How Do I Crack Satellite and Cable Pay TV?

Attacking the Digicipher 2 conditional access system used in millions of TV set-top-boxes in North America.

Chris Gerlinsky
@akacastor
chris@pdrnorth.com
Digital Television

Satellite
- Modulation:
  - DC2 QPSK ~27 Mbit
  - 8PSK Turbo FEC ~38 Mbit

Cable
- Modulation:
  - QAM 256 ~38 Mbit
- Out-of-band:
  - QPSK 2 Mbit

Video format:
- MPEG-2 or H.264 Transport Stream (MPEG TS)

Encryption:
- Digicipher 2 (not DVB standard)
MPEG Transport Stream
27 – 38 Mbit

188-byte packets
Categorized by 13-bit PID
(0 - 0x1FFF)

MPEG PES
Packetized Elementary Streams
64 Kbit – 19 Mbit
Video, audio

Service Information Tables
8-bit table ID (0 – 0xFF)
Up to 1024 bytes, with CRC32

Table 0x00 – (PID 0) Program Association Table
Table 0x01 – (PID 1) Conditional Access Table
Table 0x02 – Program Map Table

PAT contains list of programs: PID carrying PMT
PMT contains list of PIDs for video, audio, ECM

Table 0x40 – ECM40
Table 0x41 – ECM41
Table 0x95 – EMM95

ECM are sent in pairs
Cable: EMM are OOB
Genpix SkyWalker-1 USB satellite interface

https://bitbucket.org/updatelee/v4l-updatelee
Hauppauge HVR 950Q ATSC / QAM USB interface
Using dvbsnoop to view PMT PID 0x129

$ dvbsnoop -if log-11959.ts -s ts -tssubdecode 0x129

dvbsnoop V1.4.50 -- http://dvbsnoop.sourceforge.net/

<table>
<thead>
<tr>
<th>TS-Packet: 00000001</th>
<th>PID: 297 (0x0129), Length: 188 (0x00bc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>from file: 2016-05-21-full-log-11959.ts</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>TS-Packet: 00000001</td>
<td>PID: 297 (0x0129), Length: 188 (0x00bc)</td>
</tr>
<tr>
<td>from file: 2016-05-21-full-log-11959.ts</td>
<td></td>
</tr>
</tbody>
</table>

Sync-Byte 0x47: 71 (0x47)
Transport_error_indicator: 0 (0x00)  [= packet ok]
Payload_unit_start_indicator: 1 (0x01)  [= Packet data starts]
transport_priority: 1 (0x01)
PID: 297 (0x0129)  [= ]
transport_scrambling_control: 0 (0x00)  [= No scrambling of TS packet payload]
adaptation_field_control: 1 (0x01)  [= no adaptation_field, payload only]
continuity_counter: 11 (0x0b)  [= (sequence ok)]

Payload: (len: 184)

--- pointer_field: 0 (0x00)
--- Section table: 2 (0x02)  [= Program Map Table (PMT)]
Payload_unit_start_indicator: 1 (0x01)  [= Packet data starts]
transport_priority: 1 (0x01)
PID: 297 (0x0129)  [= ]
transport_scrambling_control: 0 (0x00)  [= No scrambling of TS packet payload]
adaptation_field_control: 1 (0x01)  [= no adaptation_field, payload only]
continuity_counter: 11 (0x0b)  [= (sequence ok)]

Payload: (len: 184)

--- pointer_field: 0 (0x00)
--- Section table: 2 (0x02)  [= Program Map Table (PMT)]

TS sub-decoding (1 packet(s) stored for PID 0x0129):

--- Transport_data:
--- PMT-decoding.
--- Table_ID: 2 (0x02)  [= Program Map Table (PMT)]
--- section_syntax_indicator: 1 (0x01)  
--- fixed '0': 0 (0x00)
--- reserved_1: 0 (0x00)
--- Section_length: 38 (0x0026)
--- Program_number: 15 (0x000f)
--- reserved_2: 0 (0x00)
--- Version_number: 10 (0x0a)
--- current_next_indicator: 1 (0x01)  [= valid now]
--- Section_number: 0 (0x00)
--- Last_section_number: 0 (0x00)
--- reserved_3: 0 (0x00)
--- PCR_PID: 272 (0x0110)
--- reserved_4: 0 (0x00)
--- Program_info_length: 5 (0x0005)
--- MPEG_DescriptorTag: 9 (0x09)  
--- descridtor_length: 4 (0x04)
--- CA_system_ID: 182 (0x4749)  
--- CA_PID: 297 (0x0129)
--- Stream_type loop:
--- Stream_type: 128 (0x80)  
--- Elementary_PID: 272 (0x0110)
--- ES_info_length: 3 (0x0003)
--- MPEG_DescriptorTag: 10 (0x0a)  
--- descridtor_length: 4 (0x04)
--- IS0639_language_code: eng
--- Audio_type: 0 (0x00)  [= undefined]
--- CRC: 1824817825 (0x6cc48625)

TS sub-decoding (1 packet(s) stored for PID 0x0129):

--- Transport_data:
--- PMT-decoding.
--- Table_ID: 2 (0x02)  [= Program Map Table (PMT)]
--- section_syntax_indicator: 1 (0x01)  
--- fixed '0': 0 (0x00)
--- reserved_1: 0 (0x00)
--- Section_length: 38 (0x0026)
--- Program_number: 15 (0x000f)
--- reserved_2: 0 (0x00)
--- Version_number: 10 (0x0a)
--- current_next_indicator: 1 (0x01)  [= valid now]
--- Section_number: 0 (0x00)
--- Last_section_number: 0 (0x00)
--- reserved_3: 0 (0x00)
--- PCR_PID: 272 (0x0110)
--- reserved_4: 0 (0x00)
--- Program_info_length: 5 (0x0005)
--- MPEG_DescriptorTag: 9 (0x09)  
--- descridtor_length: 4 (0x04)
--- CA_system_ID: 182 (0x4749)  
--- CA_PID: 297 (0x0129)
--- Stream_type loop:
--- Stream_type: 128 (0x80)  
--- Elementary_PID: 272 (0x0110)
--- ES_info_length: 3 (0x0003)
--- MPEG_DescriptorTag: 10 (0x0a)  
--- descridtor_length: 4 (0x04)
--- IS0639_language_code: eng
--- Audio_type: 0 (0x00)  [= undefined]
--- CRC: 1824817825 (0x6cc48625)
**DC2 Service Information Tables**

SCTE 65: Service Information Delivered Out-Of-Band For Digital Cable Television

### DC2 tables

- VCT – Virtual Channel Table
- NIT – Network Information Table
- NTT – Network Text Table
- MGT – Master Guide Table

### Many DVB standard tables are not used:

### DVB equivalent

- BAT – Bouquet Association Table
- SDT – Service Description Table
- EIT – Event Information Table
Internal connections to CableCard slot for QPSK output
SCTE 55-1 Decoding Software

MPEG Transport Stream → Descrambler → MPEG Decoder & TV

Working Key
- 56-bit DES
- Lifetime: 133 ms to 1 second

Program Key
- 56-bit DES
- Lifetime: 1 day or 1 PPV event

Category Key
- 56-bit DES
- Lifetime: 1 month, or whatever

Seed Keys
- 3x 56-bit DES
- Lifetime: STB lifespan
- Stored in battery-backed RAM

ECM40
- Incrementing Frame Count value

ECM41
- Access controls
- Encrypted Program Key

EMM95
- Access rights
- Encrypted Category Key

Each channel has unique keys

All authorized STB share same key

Each STB has unique seed keys

Pair of ECMs for each channel, all STB process the same ECM

Individual message is sent to each STB
Passive monitoring:
Using two SPI slave peripherals, MOSI is used for receiving, MISO is left unconnected.

Acting as master:
Jumpers to hold STB /RST and to connect MISO, to act as SPI master

STB controller
SPI master

ACP
SPI slave

Xmega128

SPI #1
slave / master

SPI #2
slave
SPI data log

< 80 04 00 04
> 55 04 14 10 20 FC FD 00 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 61

|| Next category epoch
|| Current category epoch
Provider ID

< 80 05 00 05
> 55 05 09 FE 16 00 10 D3 99 FA 20 00 64

UA# - Unit Address of STB

< 80 12 00 12
> 55 12 13 00 00 02 04 03 04 02 04 03 04 0C 6D 05 14 0D FC 00 00 00 80

|| Keyselect for next category epoch
|| Keyselect for current category epoch

< 80 13 02 00 00 11
> 55 13 22 00 C0 4C 61 87 11 00 51 A1 00 DF 23 80 E0 01 98 D1 3F 00 80 41 00 08 20 48 00 00 00 D0 41 15 81 00 44 B1

Subscription tiers

> 55 87 13 00 11 22 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

|| Next program key epoch
|| Current program key epoch
|| Encryption mode – ZK / FP / FW S
|| Service ID of this channel
|| 00 = Tuner0, 80 = Tuner1
07 = Response to Cmd07
87 = Unprompted response due to change in access rights (channel change etc)
Software disassembly (IDA)
Sim65:
a 65c02 simulation system

http://www.wsxyz.net/sim65/

Customizations for ACP
Personalization software is used during manufacturing to initialize the device – setup UA#, keys, etc. After initialization is complete, access to personalization software is disabled.

Application software always running after STB leaves factory. This contains all the conditional access functionality.

Vectors area contains reset and interrupt addresses.
Task switcher

**Task 0**
Priority: 1
Idle Task

**Task 1**
Priority: 5
ECM40 [#1]

**Task 2**
Priority: 5
ECM40 [#2]

**Task 3**
Priority: 4
ECM41 [#1]

**Task 4**
Priority: 4
ECM41 [#2]

**Task 5**
Priority: 3
EMM

**Task 6**
Priority: 2
TVpass

**Task 7**
Priority: 2
SPI interface

Decrypt two programs (dual tuner support)
Hardware Peripherals

**SPI slave**
- RX RAM buffer
- TX RAM buffer
- Flag set when transfer complete

**DES hardware**
- Hardware encrypt/decrypt
- 56 bit key, 64 bit data
- Standard and custom modes

**Transport Stream PID filter**
- SPI selects ECM PID
  - Tables placed in RAM and RX flag set
  - SPI selects PIDs to decrypt
  - Software selects Provider ID filter for ECM

**TS descrambler**
- Software sets working key
- Selected PIDs are decrypted
- DES standard and custom modes
System analysis

✔ MPEG Transport stream – ECM & EMM logging
✔ SPI bus – logging and understanding of ACP messages
✔ ROM dump (entire ACP firmware)
✔ Software disassembly and simulation
✔ Keys: Fixed keys found in ROM

➡ Understanding of ECM and EMM algorithms

✗ Keys: Seed keys, category keys, program keys only exist in RAM

Possible bit errors from optical extraction ruled out by valid checksum on key area
ECM40 – Working Key

Three sequential ECM40 messages (133 ms apart)

Frame Count is used to generate Working Key

Service ID  Frame Count  HW  Crypt Mode

Frame Count is used to generate Working Key
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>41 40 41 10 20 00 2D 80 07 75 6E 2A CE 13 09 E3</td>
<td>8-byte Initialization Vector</td>
</tr>
<tr>
<td>40 9E F1 9F E7 76 9A 7E BC 00 00 00 08 05 00 01</td>
<td>Hardware crypto select</td>
</tr>
<tr>
<td>35 03 00 00 06 03 00 00 83 03 00 00 A9 03 00 00</td>
<td>Bitmapped (bit 6 indicates PPV)</td>
</tr>
<tr>
<td>C0 03 00 02 54 03 01 00 AB 69 1C D1 12 A8 CE D5</td>
<td>Number of acceptable tiers</td>
</tr>
<tr>
<td>1020</td>
<td>Provider ID</td>
</tr>
<tr>
<td>2D8007</td>
<td>Service ID</td>
</tr>
<tr>
<td>75</td>
<td>Category Epoch</td>
</tr>
<tr>
<td>6E</td>
<td>Program Epoch</td>
</tr>
<tr>
<td>2ACE13</td>
<td>Validity start</td>
</tr>
<tr>
<td>09E340</td>
<td>Valid period (24 hr.)</td>
</tr>
<tr>
<td></td>
<td>8-byte Encrypted Program Key</td>
</tr>
</tbody>
</table>
Decryption using Fixed Working Key

Encrypted transport stream
\[\rightarrow\]
DES descrambler
\[
\text{decrypt TS using fixed working key}
\]
\[\rightarrow\]
Decrypted MPEG transport stream

Decryption using Fixed Program Key

Encrypted transport stream
\[\rightarrow\]
ECM40 processing
\[
\text{use fixed program key to decrypt working key from ECM40}
\]
\[\rightarrow\]
DES descrambler
\[
\text{decrypt TS using working key}
\]
\[\rightarrow\]
Decrypted MPEG transport stream
Glitch payload - RAM intercept loop
(addresses have been changed)

.SPI_RAM:4000 05 .BYTE 5 ; cmd05
.SPI_RAM:4001 40 .BYTE $40 ; length 0x40 bytes
.SPI_RAM:4002 ; ------------------------------------------------------------------------
.SPI_RAM:4002 executable_command: ; pad the beginning of the packet with some NOPs to give as big of an entry point as possible
.SPI_RAM:4002 EA NOP
.SPI_RAM:4003 EA NOP
.SPI_RAM:4004 EA NOP
.SPI_RAM:4005 EA NOP
.SPI_RAM:4006 EA NOP
.SPI_RAM:4007 EA NOP
.SPI_RAM:4008 EA NOP
.SPI_RAM:4009 EA NOP
.SPI_RAM:400A EA NOP
.SPI_RAM:400B EA NOP
.SPI_RAM:400C EA NOP
.SPI_RAM:400D A2 00 LDX #0
.SPI_RAM:400F ; CODE XREF: SPI_RAM:4018
.SPI_RAM:400F copy_intercept_loop: ; copy intercept loop from SPI_RAM to main RAM so it isn’t overwritten by next packet
.SPI_RAM:400F BD 2C 40 LDA $402C,X ; STA $1140,X
.SPI_RAM:4010 8D 40 11 ; INX
.SPI_RAM:4011 8D 2D 40 ; CPX #$20 ; ' ' ; copy_intercept_loop ; copy code from SPI_RAM to main RAM
.SPI_RAM:4012 8D 40 11 LDA #$4A5 ; ' ' ; setup response at 3114: AB 00 AB
.SPI_RAM:4013 8D 1D 41 STA $4114 ; ' ' ; setup response at 3114: AB 00 AB
.SPI_RAM:4014 8D 15 41 LDA #0 STA $1145 ; ' ' ; jump to code that was copied from SPI_RAM to main RAM
.SPI_RAM:4015 8D 16 41 LDA #$4A6 ; ' ' ; jump to code that was copied from SPI_RAM to main RAM
.SPI_RAM:4016 8A 00 LDA #0
.SPI_RAM:4017 4C 40 11 JMP $1146
.SPI_RAM:4018 ; ------------------------------------------------------------------------
.SPI_RAM:401C ; send packet we propped in SPI_RAM (calling routine from the middle to skip over undesired behaviour)
.SPI_RAM:401C #244 JSR $A224 ; reset interrupt flag and clear interrupt mask
.SPI_RAM:401D #252 JSR $A502 ; clear bit indicating response has been sent (no further packet waiting to be processed)
.SPI_RAM:401E #E1 JSR $81
.SPI_RAM:401F #04 JSR $A356 DA ; push 0x00 on stack
.SPI_RAM:4020 #E1 JSR $A537 AE 21 ; push address of flag for us to be awaken on (received SPI packet)
.SPI_RAM:4021 #2 JSR $A359 DA ; BRK #2 - sleep until SPI packet is received
.SPI_RAM:4022 #0 JSR $A35A AE 09 LDA BRK
.SPI_RAM:4023 #0 JSR $A35C 00 ; BRK #2 - sleep until SPI packet is received
.SPI_RAM:4025 JSR executable_command ; call subroutine received in RAM
.SPI_RAM:4026 BRA interrupt_loop ; loop to receive next executable command
Bit errors found in optical ROM dump

Result of ROM extraction from single sample

Errors marked in red

*By the bit*

262,144 bits total
105 bit errors
= 0.04% error rate (99.96% accuracy)

*By the byte*

32,768 bytes total
104 byte errors
= 0.32% error rate (99.68% accuracy)

Only one byte had more than a single flipped bit.
Key extraction from RAM

- All keys are stored only in RAM.
- RAM is battery-backed.
- Until now, glitching has been done on chips desoldered and put on test board.
  - This isolates the 16 VCC pins from each other and other components.
- RAM contents lost when chip is desoldered.
- If we can successfully glitch ACP with RAM intact then all keys can be read.
  - Category and Seed Keys are desirable.

Adapt glitcher to work on ACP in-circuit

or

Remove ACP from STB without losing RAM contents
dental pick
razor blade
cut to shape using Dremel tool
screwdriver handle
Transport Stream scrambling

188-byte TS packet

<table>
<thead>
<tr>
<th>47 12 34 00</th>
<th>4-byte header</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 01 01 01</td>
<td>02 02 02 02</td>
</tr>
<tr>
<td>05 05 05 05</td>
<td>06 06 06 06</td>
</tr>
<tr>
<td>09 09 09 09</td>
<td>07 07 07 07</td>
</tr>
<tr>
<td>13 13 13 13</td>
<td>08 08 08 08</td>
</tr>
<tr>
<td>17 17 17 17</td>
<td>09 09 09 09</td>
</tr>
<tr>
<td>21 21 21 21</td>
<td>10 10 10 10</td>
</tr>
<tr>
<td></td>
<td>11 11 11 11</td>
</tr>
<tr>
<td></td>
<td>12 12 12 12</td>
</tr>
</tbody>
</table>

23 blocks of 8 bytes encrypted data

Analyzing scrambling

- Flip bits in ciphertext to observe results in decrypted result (CBC/ECB/OFB modes)
  - one bit flipped corrupts one 8-byte block plus corresponding bit in next block: CBC mode
- Observe timing of decryption, look for changes in timing due to algorithm differences
  - changes to algorithm such as number of rounds should have effect on timing
- Use DES HW as oracle
  - send test data through ACP, controlling data, key, and all H/W registers
- Use DES weak keys (all 0 or all 1) and observe if behaviour matches standard DES
  - encryption and decryption operations are equivalent when using key made of all 0 or all 1 bits
DES "weak keys"

**Weak key**

- DES key: 00 00 00 00 00 00 00 00
- Plaintext: 01 23 45 67 89 AB CD EF
- Encrypted: 61 7B 3A 0C E8 F0 71 00

**Non-weak key**

- DES key: 80 00 00 00 00 00 00 00
- Plaintext: 01 23 45 67 89 AB CD EF
- Encrypted: F2 C4 69 25 D1 0D 86 BD

With DES key of all zero bits, encryption and decryption have same effect.

If even one bit is nonzero, encryption and decryption produce different effects.
Cracking Hardware Customization: DES key XOR taps

- Using H/W DES engine: Decrypt 23 blocks of 00 data, with 00 key, and customization enabled
- In software: Try decrypting 00 data with all combinations of key having 1 or more bits flipped
- Check for decryption matching any of the 23 blocks
- Positive results seen within seconds – some XOR masks have as few as 3 bits set

```c
for( flip[0] = -1; flip[8] < 56; flip[0]++ ) { // loop through all bit positions
                            for( i = 0; i < 16; i++ ) {
                                if( flip[i] == 0 ) {
                                    // XOR bits that are selected for this test
                                    des_data[i] = des_key[flip[i]/7] ^ des_data[i]; // the +1 is because bit 0 is parity bit (unused)
                                }
                            }
                            if( des_data[0] == encrypted_data ) { // check des_data[] against encrypted blocks for a match
                                for( n = 0; n < 16; n++ ) {
                                    // output the key
                                }
                            }
                        }
                    } // flip[9]
                } // flip[8]
            } // flip[7]
        } // flip[6]
    } // flip[5]
} // flip[4]
```

Cracking Hardware Customization: DES data XOR taps

DES hardware 8-bit register:

00  = standard DES
01 – FF = customization

XOR gates

DES data supplied to hardware

DES data used for decryption
Decrypted MPEG transport stream

Encrypted transport stream

Included in transport stream are tables for conditional access

softcam
software conditional access module

EMM processing
use seed keys to decrypt category key from EMM95

ECM41 processing
use category key to decrypt program key from ECM41

ECM40 processing
use program key to decrypt working key from ECM40

DES descrambler
with h/w customization (key XOR)

video playable using common MPEG decoders and software players (example: VLC)
VALID

-20dBFS
Source
16:9
1080i59

MCR'S THIS IS THE UFC 204 PAY-PER-VIEW PATH
CHANNEL 1 & 2 1KHZ TONE
CHANNEL 3 & 4 400 HZ TONE
Weaknesses

- Relatively old technology – easier for invasive analysis today
  - TQFP100 package easy to deal with compared to modern alternatives
- Voltage glitching
- Von Neumann architecture, no strong MMU protection
- No possibility for code updates for countermeasure purposes
- Hardware crypto customizations are simple
Strengths

- Key handling and decryption contained within single chip makes it difficult to do key sharing.
- Fast working key change interval (133 ms) makes key sharing difficult.
- No possibility for code update means nowhere to write code for permanent backdoor.
- Internal CLK prevents clock glitching.
- Dead addresses to prevent linear readout of keys.
- Personalization ROM appears to be inaccessible.
- Keys kept only in RAM – must maintain battery backup at all times!
  - Keys appear non-rewritable, preventing cloning units.
- No group keys for EMMs – all unit addressing is to individual units.
  - Must pull keys from a subscribed box in order to get active keys.
- Software appears generally well-designed and written.
- Although DES is used, EMMs are signed using 3 DES keys.
  - Multiple rounds of DES are used, increasing brute force complexity.
Thank you for choosing Pay Per View.