ELCE 2013

-

Secure Embedded Linux Product (A Success Story)

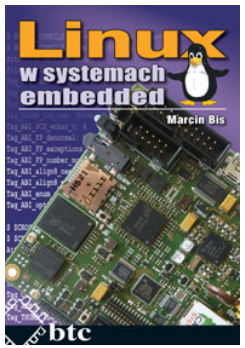
Marcin Bis

http://bis-linux.com
marcin@bis-linux.com

Edinburgh - 2013.10.25

About me

- **Marcin Bis**
- Entrepreneur
- Embedded Linux: system development, kernel development.
- Esp. Linux + Real-Time - automation (industrial- and home-).



Agenda?

I want to tell you about a success story. . .
. . . protecting added value in a product.

- A few quick words about security
  - Embedded security.
  - Attack vector and surface.
  - What is similar with standard system security?
- Practical example - secured embedded Linux system.
  - A problem (bussiness view).
  - Active & passive security.
  - Examples.

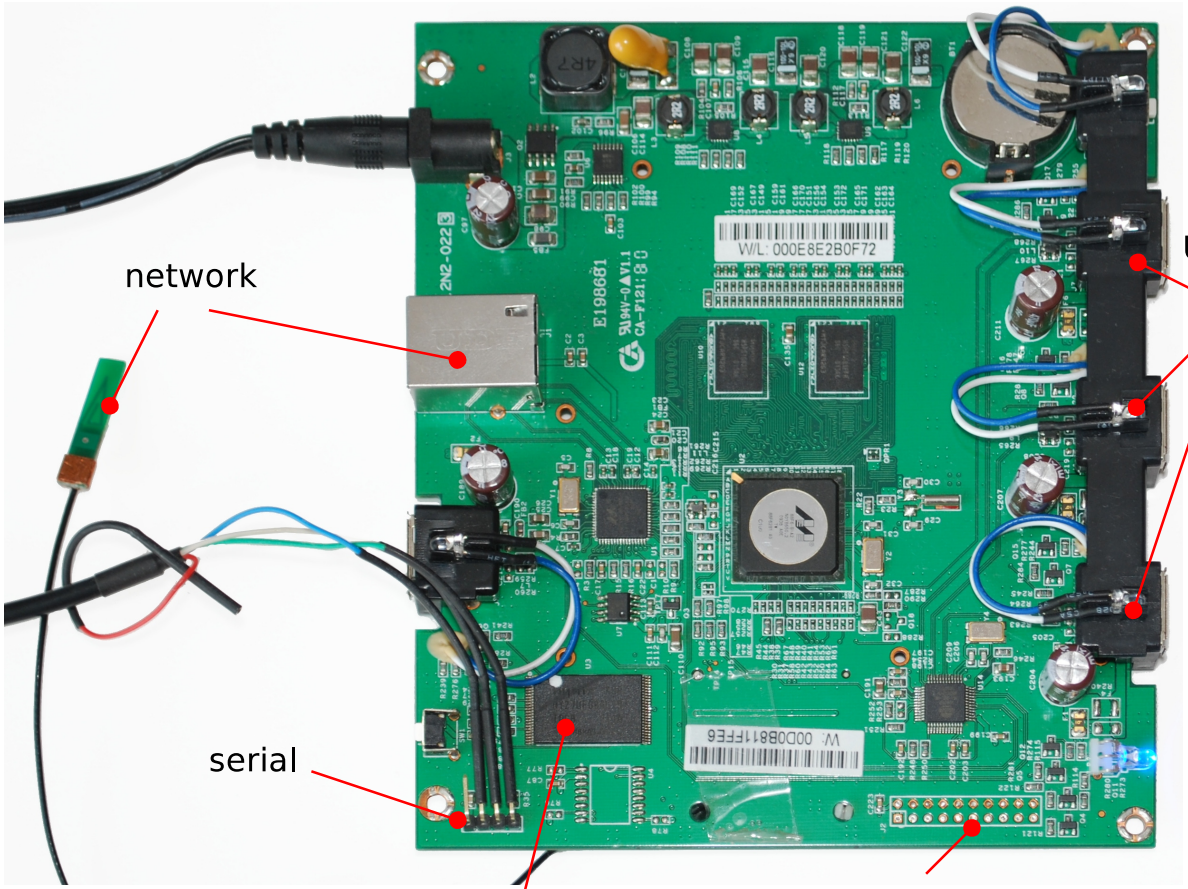
I will not talk about::

- Android
- Web apps, dedicated apps, cloud.

Attack surface

One or more input methods of the system.

- which can be accessed by untrusted user,
- or access to which can be influenced.



Attack vector

. . . to exploit a surface. Common ones:

- network (TCP/IP, Wi-Fi),
- application,
- serial port.

Less obvious:

- USB,
- I2C,
- solid state memory (FLASH),
- Bluetooth
- GPS, cellular network.

Less obvious == mode dangerous.

Embedded vs. standard

Some differences:

- Some attack vectors are unique to embedded devices.
- Problematic updates (software monoculture).
- People do not threat them as devices.

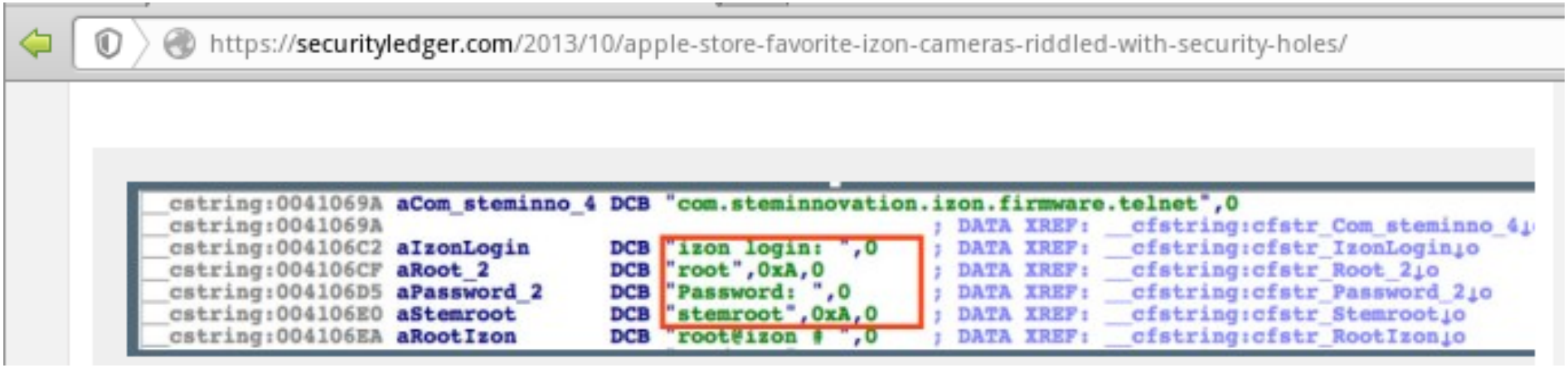
On the other hand - same programs and services.

(Wireless) network accessible.

Apache, openssh, perl, avahi, dns, openssl etc.

Some examples

- Stuxnet
- FTP access to / via root account.
- admin:default - common in network devices.
- More, easy to find: `http://lwn.net/talks/elc2009/` (2009).
- Another example of hard-coded credentials:



Make it secure (trivia)

Common methods are easy to avoid:

- Restricted shell access, eg. serial port
  - strong password,
  - use PAM to auto-logout idle shells.
- Other access methods to shell (web shell, ssh, telnet (!) etc.)
- Strong passwords (+1).
- Do not run all applications from root account.
- Bug-fix-ed components.
- Self developed vs. standard software.
  - Defensive programming.

Passive security

How the customer see the product?

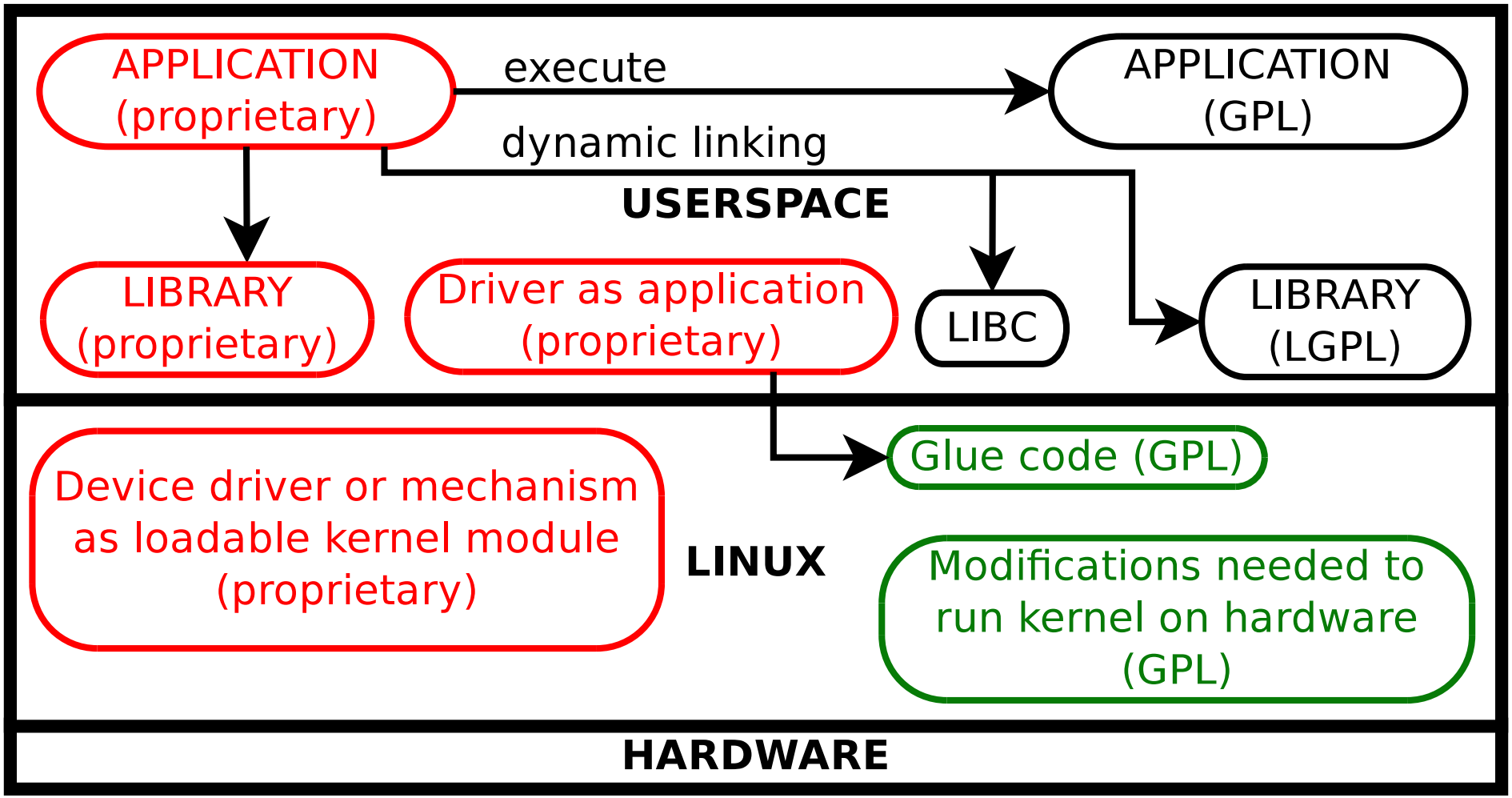
Added Value

Free Software | Open Source

How my customer see the product?

- Hardware becomes cheaper and cheaper.
- Expectations increases (let's add functionality).
- Linux and open source is a foundation of the software product.
- Open-Source and Free Software gives us all freedom:
  - Every developer has the same rights.
  - And equal chances.
- Customer will make money on added value
  - According to licences of course:
    - GPL
    - LGPL
    - BSD

Added value?


APPLICATION
(proprietary)
execute
APPLICATION
(GPL)
dynamic linking
USERSPACE
LIBRARY
(proprietary)
Driver as application
(proprietary)
LIBC
LIBRARY
(LGPL)
Glue code (GPL)
Device driver or mechanism
as loadable kernel module
(proprietary)
LINUX
Modifications needed to
run kernel on hardware
(GPL)
HARDWARE

How to secure a added value?

- possibility of "TiVo-lization", - do not go to far.
- GPLv3

Let's do it!

At first:

- Nothing will stop user (abuser) from de-soldering an element and trying to analyze logic states.
- Most SoC-s has hundreds of pins - it is difficult (but not impossible).

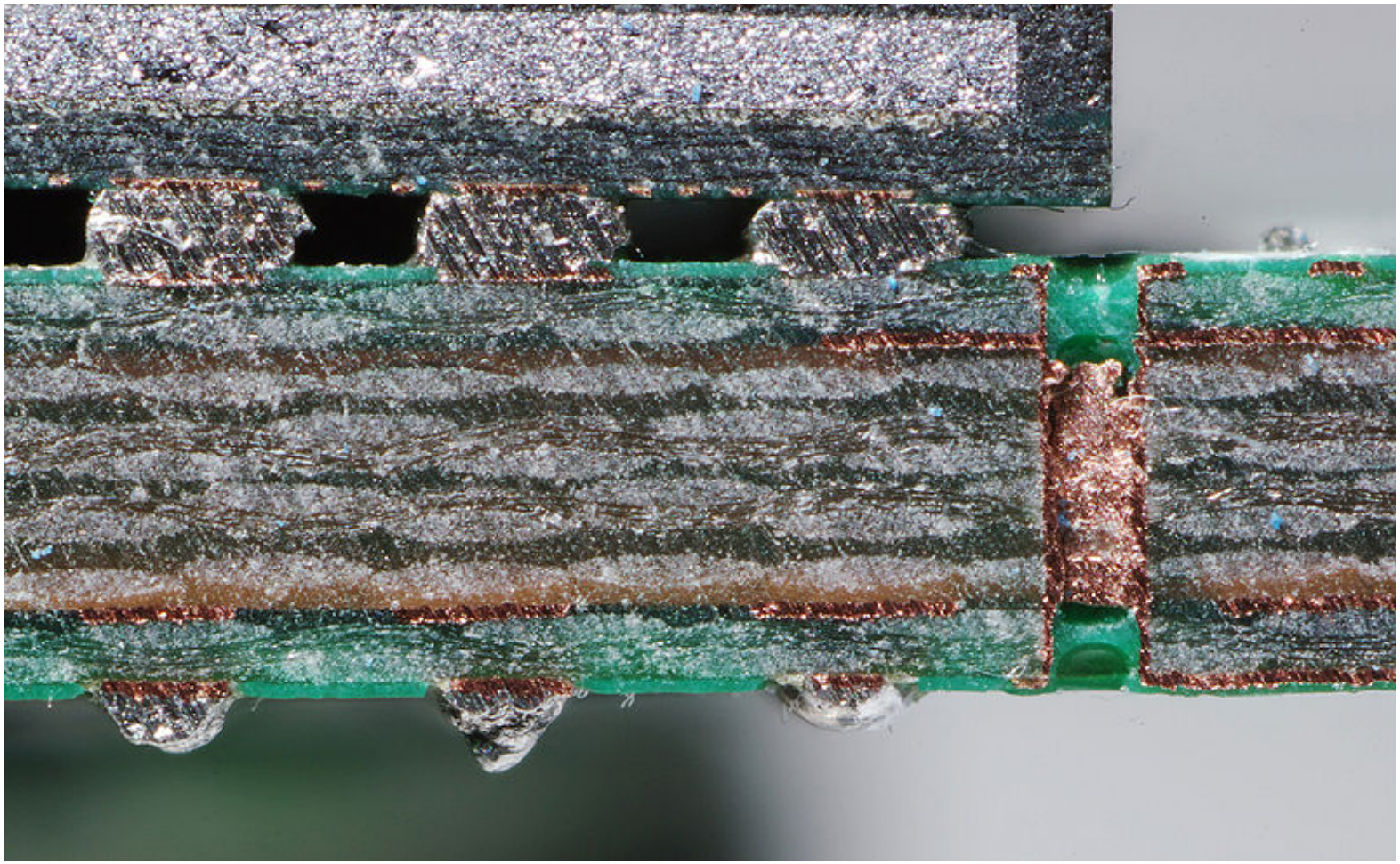
It all depends on how determined you are ($$$).

Security is a process not a product.

Hardware methods

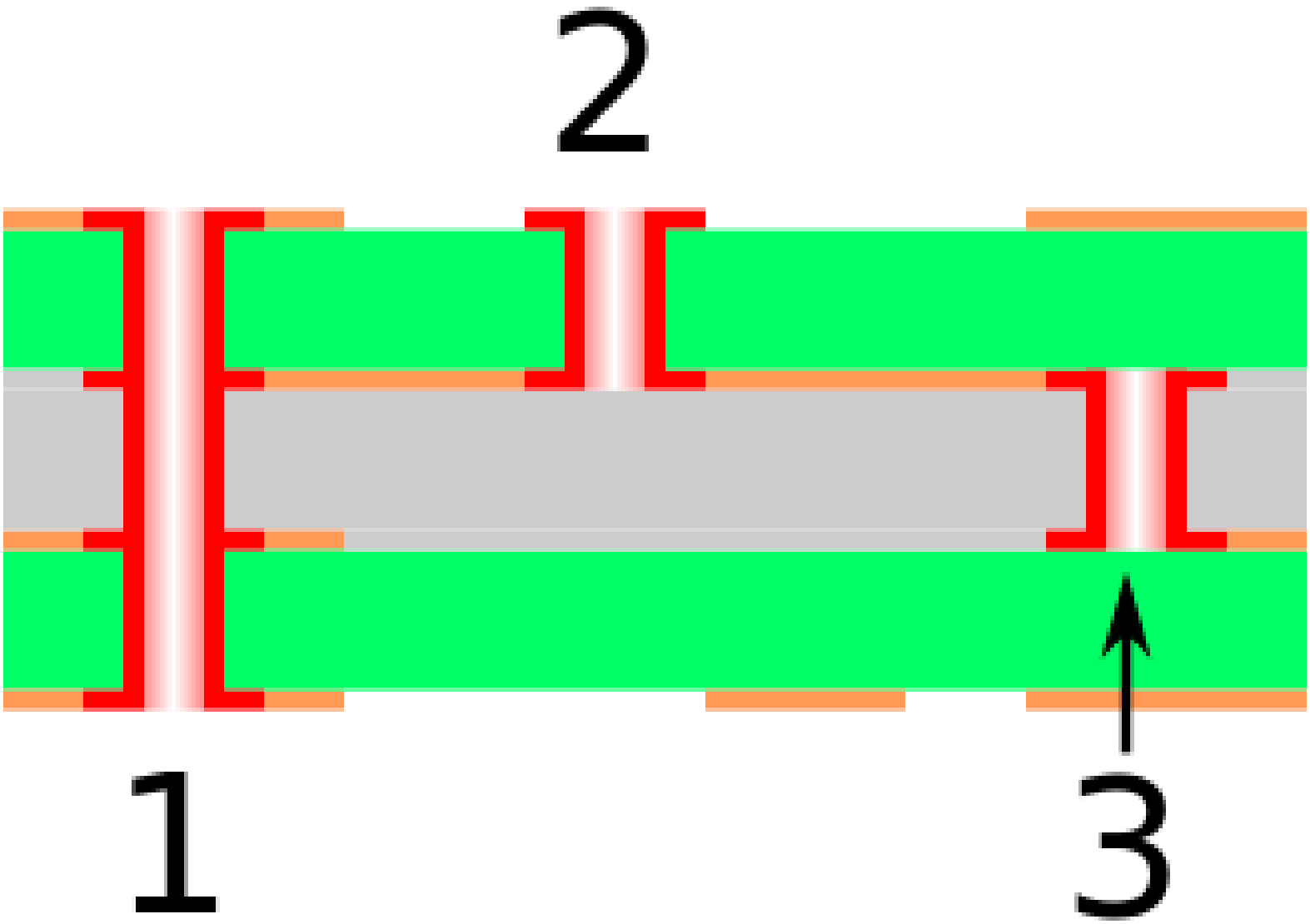
- BGP - it is harder to analyze data on bus,
- inner layers of PCB are harder to access,
- of using Application processor and external uC - add some logic to check timing (like watchdog).
- TPM chips.

VIA



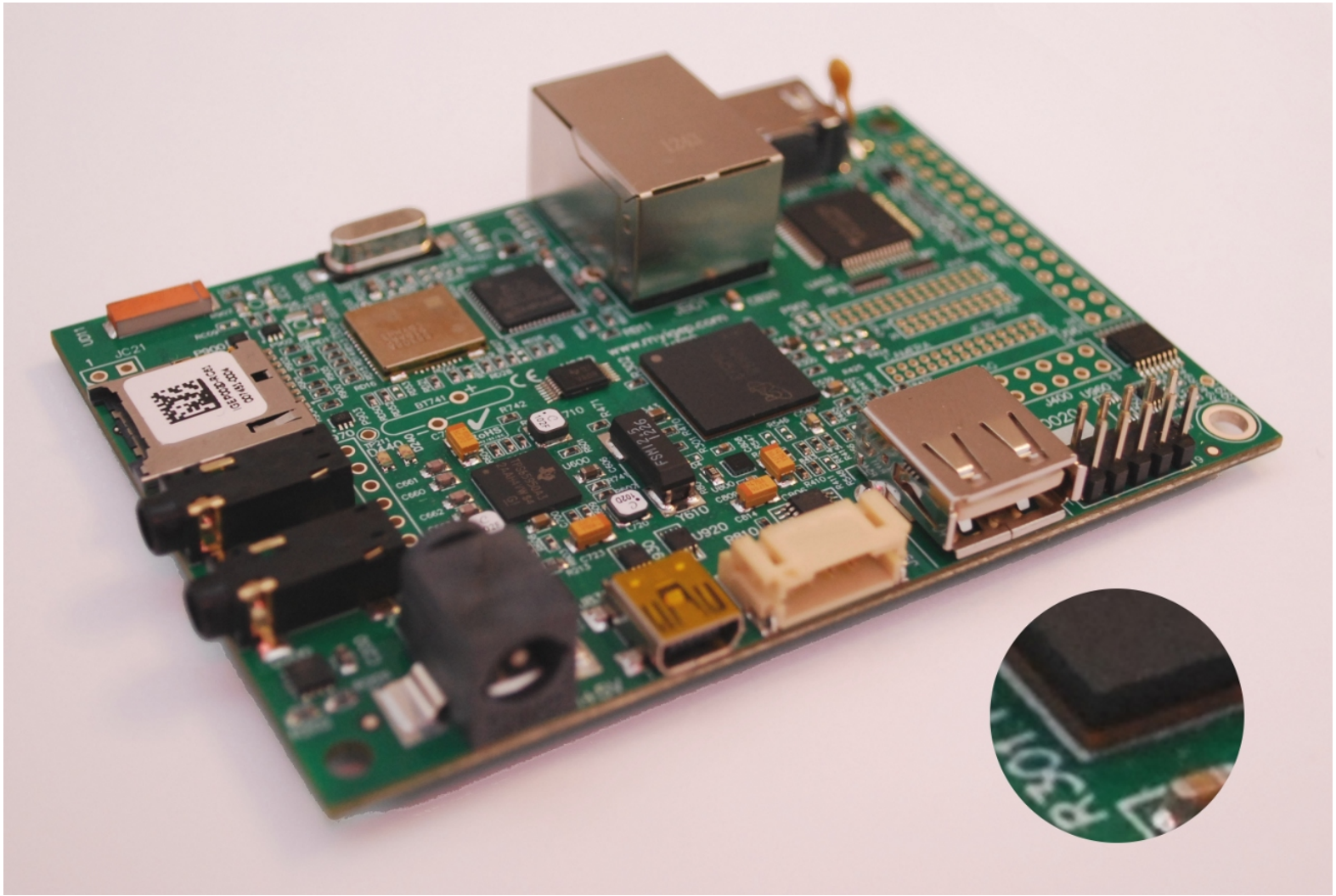
(Wikipedia)

Hidden Via (3)



(Wikipedia)

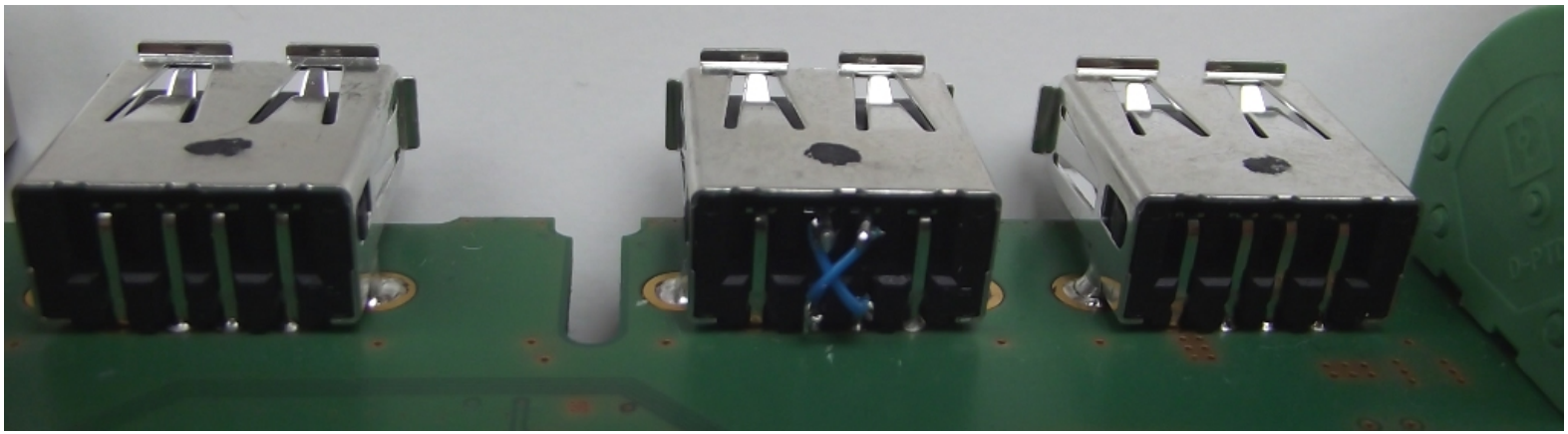
a sandwich



(Wikipedia)

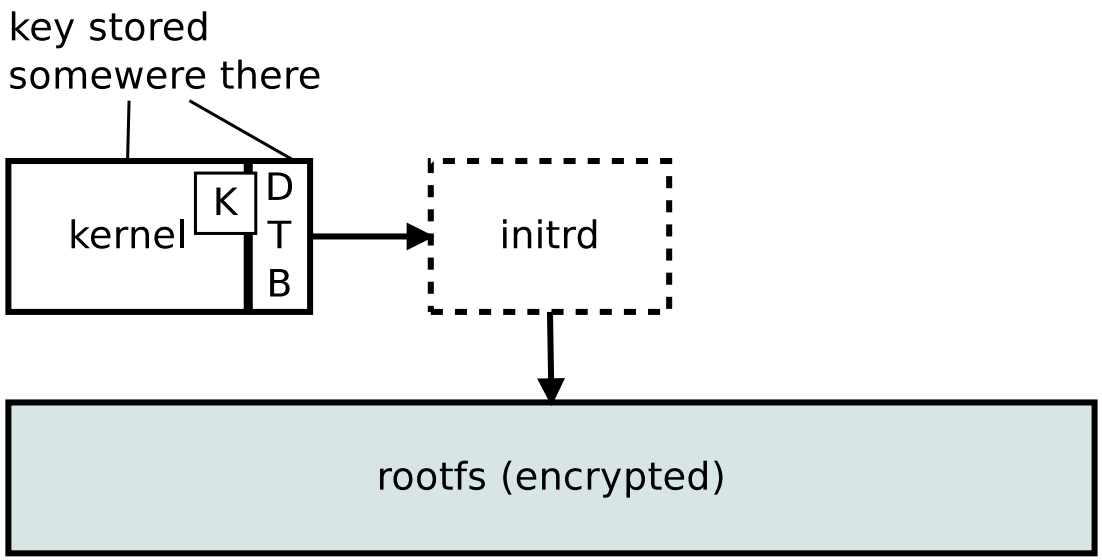
The problem. . .

- It is not easy to debug firmware.



Let's secure data

- Sign it.
  - TPM
  - HAB
- or encrypt it
  - Should be fast.
  - Performance penalty (esp. Real-Time).
  - Where to store the key.



How to encrypt rootfs

Block devices (e.g. eMMC):

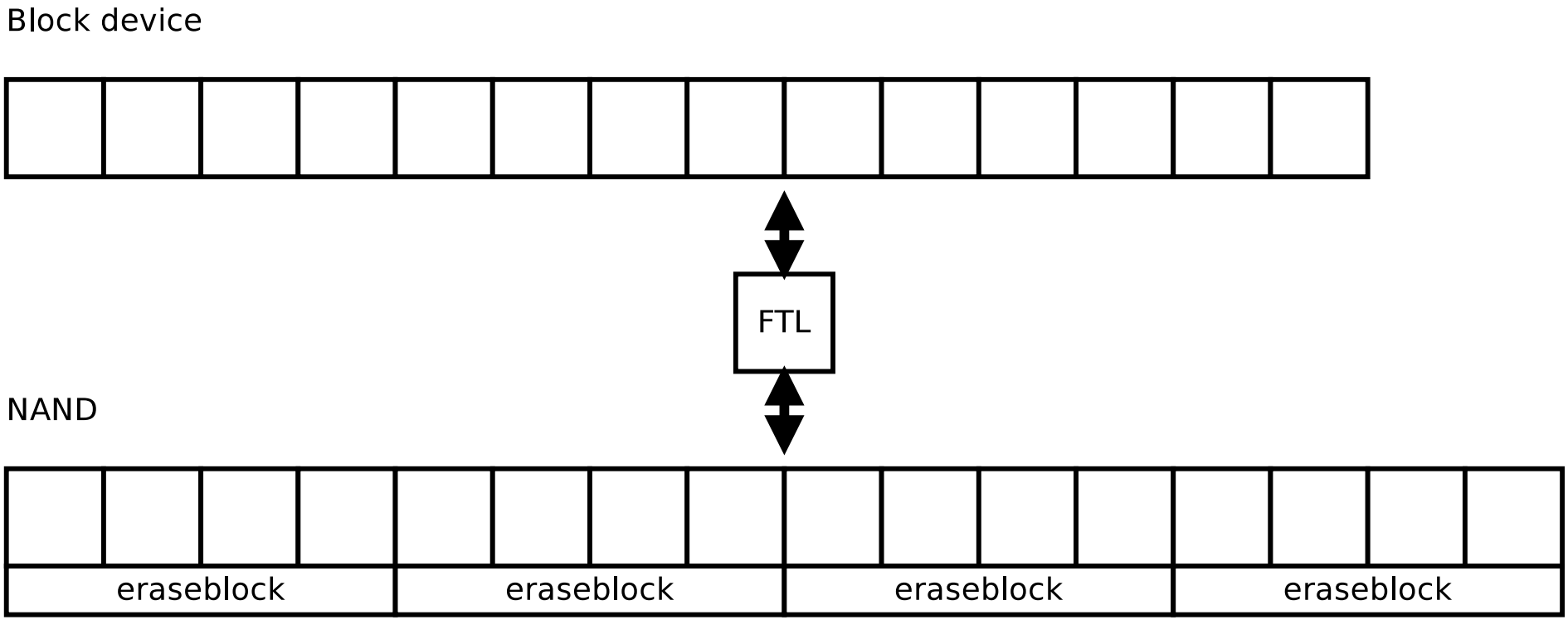
- dm-crypt
- `man cryptsetup`
- LUKS

Any filesystem

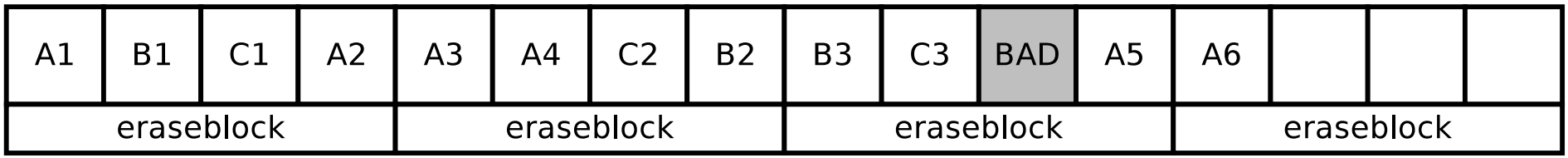
- ecryptfs
- `sudo mount -t ecryptfs tmp1 tmp2`
- problems using on rootfs (pivot_root, switch_root)
- still, can be used to encrypt parts of filesystem.

Cutomer wants to have a raw NAND device (wear leveling).

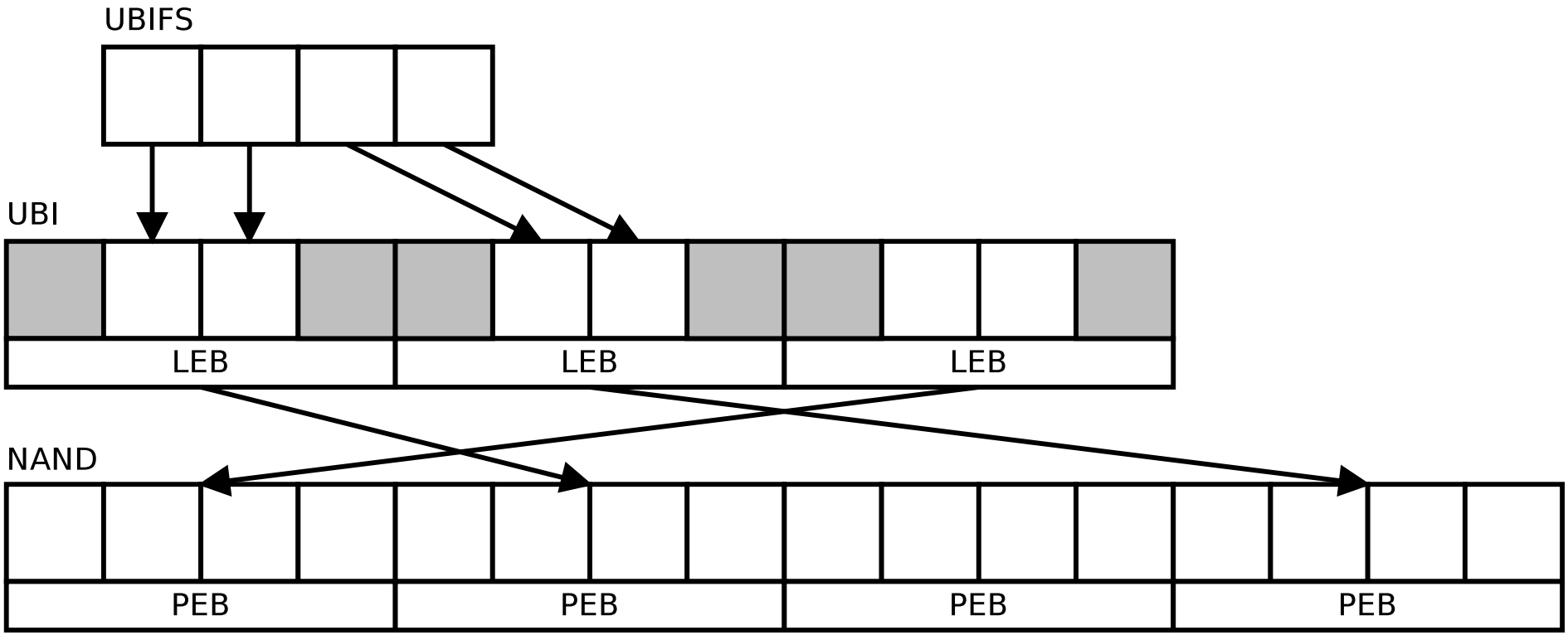
How does it work?



NAND


A1
B1
C1
A2
A3
A4
C2
B2
B3
C3
BAD
A5
A6
eraseblock
eraseblock
eraseblock
eraseblock

UBI



UBI is nice. . .

. . . for big NANDs:

- `http://elinux.org/Flash_Filesystem_Benchmarks`

How to add encryption?

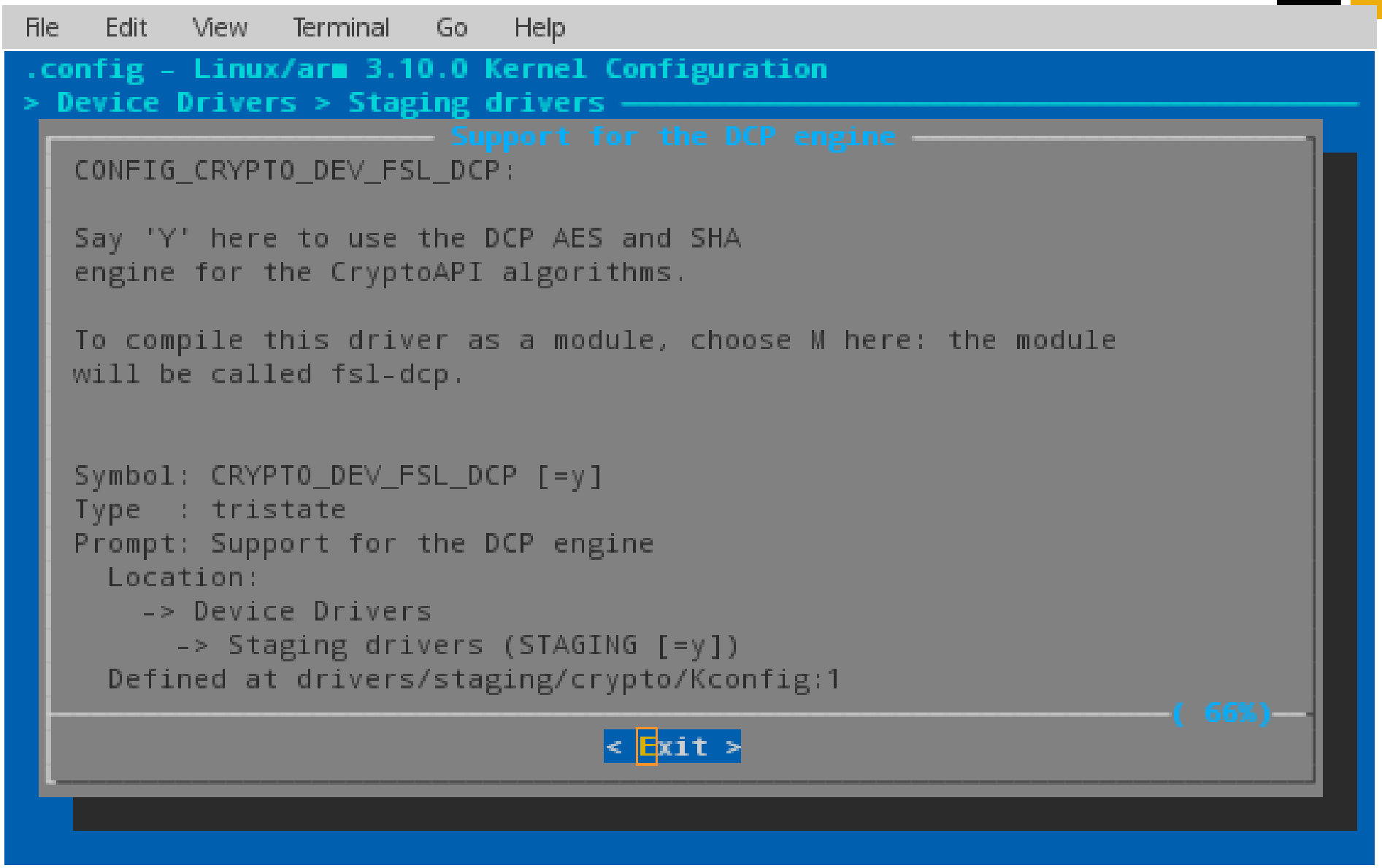
- It can emulate block device.
- Use ecryptfs.
- look at the source code.

UBIFS arleady compresses data it writes. Maeby it could encrypt it too.

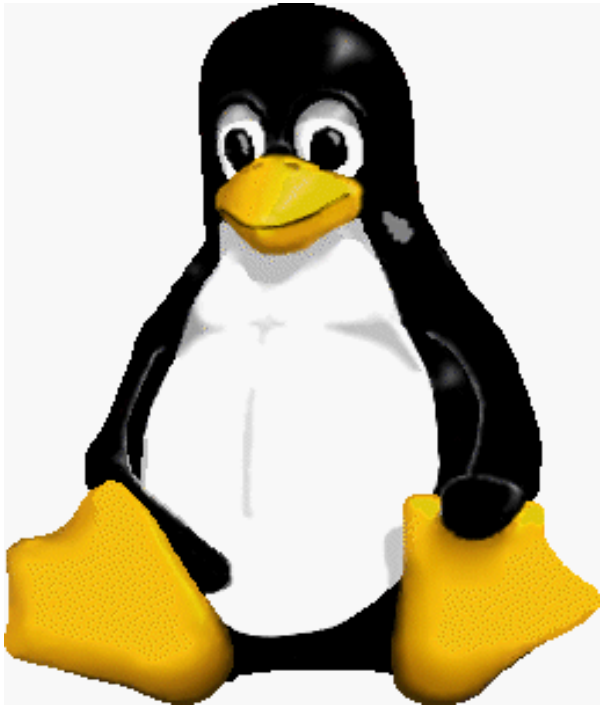
- Using Crypto-API.

```
This patch adds a function to perform AES encryption. The compress and
decompress routines use this function if they are called with a non-NULL
key parameter. It uses AES counter mode (where encryption and decryption
are the same function) and performs the operation in place on the data. It
uses a default IV of 0, since each key is only evey used to encrypt one
data item the IV does not matter.

The const qualifier was removed from the decompress routine for the
following reason. Encrypted data is not compressable, so compression is
first applied then the result is encrypted. In the reverse, decryption is
first applied and the result decompressed. This means that either the
input buffer for decompression is used to perform an in-place decryption
before decompression, or a third buffer is added and data is copied around.


Signed-off-by: Joel Reardon <reardonj at inf.ethz.ch>
---
 fs/ubifs/compress.c |   77 ++++++++++++++++++++++++++++++++++++++++++++++--
 fs/ubifs/ubifs.h    |   12 +++++++-
 2 files changed, 85 insertions(+), 4 deletions(-)
```

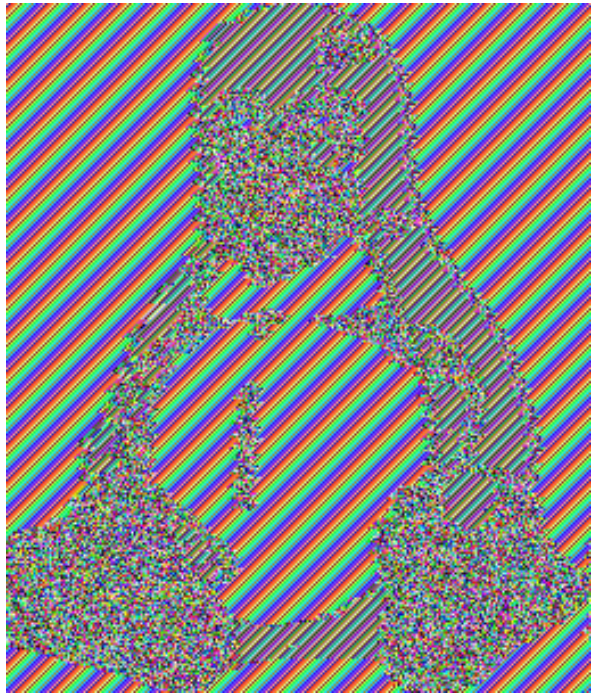
On the other side:



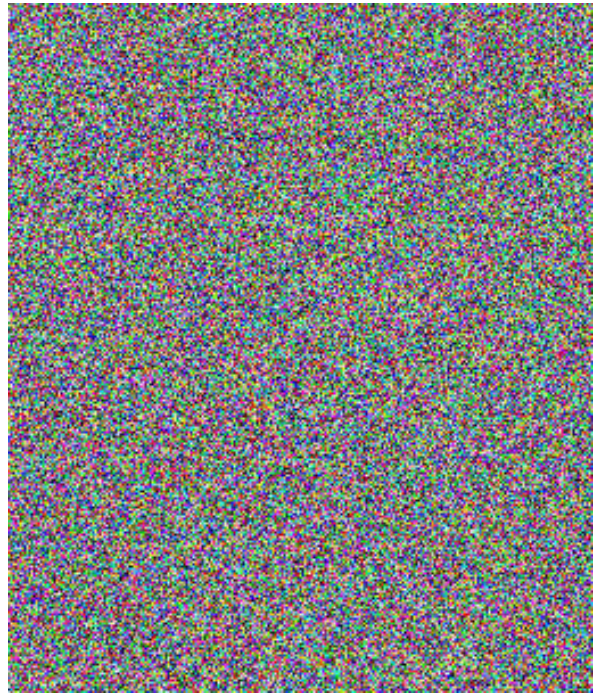
Use proper block cipher



data



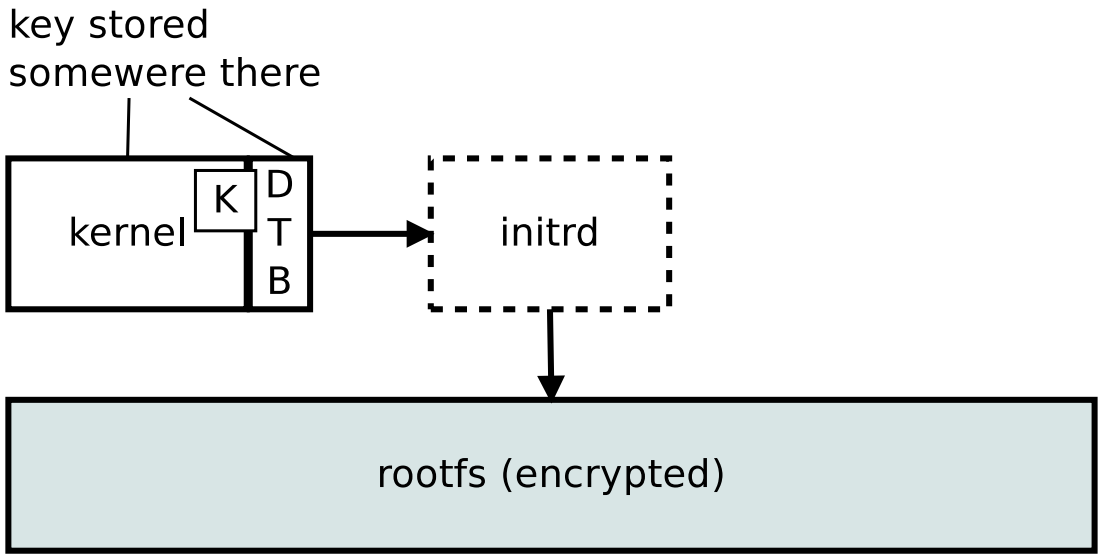
AES-ECB

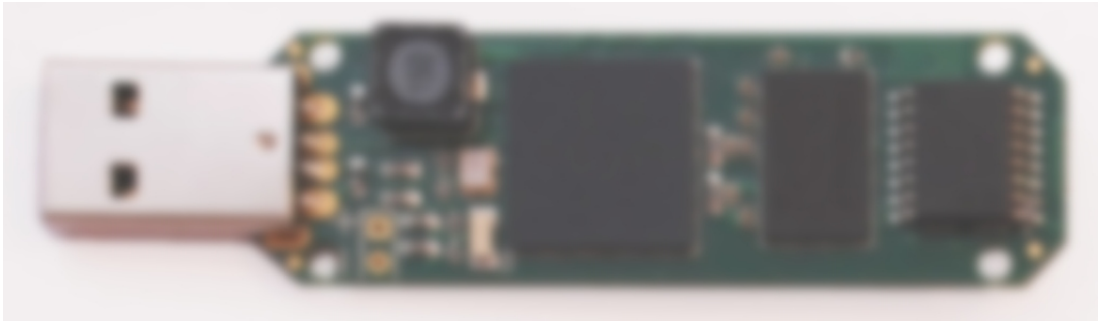


AES-CTR

```
openssl enc -aes-128-ecb -k "secret" -in logo.ppm -out out.ppm
```

Secured device





The problem of storing encryption key - still exists.

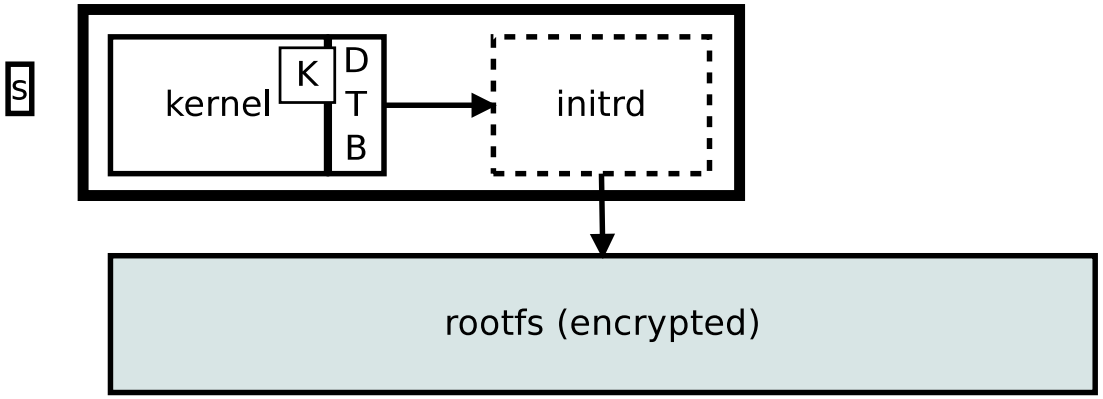
Secured device. . .

The problem of storing encryption key - still exists.

- Put it as DT attribute.
- Modify NAND driver to use it.

Encrypt kernel+DT using functions of the Chip.

i.MX28 SecureBoot



Thank You!

BIS-LINUX

What is important?

Security is not a product.

it is a process.

What else?

Internal attacks.

I do not even trust myself.

Questions?