Reverse engineering hardware for software reversers: studying an encrypted external HDD

Joffrey Czarny & Raphaël Rigo / AGI / TX5IT

2015-10-02 / Hardwear.io
Reverse engineering hardware: studying an encrypted HDD
# Introduction

**Why study encrypted hard drives?**

- Initially: audit need inside Airbus Group
- Previous work revealed vulnerabilities
- Discover how to analyze hardware based on microcontrollers

**Previous *epic fails* on this type of HW**

- Corsair Padlock: data not encrypted, reachable without PIN (2008)
- Corsair Padlock 2: brute-forceable PIN (2010)
- WD Passport (yesterday’s talk by Gunnar Alendal and Christian Kison)

**End goal**

- Analyze the actual level of protection of user data
  \[\Rightarrow\] Validate security and cryptography implementations inside the enclosure
Introduction

This talk's objectives:

- Describe the study of an external encrypted HDD:
  - Explain the methodology in details
  - Show our various failures
  - Give leads to continue the analysis

Case study: Zalman ZM-VE400

- Enclosure: HDD is replaceable
- Optional AES-256 XTS encryption (physical keyboard)
- Can “mount” ISO as USB optical drive
- *Really* a rebranded iodd 2541
Reverse engineering hardware: studying an encrypted HDD

## Context, first results

### General security checks

- Verify basic crypto properties:
  - ECB mode? statistical tests OK?
  - Fixed key?
- More tests, to verify the key is not derived directly from the PIN:
  - The same PIN, on 2 different enclosures, **must** lead to different encryption
  - The same PIN, on the same enclosure, **must** lead to different encryption
- Secret material (keys, hashes) *should* be stored in tamper resistant hardware

### VE400 results

- Basic crypto properties: OK
- Encryption does **not** depend on enclosure: *an encrypted HDD put in a new Zalman enclosure can be accessed with the right PIN*
- Activating encryption uses 10 sectors at the end of the HDD:
  - Not usable anymore
  - Contain a *blob* of 768 bytes, of high entropy, twice
## Going forward

### Important result: design failure

Everything needed to decrypt data is stored on the HDD itself.  
⇒ Efficient attacks are possible (*bruteforce*, key recovery)

### New end goal

Understand the blob stored at the end of the disk: its data and its format, to implement an offline attack

### How?

First by trying to access the *firmware* and/or by analyzing communications.  
Firmware updates are encrypted, so we need to attack the hardware
## Hardware analysis

### PCB analysis
- Components identification
- Traces and vias identification

⇒ Logical view

### Flash memories study
- Identify communication buses
- Flash content recovery

⇒ Flash content analysis (hopefully cleartext code)
PCB: component identification 1/2

**PCB: front side**

- *System on Chip (SoC)* Fujitsu MB86C311 USB3-SATA
- SPI flash EN25F80
- PIC32MX 150F128D microcontroller
PCB: component identification 2/2

PCB: back size
- SPI flash EN25F80
SoC and microcontroller

<table>
<thead>
<tr>
<th>Fujitsu MB86C311</th>
</tr>
</thead>
<tbody>
<tr>
<td>● USB3↔SATA controller</td>
</tr>
<tr>
<td>● AES-256 XTS encryption</td>
</tr>
<tr>
<td>● ARM core</td>
</tr>
<tr>
<td>● Internal ROM and external SPI firmware support (encrypted?)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PIC32MX 150F128D</th>
</tr>
</thead>
<tbody>
<tr>
<td>● MIPS32 CPU (with MIPS16e support)</td>
</tr>
<tr>
<td>● 128 Ki of internal flash</td>
</tr>
<tr>
<td>● 32 Ki of RAM</td>
</tr>
<tr>
<td>● Supports ICSP and EJTAG</td>
</tr>
<tr>
<td>● Protection bits to disable external access</td>
</tr>
</tbody>
</table>
PCB: traces analysis (1/5): Hobo mode with GIMP
PCB: traces analysis (2/5): getting real with PCBRE [5]
Reverse engineering hardware: studying an encrypted HDD

PCB: traces analysis (3/5): leveling up: optical microscope
PCB: traces analysis (4/5): level cap: X-rays
PCB: traces analysis (5/5)

In the end

- One flash dedicated to the USB-SATA controller (SoC)
- One flash dedicated to the PIC32
- One link between the SoC and the PIC, (partially) shared with the SoC flash
Reverse engineering hardware: studying an encrypted HDD

PCB: logic view

Core USB-SATA
- USB
- Fujitsu MB86C311A
- SATA
- flash

Screen/Keyboard
- PIC32 MX150F128D
- LCD screen
- Keyboard

What’s inside the flash chips?
Maybe the code is in cleartext?
⇒ Let’s get their contents!
Flash content recovery (1/2)

Reading flash content

- SPI
- Chip desoldering needed to avoid interferences
- Interface using a SOIC↔DIP adapter to keep the board working
Reverse engineering hardware: studying an encrypted HDD

Flash content recovery (2/2)

SPI tools

- GoodFET with goodfet.spiflash (recommended)
- Bus Pirate
- Raspberry Pi with spidev

Results: flashes content

USB-SATA controller:
- Plaintext configuration data (USB descriptors, etc.)
- Code, encrypted

PIC32 microcontroller:
- A font, for the LCD screen
- Code, encrypted
Results

Code access: fail
All the code is encrypted, so we cannot reverse engineer the firmware

What can we do now?
As in network reversing, we will analyze communications (black box)

How?
By using a logic analyzer to capture communications
Reverse engineering hardware: studying an encrypted HDD

Hardware and probe placement

Core USB–SATA

USB

Fujitsu MB86C311A

SATA

SPI

flash

Screen/Keyboard

PIC32 MX150F128D

LCD screen

SPI

Keyboard

flash

PROBES

Saleae Logic Pro 16 logic analyzer
Reverse engineering hardware: studying an encrypted HDD

PCB traces and components pinout
Probe placement
Reverse engineering hardware: studying an encrypted HDD

Screenshot
Analyzing flash SPI communications

**USB-SATA/PIC to flash**
- Placing the 4 probes: simply on flash pins
- SPI decoding parameters: “standard” (cf. datasheet)
- Sampling speed: 50MS/s **min**, 100MS/s recommended (25MHz quartz)

**Post-treatment**
- CSV export of decoded SPI data
- Ruby script to interpret flash commands:
  - Text display
  - Binary dump rebuilding

**Results**
- PIC never writes to its external flash
- *USB-SATA controller writes data when the PIN is validated*
Reverse engineering hardware: studying an encrypted HDD

## Analyzing SoC ↔ PIC communications

<table>
<thead>
<tr>
<th>USB-SATA controller ↔ PIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Probes placement: on the SOC flash pins (cf. PCB traces)</td>
</tr>
<tr>
<td>- Sampling speed: 50MS/s <strong>min</strong>, 100MS/s recommended</td>
</tr>
<tr>
<td>- Protocol: <em>unknown</em></td>
</tr>
</tbody>
</table>

### Post-treatment

**SPI based protocol:**
- Low level decoding with Saleae, then CSV export
- Application-layer data must be reversed engineered
Custom protocol

Reverse engineering

- Preambles: AA AA AA AA 55 (SoC → PIC) and A5 A5 A5 5A (PIC → SoC)
- *Type, Length, Value*
- Frames are numbered and acknowledged
- Unknown 16bits checksum

Ruby script to decode data from the CSV produced by Saleae

Decoded example: PIN request

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00000000</td>
<td>SoC-&gt;PIC</td>
<td>T: 0x33, ID: 0x14</td>
</tr>
<tr>
<td>0.00003861</td>
<td>PIC-&gt;SoC</td>
<td>RESP: 0x14</td>
</tr>
</tbody>
</table>
Reverse engineering hardware: studying an encrypted HDD

Summary: communication sequence
Reverse engineering hardware: studying an encrypted HDD

Summary: communication sequence
Reverse engineering hardware: studying an encrypted HDD

Summary: communication sequence

Core USB–SATA

USB

Fujitsu MB86C311A

SATA

flash

PIN request

Screen/Keyboard

PIC32 MX150F128D

SPI

Keyboard

SPI

flash

LCD screen
Reverse engineering hardware: studying an encrypted HDD

Summary: communication sequence
Reverse engineering hardware: studying an encrypted HDD

Summary: communication sequence

![Diagram of communication sequence between USB-SATA and Screen/Keyboard systems]
Reverse engineering hardware: studying an encrypted HDD

Summary: communication sequence

Core USB–SATA

- USB
- Fujitsu MB86C311A
  - BLOB
- SATA
  - BLOB
  - flash

Screen/Keyboard

- PIC32 MX150F128D
  - SPI
  - flash
  - BLOB
- LCD screen
- Keyboard

H(PIN)
Reverse engineering hardware: studying an encrypted HDD

Summary: communication sequence

Core USB–SATA

USB → H(PIN) → Validation → BLOB

SATA → SPI → flash

Screen/Keyboard

PIC32 MX150F128D

SPI → LCD screen

Keyboard → flash
Reverse engineering hardware: studying an encrypted HDD

**Summary: communication sequence**

Diagram showing the communication sequence between USB, SATA, BLOB, Keys, Fujitsu MB86C311A, SPI, flash, PIC32 MX150F128D, LCD screen, Keyboard, and Screen/Keyboard.
Reverse engineering hardware: studying an encrypted HDD

Summary: communication sequence

Core USB–SATA

- USB
- Fujitsu MB86C311A
- SATA
- flash

Screen/Keyboard

- PIC32 MX150F128D
- LCD screen
- Keyboard
- flash

Keys?
Summary: communication sequence
Reverse engineering hardware: studying an encrypted HDD

Summary: communication sequence

Core USB–SATA

USB

Fujitsu MB86C311A

SATA

SATA BLOB

Keys ?

Reading

flash

Keys ?

Screen/Keyboard

PIC32 MX150F128D

FLASH

SPI

Keyboard

LCD screen
Reverse engineering hardware: studying an encrypted HDD

Summary: communication sequence

![Diagram of communication sequence between USB, SATA, Fujitsu MB86C311A, PIC32 MX150F128D, LCD screen, and keyboard. The diagram shows the flow of information through decryption, keys, and connections between hardware components.]
And now?

Remaining questions

- Can we do a hardware bruteforcer? (PIC+Keyboard emulator)
  - No, because the hash algorithm is unknown
- What is inside the block at 0x1000 in the SoC flash?

Flash block at 0x1000

Properties:
- Written when:
  - Enabling encryption
  - Entering a valid PIN
- Erased when encryption is disabled
- Contains 3 different blocks of data of high entropy:
  1. 512 bits, AES-256-XTS key 1, encrypted?
  2. 512 bits, AES-256-XTS key 2, encrypted?
  3. SHA256 of previous data (1 and 2)
Reverse engineering hardware: studying an encrypted HDD

Designing an attack

Hypothesis
The block at 0x1000 seems to contain AES-XTS encryption keys, in an encrypted or obfuscated form

Implications?
Can we use this block to mount an attack?

The idea
Assuming the block at 0x1000 contains decryption keys:
- We will try to keep the one of the target drive intact, in the flash . . .
- while validating the PIN against a chosen blob, stored on the HDD
Reverse engineering hardware: studying an encrypted HDD

Theoretical steps

TARGET disk

SoC USB–SATA → Flash → HDD

ATTACK disk

SoC USB–SATA → Flash → HDD

SPI

DATA

G BLOB

A BLOB

PIN: 3333
Reverse engineering hardware: studying an encrypted HDD

Theoretical steps

TARGET disk

SoC USB–SATA

Flash

G keys

HDD

DATA

A BLOB

ATTACK disk

SoC USB–SATA

Flash

A keys

HDD

DATA

A BLOB

target blob crushing

2015-10-02 / Hardwear.io
Theoretical steps

1: Extraction

2: Write status register
   => read only

TARGET disk

SoC USB-SATA

Flash

G keys

HDD

DATA

A BLOB

GoodFET

Flash

G keys
Theoretical steps

Reverse engineering hardware: studying an encrypted HDD
Reverse engineering hardware: studying an encrypted HDD

Theoretical steps

- SoC USB-SATA
- Flash RO
  - G keys
- SPI
- HDD
  - DATA
  - A BLOB

TARGET disk
Theoretical steps

```
TARGET disk

SoC USB-SATA

Flash RO
  G keys

SPI

HDD
  DATA

PIN ?

PIC32 Screen Keyboard

A BLOB
```
Theoretical steps
Theoretical steps

TARGET disk

SoC USB-SATA

H(3333)

A BLOB

VALID!

SPI

Flash
RO

G keys

HDD

DATA

A BLOB
Theoretical steps

TARGET disk

SoC USB-SATA

Flash
RO
G keys

A keys

SPI:
write: silent fail

HDD

DATA

A BLOB

2015-10-02 / Hardwear.io
Reverse engineering hardware: studying an encrypted HDD

Theoretical steps

TARGET disk

SoC USB–SATA

Flash RO

G keys

SPI:
reading keys

HDD

DATA

A BLOB
### In practice

#### First fail

The flash *status register* is reset to 0 during startup

#### Attack, second version

The flash is put in read only after startup:

1. Connect the enclosure
2. Unplug flash
3. Put it in read only using GoodFET
4. Plug it back
5. Continue the attack: enter the known PIN

#### Final result

**Fail.** PIN code is not valid (*Not match on screen*)

⇒ There’s probably an unidentified check
Final attack: demo
Conclusion

**Encrypted data security**

The whole security relies on:
- The security of the blob at the end of the disk
- The security of the block at 0x1000 in the flash

⇒ Everything relies on the fact that the Fujitsu firmware is “secret”

**iodd’s feedback (original board dev)**

Firmware evolution (version 077):
- PIN hash is now non-deterministic

The rest is not fixable:
- Customer support choice: data can survive broken enclosure
- Opaque handling of the blob at the end of the HDD: binary code provided by Fujitsu
Reverse engineering hardware: studying an encrypted HDD

Conclusion: going further

Access the code of the USB-SATA controller

- Find a JTAG? (unlikely)
- The firmware encryption is the same on all chips:
  - “Buy” the SDK? (probable NDA)
  - Find someone generous ;)

Emulate the SoC SPI flash

- Allows subtle modifications of block 0x1000
- Try blind ARM code modifications

Dump PIC32 code

Use semi-invasive attack to reset protection fuse

⇒ Hardware bruteforcer by emulating the whole keyboard/screen part
Questions?
References

## Blob comparison

### bloc ssd

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>6E D1 40 A2 74 48 51 93 D1 5B 96 13 18 25 F7 67</td>
<td><a href="mailto:n.@.tHQ">n.@.tHQ</a>. .[...%.g</td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>CF 73 BB 45 0A 4F 02 11 35 34 C9 39 45 31 BE 44</td>
<td>.s.E.O.. 54.9E1.D</td>
<td></td>
</tr>
<tr>
<td>0020</td>
<td>03 93 32 E1 8A 64 69 2E D6 1B 21 9F A5 51 88 8C</td>
<td>...2..di...!0..</td>
<td></td>
</tr>
<tr>
<td>0030</td>
<td>16 57 FF 71 32 CA E8 82 69 68 6A 3A DE 77 EC 06</td>
<td>.W.q2... ihj:w...</td>
<td></td>
</tr>
<tr>
<td>0040</td>
<td>DB D9 35 2F 47 32 FC D8 30 9F 06 B7 87 C0 F3 87</td>
<td>.5/G2. 0........</td>
<td></td>
</tr>
<tr>
<td>0050</td>
<td>66 22 4D 32 C0 58 98 65 6C 50 29 E2 FE CE A5 30</td>
<td>f&quot;M2.X.e lP)....0</td>
<td></td>
</tr>
<tr>
<td>0060</td>
<td>23 D5 11 42 87 38 F5 8E 11 36 D1 8D 0C C6 67 63</td>
<td>#._B.8....6..gc</td>
<td></td>
</tr>
<tr>
<td>0070</td>
<td>C1 7B 80 63 54 21 19 9C 7D 61 CB 33 5C 29 8C 1D DE</td>
<td>{.cT!) a.3)....</td>
<td></td>
</tr>
<tr>
<td>0080</td>
<td>B8 00 83 E9 36 50 FB FE 01 66 B5 EB F9 26 D7 64</td>
<td>...6P... f...&amp;d</td>
<td></td>
</tr>
<tr>
<td>0090</td>
<td>7F FB 61 76 42 CE C7 06 74 28 EE 58 EB 3E 8C 26</td>
<td>......avB... t(.X&gt;.)&amp;</td>
<td></td>
</tr>
<tr>
<td>00A0</td>
<td>0E 6B 94 99 7B 48 97 A3 73 33 58 A6 EE B6 9B 47</td>
<td>.k...{H.. s3X...G</td>
<td></td>
</tr>
<tr>
<td>00B0</td>
<td>C8 81 F4 F8 C9 1B F5 8F FB 2F 0C 73 B5 C9 CC 8E</td>
<td>........../.s...</td>
<td></td>
</tr>
<tr>
<td>00C0</td>
<td>AC B7 1E 03 FD 6D 9D E6 46 28 7F 2A 80 E1 17 01</td>
<td>......m. F(*.....</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>97 A7 D2 8F 33 17 A2 9E 9E BE 1C C6 AB CE E7 FC</td>
<td>........3..........</td>
<td></td>
</tr>
<tr>
<td>00E0</td>
<td>4D A6 74 27 D7 C9 3A 03 64 2C 3D 52 A4 2E A6 89</td>
<td>M.t.... d.=R.....</td>
<td></td>
</tr>
</tbody>
</table>

### bloc toshiba

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>E5 91 D4 9A FD 40 12 21 1B DA 56 6D 67 AB 07 7C</td>
<td>....@.!..Vmg.</td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>86 03 F4 AF BA 4D C3 72 D9 F4 61 F6 CF F0 28 84</td>
<td>.....M.r..a..(</td>
<td></td>
</tr>
<tr>
<td>0020</td>
<td>AB EA 02 1C 08 3F 93 DB 69 BF 06 EA 8D 52 6D 16</td>
<td>.....?. i...Rm</td>
<td></td>
</tr>
<tr>
<td>0030</td>
<td>1F 7D 0A 44 7D 47 85 15 EE 43 27 74 3B CF 12 C0</td>
<td>)D)G...C't;...</td>
<td></td>
</tr>
<tr>
<td>0040</td>
<td>E8 DC 87 82 FE 8E 40 14 D1 AC 1C 13 3F D0 84 C3</td>
<td>......@. ?....</td>
<td></td>
</tr>
<tr>
<td>0050</td>
<td>84 33 44 E4 9F 72 C3 F1 60 53 58 43 C1 6A D6 AC</td>
<td>.3D..r..`SXC.j</td>
<td></td>
</tr>
<tr>
<td>0060</td>
<td>AD C8 94 88 BF 57 23 33 D4 46 77 12 3B 4C B1 AB</td>
<td>....W#3 Fw;L</td>
<td></td>
</tr>
<tr>
<td>0070</td>
<td>E0 C7 37 ED 40 15 9C 09 60 3C 06 56 F1 F9 88 DD</td>
<td>.7@.. &lt;.V...</td>
<td></td>
</tr>
<tr>
<td>0080</td>
<td>94 35 66 7B 5C 3C 01 51 DE A9 0F 20 B3 71 1D 17</td>
<td>5f{</td>
<td>&lt;.Q....q.</td>
</tr>
<tr>
<td>0090</td>
<td>52 17 6F 88 4B CF C6 E5 B8 54 C8 75 EF 93 F9 AA</td>
<td>R.o.K...T.u...</td>
<td></td>
</tr>
<tr>
<td>00A0</td>
<td>A7 74 E8 3D 66 D1 FB 4C 91 3F D5 2A 98 8C 75 B3</td>
<td>.t.=f..L.?.*.u</td>
<td></td>
</tr>
<tr>
<td>00B0</td>
<td>04 C7 5C 53 53 7A 8E E3 AB FB 2B 2E 44 E1 98 27</td>
<td>.\SSz...+D'</td>
<td></td>
</tr>
<tr>
<td>00C0</td>
<td>9B 96 58 07 8A A8 60 19 DB 32 DE BF 26 58 1E 2A</td>
<td>...X... 2..&amp;X.</td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>F4 05 34 88 2F F6 6B A1 50 01 FE 80 BA B8 1F 26</td>
<td>.4/.k. P....&amp;</td>
<td></td>
</tr>
<tr>
<td>00E0</td>
<td>CD CC DD 80 77 EC 91 50 EE 25 50 79 56 18 DC C9</td>
<td>....w.P %PyV...</td>
<td></td>
</tr>
</tbody>
</table>
### Firmware comparison: Zalman vs PS4

<table>
<thead>
<tr>
<th>flash_controllerSATA</th>
<th>0000 20C0: A2 3E 19 19 F5 C8 85 41 B9 E4 92 15 9F F2 CA 77</th>
<th>.&gt;.... A .......w</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 20D0: 6C D3 BE 77 6F 17 0A 85</td>
<td>88 14 2E 49 3E 22 F5 05</td>
<td>l...wo... I&gt;&quot;.</td>
</tr>
<tr>
<td>0000 20E0: 96 B0 C1 3A 93 23 4C 51</td>
<td>7C 7A BB CD C3 19 13 7F</td>
<td>...#LQ</td>
</tr>
<tr>
<td>0000 20F0: B2 8F 34 59 B7 0E B4 F2</td>
<td>75 43 10 D5 5B 22 7D 86</td>
<td>...4Y... uC...[.}</td>
</tr>
<tr>
<td>0000 2100: 0E 93 D1 D3 4E 37 BB D1</td>
<td>1C C9 DF 95 EC 7C 73 37</td>
<td>...N7...</td>
</tr>
<tr>
<td>0000 2110: 83 90 A9 EF 89 A1 2B 12 BB 52 38 C2</td>
<td>4F 6B 8F DC</td>
<td>+. R8.0k.</td>
</tr>
<tr>
<td>0000 2120: 01 31 47 9B 97 4F F1</td>
<td>3A 01 87 DC C6 50 18 95</td>
<td>1G...0. ...P.</td>
</tr>
<tr>
<td>0000 2130: D7 0E 75 E0 17 83 32 A0</td>
<td>19 3D 46 5A DC 44 88 DF</td>
<td>...u...2. =FZ.D...</td>
</tr>
<tr>
<td>0000 2140: E4 D0 84 89 B6 FA 9B BD</td>
<td>FA D7 F1 BE C5 79 EF C4</td>
<td>...............y.</td>
</tr>
<tr>
<td>0000 2150: 96 2D D2 5C 5C F4 4C E8</td>
<td>24 83 93 CB 12 B1 18 04</td>
<td>-\L $...</td>
</tr>
<tr>
<td>0000 2160: 94 BD 16 44 49 C3 54 36</td>
<td>76 A6 4A D1 5D 4C BE E0</td>
<td>...DI.T6 v.J.]L.</td>
</tr>
<tr>
<td>0000 2170: FF 60 7D 96 D3 DD 9C C7</td>
<td>9A 69 C0 60 C7 7F EB 8F</td>
<td>`}...... .i.</td>
</tr>
<tr>
<td>0000 2180: DE F1 0E CB 7F C9 55 28</td>
<td>D7 23 7E 1F 98 10 00 4D</td>
<td>...U( .#~...M</td>
</tr>
<tr>
<td>0000 2190: 53 8D CF 14 50 32 6C 6E</td>
<td>82 C6 E1 06 2B C6 22 B4</td>
<td>S...P2ln ...+.</td>
</tr>
<tr>
<td>0000 21A0: 8A 23 ED EB F4 46 0F 15</td>
<td>02 EF 45 0A 77 59 A3 9B</td>
<td>#.F...E.wY...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PS4 dump.bin</th>
<th>0000 20C0: C0 15 19 19 81 19 85 41</th>
<th>09 6D 92 15 9F F2 CA 77</th>
<th>....... A .m ...w</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 20D0: 60 EC BF 77 E7 90 0A 85</td>
<td>88 14 2E 49 3E 22 F5 05</td>
<td>...w...... I&gt;&quot;.</td>
<td></td>
</tr>
<tr>
<td>0000 20E0: 96 B0 C1 3A 93 23 4C 51</td>
<td>7C 7A BB CD C3 19 09 7F</td>
<td>...#LQ</td>
<td>z...</td>
</tr>
<tr>
<td>0000 20F0: B2 8F 34 59 B7 0E B4 F2</td>
<td>75 43 10 D5 5B 22 7D 86</td>
<td>...4Y... uC...[.}</td>
<td></td>
</tr>
<tr>
<td>0000 2100: 0E 93 D1 D3 4E 37 BB D1</td>
<td>1C C9 DF 95 EC 7C 73 37</td>
<td>...t7...</td>
<td>s7</td>
</tr>
<tr>
<td>0000 2110: 83 90 A9 EF 89 A1 2B 12 BB 52 38 C2</td>
<td>FB 08 8F DC</td>
<td>+. R8...</td>
<td></td>
</tr>
<tr>
<td>0000 2120: 55 52 47 D6 9B 97 4F F1</td>
<td>3A 01 87 DC C6 50 18 95</td>
<td>URG...O. ...P.</td>
<td></td>
</tr>
<tr>
<td>0000 2130: D7 0E 75 E0 17 83 32 A0</td>
<td>19 3D 46 5A DC 44 88 DF</td>
<td>...u...2. =FZ.D...</td>
<td></td>
</tr>
<tr>
<td>0000 2140: E4 D0 84 89 B6 FA 9B BD</td>
<td>FA D7 F1 BE C5 79 EF C4</td>
<td>...............y.</td>
<td></td>
</tr>
<tr>
<td>0000 2150: 96 2D D2 5C 5C F4 4C E8</td>
<td>24 83 93 CB 12 B1 18 04</td>
<td>-\L $...</td>
<td></td>
</tr>
<tr>
<td>0000 2160: 94 BD 16 44 49 C3 54 36</td>
<td>76 A6 4A D1 5D 4C BE E0</td>
<td>...DI.T6 v.J.]L.</td>
<td></td>
</tr>
<tr>
<td>0000 2170: FF 60 7D 96 D3 DD 9C C7</td>
<td>9A 69 C0 60 C7 7F EB 8F</td>
<td>`}...... .i.</td>
<td></td>
</tr>
<tr>
<td>0000 2180: DE F1 0E CB 7F C9 55 28</td>
<td>D7 23 7E 1F 98 10 00 4D</td>
<td>...U( .#~...M</td>
<td></td>
</tr>
<tr>
<td>0000 2190: 53 8D CF 14 50 32 6C 6E</td>
<td>82 C6 E1 06 2B C6 22 B4</td>
<td>S...P2ln ...+.</td>
<td></td>
</tr>
<tr>
<td>0000 21A0: 8A 23 ED EB F4 46 0F 15</td>
<td>02 EF 45 0A 77 59 A3 9B</td>
<td>#.F...E.wY...</td>
<td></td>
</tr>
</tbody>
</table>