Cryp driver Documentation

Release 0.9.0

ANSSI

Apr 28, 2020

TABLE OF CONTENTS

This library is an implementation of the STM32F4 Cryp cryptographic HW accelerator.

It provide an abstraction of the Cryp device interactions through high level API in order to request (de)cryption of any data using one of the supported cryptographic algorithm of the hardware device.

CHAPTER

ONE

ABOUT THE CRYP DRIVER

1.1 Principles

The STM32F4x9 cryptographic coprocessor support various symmetric algorithms, that are configured by the Cryp driver during its initialization step.

The driver is also able to separate two independent roles:

- The CRYP configurator, allowing key injection, but no (de)cryption requests
- The CRYP user, allowing (de)cryption on a previously configured CRYP device by a CRYP injector, but refusing any key injection

These two roles permit to separate a (de)cryption data plane from a cryptographic control dataplane, which can be held in two independent tasks.

It is also possible to handle the overall CRYP manipulation in the same task.

The STM32F4x9 Cryptographic coprocessor supports the following algorithms:

- 3DES (ECB mode)
- 3DES (CBC mode)
- DES (ECB mode)
- DES (CBC mode)
- AES (ECB mode)
- AES (CBC mode)

For AES algorithm, the device support three key sizes:

- 128 bits length
- 192 bits length
- 256 bits length

CHAPTER

TWO

CRYP DRIVER API

2.1 Initializing the cryp driver

Initializing the hash driver is done with the following API:

```
#include "libcryp.h"

enum crypto_usage {
  CRYP_USER = 0,
  CRYP_CFG,
};

typedef enum {
    CRYP_MAP_AUTO,
    CRYP_MAP_VOLUNTARY
} cryp_map_mode_t;


enum crypto_key_len {
    KEY_128,
    KEY_192,
    KEY_256
};

enum crypto_algo {
    TDES_ECB,
    TDES_CBC,
    DES_ECB,
    DES_CBC,
    AES_ECB,
    AES_CBC,
    AES_CTR,
    AES_KEY_PREPARE
};

enum crypto_dir {
    ENCRYPT,
    DECRYPT
};


/* Cryp device declaration */
int cryp_early_init(bool               with_dma,
                    cryp_map_mode_t    map_mode,
```

(continues on next page)

(continued from previous page)

```
                    enum crypto_usage usage,
                    int *             dma_in_desc,
                    int *             dma_out_desc);

/* per role initialization */
void cryp_init_user(      enum crypto_key_len key_len,
                    const uint8_t *           iv,
                          uint32_t            iv_len,
                          enum crypto_algo    mode,
                          enum crypto_dir     dir);

void cryp_init_injector(const uint8_t *           key,
                              enum crypto_key_len key_len);

void cryp_init(const uint8_t *           key,
                     enum crypto_key_len key_len,
               const uint8_t *           iv,
                     uint32_t            iv_len,
                     enum crypto_algo    mode,
                     enum crypto_dir     dir);

/* configure the DMA streams with proper informations (handlers, buffers...) */
int cryp_init_dma(void *     handler_in,
                  void *     handler_out,
                  int        dma_in_desc,
                  int        dma_out_desc);
```

2.1.1 About early init

The cryp driver early initialization must be executed before the end of the task initialization phase (see EwoK kernel API). This initialization declare the cryp device against the kernel at boot time. It also get back some information on the say the task wish to use the device:

- **with_dma**: specify if the Cryp device input and output memory transfers should be accelerated by the associated DMA controler
- **map_mode**: specify if the cryp device is automatically mapped on the task's memory space or is manually mapped/unmapped using the corresponding sys_cfg(CFG_MAP)/sys_cfg(CFG_UNMAP) syscalls. This permit to schedule device mapping during the task livecycle
- **usage**: specify the role of the task (crypto user or configurator)
- **dma_in_desc**: the input (from memory to device) DMA stream descriptor that will be fulfill by the function if the DMA mode is required
- **dma_out_desc**: the output (from device to memory) DMA stream descriptor that will be fulfill by the function if the DMA mode is required

Hint: The CRYP device is using two DMA streams, one to load data from the main memory, another to write (de)crypted data into memory

Caution: At eary initialization time, the DMA streams are declared but are not yet operational. Input and output buffer and buffer lengths are not set

2.1.2 About init

The init step **map** the cryp device only if the chosen map mode is CRYP_MAP_AUTO. If CRYP_MAP_VOLUNTARY is chosen, be sure to handle the map step before using the other functions of the CRYP API.

> **Danger:** Any access to an unmapped device leads to a memory fault of the task. In paranoid mode of EwoK, the device reboots

At initialization time, the task uses one of the initialization functions depending on the role chossen through the *usage* argument of the early initialization function.

In injector mode, the initialization function is the following

```
void cryp_init_injector(const uint8_t *           key,
                        enum crypto_key_len key_len);
```

In injector mode, the task inject the private key into the cryp engine.

> **Danger:** The injector initialization must be done **before** the user initialization

In user mode, the initialization function is the following

```
void cryp_init_user(      enum crypto_key_len key_len,
                    const uint8_t *           iv,
                          uint32_t            iv_len,
                          enum crypto_algo    mode,
                          enum crypto_dir     dir);
```

When a task using the user role, it configures the following:

- **key_len**: the cryptographic key length
- **iv**: The IV value
- **iv_len**: The IV length
- **mode**: The cryptographic algorithm
- **dir**: The cryptographic device direction (encrypting or decrypting)

This function set the Cryp device with the correct values (including IV, IV length, and direction). From now on, if the private key has been set by the injector, the Cryp engine can be used.

Hint: Depending on the way the Cryp device is handled, the IV field may be regulary updated. There is a dedicated API for this

---

> **Caution:** This function can be called multiple time, to change IV, mode, and/or direction

2.1.3 Configuring DMA streams

Now that the Cryp engine has been configured, the DMA controler has to be set up. Here, we only set DMA handlers for each stream. Setting handlers is required in order to schedule correctly the Cryp engine execution in association with the overall data stream:

- The task is responsible for fulfill the input buffer (alone or in association with another task)
- Then, the task launch the Cryp DMA engines to (de)crypt the buffer content
- When the two DMA streams have finished their work, the output buffer is fulfill. The task is responsible for setting a flag when the DMA handlers are called, signifying that the DMA transfers are finished
- The task is then responsible for any usage of the output buffer (storage write, I/O transfer, etc.)

Initializing the DMA handlers to be informed of the DMA transfer terminaison is done using the following API

```
int cryp_init_dma(void *     handler_in,
                  void *     handler_out,
                  int        dma_in_desc,
                  int        dma_out_desc);
```

The DMA initialization uses the following parameters:

- **handler_in**: the address of the handler called at the end of the input Cryp transfer
- **handler_out**: the address of the handler called at the end of the output Cryp transfer
- **dma_in_desc**: input DMA descriptor, set by *cryp_early_init()*
- **dma_out_desc**: output DMA descriptor, set by *cryp_early_init()*

The DMA input and output buffers are set later, at each DMA transfer time.

2.2 Mapping and unmapping the Cryp device

Mapping and unmapping the Cryp engine, when using the CRYP_MAP_VOLUNTARY mode, is done using the following API

```
#include "libcryp.h"

int cryp_map(void);
int cryp_unmap(void);
```

Danger: Don't use any of the libcryp API other than these functions when the Cryp device is not mapped

2.3 Using the Cryp engine

2.3.1 Updating the Cryp data

Until the Cryp engine initialization is done, it is still possible to update:

- the private key (in CRYP_CFG mode)
- the IV (in both CRYP_CFG and CRYP_USER mode)

Updating these fields can be done using the following

```
#include "libcryp.h"

void cryp_set_key(const uint8_t * key, enum crypto_key_len key_len);
void cryp_set_iv(const uint8_t * iv, unsigned int iv_len);
void cryp_get_iv(uint8_t * iv, unsigned int iv_len);
enum crypto_dir cryp_get_dir(void);
```

2.3.2 (de)cyphering content

(De)cyphering data using the Cryp engine is done using two independent API. One using the DMA, the other using direct access. The second one is smaller as the data transfer is made by the task instead of the DMA controller. (De)cyphering data is done using the following API

```
#include "libcryp.h"

int cryp_do_no_dma(const uint8_t * data_in,
                         uint8_t * data_out,
                         uint32_t  data_len);
int cryp_do_dma(const uint8_t * bufin,
                const uint8_t * bufout,
                      uint32_t  size,
                      int       dma_in_desc,
                      int       dma_out_desc);
void cryp_enable_dma(void);
```

cryp_do_no_dma() is used when using the Cryp in direct access mode (without DMA). It (de)cypher data of *data_len* bytes from *data_in* to *data_out*.

When using DMAs, the transfer function to use is *cryp_do_dma()*. This function:

- it get back the two DMA descriptors from the task
- it configure the DMA streams with the given buffers and buffer length

Caution: This function does not start the DMA streams. This is done by the cryp_dma_enable()

Danger: When changing the Cryp engine direction in AES mode (using cryp_init_user()), the private key has to be injected again, as the device drop the key due to internal limitations

The task must wait for the dma_out_handler to be executed to manipulate the output buffer content.

CHAPTER

THREE

CRYP DRIVER FAQ