

# Capire gli ECM/EMM

## - Seconda Edizione -

*Questo documento è un tentativo di raccolta e riorganizzazione dei vari post & thread esistenti sull'argomento SEKA 2, senza la pretesa di essere un trattato esaustivo. Per la comprensione dell'argomento affrontato e' opportuna la conoscenza del SEKA 1, attraverso la lettura di FAQ/Prontuari comunemente reperibili in rete. Tutto quello che trovate scritto in questo documento serve per soli scopi di ricerca e di studio. La visione dei programmi delle Pay-Tv senza la sottoscrizione di un regolare abbonamento è vietata. L'autore non può essere ritenuto responsabile di eventuali danni o abusi commessi da terze parti.*

### La novita' del SEKA 2

A partire da un ECM loggato e' possibile effettuare una prima caratterizzazione del nuovo sistema:

C1 3C 01 BE 5C

10 01 CD 99 F1 E7 88 F1 E9 00 50 F9 09 5B 02 43  
DD 03 35 39 1B C2 41 97 AF B3 8A A7 F7 9A FA 78  
12 C9 BA 23 16 42 99 0E 9A F3 31 72 22 BC B8 C6  
62 45 37 F9 7F 68 15 7C BB 7C A7 F2 3D F8 27 82  
52 49 54 56 98 0C AB 94 26 95 74 A7 12 B6 83 4D  
23 46 03 F1 E1 A5 66 31 05 96 46 48 [90 00]

Tra due ECM diversi cambiano tutti i byte, ad eccezione di quelli evidenziati; dopo di essi non è riconoscibile nessun Nanocomando e non vi e' traccia della Signature: la ragione sta nel fatto che i dati sono stati criptati.

E' ormai noto che nel SEKA 2 siano presenti due passaggi di crypt: la **SuperEncryption** (che opera su blocchi di byte) e la Secondary SuperEncryption (**SSE** oppure **ENVELOPE**), che agisce sui dati nella loro interezza. Questi processi di crypt sono obbligatori per i dati delle INS 3C/38/40.

L'algoritmo utilizzato per l'Envelope non e' noto, anche se e' stato ipotizzato un (de)crypt tipo RSA (per chi e' interessato esistono in Rete molti doc sull'argomento), la SuperEncryption, invece, dovrebbe somigliare a quella del SEKA 1.

#### **SuperEncryption (SEKA 1)**

Si prende il corpo dell'istruzione ( compresa la signature ) e si suddivide in blocchi di 8 byte; a ciascun blocco si applica l'algoritmo di criptaggio con la chiave indicata da P2. Se rimangono dei byte in numero minore di 8 che non completano un blocco restano non codificati.

NOTA:

*Esisterebbero nell'EEPROM della card delle locazioni di memoria che, opportunamente settate, renderebbero la SuperEncryption facoltativa... per adesso non e' noto un modo per agire su di esse !!!*

## La struttura delle INS 38/3C/40

E' stata mantenuta l'impostazione classica del SEKA 1, in pieno rispetto dello standard ISO 7816:

**C1 38/3C/40 P1 P2 LEN + DATA PACKET**

### **P1**

- Bit 0...3 : Indice del Provider a cui inviare il comando
- Bit 4 : Utilizzo di Key primaria oppure Key primaria + secondaria
- Bit 5,6 : Indicano come inizializzare l'hashbuffer per il calcolo Signature
- Bit 7 : Non utilizzato

### **P2**

- Bit 0...3 : Indice della key da usare per il (de)crypt SuperEncryption
- Bit 4 : *Significato sconosciuto (deve essere = 1)*
- Bit 5,6 : Selezionano le tabelle per il (de)crypt e l'eventuale user ALGO (non attivo su V7)
- Bit 7 : Indicatore della SuperEncryption (deve essere = 1)

### **LEN**

E' la lunghezza del "data packet"

Esempio:

C1 40 01 B1 5C

**10 01** 12 08 F5 56 DA F0 11 12 D0 D8 40 3B 34 7A  
EB A5 B7 30 41 50 5F 02 6D B2 03 AB 29 2B 29 7A  
05 4F AF 83 18 75 1F 33 49 67 29 0C C0 22 C7 44  
E3 BA 45 6D 1B 3A F3 56 07 A9 89 5D B4 5E 8A D1  
1F 40 F4 50 D1 57 D0 96 88 5B EB 93 2A 10 CE E8  
4D 36 1F 80 A7 65 A6 9C 3E 03 78 49

Come già accennato, i due byte evidenziati si mantengono uguali per tutte le INS loggate, essi rappresentano dei parametri per il de-envelope e vengono gestiti in maniera diversa rispetto a tutti gli altri dati. E' stata data loro la definizione di parametri **P3** e **P4**. All'interno dei dati e' presente un ulteriore parametro che non e' visibile perche' criptato. La sua funzione sara' spiegata in seguito, per adesso e' sufficiente conoscerne l'esistenza, la posizione (e' l'ultimo byte dei dati) ed il nome (e' stato chiamato **P5**). Alla luce di tutto cio' possiamo dare una nuova rappresentazione della struttura delle INS:

**C1 38/3C/40 P1 P2 LEN P3 P4 + DATA PACKET + (P5)**

## Il processo di esecuzione delle INS

A grandi linee dovrebbe avere questa sequenza logica (e temporale):

- 1 - Normali controlli sul protocollo ISO7816 CLA/INS/LEN (possibili status d'errore 6D00, 6E00, 6700)
- 2 - Controllo su P1, esistenza del Provider (status 9004 se inesistente)
- 3 - Controllo sul bit 4 di P2 (deve essere = 1, status 9024 in caso contrario)
- 4 - Controllo sul bit 0 di P4 (deve essere = 1, status 9024 in caso contrario)
- 5 - Controllo su P3

Il trattamento riservato a P3 e' simile a quello di un Nanocomando, pero' :

- Deve avere come valore minimo 10 (status 9024 per valori inferiori)
- Viene completamente ignorato, assieme ai dati ad esso associati, per i seguenti processi:
  - > Decrypt SSE (de-envelope)
  - > Decrypt SE
  - > Parsing ed esecuzione*("Parsing", tradotto letteralmente, vuol dire "Analisi del periodo"; nel nostro contesto significa "esame di un Nanocomando e dei suoi parametri")*
- Fa parte dei dati per il calcolo Signature

Per sapere quanti sono i dati associati a P3, si segue la classica tabella della LEN dei nanocomandi:

High Nibble | Nano Length

-----+-----

0 ... C		0 ... 12	(IMPORTANTE: '0' non e' un
D		16 byte	valore ammissibile per P3)
E		24 byte	
F		32 byte	

Osservazione: P4 e' il primo byte di dati per P3.

Al termine di questo controllo si puo' ottenere lo status 6700 (LEN errata) perche', dopo i dati di P3, devono esserci almeno 5A byte (90, in decimale), il motivo e' spiegato piu' avanti.

- 6 - Controllo sui bit 1,2 di P4 (devono essere = 0, status 9036 in caso contrario)
- P4 e' un indicatore per il de-envelope, ma la precisa funzione e' sconosciuta.*

## 7 - Decrypt SSE (o de-envelope)

Pur non essendo conosciuto l'algoritmo, si e' potuto osservare che esso utilizza chiavi differenti per ciascun provider (ma uguali per tutte le card) e che opera sempre e solo sugli ultimi 5A byte di dati. Le conseguenze sono che:

- Un ECM/EMM indirizzato ad un provider non e' applicabile ad un provider diverso
- I dati di P3 (che non prendono parte al de-envelope) devono essere seguiti da almeno 5A byte (ecco spiegato il perche' dello status 6700 durante i controlli su P3)
- Eventuali byte aggiunti tra i dati di P3 e gli ultimi 5A byte non sono sotto SSE/envelope (*importante*)

Esempio:

*(NOTA: i comandi mostrati sono tutti fittizi,  
non corrispondono a dati realmente loggati)*

C1 40 01 B1 6A

```
43 51 6E B1 92 0F 87 17 D3 5C 87 47 34 5E 39 79
12 08 F5 56 DA F0 11 12 D0 D8 40 3B 34 7A EB A5
B7 30 41 50 5F 02 6D B2 03 AB 29 2B 29 7A 05 4F
AF 83 18 75 1F 33 49 67 29 0C C0 22 C7 44 E3 BA
45 6D 1B 3A F3 56 07 A9 89 5D B4 5E 8A D1 1F 40
F4 50 D1 57 D0 96 88 5B EB 93 2A 10 CE E8 4D 36
1F 80 A7 65 A6 9C 3E 03 78 49
```

In questo comando i byte evidenziati in giallo rappresentano P3 e relativi dati (high-nibble di P3 = 4, quindi 4 byte di dati), i byte evidenziati in celeste rappresentano i 5A byte sotto envelope, gli altri sono byte fuori dall'envelope (quindi sono byte in chiaro oppure sotto SuperEncryption).

*Cosa troviamo sotto il de-envelope ?*

Si nota immediatamente una grossa differenza rispetto al SEKA 1: la Signature e' esterna alla SuperEncryption e non e' in posizione fissa. Riprendendo il comando dell'esempio precedente ed ipotizziamone un possibile de-envelope:

C1 40 01 B1 6A

```
43 51 6E B1 92 0F 87 17 D3 5C 87 47 34 5E 39 79
D7 C6 1A C9 4F E8 40 6C 67 E3 AB 84 4C 29 3F DD
A3 F0 E6 37 86 6D 8A 23 CB 88 55 9D 47 78 B6 71
54 9F 82 D8 85 1C 3E 05 34 36 27 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 P5
```

← (P5 ora e' in chiaro)

Possiamo vedere P3 con i suoi dati (in giallo), a seguire ci sono i byte criptati con la SuperEncryption (in verde, da notare la parte sottolineata non ha subito de-envelope), dopo e' presente il Nano 82 + la Signature in chiaro, il resto sono byte di riempimento fino al penultimo. Il comando si conclude col parametro **P5**.

Questo parametro ha la funzione di indicare dove e' posizionato il Nano 82, questo significa che la card non ricerca la signature scandendo tutto il comando alla ricerca del Nano, ma va "a botta sicura" alla posizione puntata da P5. Si hanno due **vincoli** sulla posizione dell'82:

- La Signature e' lunga 8 byte, non e' quindi ammissibile che il nano 82 possa stare nelle ultime 8 posizioni del comando;
- I dati per il calcolo Signature devono essere almeno 8, in altre parole il Nano 82 deve essere preceduto da almeno 8 byte;

#### 8 - Controlli sul valore assunto da P5

Il conto che fa la card per determinare la posizione del Nano 82 e' il seguente (calcolo ad 8 bit):

$$\text{Posiz.82} = \text{LEN} - (\text{P5} + 8)$$

I valori di P5 compresi tra 128 e 255 non sono ammissibili (si ha lo status 9037). Se P5 vale 0 si ha lo status 9038. Se P5 assume valori nel range 1÷127, ma il risultato del calcolo non rispetta i **vincoli** del punto precedente si ha lo status 9037.

#### 9 - Controllo sul bit 7 di P2 (deve essere = 1, status 9035 in caso contrario)

E' il bit indicante la SuperEncryption.

#### 10 - Ricerca Key per il calcolo Signature

Per il calcolo Signature e' necessaria la key indicata da P2; a seconda della INS considerata non tutte le key sono utilizzabili:

INS 40 : Solo Management Key (Key Index nel range 0...B), in caso contrario si ha status 9013

INS 3C : Solo Operational Key (Key Index nel range C...F), in caso contrario si ha status 904A

INS 38 : Nessun vincolo sulla Key utilizzabile

A questo punto inizia la ricerca della Key in EEPROM, se non e' presente si ottiene lo status 901D, se non e' stata trovata la Key primaria, oppure lo status 901F, se non e' stata trovata la Key secondaria. Una volta reperita la Key, ne viene verificato il checksum, se questo controllo fallisce si ha lo status 9028.

#### ATTENZIONE !!!

Lo status 9028 non compare solo a causa del fallimento del controllo-checksum !!! Esso puo' comparire anche durante i controlli che precedono il de-envelope, ma il motivo per cui cio' accade non e' stato ancora capito. Per distinguere i due casi e' necessario analizzare il tempo di risposta.

9028 pre-SSE : circa 73000 cicli di clock

9028 key-checksum : oltre 190000 cicli di clock

## 11 - Verifica presenza Nano 82 e calcolo signature

La card prende il valore calcolato al punto 8 e va a verificare se il Nano 82 e' effettivamente presente in tale posizione. Se non viene trovato si ha lo status 9002 (chiamato anche 9002\_A oppure 9002 di tipo 1). Se il Nano 82 e' presente, allora si procede col calcolo Signature. I byte da cui dipende il valore della Signature comprendono tutti e soli i dati che precedono il Nano 82 fino ad arrivare a P3 incluso.

C1 40 01 B1 6A

```
43 51 6E B1 92 0F 87 17 D3 5C 87 47 34 5E 39 79
D7 C6 1A C9 4F E8 40 6C 67 E3 AB 84 4C 29 3F DD
A3 F0 E6 37 86 6D 8A 23 CB 88 55 9D 47 78 B6 71
54 9F 82 D8 85 1C 3E 05 34 36 27 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 P5
```

Nell'esempio, tutti i byte evidenziati in celeste prendono parte al calcolo signature.

La key per il calcolo signature, come gia' detto, e' indicata dal nibble basso di P2, mentre i bit 5,6 indicano le tabelle hash da usare.

*Le "Tabelle hash" sono delle tabelle di conversione dati utilizzate dall'algoritmo di (de)Crypt, cambiando le tabelle, cambia il risultato del (de)Crypt.*

Bit 6	Bit 5	
0	0	Tabelle ROM (uguali al SEKA1)
0	1	Tabelle EEPROM (A)
1	0	Algoritmo "alternativo"
1	1	Tabelle EEPROM (B)

Se (bit 6, bit 5) = (1,0) si richiede l'esecuzione del (de)crypt tramite un algoritmo alternativo, memorizzabile nell'EEPROM della card. Nelle V7 tale algoritmo non e' presente: un'eventuale INS inviata con (bit 6, bit 5) = (1,0) otterrebbe lo status 9034.

*L'analisi dei bit 6,5 avviene immediatamente prima della verifica della presenza del Nano 82.*

E' giunto il momento del calcolo Signature: e' interessante notare che la lunghezza del comando non influenza il tempo impiegato per il calcolo, questo lascia pensare che esso sia eseguito tramite hardware dedicato (crypto-processore).

Se la Signature calcolata e' diversa da quella presente nel comando si ha lo status 9002 (chiamato anche 9002\_B oppure 9002 di tipo 2).

9002\_A e 9002\_B differiscono per i tempi di risposta: **time(9002\_B) > time(9002\_A)**

## 12 - Decrypt SuperEncryption

Il decrypt SE avviene ad ottetti, agendo sui byte compresi tra i dati di P3 ed il Nano 82.

C1 40 01 B1 6A

43	51	6E	B1	92	0F	87	17	D3	5C	87	47	34	5E	39	79
D7	C6	1A	C9	4F	E8	40	6C	67	E3	AB	84	4C	29	3F	DD
A3	F0	E6	37	86	6D	8A	23	CB	88	55	9D	47	78	B6	71
54	9F	82	D8	85	1C	3E	05	34	36	27	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Osservazione: il decrypt SE agisce in maniera lievemente diversa per le INS 3C, analogamente a quanto accadeva nel SEKA 1.

### SuperEncryption

In caso di encryption delle C1 3C, non viene decryptato il primo ottetto di dati.

Osservazione 2: a differenza della SE del SEKA 1, dove gli ottetti incompleti erano lasciati in chiaro, qui si ha un processo di pseudo-crypt degli 8 byte precedenti il Nano 82, in modo da non lasciare nessun dato visibile.

Dopo la SuperEncryption abbiamo una normalissima INS tipo SEKA 1, con i canonici nanocomandi, ma non si e' ancora giunti alla loro esecuzione. C'e' una sorta di pre-analisi dei dati, che dovrebbe controllare se, saltando di nano in nano si arriva sull'82 (Nano signature). Un esempio rendera' tutto piu' chiaro:

```

10 01 [F0] FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF [22] 1A 3F [21] 1A 7F [90] 5D B9 5D 66 DF AC FF 00 45
[90] 5C 55 A8 EE 6F 9F 67 AA 92 [90] 5E EF AB 5A 97 FA BC 6F 91 [82] C9 2D C3 6C
C4 77 74 8B ... ecc. ecc.

```

1 + 1 byte	Nano <b>10 01</b> - P3 e P4, non considerati nell'esecuzione del comando, vengono saltati
1 + 32 byte	Nano <b>F0</b> piu' 32 byte di dati (Customer Word Pointer bitmap)
1 + 2 byte	Nano <b>22</b> - Controllo data fine abbonamento
1 + 2 byte	Nano <b>21</b> - Impostazione data fine abbonamento
1 + 1 +8 byte	Nano <b>90</b> - Scrittura Key primaria (Indice <b>5D</b> )
1 + 1 +8 byte	Nano <b>90</b> - Scrittura Key primaria (Indice <b>5C</b> )
1 + 1 +8 byte	Nano <b>90</b> - Scrittura Key primaria (Indice <b>5E</b> )

```

10 01 [F0] FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF [22] 1A 3F [21] 1A 7F [90] 5D B9 5D 66 DF AC FF 00 45
[90] 5C 55 A8 EE 6F 9F 67 AA 92 [41] 5E EF AB 5A [90] 5E BC 6F 91 [82] C9 2D C3
6C C4 77 74 8B ... ecc. ecc.

```

Qui si giungerebbe al Nano 90, che richiederebbe 9 byte di dati, oltrepassando il Nano 82 !!! Questo e' un esempio di parsing errato. In tal caso si dovrebbe avere lo status 9600 (questo controllo e' stato chiamato "pre-parsing"). Se il pre-parsing viene superato, il comando va (finalmente) in esecuzione.



## Lo status 9600 & l'INS 40

Il pre-parsing per l'INS 40 funziona grosso modo come descritto sopra, cioè la card esamina tutti i Nano fino ad arrivare al Nano 82 (oppure no, ed allora status 9600). Da notare che non è necessario che un Nanocomando incontrato sia significativo (cioè che abbia un significato), la card semplicemente lo ignora e va ad esaminare il Nano successivo, in funzione della lunghezza teorica del Nano non riconosciuto.

## Lo status 9600 & l'INS 3C - Il funny-bug 2

### **Con l'INS 3C un parsing errato non genera sempre lo status 9600**

*Questo dato sperimentale contrasta con la teoria precedentemente esposta sul pre-parsing. E' ragionevole pensare ad una diversa gestione dei dati dell' INS 3C.*

Sono stati osservati comportamenti diversi al variare del corpo dell'INS, spesso compare lo status 9A00, anch'esso legato al pre-parsing (si ottiene quando i dati della C1 3C contengono i Nano 24, 81, 90, 91, F6, F7), ma la cosa interessante si ha quando il parsing provoca lo "scavalco" della signature: anziché avere 9600, si ha un bug simile a quello presente sulle precedenti versioni di card: il **funny-bug**.

#### **Funny bug SEKA 1 (citazione)**

Ci troviamo alla conclusione del processo di esame dei nanocomandi relativi all'istruzione 3C; in particolare si tratta di sommare il numero di byte dell'ultimo nanocomando al numero totale di byte processati e verificare se si sia raggiunta o meno la lunghezza complessiva del comando (LEN). Osservando la ROM notiamo tale comparazione ma, al contrario di un corretto algoritmo, non segue una routine di controllo ma si passa direttamente alle fasi di chiusura (a differenza dell'ins 40). Infatti segue la copia del buffer dei record in eeprom e il mascheramento del buffer dati. E' proprio il passaggio diretto a tale fase di chiusura che attiva il bug: **ne deriva la scrittura del record buffer in eeprom, nel primo record**. Solitamente nel record buffer si trova la chiave usata nel comando, mentre i registri che puntano al numero di record sono resettati, indirizzando al primo. Se viene processato un nanocomando sconosciuto tale da provocare il superamento della lunghezza totale del comando, la chiave (o eventualmente qualsiasi altra struttura presente nel buffer dei record) viene copiata sul record 1 della eeprom.

Nelle nuove card il bug funziona in maniera leggermente diversa, si osserva che sul primo record viene copiata la MK00 primaria del provider cui si riferisce la C1 3C... chiamiamolo **funny-bug 2** ☺

*Potrebbe essere interessante provare il bug su un provider senza MK00 e vedere cosa succede.*

## Lo status 9600 e l'INS 38 - Il bug 36/38

Qui si osserva un bug macroscopico, che e' stato prontamente corretto sulle card 7.1. La routine di controllo della vecchia C1 38 e' stata rimossa ed al suo posto viene utilizzata la stessa routine di pre-parsing dell'INS 40. In pratica, il controllo sul pre-parsing viene superato se i dati seguono i canoni dell'INS 40 e non quelli dell'INS 38 ! Vediamo come funzionava il pre-parsing della C1 38 in SEKA 1:

### Istruzioni 36 e 38 - Lettura record e signature (citazione)

Questa coppia di istruzioni esegue la lettura dei record presenti sulla carta, con la gestione della signature sia sul comando inviato, sia sulla risposta generata. Inoltre permette di modificare alcuni byte dei record Bx eventualmente presenti. Prima va eseguita la 38 ( invio dati ) e poi la 36 ( richiesta dati ).

**C1 38 P1 P2 1A 16 xx 2A mm mm 2B nn nn 86 K0 K1 K2 K3 K4 K5 K6 K7 82 SIGNATURE**

I valori **16**, **2A**, **2B**, **86** non sono Nanocomandi veri e propri, è necessario che siano nella posizione indicata, altrimenti si ha errore **96 00**. *(Ecco quindi come funzionava il pre-parsing dell'INS 38 in SEKA 1, si trattava di un controllo sulla presenza degli pseudo-nano in ben precise posizioni, NdA)*

P1 e P2 hanno lo stesso significato che per l'istruzione 40, si possono usare PK oppure PK+SK ed è inoltre utilizzabile la SUPERENCRYPTION.

Il valore 'xx' indica il tipo di record da dumpare, mentre i byte 'mm mm' indicano un offset ( in pratica si comportano in maniera analoga dati dell'INS 34 ):

<b>00 vv vv</b>	Provider Package Bitmap record	
<b>01 vv vv</b>	Provider PPV Credit record	<b>vv vv</b> valori qualunque
<b>03 xx xx</b>	Provider PPV record	<b>xx xx</b> contiene l'EventID della trasmissione PPV
<b>04 00 yy</b>	Record Ex	<b>yy</b> specifica il tipo Record Ex da cercare
<b>06 zz zz</b>	Record generici	<b>zz zz</b> è il <b>Numero Record</b> da cui iniziare la lettura

Attenzione: nel caso in cui si abbia 'xx' uguale a **03** oltre a mostrare i PPV record, l'istruzione va a modificare il PPV-event spot, cioè il decimo ed undicesimo byte ( contando da sinistra ) del record trattato inserendo due valori che seguono **2B**. Dopo c'è il valore **86** seguito da 8 byte.

Il dump vero e proprio viene eseguito con l'istruzione 36

**C1 36 P1 P2 LEN**

P1 deve essere uguale a P1 dell'istruzione precedente con, in più, il sesto bit posto ad uno.  
P2 indica, come sempre, la Key. Se il bit più significativo è settato, la risposta della carta è criptata.  
LEN deve essere maggiore di 13 ( esadecimale ).

L'esecuzione di questa istruzione va a buon fine solo se è preceduta da un'istruzione 38. La risposta all'istruzione 36 ha la seguente forma: il primo byte è la lunghezza della risposta, si ha poi il valore 86 seguito dagli 8 byte inseriti con l'istruzione 38, infine si ha il dump dei record secondo lo stesso significato dell'istruzione 32 seguiti dal valore 82 e dalla signature calcolata sulla risposta. E' importante notare che questa è l'unica istruzione che ha la risposta con signature.

Adesso non e' piu' necessaria la presenza dei valori 16, 2A, 2C, 86, da cui il **bug 36/38**. Infatti, se e' sufficiente seguire i canoni dell'INS 40, allora un'INS 38 col corpo di una C1 40 e' eseguita senza errori !

C1 40 01 B1 5C

10 01 7E 3C 29 03 1B AC 43 31 82 A1 7E AE C4 26  
96 F2 3E 0A C3 9E 76 3E C6 20 86 79 2E 4A 8D D9  
18 14 95 3B 2E B6 54 D2 89 5F 76 0B 23 3B 63 4D  
BF 00 EA 81 E5 24 BB 93 CA D4 D0 6F F8 38 5F 9C  
5B EF AB 11 33 9B 17 8A EC CC 1D 6B 90 5D CD 2F  
50 2C 90 A7 BE 29 68 37 7C 56 6F 33 [90 00]

Cambiando 40 in 38 ...

C1 38 01 B1 5C

10 01 7E 3C 29 03 1B AC 43 31 82 A1 7E AE C4 26  
96 F2 3E 0A C3 9E 76 3E C6 20 86 79 2E 4A 8D D9  
18 14 95 3B 2E B6 54 D2 89 5F 76 0B 23 3B 63 4D  
BF 00 EA 81 E5 24 BB 93 CA D4 D0 6F F8 38 5F 9C  
5B EF AB 11 33 9B 17 8A EC CC 1D 6B 90 5D CD 2F  
50 2C 90 A7 BE 29 68 37 7C 56 6F 33 [90 00]

Attenzione: provando a fare la stessa cosa con un INS 3C non si ottiene analogo risultato:

C1 3C 01 BD 5C

10 01 E6 12 CC 77 60 AE 96 FF F4 4D 58 03 99 F1  
3F EF F3 A4 44 E6 1B 16 18 21 EB 40 D4 65 BC F5  
66 60 6D 7D 54 29 28 06 52 95 29 D6 07 E0 11 23  
1C 8E 89 29 89 2A 43 4E 11 98 2A 63 35 DB 56 F4  
4B 03 82 B2 66 29 69 80 69 50 68 FC EC 0B 2E 3B  
F2 A7 BC 04 CA A8 15 D9 60 7D 28 47 [90 00]

C1 38 01 BD 5C

10 01 E6 12 CC 77 60 AE 96 FF F4 4D 58 03 99 F1  
3F EF F3 A4 44 E6 1B 16 18 21 EB 40 D4 65 BC F5  
66 60 6D 7D 54 29 28 06 52 95 29 D6 07 E0 11 23  
1C 8E 89 29 89 2A 43 4E 11 98 2A 63 35 DB 56 F4  
4B 03 82 B2 66 29 69 80 69 50 68 FC EC 0B 2E 3B  
F2 A7 BC 04 CA A8 15 D9 60 7D 28 47 [90 02]

Nelle C1 40 d'esempio gli status byte sono sempre **90 00**, in realta' esistono tantissime C1 40 con status diversi ed ugualmente utilizzabili per il bug. Gli status piu' frequenti sono questi:

- **97 xy**
- **90 19**
- **93 01**
- **90 09**

Oppure [90 37]

Conclusione: a parita' di byte cryptati, il de-envelope delle INS 3C porge un risultati diverso da quello delle INS 40/38. Potrebbe essere diversa la key utilizzata, oppure l'inizializzazione dell'algoritmo, oppure diversa la gestione di P5 che, comunque, si trova nella solita posizione (*dimostrabile, NdA*).

Torniamo alla C1 38 con status 90 00 ... questo risultato ci permette di lanciare con successo una C1 36 (si avrebbe altrimenti lo status 9014).

**C1 36 P1 P2 LEN**

Analizziamo vincoli e gradi di liberta' relativi a questa INS:

**P1** deve essere uguale a **2y** oppure **3y** dove “y” deve essere uguale al nibble basso di P1 della C1 38 precedentemente inviata (se tali vincoli non sono rispettati si ha lo status 9401).

**P2** indica la Key e le tabelle hash da usare per criptare la risposta (stesso significato delle C1 38/3C/40).

**LEN** deve essere maggiore di 13 ( esadecimale ), altrimenti si ha lo status 6700.

La risposta che si ottiene dipende dal contenuto del corpo dell'INS 38 che, se inviata seguendo rigorosamente la sintassi richiesta, avrebbe questa forma (dopo che la card ha decriptato il comando):

**C1 38 P1 P2 LEN**

```
P3 + dati di P3 + 16 d1 2A d2 d3 2B d4 d5  
86 K0 K1 K2 K3 K4 K5 K6 K7 82 s0 s1 s2 s3 s4 s5  
s6 s7 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 P5
```

Ricordando che, a causa di un bug, non c'e' necessita' di seguire tale sintassi, la card non controlla la presenza dei “nano”, ma preleva direttamente i dati “**d1**”, “**d2 d3**”, “**d4 d5**”, “**K0 ... K7**”:

Iniziando a contare i byte a partire da quello immediatamente successivo ai dati di P3, si prelevano i dati seguendo questo criterio...

- il secondo per d1
- il quarto e quinto per d2 d3
- il settimo e ottavo per d4 d5
- dal decimo al diciassettesimo (compresi) per K0 ... K7

Esempio:

C1 40 01 B0 61

10 01 92 BD 1F C2 E6 A6 C0 8B 1E F7 02 1E 7D F0  
07 F2 31 2B 31 46 9E E9 1F 28 65 0D 12 68 F6 F1  
25 B2 91 66 63 C0 EA 48 7D E3 1F EB E0 99 92 37  
26 86 DB 0E C6 92 13 CD 61 E7 4C 70 45 27 2F A6  
D9 B2 74 0C DF 7C E4 07 5D 62 B1 DD B0 13 F5 8C  
9E 2A F2 28 12 55 1E 8D 76 43 19 31 01 6E B1 92  
11

Trasformo in C1 38...

C1 38 01 B0 61

10 01 92 BD 1F C2 E6 A6 C0 8B 1E F7 02 1E 7D F0  
07 F2 31 2B 31 46 9E E9 1F 28 65 0D 12 68 F6 F1  
25 B2 91 66 63 C0 EA 48 7D E3 1F EB E0 99 92 37  
26 86 DB 0E C6 92 13 CD 61 E7 4C 70 45 27 2F A6  
D9 B2 74 0C DF 7C E4 07 5D 62 B1 DD B0 13 F5 8C  
9E 2A F2 28 12 55 1E 8D 76 43 19 31 01 6E B1 92  
11 [90 00]

I dati vengono prelevati dopo che sono stati messi in chiaro. Ipotizzando che il decrypt sia il seguente...

10 01 10 02 17 00 10 03 80 00 00 00 00 00 22 00  
80 90 51 DE E2 04 D1 AF B3 FB B6 91 51 A1 37 9E  
4B D0 49 4C 51 21 1A 7F 90 5C 75 13 DE 05 65 03  
C4 57 B0 66 75 63 6B 20 26 20 73 75 63 6B 90 5D  
9D 33 0E A2 6C 2D 3C 05 90 5E 06 73 8A B7 5D 9A  
4C 5C 41 09 C3 22 21 82 D3 E8 54 CA 78 CD D2 61  
P5

... si capisce quali sono i dati prelevati dalla card (in giallo) ed utilizzati per la successiva C1 36. Essa puo' generare una risposta in chiaro oppure criptata, variabile a seconda del valore assunto da **d1**.

Risposta con d1 = 00 : Provider Bitmap Package Record

```
C1 38 01 B1 5C
10 01 7E 3C 29 03 1B AC 43 31 82 A1 7E AE C4 26
96 F2 3E 0A C3 9E 76 3E C6 20 86 79 2E 4A 8D D9
18 14 95 3B 2E B6 54 D2 89 5F 76 0B 23 3B 63 4D
BF 00 EA 81 E5 24 BB 93 CA D4 D0 6F F8 38 5F 9C
5B EF AB 11 33 9B 17 8A EC CC 1D 6B 90 5D CD 2F
50 2C 90 A7 BE 29 68 37 7C 56 6F 33 [90 00]
```

```
C1 36 21 00 LEN
36
86 K0 K1 K2 K3 K4 K5 K6 K7
83 YY YY YY YY YY YY YY YY
04 82 + bytes ad FF [90 35]
```

In questo caso la risposta e' in chiaro, sono presenti alcuni parametri caratteristici (detti pseudo-nano):

Pseudo-nano 86 - E' seguito dai byte K0...K7 dell'INS 38 (vedi)

Pseudo-nano 83 - Precede il Bitmap Package per il Provider indicato dal low nibble di P1

Pseudo-nano 04 - Indica che non ci sono ulteriori informazioni

Nano 82 - E' quello della Signature (se la risposta e' in chiaro, la Signature non e' presente)

Dopo il Nano 82 sono trasmessi tanti byte a FF quanti sono necessari ad arrivare alla giusta lunghezza della risposta.

```
C1 38 01 B1 5C
10 01 7E 3C 29 03 1B AC 43 31 82 A1 7E AE C4 26
96 F2 3E 0A C3 9E 76 3E C6 20 86 79 2E 4A 8D D9
18 14 95 3B 2E B6 54 D2 89 5F 76 0B 23 3B 63 4D
BF 00 EA 81 E5 24 BB 93 CA D4 D0 6F F8 38 5F 9C
5B EF AB 11 33 9B 17 8A EC CC 1D 6B 90 5D CD 2F
50 2C 90 A7 BE 29 68 37 7C 56 6F 33 [90 00]
```

```
C1 36 21 90 LEN
36
1C 2A 06 F8 58 E1 46 E3
B1 C6 53 51 6E 92 56 21
46 DE 4A F3 82 0E 0F F1
E4 1E 70 F0 7E ... [90 00]
```

Questa e' una risposta criptata, rimane riconoscibile il solo Nano 82, seguito dalla signature. E' inoltre presente un byte in piu' rispetto alla C1 36 in chiaro, che e' quello immediatamente evidenziato in giallo (1C, in decimale 28). Tale valore indica quanti dati significativi ci sono nella risposta e che sono tutti quelli che lo seguono fino alla signature compresa.

Risposta con d1 = 01 : Provider PPV Credit Record

```
C1 38 01 B1 5C
10 01 E9 2B BD 29 C1 89 4A DC 74 40 E8 8B 62 27
04 85 DD 6B 0A 8B AB 72 5A D5 18 36 5A 7B 1F AD
D4 89 C9 D7 CC 1A E9 94 35 8B 0F 10 96 09 9C DD
89 AB F7 CA F4 2A 41 87 32 35 7C 70 C7 3B DA 33
16 2D BB 2B DF C2 A2 4B 86 F5 92 BC C5 B3 92 A7
FC A7 5B 62 22 8E 42 12 DE 5D EE 25 [90 00]
```

Richiedendo una risposta alla C1 36 in chiaro...

```
C1 36 21 00 LEN
36
86 K0 K1 K2 K3 K4 K5 K6 K7
84 YY YY YY YY YY YY YY YY
04 82 + bytes ad FF [90 35]
```

Pseudo-nano 86 - E' seguito dai byte K0...K7 dell'INS 38 (vedi)

Pseudo-nano 84 - Precede il Credit Record per il Provider indicato dal low nibble di P1

Pseudo-nano 04 - Indica che non ci sono ulteriori informazioni

Nano 82 - E' quello della Signature (se la risposta e' in chiaro, la Signature non e' presente)

Dopo il Nano 82 sono trasmessi tanti byte a FF quanti sono necessari ad arrivare alla LEN richiesta.

Richiedendo una risposta alla C1 36 criptata (ho ommesso la C1 38)...

```
C1 36 21 90 LEN
36
1C 7B AA 06 E3 11 D4 D3
24 83 7B AD 4A 16 C7 1C
CD A8 A9 B8 82 19 58 38
84 ED 84 B1 F2 ... [90 00]
```

Dall'esame delle risposte criptate non si riesce a distinguere il caso d1 = 00 dal caso d1 = 01.

Risposta con d1 = 03 : Provider PPV Record

C1 38 01 B1 5C

10 01 37 FA 7D D3 E0 B2 3B 1C 9B 04 D8 FF C1 38  
8D C6 5C E1 38 7F FD C5 84 35 A8 1E D4 9C 25 48  
C1 76 9C 4D A8 21 DE 66 E0 C8 CC 29 AB 50 8D 20  
F5 AD 61 13 2C 5E A2 E3 56 5B F8 33 5A FA FC AD  
97 85 84 49 C1 56 10 AD 4B 9C C1 9F BB 70 B5 CC  
67 69 61 9C 70 34 3E 90 4E 39 2E 10 [90 00]

Al momento dell'invio della C1 36, la card esegue una ricerca in EEPROM. Tale ricerca ha lo scopo di reperire un Record Bx...

Record Bx : PPV Record (citazione)

**00 ID ID dn cv 00 00 dd dd sp sp Bx**

Il record Bx viene creato al momento dell'acquisto di un evento PPV. In pratica, con un'apposita richiesta dati il decoder verifica la presenza di questo record per un dato PPV Event Id, se presente è possibile visionare la trasmissione PPV.

Il primo byte non ha significato ed è sempre a 00, i successivi due indicano l'Event Id a cui si riferisce il record. Il byte indicato con **dn** è il Numero di Diffusione. Quando uno stesso evento viene trasmesso su più canali, il PPV Event ID è lo stesso su tutti i canali mentre il numero di diffusione cambia. Il numero di diffusione cambia anche fra due successive trasmissioni dell'evento sul solito canale. Serve quindi per identificare l'evento temporalmente e spazialmente ( il canale ). Ad ogni incremento del numero di diffusione corrisponde una diminuzione del numero di visioni ancora disponibili. Se l'evento non è stato ancora visionato, assume il valore di FF Il byte **cv** è il numero di visioni ancora disponibili. Seguono due byte Seguono due byte a 00, senza un particolare significato e due byte indicanti la data della prima visione dell'evento ( **dd dd** ). Se l'evento non è ancora stato visionato, assumono il valore FF FF. Infine c'e' il PPV event-spot (due byte normalmente a 00), modificabili con le INS 36/38.

Esempio :

Supponiamo di avere il PPV Event **0E 38**

**00 0E 38 FF 04 00 00 FF FF 00 00 B1**

Quando il Film è stato acquistato, ma non è ancora stato visionato

**00 0E 38 0F 04 00 00 16 A1 00 00 B1**

Quando il Film è disponibile ed è stato visionato almeno una volta.

**0E 38** = PPV-EventID

**0F** = Numero di diffusione / **FF** ancora nessun numero di diffusione è stato fornito

**04** = Numero di visioni disponibili

**16 A1** = Data della prima visione / **FF FF** = Evento non ancora visionato



Nella C1 38 e' indicato l'Event ID da ricercare tramite i byte **d2 d3**, mentre i dati **d4 d5** rappresentano il PPV event-spot. Se nell'istruzione viene specificato un Event ID che non è stato memorizzato nella carta, sarà considerato l'Event ID che ha il valore ( esadecimale ) immediatamente superiore a quello specificato. Di conseguenza l'Event-ID 00 00 individua in ogni caso un PPV Event memorizzato.

Se e' stato trovato l'event ID si ha la seguente risposta in chiaro...

```
C1 36 21 00 LEN
36
86 K0 K1 K2 K3 K4 K5 K6 K7
B1 YY YY YY YY YY YY YY YY YY YY YY
04 82 + bytes ad FF [90 35]
```

Pseudo-nano 86 - E' seguito dai byte K0...K7 dell'INS 38 (vedi)

Pseudo-nano B1 - Precede il PPV record

Pseudo-nano 04 - Indica che non ci sono ulteriori informazioni

Nano 82 - E' quello della Signature (se la risposta e' in chiaro, la Signature non e' presente)

Dopo il Nano 82 sono trasmessi tanti byte a FF quanti sono necessari ad arrivare alla LEN richiesta.

Risposta criptata...

```
C1 36 21 90 LEN
36
1F 69 FE 5E 15 DC F3 62
AF FA DD 5D 07 06 55 A0
3C C8 58 28 45 AD 33 82
A5 6F DA 72 9F 99 D4 70 ...
[97 04]
```

La risposta e' piu' lunga rispetto ai casi precedenti. ATTENZIONE: se sono presenti piu' PPV record e' possibile richiederli "a gruppi" con una singola C1 36; e' sufficiente utilizzare valori sufficientemente grandi per LEN.

E' necessario a questo punto commentare lo status che si e' ottenuto, perche' e' indicativo di una *tentata* scrittura su EEPROM.

#### Citazione

Nel caso in cui si abbia **d1** uguale a 03 oltre a mostrare il PPV record, l'istruzione va a modificare il PPV-event spot, cioe' il decimo ed undicesimo byte ( contando da sinistra ) del record trattato inserendo i valori **d4 d5**.

### Risposta con d1 = 04 : Record Ex

Contrariamente a quanto scritto in molti doc, per ottenere questa risposta non e' necessario indirizzare il comando al Provider 00 00 (provider SEKA).

```
C1 38 01 B1 5C
10 01 B0 DF 71 ED 4E 3E 40 A0 CA B4 34 9F 49 84
B2 22 56 D6 AD 5C 31 2D AE 51 AE CC F6 97 F1 10
5D 56 D8 EA CB B0 F4 5B D5 01 1B 40 01 C3 CC 7B
2B F7 26 44 BE D6 04 14 95 FB F6 5B 6C 5B BC C1
51 54 5B A0 7A C3 47 78 C4 EB 5D EC 3C A7 A3 CF
1F D5 32 87 A6 D0 C9 DD CD 8D 1B 01 [90 00]
```

A seconda del valore di **d3** , si seleziona il Record Ex di cui interessano i dati (d3 deve essere uguale al primo byte del record)

Richiesta in chiaro...

```
C1 36 21 00 LEN
36
86 K0 K1 K2 K3 K4 K5 K6 K7
B2 YY YY YY YY YY YY YY YY YY YY YY
04 82 + bytes ad FF [90 35]
```

Pseudo-nano 86 - E' seguito dai byte K0...K7 dell'INS 38 (vedi)

Pseudo-nano B2 - Precede il SEKA Record (i primi 0B byte di un record E0)

Pseudo-nano 04 - Indica che non ci sono ulteriori informazioni

Nano 82 - E' quello della Signature (se la risposta e' in chiaro, la Signature non e' presente)

Risposta criptata...

```
C1 36 21 90 LEN
36
1F 22 ED 88 F1 77 36 13
00 CD EB 42 42 4C 95 69
DA 8D 13 B1 AB 48 5A 82
D4 17 33 69 FE 5E 15 32 ...
[90 00]
```

Da notare che la lunghezza della risposta e' la stessa del d1 = 03, e' pero' possibile distinguere i due casi osservando gli status byte che, in questo caso, assumono sempre valore 9000.

### Risposta con d1 = 06 : Record generico

In questo caso si vanno a leggere i record "generici", cioè nel formato in cui sono memorizzati sulla carta (leggibili anche con l'INS 32/34). Il record visualizzato e' il primo che contiene dati relativi al provider indicato dal low-nibble di P1, la ricerca del record parte da quello specificato nell'INS 38 attraverso i byte **d2 d3**.

```
C1 38 01 B1 5C
10 01 00 4C 0D F3 5C BE 9E 54 66 A7 8D 33 49 BF
DC C0 30 ED 74 8B 49 64 83 6F F0 0F F5 FC 33 2E
C3 C9 86 B4 59 27 D8 93 36 7C 9D C7 C6 34 82 8B
43 7F 4A 20 43 36 28 D7 45 CA 38 69 FA 10 11 94
FE 9E A8 C3 4A 5E 7A C1 3A B0 61 3C E9 2B 80 98
43 90 81 59 CA E2 9F 68 1E 2B 2C 47 [90 00]
```

Risposta in chiaro...

```
C1 36 21 00 LEN
36
86 K0 K1 K2 K3 K4 K5 K6 K7
D2 ii ii yy yy yy yy yy yy yy yy yy yy yy yy 00 00
03 82 + bytes ad FF [90 35]
```

Pseudo-nano 86 - E' seguito dai byte K0...K7 dell'INS 38 (vedi)

Pseudo-nano D2 - Precede il Record

I dati associati a D2 sono organizzati in questo modo:

- 02 byte usati per mostrare l'indice del record
- 0C byte indicanti il contenuto del record
- 02 byte uguali a 00 00

Pseudo-nano 03 - Possibili ulteriori informazioni

Nano 82 - E' quello della Signature (se la risposta e' in chiaro, la Signature non e' presente)

Risposta criptata...

```
C1 36 21 90 LEN
36
24 1D 02 62 22 02 76 60
BC E1 C5 7A DE CA 5C 3B
77 9F CC F0 E5 7B 7C 79
A1 44 D4 01 82 B4 A0 41
40 49 4D 5A 97 ... [90 00]
```

Il caso piu' frequente: le risposte "corte"

Capita molto spesso che, in risposta ad una C1 36, si ottenga qualcosa del genere:

```
C1 36 21 90 LEN
36
13 A2 B1 24 F9 81 F9 33
C3 DC B7 82 DF 72 51 44
2D AE 60 BA ... [90 00]
```

Si parla in questi casi di "risposte corte", cioe' con pochi dati significativi. Il corrispondente contenuto in chiaro puo' assumere le seguenti forme

```
C1 36 21 00 LEN
36
86 K0 K1 K2 K3 K4 K5 K6 K7
04 82 + bytes ad FF [90 35]
```

```
C1 36 21 00 LEN
36
86 K0 K1 K2 K3 K4 K5 K6 K7
03 82 + bytes ad FF [90 35]
```

La prima risposta si ottiene nelle condizioni indicati dalla seguente tabella:

Valore assunto da "d1"	Valore assunto da LEN (hex)	Condizione al contorno
00	maggiore di 1D	Bitmap package non presente
01	maggiore di 1D	Credit record non presente
03	maggiore di 20	Evento PPV con ID maggiore o uguale a d2 d3 non presente
04	maggiore di 20	Record Ex con primo byte uguale a d3 non presente
06	maggiore di 25	Record con index maggiore o uguale a d2 d3 non presente

La seconda risposta si ottiene in vari modi. Il primo e' indicato dalla seguente tabella:

Valore assunto da "d1"	Valore assunto da LEN (hex)	Condizione al contorno
00	minore o uguale a 1D	Bitmap package non presente
01	minore o uguale a 1D	Credit record non presente
03	minore o uguale a 20	Evento PPV con ID maggiore o uguale a d2 d3 non presente
04	minore o uguale a 20	Record Ex con primo byte uguale a d3 non presente
06	minore o uguale a 25	Record con index maggiore o uguale a d2 d3 non presente

Il secondo modo e' quello di utilizzare valori di d1 diversi da 00, 01, 03, 04, 06.

*Nella descrizione delle possibili risposte alla C1 36 e' stata fatta la distinzione tra risposta criptata e risposta in chiaro. Vediamo adesso come e' possibile generare quest'ultima.*

### C1 36 in chiaro di tipo 1 - Flag SuperEncryption

Se il bit 7 di P2, indicante la SuperEncryption, e' uguale a 0 si ha una condizione di errore, perche' sulle 7.0 la SuperEncryption e' obbligatoria anche per le C1 36 ed infatti la card risponde con lo status 90 35. Prima dello status, pero', ci viene "regalata" dalla card la risposta in chiaro, fino al Nano 82 (la Signature e' calcolata sui dati criptati, in assenza di essi dopo il Nano 82 troviamo solo byte di riempimento uguali a FF).

Esempio:

```
C1 36 21 00 14
36 86 11 21 AA 2A 12 20
22 2F 03 82 FF FF FF FF
FF FF FF FF [90 35]
```

### C1 36 in chiaro di tipo 2 - Crypt con user algo

Se il bit 7 di P2 e' correttamente settato, ma i bit 5,6 sono tali che  $P2 = Cx$  oppure  $Dx$ , il processo di crypt della risposta non puo' avere luogo, perche' l'user algo non e' presente/attivo sulle 7.0. La card allora risponde con la risposta in chiaro, seguita dallo status 9034.

Esempio:

```
C1 36 21 D0 14
36 86 11 21 AA 2A 12 20
22 2F 03 82 FF FF FF FF
FF FF FF FF [90 34]
```

### C1 36 in chiaro di tipo 3 - Il bug sulla LEN

Puo' capitare che, pur trovandosi nei due casi precedenti (oppure richiedendo una normale risposta criptata) la card risponda ugualmente con l'INS in chiaro, seguita dallo status 9015. Questo accade quando si utilizza per **d1** un valore diverso da 00, 01, 03, 04, 06 e si supera una certa LEN (dipendente dal **d1** specificato).

Esempio:

```
C1 36 23 F0 14
36 86 11 21 AA 2A 12 20
22 30 FF FF FF FF FF FF
FF FF FF FF [90 15]
```

Questo comportamento, noto col nome di "bug sulla LEN", permette il dump di una piccola parte di ROM.

### Il bug sulla LEN (citazione)

Il bug presente sulle V7 e' discendente diretto del bug sulla len presente nel C134/32 delle card V6.0 e inferiori. Il bug era il seguente:

```
C1 34 00 00 03 XX 00 00
C1 32 00 00 YY
```

A seconda del valore assunto da 'XX' si ha un valore minimo ammissibile per 'YY', in funzione di una tabella contenuta in ROM. Sappiamo che i valori standard per 'XX' sono 00,01,03,04,06 ma la ROM non ci vieta di usarne altri!

Per valutare la lunghezza minima si usa una tabella i cui elementi sono letti sommando all'indirizzo base (della tabella) il valore XX, senza nessun controllo.

Se ad XX sostituisco valori > 06 vado a leggere 'indirettamente' un pezzo di ROM (i byte successivi alla tabella)...ho detto indirettamente perche' devo fare in questo modo:

```
C1 34 00 00 03 07 00 00
C1 32 00 00 00
```

```
C1 34 00 00 03 07 00 00
C1 32 00 00 01
```

```
C1 34 00 00 03 07 00 00
C1 32 00 00 02
```

Incrementare la LEN della C1 32 finche' non cambia la risposta della card...*(Finche non si ha lo status 9015, NdA)* quando cio' avviene ho trovato il valore del byte (il byte vale 'YY-1').

Sulle V7 il bug e' stato tolto dalla C1 34/32 ma e' rimasto nella C1 36/38! Si puo' azzardare un dump di parte della ROM, ma ci vuole molta pazienza ed un bel numero di INS. La C1 38 seka 1 aveva la seguente struttura:

```
C1 38 P1 P2 LL 16 XX 2A mm mm 2B nn nn 86 K1 K2 K3 K4 K5 K6 K7 K8 82 + SIG
```

dove 'XX' ha lo stesso significato della C1 34, quindi dobbiamo 'giocare' sul suo analogo presente nel comando seka2.

L'unica cosa da ricordarsi e' che la C1 36 ammette come LEN minima 0x14, perche' con valori inferiori si ha lo status 67 00, e che ,a differenza delle precedenti versioni di kard, si puo' superare tranquillamente il valore #5Fh.

*(omissis)*

il calcolo della LEN minima avviene pescando i dati da una tabella in ROM. A questo punto la gestione del valore trovato segue due strade diverse, a seconda che si tratti di un INS 32 oppure un INS 36, perche' quest'ultima restituisce in output gli stessi dati della 32 piu' altra roba (...) in totale fanno 20 byte in piu' (0x13 in esadecimale). Per le C1 36 ci si riporta al valore presente in ROM prendendo la LEN che da' come status 90 15 e togliendo 0x14.

*(omissis)*

- 9015 non e' contemplato per i valori 'normali' (00,01,03,04,06) di **d1**.

- Quando hai una C1 36 che non da' 9015 per **nessuna** LEN vuol dire che **d1** punta ad un valore in ROM maggiore o uguale ad EC oppure per il motivo del punto precedente. I due casi sono normalmente distinguibili, verificando la transizione (o meno) nano 03 --> nano 04 a partire da una certa LEN.

## Costruzione di una C1 38/40 "custom" a partire da un EMM

Per questa esercitazione sono necessari :

SmartTimer 1.6 o versioni superiori

EMM di attivazione o EMM di aggiornamento chiavi operative antecedenti al 31/01/2003

### **Fase 1 : Preparazione del comando iniziale :**

Nel caso si disponga di un EMM loggato precam esso ci appare nelle seguente forma :

86 00 65 0000 06xxxyzz 0070 00 B0 0203 509392031...

Senza addentrarsi troppo nella struttura di un EMM precam, possiamo distinguerne le parti piu' importanti e necessarie a ricostruirne l'equivalente destinato alla card:

- 65** Lunghezza del comando precam
- 06xxxyzz** PPUA a cui è indirizzato l'EMM
- 0070** Provider a cui è indirizzato l'EMM (in questo caso prov **01**; per la relazione  
Provider ID ↔ Numero Provider, vedi tabella)
- B0** Chiave con cui è criptato e firmato l'EMM in questo caso MK 00

### Tabella Provider

Nazionalita' della card	Numero Provider (Low-Nibble P1)				
	00	01	02	03	04
<b>Italiana</b>	<b>0000</b> SEKA	<b>0070</b> TEL£+DIGIT@L£	<b>0071</b> +C@LCIO	<b>0072</b> STREAM	<b>0073</b> P@LC0
<b>Francese (V7.1)</b>	<b>0000</b>	<b>0080</b> C@N@LSATELLIT£	<b>0081</b> C@NAL+	<b>0082</b> SPARE-A	Non presente
<b>Polacca</b>	<b>0000</b>	<b>0065</b> CYFR@+			Non presente
<b>Spagnola</b>	<b>0000</b>	<b>0064</b> C@N@LSATÉLIT£	<b>0066</b> C@N@LSATÉLITE2	<b>0067</b> C@N@LSATÉLITE3	Non presente

(Altri Provider ID esistenti : 0084 - 0086 - 0088)

La conversione dell'EMM nella forma equivalente destinata card si ottiene in questa maniera: prima di tutto si calcola la lunghezza reale del comando, sulla base della lunghezza del comando precam (nell'esempio, 65):

$$\text{LEN} = (\text{Lunghezza comando precam} - 4)$$

Poi si compila il comando:

C1 38 01 B0 61 10 01 509392031...

Nel caso, invece, si disponga di un EMM loggato sara' sufficiente cambiare la INS 40 in INS 38, eliminare gli status byte ed il byte di ACK.

*A seconda del software di logging utilizzato l'ACK puo' non essere presente. Tale byte si trova tra la LEN del comando ed i dati, esso assume lo stesso valore dell'INS.*

#### Esempio con ACK-byte

C1 40 01 B0 61 40 10 01 58 65 55 88 99 15 22....86 → 97 FA  
C1 38 01 B0 61 10 01 58 65 55 88 99 15 22....86

#### Esempio senza ACK-byte

C1 40 01 B0 61 10 01 58 65 55 88 99 15 22....86 → 97 FA  
C1 38 01 B0 61 10 01 58 65 55 88 99 15 22....86

A questo punto possiamo inviare il comando:

C1 38 01 B0 61  
10 01 58 65 55 88 99 15 22 D3 06 AA D2 F8 E6 19  
29 E3 5F DF 96 6A 1C 42 4F 94 0D B8 01 43 2F D0  
7E FB 90 DE 42 2B C7 14 A7 A8 E2 56 AD F6 10 56  
71 EB 38 6B 5B 6E 5E 0E CE 8B B9 BC 0E 30 94 60  
39 F2 AC 73 20 1B 6B 3E C5 CF 82 29 2C 14 73 E6  
23 F2 D7 99 21 7E 73 E3 90 27 40 A0 A5 16 55 C6  
86 [90 00]

...a seguito del quale invieremo la INS complementare alla 38 ovvero la INS 36. Essa ha la possibilita' di restituire, sia in forma criptata che in chiaro, dei dati che sono dipendenti dalla INS 38 immediatamente precedente ed in funzione della chiave e delle tabelle di crypt selezionate dai parametri P1 e P2. Tramite un semplice gioco di variazione dei parametri P1 e P2 andiamo alla ricerca di una risposta che soddisfi le condizioni imposte dai nuovi algoritmi di crypt.



## Fase 2 : Esecuzione ciclo INS 38 e INS 36 :

Come già accennato nella descrizione delle INS 36/38, esistono dei vincoli per l'esecuzione dei comandi.

### Parametri di P1 e P2 ammessi nella INS 36

Per i valori ammissibili si farà riferimento ai casi utili nella pratica, per una trattazione rigorosa della sintassi si faccia riferimento alla descrizione dell'INS (pag.10).

P1 High-nibble	Puo' assumere il valore <b>2</b> per tutti i valori del Low-Nibble di P2 Puo' assumere il valore <b>3</b> solo per i valori del Low-Nibble di P2 uguali a 0, 1
<b>P1 Low-nibble</b>	<b>deve assumere lo stesso valore dei P1 Low-nibble della C1 38</b>
P2 High-nibble	Puo' assumere i valori 9, B, D, F (NdA:con 'D' non si può avere risposta criptata Vedi pag.21 : C1 36 in chiaro di Tipo 2)
P2 Low-nibble	Puo' assumere i valori 0, 1, (2, 3), C, D, E tenendo conto che C, D, E non sono poi utilizzabili per creare EMM, ma solo ECM...

Ovvero valori ammissibili di P1 e P2 (esempio con **C1 38 01 B0 LL 10 01 ...**)

C1 36 **21 90** , C1 36 **21 B0** , C1 36 **21 D0** , C1 36 **21 F0**  
C1 36 **21 91** , C1 36 **21 B1** , C1 36 **21 D1** , C1 36 **21 F1**  
C1 36 **21 9C** , C1 36 **21 BC** , C1 36 **21 DC** , C1 36 **21 FC**  
C1 36 **21 9D** , C1 36 **21 BD** , C1 36 **21 DD** , C1 36 **21 FD**  
C1 36 **21 9E** , C1 36 **21 BE** , C1 36 **21 DE** , C1 36 **21 FE**  
C1 36 **31 90** , C1 36 **31 B0** , C1 36 **31 D0** , C1 36 **31 F0**  
C1 36 **31 91** , C1 36 **31 B1** , C1 36 **31 D1** , C1 36 **31 F1**

E, qualora se ne disponga :

C1 36 **31 92** , C1 36 **31 B2** , C1 36 **31 D2** , C1 36 **31 F2**  
C1 36 **31 93** , C1 36 **31 B3** , C1 36 **31 D3** , C1 36 **31 F3**

Il ciclo di invio INS 38 + INS 36 sarà quindi:

**Invio C1 38 P1 P2 LEN P3 P4 + DATI**  
**Invio C1 36 P1 P2 LL**

Per **LL** useremo il valore **14** (hex):  
in questo modo si otterranno solo  
risposte "corte".

Tale ciclo terminerà una volta che tutte le combinazioni possibili riguardanti le INS 36 sono state inviate. Le risposte ad ogni singola INS 36 vanno annotate: al termine della prova si utilizzeranno solo quelle che avranno la forma imposta dai parametri P3 e P4 della nuova decodifica.

### Forma ammissibile della risposta alla C1 36

#### 1) La forma della risposta deve essere obbligatoriamente criptata;

Una veloce verifica per sapere se si e' ottenuto una risposta criptata e' tramite l'analisi degli status byte: se assumono il valore 90 00, la risposta e' sicuramente criptata.

2) I primi due byte della risposta (che diventeranno il P3 e P4 della C1 38) devono obbedire alla regola :

**P3 Nibble alto maggiore di 1 e minore di 9**

**P4 Nibble basso uguale a 1 o uguale a 9**

Ovvero valori ammissibili della risposta:

C1	36	P1	P2	14	(36)	13	1x	x1	dd	dd	dd	dd	...
C1	36	P1	P2	14	(36)	13	1x	x9	dd	dd	dd	dd	...
C1	36	P1	P2	14	(36)	13	2x	x1	dd	dd	dd	dd	...
C1	36	P1	P2	14	(36)	13	2x	x9	dd	dd	dd	dd	...
C1	36	P1	P2	14	(36)	13	3x	x1	dd	dd	dd	dd	...
C1	36	P1	P2	14	(36)	13	3x	x9	dd	dd	dd	dd	...
C1	36	P1	P2	14	(36)	13	4x	x1	dd	dd	dd	dd	...
C1	36	P1	P2	14	(36)	13	4x	x9	dd	dd	dd	dd	...
C1	36	P1	P2	14	(36)	13	5x	x1	dd	dd	dd	dd	...
C1	36	P1	P2	14	(36)	13	5x	x9	dd	dd	dd	dd	...
C1	36	P1	P2	14	(36)	13	6x	x1	dd	dd	dd	dd	...
C1	36	P1	P2	14	(36)	13	6x	x9	dd	dd	dd	dd	...
C1	36	P1	P2	14	(36)	13	7x	x1	dd	dd	dd	dd	...
C1	36	P1	P2	14	(36)	13	7x	x9	dd	dd	dd	dd	...
C1	36	P1	P2	14	(36)	13	8x	x1	dd	dd	dd	dd.....	...
C1	36	P1	P2	14	(36)	13	8x	x9	dd	dd	dd	dd.....	...

### Fase 3 : Costruzione nuova INS "custom " ed aggiramento Envelope

Nel caso si abbia ottenuto una risposta valida, cioè corrispondente ai vincoli della Fase 2, possiamo andare a creare la nuova INS "custom", come esempio utilizziamo una risposta del tipo:

C1 36 21 B0 14

36

13 65 31 D3 B2 97 A5 D2

86 81 DE 82 77 B6 85 8D

A9 00 C0 AD [90 00]

Eliminiamo l'ACK-byte (36, se presente), la length della risposta (13) e status byte (90 00), in modo da avere

65 31 D3 B2 97 A5 D2 86 81 DE 82 77 B6 85 8D A9 00 C0 AD

Componiamo il nuovo comando:

C1 38 21 B0 LL 65 31 D3 B2 97 A5 D2 86 81 DE 82 77 B6 85 8D A9 00 C0 AD

ed aggiungiamo almeno 0x5A byte con valore pari a 0x00: essi rappresentano i byte che subiranno il de-envelope ma che non prenderanno parte al calcolo Signature ed esecuzione, grazie alla possibilità di aggiramento dell'-envelope, tramite la procedura spiegata più avanti.

E' consigliabile, anche per ciò che si andrà a realizzare nelle successive prove, partire con una length **LL = 0x76** ovvero occorre aggiungere 0x63 byte di valore pari a 0x00. Di questi 0x63 byte i **primi 8 post-signature saranno in "chiaro" e fuori envelope**, i **restanti 0x5A** saranno i byte in envelope (vedere la teoria su SSE/Envelope per ulteriori chiarimenti su length minime, LB, NLB, RSA ecc. ecc.).

Alla fine avremo :

C1 38 21 B0 76

65 31 D3 B2 97 A5 D2 86 81 DE 82 77 B6 85 8D A9

00 C0 AD 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00

Se inviamo questa INS otterremo lo status 9002 (possibili anche 9037 e 9038, a seconda del provider), perche' l'envelope non è stato ancora aggirato. L'aggiramento si ottiene facilmente cambiando il valore dell'ultimo byte della INS da inviare incrementandolo fino a che si passi dallo status 9002/9037/9038 allo status 9600. Cio' accade perche' la variazione di un byte si riflette sul risultato di tutto il de-envelope, in particolare sul valore assunto da **P5**, quando si ha il valore "giusto" (tale che si indirizzi correttamente il Nano 82 del comando che abbiamo costruito) allora siamo riusciti ad aggirare l'envelope. (NOTA: Potremmo cambiare uno qualsiasi dei byte in envelope - gli ultimi 0x5A byte - ma per comodita' e un po' per convenzione si utilizza l'ultimo byte. Non sempre esiste un valore per l'ultimo byte tale da generare l'aggiramento dell'envelope, allora si varia anche il penultimo byte).

*A pagina 29 e' presente la tabella che indica i giusti valori da assegnare al penultimo ed ultimo byte (NLB/LB) in funzione del numero di byte in chiaro (quelli tra signature ed Envelope) dell'INS.*

Consultando la tabella deriviamo :

INS = **38**, Byte in chiaro = **08**, Provider **1** → NLB = **0x00**, LB = **0x09**

```
C1 38 21 B0 76
65 31 D3 B2 97 A5 D2 86 81 DE 82 77 B6 85 8D A9
00 C0 AD 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 09
```

→ [96 00]

Qualora ci si trovi in una situazione non prevista in tabella si puo' fare la variazione a "mano" oppure ci si appoggia alla funzione "Find P5" del programma SmartTimer 1.7 (finestra CMD studio).

		INS 38 & 40					INS 3C				
		00 00	00 70	00 71	00 72	00 73	00 00	00 70	00 71	00 72	00 73
Numero di byte in chiaro dopo la signature e prima dei 5A byte in SSE	00	00 59	00 53	01 0D	01 14	01 A8	00 3A	00 63	02 AC	00 7B	01 9B
	01	00 58	00 08	00 DA	00 13	01 01	00 26	01 87	00 BB	00 2F	04 2E
	02	00 35	00 77	00 B0	00 55	00 69	00 4D	00 6D	01 39	00 42	00 2F
	03	00 D2	00 40	00 04	00 40	00 59	00 29	00 0E	01 75	00 04	00 10
	04	00 E3	00 27	00 8C	01 44	00 7C	00 2C	00 E9	00 56	00 62	01 C2
	05	00 2B	00 54	00 D4	01 DD	00 80	00 77	00 04	00 13	00 B4	00 14
	06	02 E9	00 8D	00 83	00 36	01 74	00 A3	02 05	01 1F	00 9B	00 3C
	07	00 66	00 47	01 11	00 D2	00 96	00 23	00 73	01 2B	00 EE	01 03
	08	00 09	00 3C	00 9B	00 09	00 0C	00 0E	00 08	00 DF	00 9A	01 45
	09	00 27	00 09	00 62	00 60	00 8D	00 6E	01 02	01 09	00 08	00 60
	0A	00 0B	01 05	01 BD	04 94	02 A6	00 E7	00 05	00 1F	00 A5	00 4E
	0B	00 B9	00 2E	00 B1	00 10	01 AE	00 35	00 10	00 3E	00 11	00 F3
	0C	01 38	00 83	00 EB	02 1C	00 36	00 02	00 77	00 75	00 05	00 A4
	0D	02 80	00 19	00 36	00 3A	00 10	00 27	00 B4	00 20	00 23	01 32
	0E	00 7E	00 17	00 42	00 F3	00 EA	00 11	01 76	00 21	00 51	00 51
	0F	00 54	00 2B		00 B3	00 99	00 06	01 0F	02 48	01 1F	00 EE
	10	00 85	01 5F	00 91	00 48	00 27	00 0D	00 85	00 68	00 66	00 DA
	11	00 B4	01 63	00 6A	01 69	01 1B	00 61	00 5C	01 C9	00 2C	01 20
	12	00 DC	01 01	00 B3	00 B2	00 09	00 42	00 86	01 6A	00 87	01 84
	13	00 74	00 69	00 96	00 B7	00 77	00 58	00 A7	00 E4	00 E5	00 24
	14	02 B0	00 0A	00 C6	00 46	00 82	00 0A	01 B2	00 09	00 52	01 C7
	15	02 2B	00 33	02 43	00 49	00 04	00 49	00 22	00 31	00 39	00 75
	16	01 11	00 59	00 33	01 04	00 4F	00 1D	00 3F	00 0A	01 F5	01 5E
	17	03 5C	01 49		00 7D	00 7E	00 A0	00 0D	00 33	00 92	00 1A
	18	00 1C	00 2A	00 3C	00 04	01 D3	00 2F	00 B9	00 64	00 5E	00 C3
	19	00 76	00 1E	01 1C	00 2C	00 34	00 51	00 01	00 23	00 5B	00 BF
	1A	00 37	01 0E	00 66	00 8D	00 2B	01 15	00 03	00 2C	00 38	00 84
	1B	01 9F	01 92	01 46	00 50	00 79	00 2B	00 8B	00 4E	00 8A	00 CF
	1C	00 0A	00 73	00 26	00 67	00 08	00 16	00 C5	00 22	00 7E	00 65
	1D	00 80	01 6F	01 DD	00 16	00 B3	00 AE	00 17	00 BD	00 9E	00 2B
	1E	00 B3	00 42	00 10	00 75	01 67	00 39	00 69	00 A4	00 7A	00 BB
	1F	01 49	00 03	00 8A	00 0A	00 66	01 1A	00 32	01 99	00 B1	00 AD
	20	00 69	00 15	00 5D	00 38	01 46	00 43	00 4C	00 69	00 8C	01 04
	21	02 16	00 0E	00 E7	00 23	00 A0	00 21	01 0A	00 9A	00 3B	00 42
	22	00 32	01 20	01 C3	00 C5	00 B6	01 76	00 84	00 6B	00 2A	01 07
	23	01 6A	00 66	00 20	00 81	00 70	00 0F	00 65	00 76	00 71	00 15
	24	00 3F	00 13	00 39	00 8E	01 D1	00 71	00 66	00 74	01 9C	00 13
	25	00 D1	00 36	01 05	00 35	00 54	00 73	01 2C	00 C5	00 15	00 4A

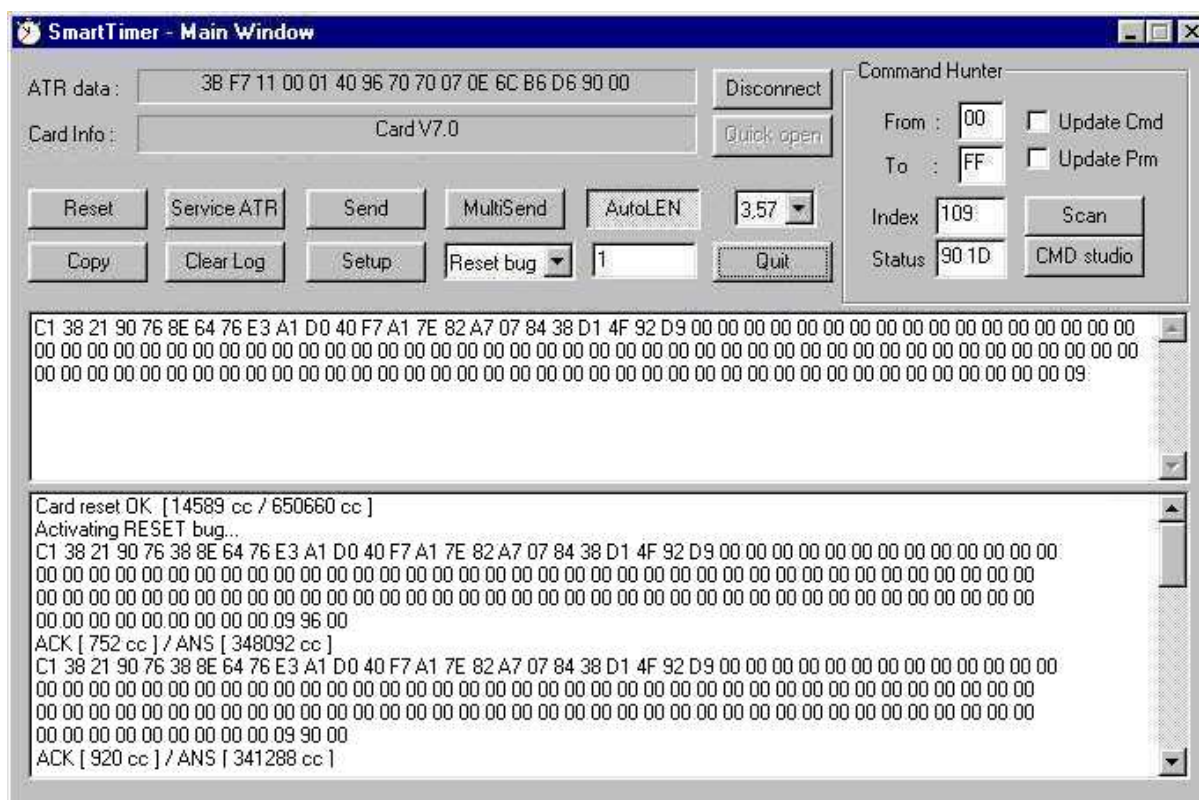
#### ***Fase 4 : Reset Bug e ottenimento dello status 9000 su di una INS "custom"***

Per passare dallo status 9600 allo status 9000 occorre semplicemente eseguire un ciclo che, per il cultori del “faccio a manina “ vuol dire:

## Reset card

**Invio INS**

Per i piu' pigri il tasto Multisend di SmartTimer (modalita' "Reset Bug") risolve il problema da se'. Ecco un piccolo LOG d'esempio :



### Che cos'è il RESET-bug

Lo status 9600 indica che il controllo sul pre-parsing non è stato superato, allora com'è possibile che dopo un certo numero di RESET si ottenga 9000?

La causa è da ricercarsi in un bug presente nella gestione della SuperEncryption, riprendiamo il comando utilizzato nella precedente Fase 3:

C1 38 21 B0 76

65 31 D3 B2 97 A5 D2 86 81 DE 82 77 B6 85 8D A9

00 C0 AD 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

00 00 00 00 00 09

I byte in verde, trattandosi di P3 e relativi dati, non vanno considerati per il decrypt, allora prima del Nano 82 rimangono solamente 3 byte, insufficienti a formare un ottetto completo. Quando i dati per il decrypt sono inferiori ad 8 la SuperEncryption "va in crisi", nel senso che il decrypt diviene dipendente non solo dai dati, ma anche da delle quantità ignote pescate dalla card chissà dove (il bug è tuttora oggetto di studio). Queste quantità variano ad ogni reset, di conseguenza, ad ogni RESET si ha un diverso risultato del decrypt, con la possibilità di indovinare un parsing-corretto (da cui lo status 9000).

Il numero di byte disponibili per il decrypt dipendono da P3: a parità di lunghezza del comando, più è grande l'high-nibble di P3 e minori sono i dati per il decrypt. Si possono avere addirittura **zero dati da decryptare**: in tal caso la card non porta a normale compimento l'esecuzione dell'INS (o risponde con l'ATR oppure si "blocca" finché non si invia un RESET).

Questo metodo ottiene lo stesso risultato del Reset Bug ma ha con se' delle proprieta' che lo rendono piu' appetibile. Il metodo consiste nella variazione del valore di uno o piu' byte posti tra l'ultimo byte della signature ed il primo byte in envelope. A titolo d'esempio, variamo il byte in verde della nostra INS:

Anche in questo caso, per i cultori del metodo a “manina”, il programma SmartTimer ci viene in aiuto. Nel nostro caso imponiamo come "Index byte" da variare il byte 20 (nel programma è espresso in decimale), settiamo la casellina “From” a 00 e la casellina “To” ad FF , la casellina “Status” la settiamo a 9000 che e' la condizione di uscita del nostro test, avviamo la funzione “Scan” ed aspettiamo che SmartTimer finisca ...





*(Copia & Incolla da vari post trovati in rete)*

Senza resettare mai si vari uno dei byte "in chiaro" dopo la signature finche' non si ottenga lo status 90 00. Detto i tale byte, si modifichi ora il byte **i+2n** o **i-2n**. Per ogni n tale che il nuovo byte variato sia ancora nei limiti sopra esposti (e cioè tra l'ultimo byte di signature e il primo byte antecedente gli ultimi 90) si notano delle cose molto interessanti:

- La sequenza degli status 9600/9000 ottenuta è identica.
- C'e' una ripetitività del Low-nibble ogni 4 High-nibble.
- Ripetendo la stessa procedura con un corpo sotto SSE differente il fenomeno si ripete ma per altri valori.

```
...82 s1 s2 s3 s4 s5 s6 s7 s8 00 00 00 00 00 00 00 00 00 00 00 00 .... 96 00
```

...82	s1	s2	s3	s4	s5	s6	s7	s8	00	3C	00	00	00	00	00	00	00	00	....	90	00
...82	s1	s2	s3	s4	s5	s6	s7	s8	00	7C	00	00	00	00	00	00	00	00	....	90	00
...82	s1	s2	s3	s4	s5	s6	s7	s8	00	BC	00	00	00	00	00	00	00	00	....	90	00
...82	s1	s2	s3	s4	s5	s6	s7	s8	00	FC	00	00	00	00	00	00	00	00	....	90	00

$$(\dots)$$
[illegible]

C1 38 21 B0 86

C1 38 21 B0 86

C1 38 21 B0 86

Ora facciamo questo esperimento...

C1 38 21 F0 71

```

65 11 A4 96 FA 52 EF 7A 88 7E 82 0E ED 18 95 B0
CA 14 97 XX YY 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
27

```

Faccio variare una WORD in chiaro del comando (i byte **XX YY**) e annoto i valori ai quali corrisponde lo status 9000 (costruisco la cosiddetta **"tabella dei 9000"**). Dopodichè invio un RESET e ripeto il procedimento. La prova è fatta provando tutte le WORD da **0000** a **3FFF**: data la periodicità di comportamento (secondo la somma modulo 0x4000) non importa andare oltre. Ottengo due successioni di valori, il primo risultato è che il numero totale di WORD che danno luogo a 9000 è costante ad ogni RESET. Il secondo risultato, ben più importante, è che le due successioni sono legate tra loro. Prendiamo i valori ricavati e calcoliamoci i **"delta"**, cioè le differenze tra WORD consecutive. Abbiamo così costruito le **"successioni delle delta"**. Confrontando le due successioni si vede che si ottengono le stesse differenze, ma cambia il punto di partenza (è come se fossero tra loro "traslate").

Successione (A)		Successione (B)	
WORD	DELTA	WORD	DELTA
0110	17	35CD	17
011D	13	35DA	13
0127	10	35E4	10
012E	7	35EB	7
0130	2	35ED	2
0135	5	35F2	5
015F	42	361C	42
0162	3	361F	3
0189	39	3646	39
019D	20	365A	20
019F	2	365C	2
01BD	30	367A	30
01C0	3	367D	3
01C1	1	367E	1
01C9	8	3686	8
01DD	20	369A	20
01E6	9	36A3	9
01EE	8	36AB	8
01F7	9	36B4	9
01FC	5	36B9	5
020B	15	36C8	15
(eccetera)		(eccetera)	

Alla luce del fatto che:

- Tra la successione (A) e (B) c'e' stato un RESET, ma non sono stati variati byte del comando
- Lo status 9600/9000 dipende dalla **(somma WORD) mod 0x4000**

si conclude che e' possibile legare assieme RESET-bug e BYTE-bug, in pratica, agli effetti del pre-parsing la variabilita' dovuta al RESET puo' essere considerata come una WORD incognita da aggiungere nella somma. Infatti, una volta allineate le successioni delle 'delta' si vede che la differenza tra le WORD omologhe e' costante per tutta la sequenza (ma CAMBIA ad ogni reset):

```
35CD - 0110 = 34BD
35DA - 011D = 34BD
35E4 - 0127 = 34BD
35EB - 012E = 34BD
```

e cosi' via...

(...)

Riassumendo, abbiamo visto che e' possibile costruire comando "validi" (cioe' che superano il controllo-signature), ma **rimane tuttora non controllabile cio' che si va ad eseguire**, dato che non conosciamo il contenuto dei byte in SE; si e' pero' visto che il comportamento dipende da:

- Valore assunto da P3 (9600 / 9000 / ATR / NO ANSWER)
- Numero di reset della carta
- Contenuto dei byte in SE
- Contenuto dei byte in ENVELOPE
- Contenuto dei byte in chiaro

Come gia' detto, un primo condizionamento del risultato viene dai **byte in chiaro**, secondo la regola della "somma WORD modulo 0x4000":

C1 38/40 P1 P2 LEN

P3 P4 x0 x1 x2 x3 x4 x5 x6 x7

82 s0 s1 s2 s3 s4 s5 s6 s7

q0 q1 q2 q3 q4 q5 q6 q7 q8 q9 q10 q11 q12 q13 q14

q15 q16 q17 q18 q19 q20 q21 q22 q23 q24

+ 90 byte in SSE/Envelope

Tenendo presente la distinzione tra high-nibble di P3 pari e dispari, i byte in chiaro (nell'esempio q0...q24, ovviamente possono essere di piu' o di meno, dipende dalla LEN del comando) vanno suddivisi in WORD (16 bit), secondo la seguente tabella...

High Nibble P3 dispari	High Nibble P3 pari
(q1 q2)	(q0 q1)
(q3 q4)	(q2 q3)
(q5 q6)	(q4 q5)
(q7 q8)	(q6 q7)
(q9 q10)	(q8 q9)
(q11 q12)	(q10 q11)
(q13 q14)	(q12 q13)
(q15 q16)	(q14 q15)
(q17 q18)	(q16 q17)
(q19 q20)	(q18 q19)
(q21 q22)	(q20 q21)
(q23 q24)	(q22 q23)
Il byte "spaiato" va considerato come (00 q0)	Il byte "spaiato" va considerato come (q24 00)

Fissati i byte **P3 P4 x0 x1 x2 x3 x4 x5 x6 x7 82 s0 s1 s2 s3 s4 s5 s6 s7** e variando i byte in chiaro si ottengono al massimo 16384 decrypt diversi (in esadecimale: 0x4000), e' inoltre possibile identificare l'insieme di tutte le combinazioni che danno luogo a **medesimo** decrypt tramite una semplice relazione:

#### High-Nibble P3 dispari

$[(00\ q0) + (q1\ q2) + (q3\ q4) + (q5\ q6) + (q7\ q8) + (q9\ q10) + (q11\ q12) + (q13\ q14) + (q15\ q16) + (q17\ q18) + (q19\ q20) + (q21\ q22) + (q23\ q24)]$  modulo 0x4000

#### High-Nibble P3 pari

$[(q0\ q1) + (q2\ q3) + (q4\ q5) + (q6\ q7) + (q8\ q9) + (q10\ q11) + (q12\ q13) + (q14\ q15) + (q16\ q17) + (q18\ q19) + (q20\ q21) + (q22\ q23) + (q24\ 00)]$  modulo 0x4000

Fin qui tutto bene, peccato che ... basti un RESET per scombinare tutto ! Ad ogni RESET sembrano cambiare gli effetti del comando! Ho scritto "sembrano" alla luce della prova relativa alla "**successione delle delta**" (*Vedi pag. 35, NdA*) :

**i byte RANDOM entrano a far parte della sommatoria delle WORD**

*High-Nibble P3 dispari*



### Fase 5 : Utilizzi di una Ins "custom"

Dopo che si sia ottenuto uno status 9000 mediante "Reset Bug" o con "Byte Bug" si aprono varie possibilità: generare ulteriori INS "custom" mediante utilizzo di INS 38, INS 36 (in particolare per cercare delle risposte "lunghe") oppure convertire la nostra INS 38 nella corrispettiva INS 40. Analizziamo il secondo caso che ai fini "pratici" potrebbe sembrare piu' interessante...

Ricordiamo che la nostra INS "custom" e' cosi formata (caso di status 9000 ottenuto con "BYTE bug"):

```
C1 38 21 B0 76 65 31 ... 00 09
```

Sostituiamo il byte 38 con il byte 40 ed avremo:

```
C1 40 21 B0 76 65 31 ... 00 09
```

Ovvero nella sua forma estesa :

```
C1 40 21 B0 76
65 31 D3 B2 97 A5 D2 86 81 DE 82 77 B6 85 8D A9
00 C0 AD 02 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00
```

Inviemo il comando **senza resettare** ed otterremo:

```
C1 40 21 B0 76
65 31 D3 B2 97 A5 D2 86 81 DE 82 77 B6 85 8D A9
00 C0 AD 02 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 09 [90 00]
```

#### ATTENZIONE !!!

I C1 40 cosi' costruiti, se inviati, possono causare l'alterazione o cancellazione dei dati memorizzati nella card, col rischio di renderla inutilizzabile !!!

(Si possono avere anche altri status)

In conclusione, l' EMM custom e' stato accettato.



## Welcome to Las Vegas - Probabilità di avere un parsing corretto

(Traduzione ed adattamento da uno studio condotto in terra straniera)

Dopo le voci insistenti di "programmini miracolosi" ho deciso di capire se fossero presenti dei bug non ancora scoperti dalla gente normale o se, effettivamente, tutte le informazioni fossero di dominio pubblico. Mi sono veramente sorpreso per il risultato ottenuto: le condizioni di parsing corretto pongono una restrizione molto forte sulla struttura delle INS generate in maniera casuale:

Facciamo l'ipotesi che le proprietà statistiche dei byte in chiaro e di quelli criptati siano le stesse (cioè che la SuperEncryption abbia una buona proprietà di diffusione).

Prendendo un comando lungo **n**, è interessante cercare di scoprire tutte le combinazioni di nanocomandi SEKA che possono generarsi nel comando stesso, in modo tale da avere un parsing corretto e quindi l'esecuzione da parte della card. Tutte le possibilità possono essere calcolate applicando ricorsivamente una "forma grammaticale" nella quale il simbolo **n** è espresso in **n → (n-1) & 0**, o in altri termini, un nano **Mx** può essere sostituito dal nano **(M-1)x** più il nano **0w**.

Applicando quanto detto a un nano qualunque "**Nx yy ... yy**" (nano + dati) e tenendo conto della non espandibilità del nano **0w**, possiamo costruire tutte le possibilità (NOTA: nella trattazione non saranno considerati i Nanocomandi Dx, Ex, Fx, ma con poche considerazioni aggiuntive possono comunque rientrare nella procedura).

Si definisce "**Type**" questa struttura, di conseguenza una INS **Type 110** conterrà i seguenti dati:

**1x mm 1y nn 0z**

Invece, per una INS **Type 100**, si ha:

**1x mm 0y 0z**

Consideriamo quest'ultima struttura, quante INS possono essere costruite con essa? Avendo 3 byte con un nibble fisso ed un byte a valori qualunque, risulterà:

$$T = 16^m * 256^{(n-m)}$$

Dove:

<b>m</b>	è il numero di Nanocomandi della <b>Type</b> ( 3 in questo caso )
<b>n</b>	è la lunghezza dell'INS
<b>(n - m)</b>	è il numero di byte a valori qualunque

L'espressione precedente non rende conto di tutte le possibili **permutazioni** dei nanocomandi. Questa proprietà di permutazione si definisce "**molteplicità**". Dovremo quindi calcolare le permutazioni con ripetizioni di **k** elementi (cioè i nanocomandi), avendone **ki** per ogni tipo:

$$PERM( k, (k_1, k_2, ..., k_n) ) = k! / ( (k_1!) * (k_2!) * ... * (k_n!) )$$

Nel caso trattato (**Type 110**) avremo:

$$\text{PERM}(3, (2, 1)) = 3! / (2! * 1!) = 3$$

Infatti i possibili risultati della permutazione sono "110", "101", "011". Con tutto quanto enunciato ora possiamo calcolare la probabilità di ogni **Type** su una INS generata casualmente e, conseguentemente, la probabilità di avere un parsing corretto. Nella seguente tabella si potranno osservare le probabilità per dati dell'INS di lunghezza variabile da 2 a 9 byte.

**LEN = 2**

Type	Count	Multip.	Total	%	% Parsing OK
1	4096	1	4096	6,25%	94,12%
00	256	1	256	0,39%	5,88%
Total	65536		4352	6,64%	

**LEN = 3**

Type	Count	Multip.	Total	%	% Parsing OK
2	1048576	1	1048576	6,25%	88,89%
10	65536	2	131072	0,78%	11,11%
000	4096	1	4096	0,02%	0,35%
Total	16777216		1179648	7,03%	

**LEN = 4**

Type	Count	Multip.	Total	%	% Parsing OK
3	268435456	1	268435456	6,25%	83,37%
20	16777216	2	33554432	0,78%	10,42%
11	16777216	1	16777216	0,39%	5,21%
100	1048576	3	3145728	0,07%	0,98%
0000	65536	1	65536	0,00%	0,02%
Total	4294967296		321978368	7,50%	

**LEN = 5**

Type	Count	Multip.	Total	%	% Parsing OK
4	68719476736	1	68719476736	6,25%	78,47%
30	4294967296	2	8589934592	0,78%	9,81%
21	4294967296	2	8589934592	0,78%	9,81%
200	268435456	3	805306368	0,07%	0,92%
110	268435456	3	805306368	0,07%	0,92%
1000	16777216	4	67108864	0,01%	0,08%
00000	1048576	1	1048576	0,00%	0,00%
Total	1,09951E+12		87578116096	7,97%	

**LEN = 6**

Type	Count	Multip.	Total	%	% Parsing OK
5	1,75922E+13	1	1,75922E+13	6,25%	74,08%
40	1,09951E+12	2	2,19902E+12	0,78%	9,26%
31	1,09951E+12	2	2,19902E+12	0,78%	9,26%
22	1,09951E+12	1	1,09951E+12	0,39%	4,63%
300	68719476736	2	1,37439E+11	0,05%	0,58%
210	68719476736	6	4,12317E+11	0,15%	1,74%
111	68719476736	1	68719476736	0,02%	0,29%
2000	4294967296	3	12884901888	0,00%	0,05%
1100	4294967296	6	25769803776	0,01%	0,11%
10000	268435456	5	1342177280	0,00%	0,01%
000000	16777216	1	16777216	0,00%	0,00%
Total	2,81475E+14		2,37482E+13	8,44%	

LEN = 7

Type	Count	Multip.	Total	%	% Parsing OK
6	4,50E+15	1	4,50E+15	6,25%	69,51%
50	2,81475E+14	2	5,63E+14	0,78%	8,69%
41	2,81475E+14	2	5,63E+14	0,78%	8,69%
32	2,81475E+14	2	5,63E+14	0,78%	8,69%
400	1,75922E+13	3	5,28E+13	0,07%	0,81%
310	1,75922E+13	6	1,06E+14	0,15%	1,63%
220	1,75922E+13	3	5,28E+13	0,07%	0,81%
211	1,75922E+13	3	5,28E+13	0,07%	0,81%
3000	1,09951E+12	4	4,40E+12	0,01%	0,07%
2100	1,09951E+12	12	1,32E+13	0,02%	0,20%
1110	1,09951E+12	4	4,40E+12	0,01%	0,07%
20000	68719476736	5	3,44E+11	0,00%	0,01%
11000	68719476736	10	6,87E+11	0,00%	0,01%
100000	4294967296	6	2,58E+10	0,00%	0,00%
0000000	268435456	1	2,68E+08	0,00%	0,00%
Total	7,20576E+16		6,48E+15	8,99%	

LEN = 8

Type	Count	Multip.	Total	%	% Parsing OK
7	1,15E+18	1	1,15E+18	6,25%	65,42%
60	7,21E+16	2	1,44E+17	0,78%	8,18%
51	7,21E+16	2	1,44E+17	0,78%	8,18%
42	7,21E+16	2	1,44E+17	0,78%	8,18%
33	7,21E+16	1	7,21E+16	0,39%	4,09%
500	4,50E+15	3	1,35E+16	0,07%	0,77%
410	4,50E+15	6	2,70E+16	0,15%	1,53%
320	4,50E+15	6	2,70E+16	0,15%	1,53%
311	4,50E+15	3	1,35E+16	0,07%	0,77%
221	4,50E+15	3	1,35E+16	0,07%	0,77%
4000	2,81475E+14	4	1,13E+15	0,01%	0,06%
3100	2,81475E+14	12	3,38E+15	0,02%	0,19%
2200	2,81475E+14	6	1,69E+15	0,01%	0,10%
2110	2,81475E+14	12	3,38E+15	0,02%	0,19%
1111	2,81475E+14	1	2,81E+14	0,00%	0,02%
30000	1,75922E+13	5	8,80E+13	0,00%	0,00%
21000	1,75922E+13	10	1,76E+14	0,00%	0,01%
11100	1,75922E+13	10	1,76E+14	0,00%	0,01%
200000	1,09951E+12	5	5,50E+12	0,00%	0,00%
110000	1,09951E+12	15	1,65E+13	0,00%	0,00%
1000000	68719476736	7	4,81E+11	0,00%	0,00%
00000000	4294967296	1	4,29E+09	0,00%	0,00%
Total	1,84467E+19		1,76E+18	9,55%	

LEN = 9

Type	Count	Multip.	Total	%	% Parsing OK
8	2,95E+20	1	2,95E+20	6,25%	61,58%
70	1,84E+19	2	3,69E+19	0,78%	7,70%
61	1,84E+19	2	3,69E+19	0,78%	7,70%
52	1,84E+19	2	3,69E+19	0,78%	7,70%
43	1,84E+19	2	3,69E+19	0,78%	7,70%
600	1,15E+18	3	3,46E+18	0,07%	0,72%
510	1,15E+18	6	6,92E+18	0,15%	1,44%
420	1,15E+18	6	6,92E+18	0,15%	1,44%
330	1,15E+18	3	3,46E+18	0,07%	0,72%
411	1,15E+18	3	3,46E+18	0,07%	0,72%
321	1,15E+18	6	6,92E+18	0,15%	1,44%
222	1,15E+18	1	1,15E+18	0,02%	0,24%
5000	7,21E+16	4	2,88E+17	0,01%	0,06%
4100	7,21E+16	12	8,65E+17	0,02%	0,18%
3200	7,21E+16	12	8,65E+17	0,02%	0,18%
3110	7,21E+16	12	8,65E+17	0,02%	0,18%
2210	7,21E+16	12	8,65E+17	0,02%	0,18%
2111	7,21E+16	4	2,88E+17	0,01%	0,06%
40000	4,50E+15	5	2,25E+16	0,00%	0,00%
31000	4,50E+15	10	4,50E+16	0,00%	0,01%
22000	4,50E+15	10	4,50E+16	0,00%	0,01%
21100	4,50E+15	30	1,35E+17	0,00%	0,03%
11110	4,50E+15	5	2,25E+16	0,00%	0,00%
300000	2,81475E+14	6	1,69E+15	0,00%	0,00%
210000	2,81475E+14	30	8,44E+15	0,00%	0,00%
111000	2,81475E+14	20	5,63E+15	0,00%	0,00%
2000000	1,75922E+13	7	1,23E+14	0,00%	0,00%
1100000	1,75922E+13	21	3,69E+14	0,00%	0,00%
10000000	1,09951E+12	8	8,80E+12	0,00%	0,00%
000000000	68719476736	1	6,87E+10	0,00%	0,00%
Total	4,72237E+21		4,79E+20	10,15%	

Diverse cose possono essere estrapolate dai numeri rappresentati in tabella: per prima cosa, si vede che la probabilità di avere un parsing corretto varia da 6.46% per INS con lunghezza di 2 byte al 10.15% per INS con lunghezza di 10 byte, un dato riscontrabile anche sperimentalmente. Un'altra importante proprietà che si ricava è la seguente:

*Per una INS di lunghezza (N+1) la probabilità di avere il Nano Nx è molto maggiore di ogni altra configurazione a parsing corretto.*

In conclusione, se assumiamo che la INS con Type che includono "1" "0/1" "0y" sono le più pericolose, ed escludendo la possibilità di generare cancellazioni multiple di chiavi e/o la rimozione di un provider (nano 25, quest'ultima ammissibile solo dal provider SEKA) è possibile stimare la proporzione tra la probabilità di produrre INS dagli effetti "positivi" e probabilità di produrre INS dagli effetti "distruttivi".

*Per esempio, un 80 xx xx xx... è circa 130 volte più probabile della cancellazione di una chiave ...*

E' finalmente "rivelato" il mistero della generazione dei nano e questo porta ad un grosso passo avanti, perché possiamo usare altri nanocomandi (diversi dai soliti 80, 41 o 21) che potranno aiutarci a capire meglio il funzionamento della card. La situazione e' analoga a quella del passaggio dal mondo non deterministico ad un piu' controllabile mondo "classico" dove il determinismo ci permette di prevedere il risultato delle nostre azioni.

*That's all folks !*

*Icon of Coil*

*Coming soon...*

*INS 36 - Metodi di ottenimento di risposte lunghe*

*- Procedura per tentativi (per risposte lunghe 1C)*

*- Procedura per tentativi "pilotati" per ottenere lunghezze >1C (uso della signature per "d2 d3")*

*Costruzione di una C1 38/40/3C a partire da una C1 36 "lunga"*

*- Problemi di instabilita' dello status 9000*

*- I rischi di "funny-bug 2"*

*Decrypt SE*

*- Probabile modello di funzionamento*

*- Il processo subito dall'ultimo ottetto come causa del bug9600*