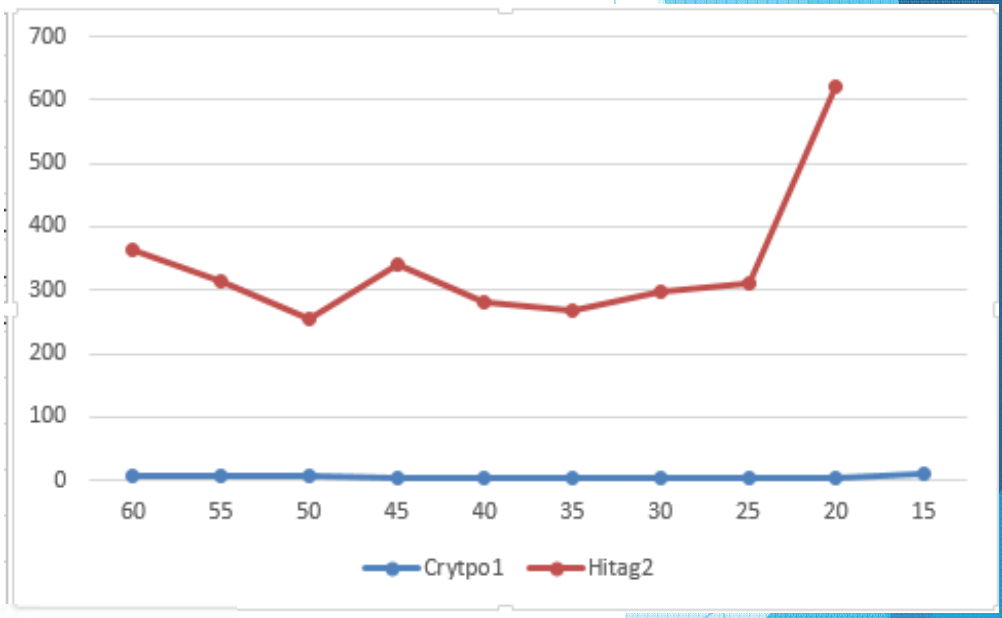
代數差分攻擊 - 實驗結果分析

- 使用不同traces數量做比較分析
  - 約25~35traces左右表現最佳

Differential lab with 5 rule			
trace ↓	Crytpo1	Hitag2	相差倍率
60	6m9.008s	364m9.450s	52.88444818
55	6m6.463s	312m7.960s	47.06317801
50	5m3.827s	255m1.909s	47.40945994
45	4m9.368s	341m42.887s	69.15995584
40	4m5.236s	280m15.088s	61.93095764
35	3m8.508s	268m32.282s	69.67976005
30	3m6.101s	297m13.072s	82.30539874
25	4m4.129s	312m24.711s	70.75780326
20	3m8.999s	622m53.323s	159.6279982
15	8m7.7174s	5066m4.634s	577.5893266

代數差分攻擊 - 實驗結果分析





差距最大拿掉

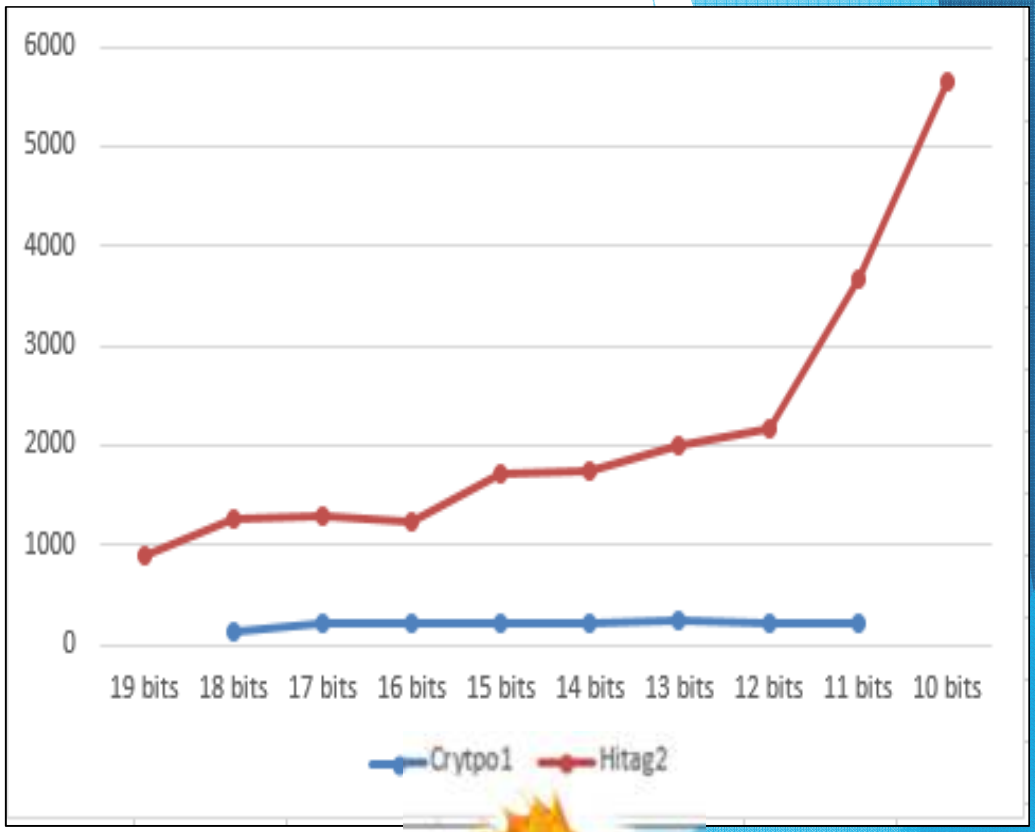
My cnf 2	Crypto1	Hitag2
Solver time	< 10 min	> 300 min

代數差分攻擊 - HELP α33 bits

- 承上條件下，同樣的再給α33的HELP bits，看看結果如何
- 經多次實驗觀察20與30trace於crypto1和Hitag2會比較有效，因此本實驗使用了20traces和30traces的兩種case
- 將給予的α33 HELP bit 逐漸減少(from 48 to 1)

代數差分攻擊 - HELP α33 bits

Differential lab with 5 rule _ alpha33 HELP _20 trace			
alpha33 HELP bits ↓	Crytpo1	Hitag2	相差倍率
19 bits	NA	904.785	NA
18 bits	142.6424	1257.201	8.813655
17 bits	221.3709	1286.426	5.811181
16 bits	209.7559	1221.591	5.823869
15 bits	214.7431	1728.835	8.050712
14 bits	220.2315	1748.915	7.941258
13 bits	229.6657	2009.724	8.750648
12 bits	211.6633	2177.903	10.28947
11 bits	214.4096	3677.345	17.15102
10 bits	NA	5649.662	NA

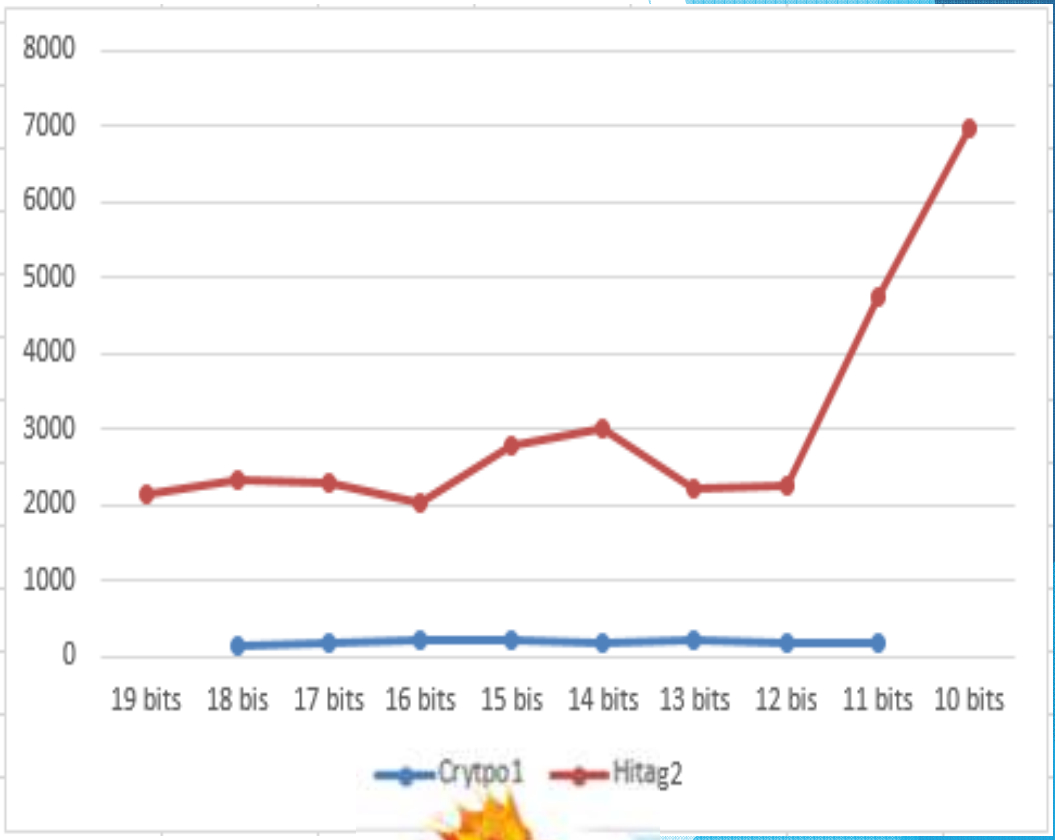


單位：秒

My cnf 2	Crypto1	Hitag2
Solver time	大約 < 220 sec ≒ 3.6min	大約 > 1000 sec≒16.6min

代數差分攻擊 - HELP α33 bits

Differential lab with 5 rule _ alpha33 HELP _30 trace			
alpha33 HELP bits ↓	Crytpo1	Hitag2	相差倍率
19 bits	NA	2153.204	NA
18 bis	130.6564	2342.147	17.926
17 bits	162.8662	2290.322	14.0626
16 bits	196.7488	2033.13	10.33363
15 bis	196.5496	2769.023	14.08817
14 bits	190.8321	2995.134	15.69512
13 bits	203.8462	2193.108	10.75864
12 bis	160.0047	2253.522	14.0841
11 bits	174.7519	4758.851	27.23205
10 bits	NA	6985.271	NA



單位：秒

My cnf 2	Crypto1	Hitag2
Solver time	大約 < 200 sec ≒ 3.3min	大約 > 1000 sec≒16.6min

勝

代數差分攻擊- 比較分析

	Crypto1	Hitag2
no α33 HELP _ 30trace	< 10 min	> 300 min
HELP α33 _ 20trace	大約 < 220 sec ≒ 3.6min	大約 > 1000 sec≒16.6min
HELP α33 _ 30trace	大約 < 200 sec ≒ 3.3min	大約 > 1000 sec≒16.6min

雖然Hitag2有比較好的防禦能力，但是也有機會在容許的時間範圍內被解出來

結論

- 1. 代數差分攻擊抵抗能力：
  - Hitag2 >> Crypto1
- 2. 兩者間 Filter Function input的Hamming distance為13，但實驗解的速度並沒有增加2^{13}
- 3. 代數差分攻擊的確會對Stream Cipher與Mifare Classic protocol造成影響
- 4. 建議停止使用Mifare Classic，並改用其他架構設計，以免造成更大資的安損失。

結論

- Hitag2是一個比Crypto1強的cipher，套用上了Mifare classic protocol，以我們實驗結果其安全性並不很高
- 以代數差分攻擊角度來看，對於兩種不一樣的stream cipher 都可造成有效的攻擊，本攻擊證明對於protocol方面也是有影響的效力
- 在未來的研究可以繼續討論是否代數差分攻擊可以針對於所有stream cipher加上任何的protocol有顯著的效果

Extra: Non-NIST Ciphers

- Lavabit, TrueCrypt, SilentMail etc
  - Close any good (tough) services or companies as you can ?
- The link between NIST and NSA ? RSA and NSA ?
  - ECC ?
  - AES ?
  - SHA-1 ?
  - OpenSSL Heartbleed ?

Extra: Toward a New Good/Bad World !?

- Prepare for the Good New World ?
  - Government cannot censor anymore ?
- Prepare for the Cryptopocalypse ?
  - Discrete Logarithm Problem(DLP) is easier than we thought before ?
  - Eurocrypt2014
    - **A Heuristic Quasi-Polynomial Algorithm for Discrete Logarithm in Finite Fields of Small Characteristic**
  - ECC is ok ?

David Hilbert

We must know
We will know

謝謝聆聽

Q and A