arm

Under Lock and Key: Using Hardware Protected Keys with the Linux Crypto API

Embedded Linux Conference
Europe

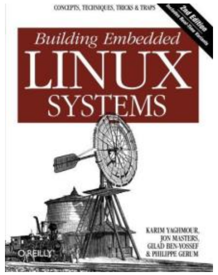
Gilad Ben-Yossef
30 October 2019

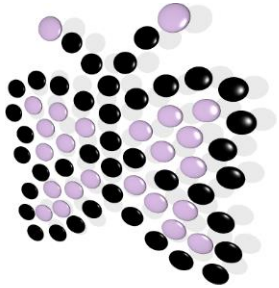
Who Am I?

- I'm Gilad ben-Yossef.
- I'm a principal Software Engineer at Arm.
- I work on applied cryptography and security of the upstream Linux kernel in general and maintain the arm® TrustZone® CryptoCell® Linux device driver.
- I have been working in various forms with and on the Linux kernel and other Open Source projects for over twenty years.
- I have co-authored “Building Embedded Linux Systems” 2nd edition from O’Reilly.

Should we use threads?

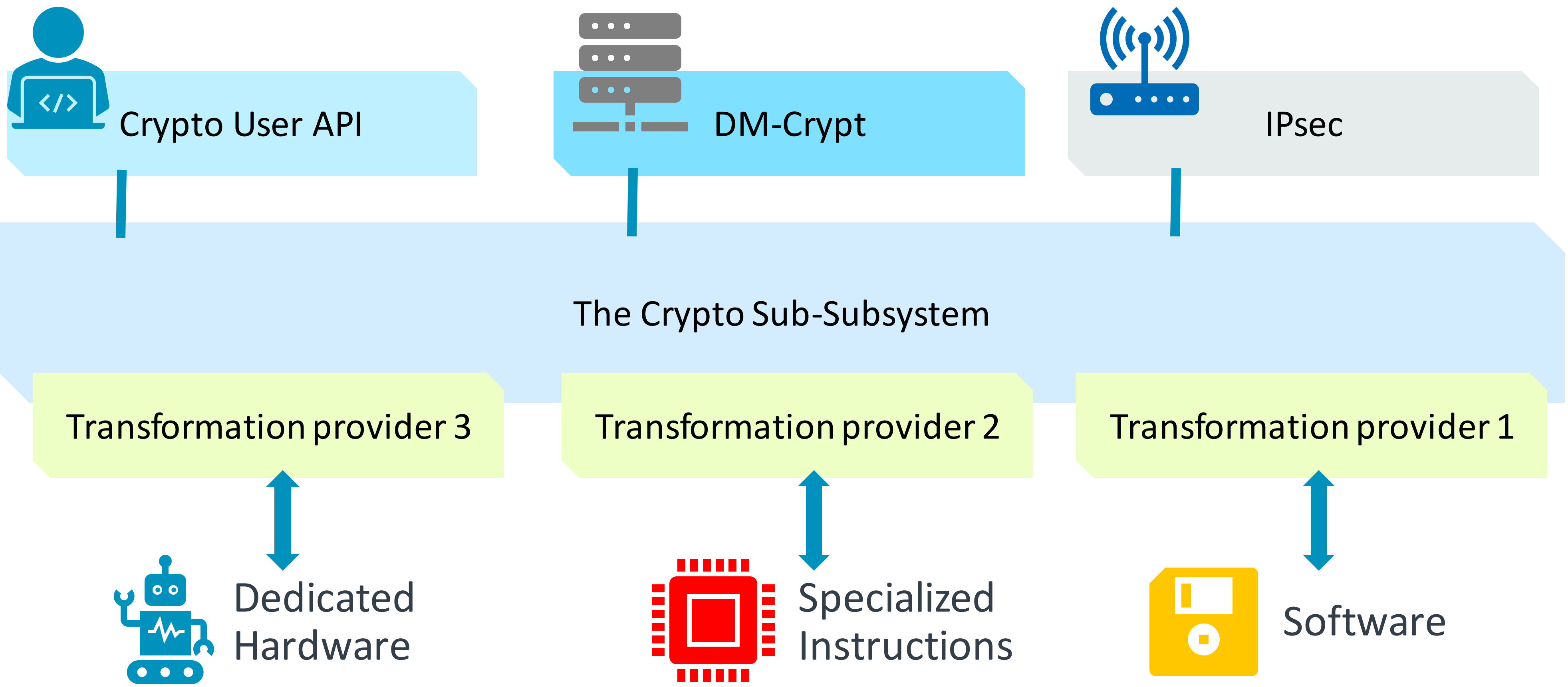
Given an application that calls for several tasks, should we implement each task as a thread in the same processes or in a separate processes?





The Linux Cryptography Sub-System

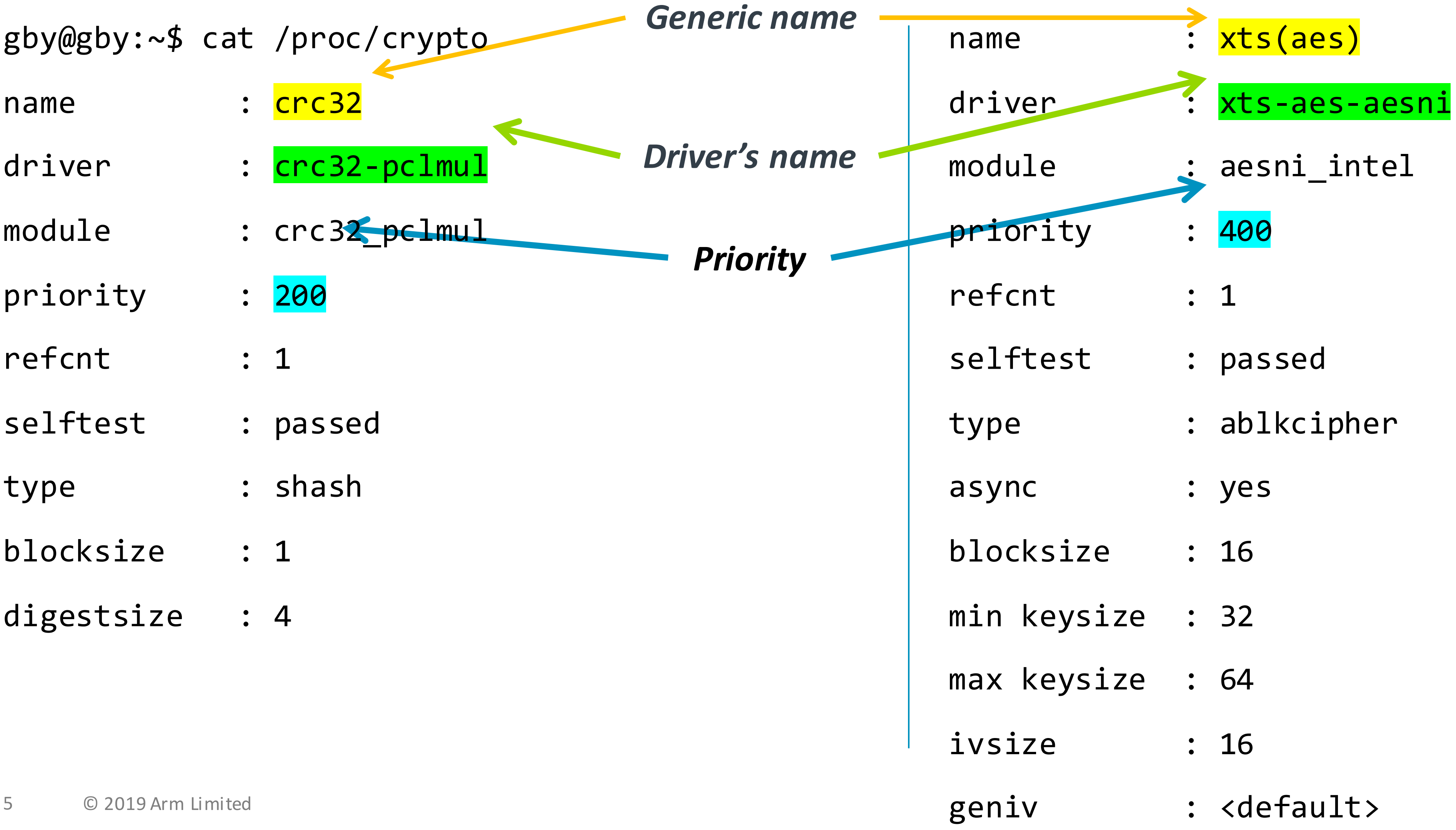
Or the Linux Crypto API, in short



Crypto API Usage Example

```
tfm = crypto_alloc_skcipher("xts(aes)", 0, 0); // Get a handle of a transformation that handles XTS mode of AES.

err = crypto_skcipher_setkey(tfm, key, sizeof(key)); // Set the key to be used for all subsequent operations

req = skcipher_request_alloc(tfm, GFP_KERNEL); // Get a request handle

skcipher_request_set_callback(req, CRYPTO_TFM_REQ_MAY_BACKLOG | CRYPTO_TFM_REQ_MAY_SLEEP,
                                            crypto_req_done, &wait); // Set the callback function to be called when done

skcipher_request_set_crypt(req, &sg, &sg, datasize, iv); // Set the input, output and initial vector buffers
ret = crypto_skcipher_encrypt(req); // Start the operation

err = crypto_wait_req(ret, &wait); // Wait for the operation to finish
crypto_free_skcipher(tfm); // Free things up
skcipher_request_free(req);
```

/proc/crypto



```
gby@gby:~$ cat /proc/crypto
name            : crc32
driver          : crc32-pclmul
module          : crc32_pclmul
priority        : 200
refcnt          : 1
selftest        : passed
type            : shash
blocksize       : 1
digestsize      : 4
```

```
name            : xts(aes)
driver          : xts-aes-aesni
module          : aesni_intel
priority        : 400
refcnt          : 1
selftest        : passed
type            : ablkcipher
async           : yes
blocksize       : 16
min keysize     : 32
max keysize     : 64
ivsize          : 16
geniv           : <default>
```

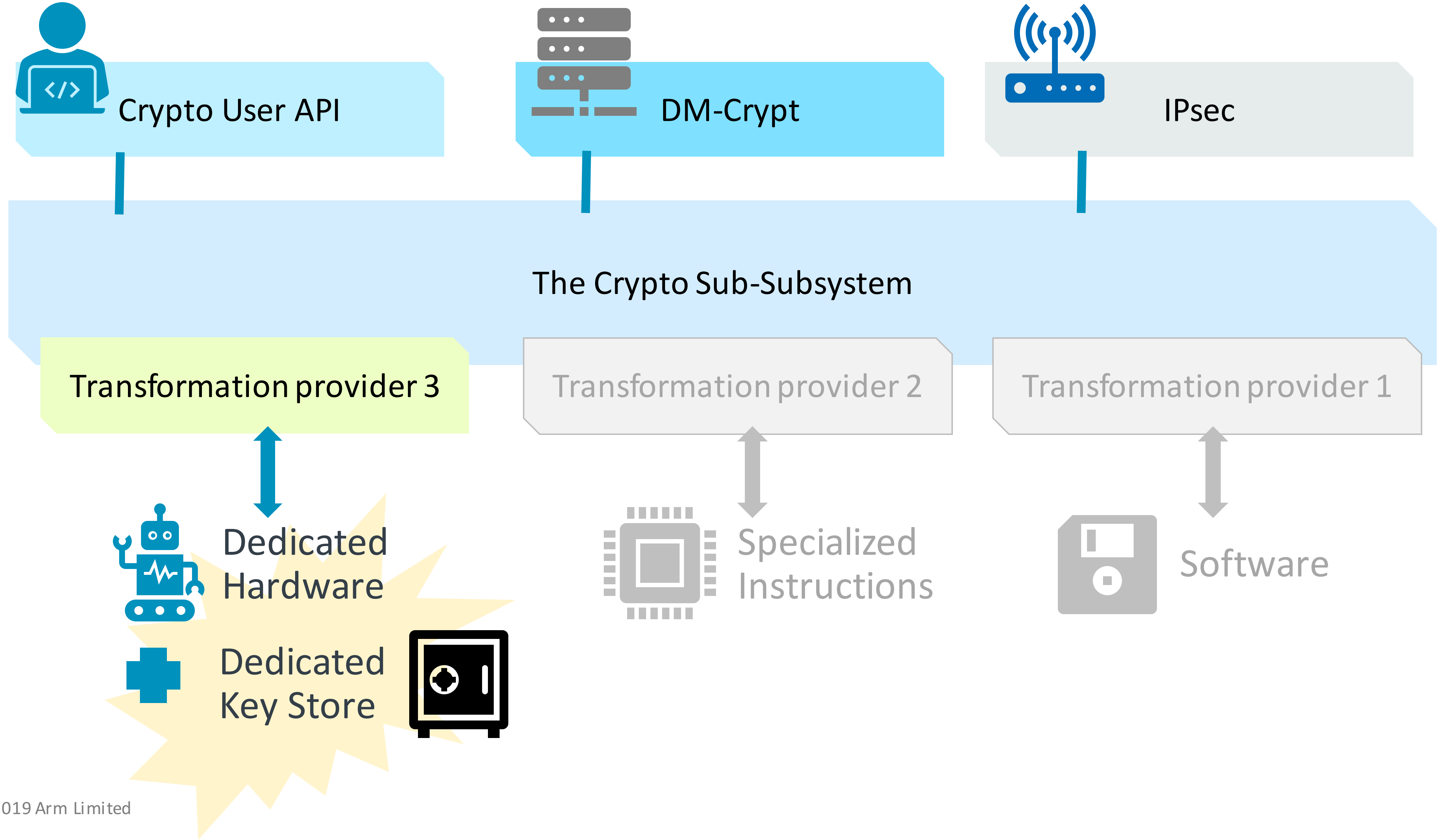
Wait, back up a little…

```
// Set the key to be used for all subsequent operations
err = crypto_skcipher_setkey(tfm, key, sizeof(key));
```

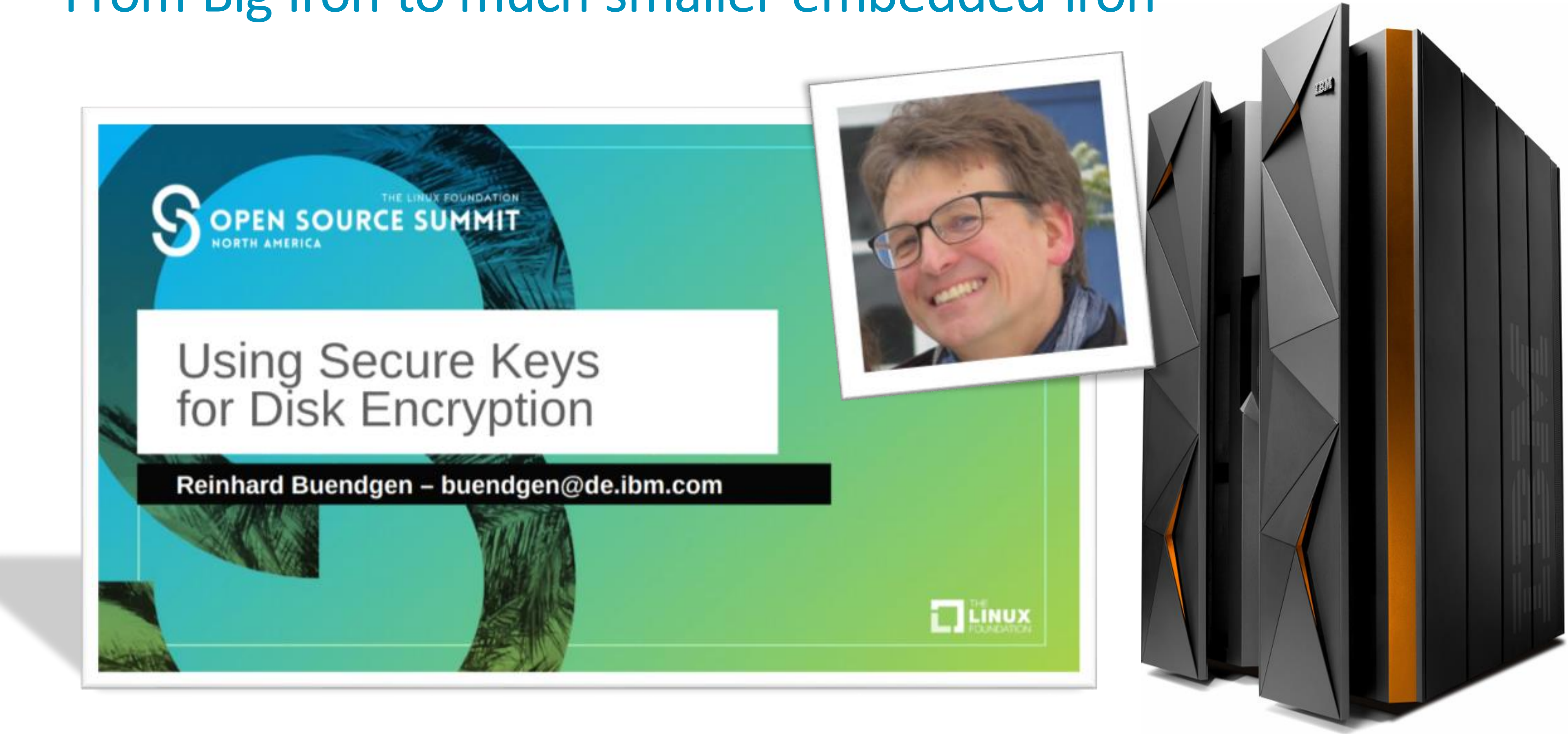
… where is the key stored?
In RAM, like everything else.

What if someone gains access to my device and steals my key?

Hardware Protected keys



From Big Iron to much smaller embedded iron



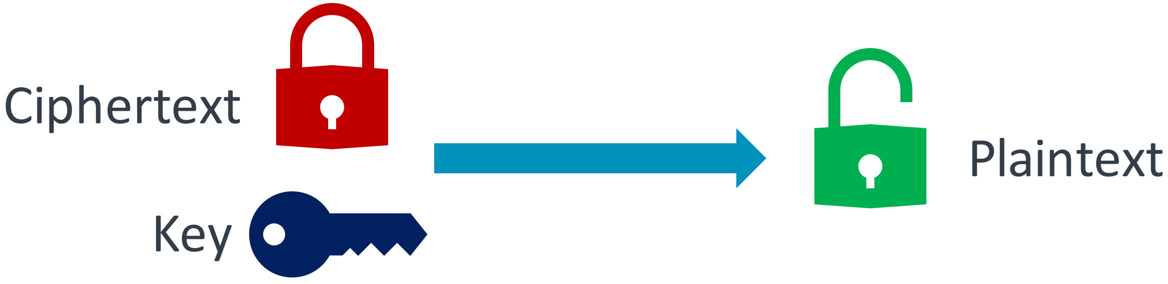
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

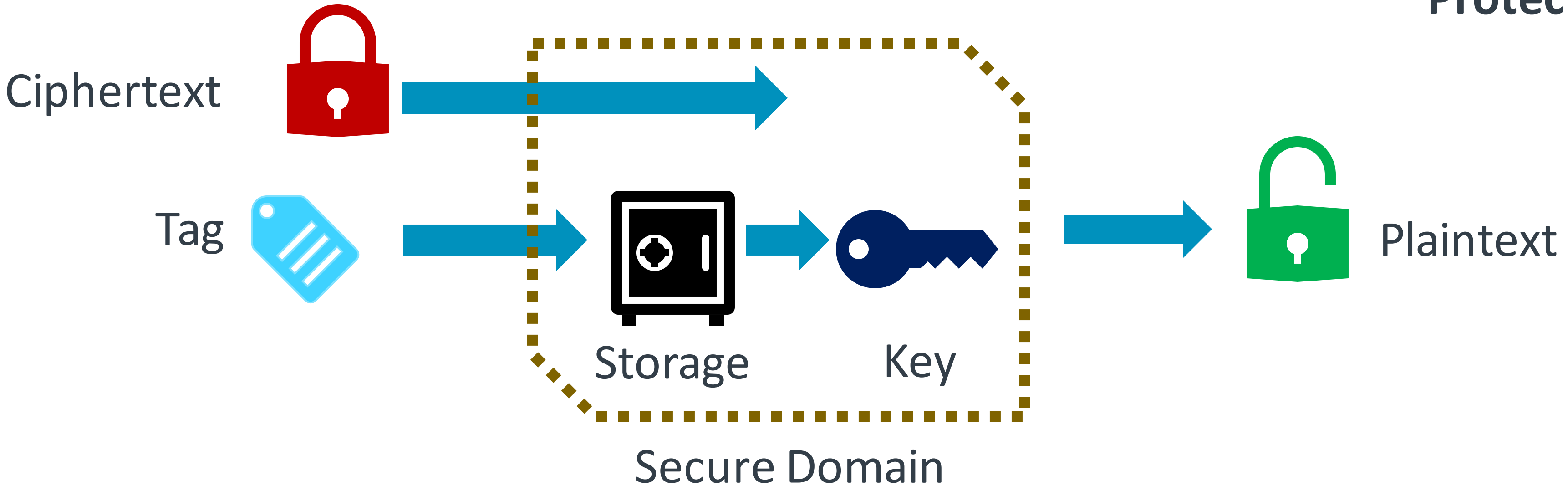
Using Secure Keys
for Disk Encryption

Reinhard Buendgen – buendgen@de.ibm.com

THE LINUX FOUNDATION

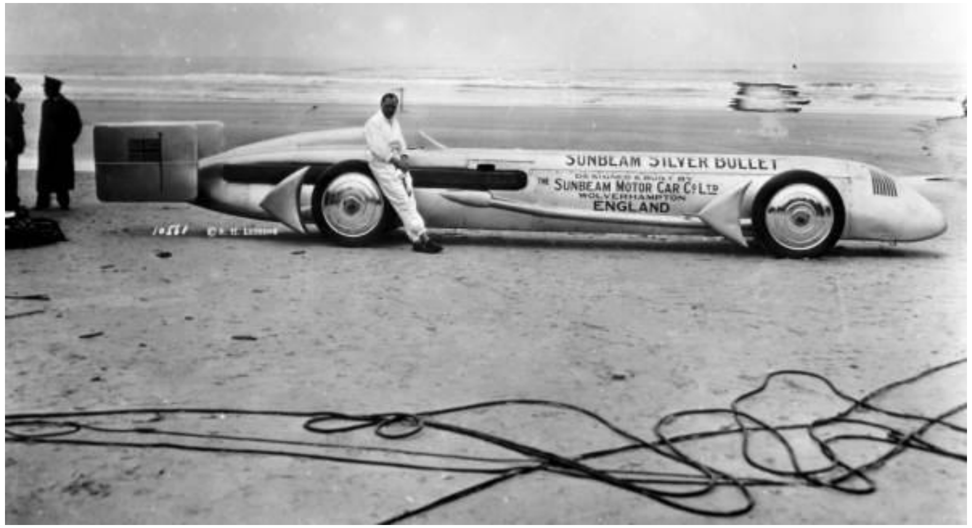
Protected Keys Usage





Not a silver bullet

- Actual security depends on the security of the so called "Secure Domain"…
- Ability to use the key might be enough for attacker.
- Key provisioning and management is a problem as always.
- **A good component in a "Defense in Depths" strategy.**



*This work is from the **Florida Memory Project** hosted at the State Archive of Florida, and is released to the **public domain** in the United States under the terms of Section 257.35(6), Florida Statutes.*

Interface Details

Note: some parts are Arm CryptoCell specific

- The letter "p" is used as prefix to the generic algorithms name.
  - E.g. Use "paes" for Protected Key AES
- Because the tag value are implementation specific, use of a driver specific name is preferred over generic name, where possible.
  - E.g. Use "xts-paes-ccree" instead of "xts(paes)"
- Instead of the normal key, provide a tag appropriate for the implementation
  - E.g.

```
struct cc_hkey_info {
   u16 keylen;       /* Length of actual key in bytes*/
   u8 hw_key1;       /* First key index */
   u8 hw_key2;       /* Second key index (optional) */
} __packed;

#define CC_HW_KEY_SIZE sizeof(struct cc_hkey_info)
```

Example: DM-Crypt with protected keys

```
# dmsetup create my_encrypted_volume \ <-- name of volume
  --table "0 $(blockdev --getsz /dev/sdb) \ <-- start and end offset of volume
          crypt \                           <-- use the DM-Crypt target
           capi:xts(paes)-plain64 \         <-- use the xts(paes) kernel cipher with 64 bit IV
           00200100 \                       <-- use 256 bit AES keys from indices 0 and 1
           0 /dev/sdb 0"                    <-- No IV offset, no volume offset
```

Note: cryptsetup can also be used if you want to password protect the key index with PKDF

Thank You

Danke

Merci

谢谢

ありがとう

Gracias

Kiitos

감사합니다

धन्यवाद

شكرًا

תודה

arm