AES

Link

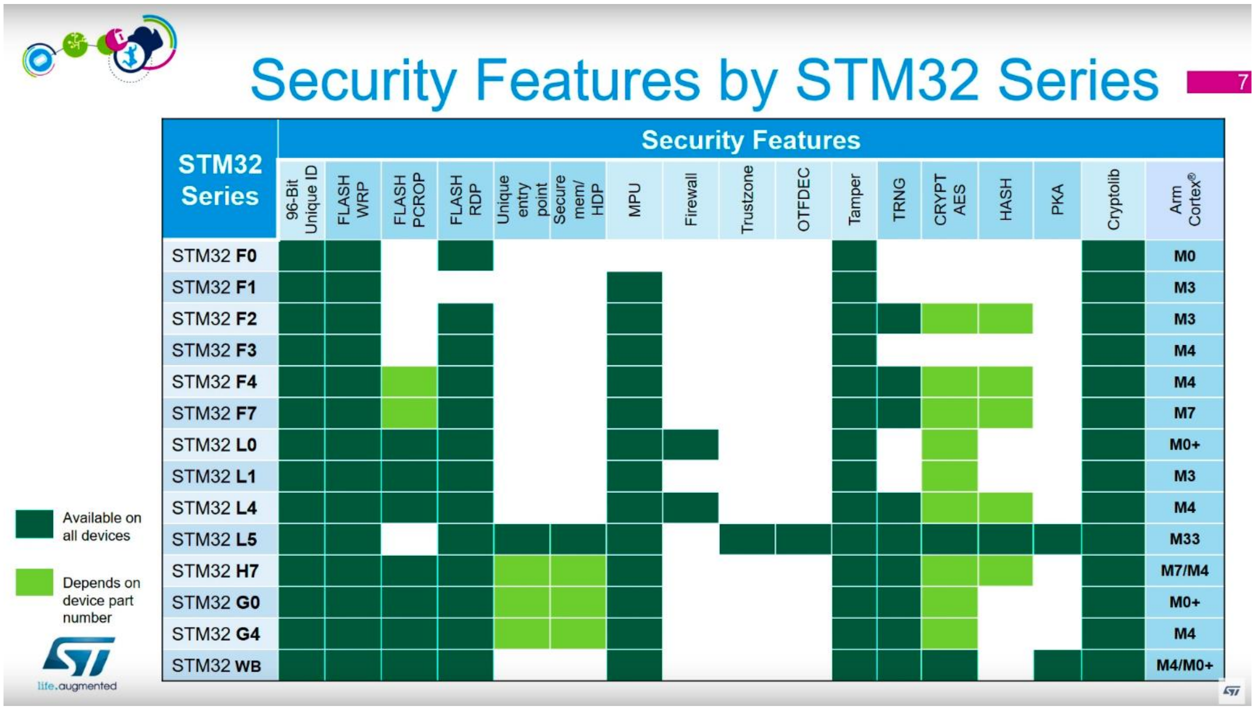
https://www.youtube.com/watch?v=qCuVBD2dmTA&list=PLnMKNibPkDnFzux3PHKUEi14ftDn9Cbm7&index=25

Description

In this paperwork, we will learn how to use symmetric encryption on our board.

Contents

Prerequisites



Security Features by STM32 Series

7

STM32 Series	Security Features															
	96-Bit Unique ID	FLASH WRP	FLASH PCROP	FLASH RDP	Unique entry point	Secure mem/ HDP	MPU	Firewall	Trustzone	OTFDEC	Tamper	TRNG	CRYPT AES	HASH	PKA	Cryptolib	Arm Cortex®
STM32 **F0**	■	■		■							■					■	**M0**
STM32 **F1**	■	■					■				■					■	**M3**
STM32 **F2**	■	■		■			■				■	■	▨	▨		■	**M3**
STM32 **F3**	■	■		■			■				■					■	**M4**
STM32 **F4**	■	■	▨	■			■				■	■	▨	▨		■	**M4**
STM32 **F7**	■	■	▨	■			■				■	■	▨	▨		■	**M7**
STM32 **L0**	■	■	■	■			■	■			■		▨			■	**M0+**
STM32 **L1**	■	■	■	■			■				■		▨			■	**M3**
STM32 **L4**	■	■	■	■			■	■			■	■	▨	▨		■	**M4**
STM32 **L5**	■	■		■	■	■	■		■	■	■	■	■	■	■	■	**M33**
STM32 **H7**	■	■	■	■	▨	▨	■				■	■	▨	▨		■	**M7/M4**
STM32 **G0**	■	■	■	■	▨	▨	■				■	■	▨			■	**M0+**
STM32 **G4**	■	■	■	■	▨	▨	■				■	■	▨			■	**M4**
STM32 **WB**	■	■	■	■			■				■	■	■		■	■	**M4/M0+**

■ Available on all devices

▨ Depends on device part number

life.augmented

STM32 Board

ST-Link cable

STM32CubeMX

STM32CubeIDE

Walkthrough

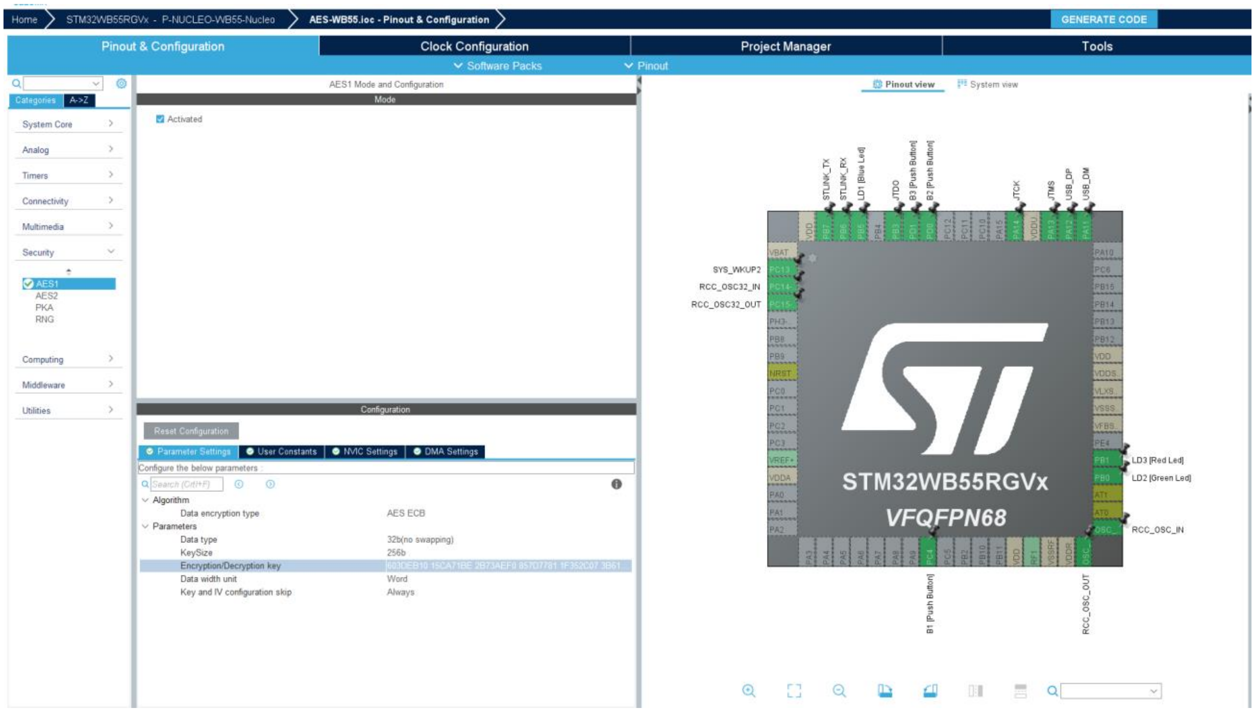
Step 1 : Launch STM32CubeMX and generate the code

Launch STM32CubeMX and select the right board depending on the one you are using. In my case I use the WB55 Nucleo board. Then you can generate the code of your project.

Then you have to activate AES1.

In our example we will use a 256b key size with the following Encryption and Decryption key :

`603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4`



AES1 Mode and Configuration

Mode

☑ Activated

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | DMA Settings

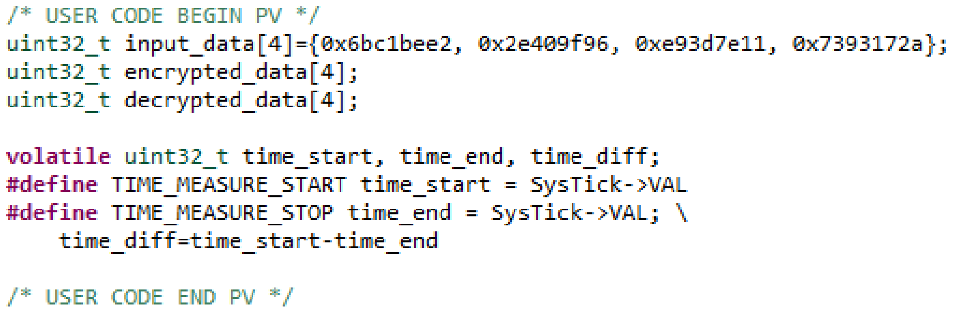
Configure the below parameters :

Search (Crtl+F)

```
Algorithm
    Data encryption type            AES ECB
Parameters
    Data type                       32b(no swapping)
    KeySize                         256b
    Encryption/Decryption key       603DEB10 15CA71BE 2B73AEF0 857D7781 1F352C07 3B61
    Data width unit                 Word
    Key and IV configuration skip   Always
```

Step 2 : Write the main

First thing we have to declare variables in the main.c.
So we will set an input data as an array, and put two empty arrays for the data we will encrypt and decrypt.

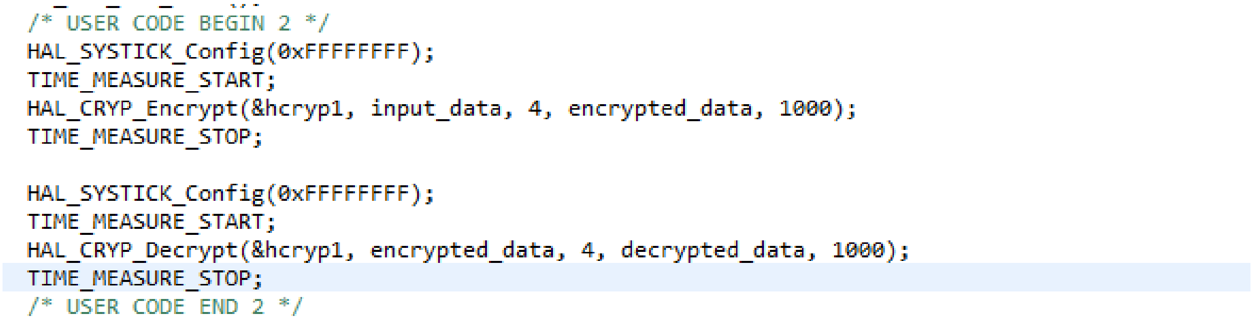


```
/* USER CODE BEGIN PV */
uint32_t input_data[4]={0x6bc1bee2, 0x2e409f96, 0xe93d7e11, 0x7393172a};
uint32_t encrypted_data[4];
uint32_t decrypted_data[4];

volatile uint32_t time_start, time_end, time_diff;
#define TIME_MEASURE_START time_start = SysTick->VAL
#define TIME_MEASURE_STOP time_end = SysTick->VAL; \
    time_diff=time_start-time_end

/* USER CODE END PV */
```

Then we will write in the main the following code. The Measure functions are made to calculate the number of ticks of the clock needed to realize those functions.



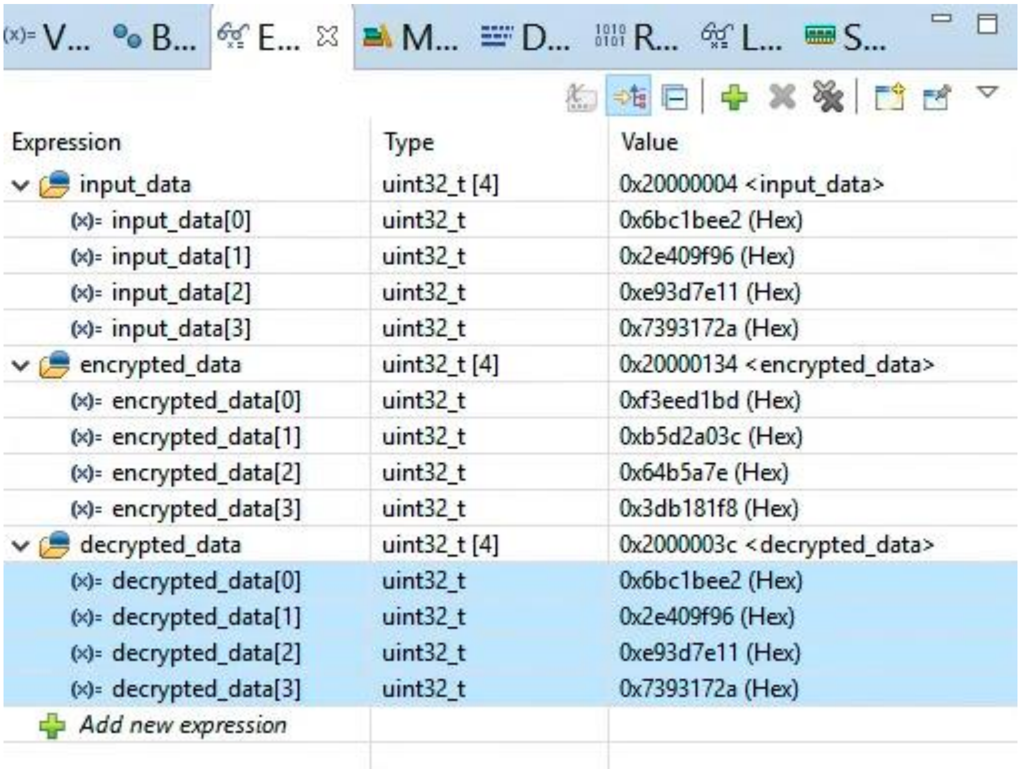
```
  /* USER CODE BEGIN 2 */
  HAL_SYSTICK_Config(0xFFFFFFFF);
  TIME_MEASURE_START;
  HAL_CRYP_Encrypt(&hcryp1, input_data, 4, encrypted_data, 1000);
  TIME_MEASURE_STOP;

  HAL_SYSTICK_Config(0xFFFFFFFF);
  TIME_MEASURE_START;
  HAL_CRYP_Decrypt(&hcryp1, encrypted_data, 4, decrypted_data, 1000);
  TIME_MEASURE_STOP;
  /* USER CODE END 2 */
```

If the code is completed, just compile and launch the debug.

Step 3 : Check the encryption

Finally we just have to put our variables in the debug expressions and check them after the program has been launched.



Expression	Type	Value
input_data	uint32_t [4]	0x20000004 <input_data>
input_data[0]	uint32_t	0x6bc1bee2 (Hex)
input_data[1]	uint32_t	0x2e409f96 (Hex)
input_data[2]	uint32_t	0xe93d7e11 (Hex)
input_data[3]	uint32_t	0x7393172a (Hex)
encrypted_data	uint32_t [4]	0x20000134 <encrypted_data>
encrypted_data[0]	uint32_t	0xf3eed1bd (Hex)
encrypted_data[1]	uint32_t	0xb5d2a03c (Hex)
encrypted_data[2]	uint32_t	0x64b5a7e (Hex)
encrypted_data[3]	uint32_t	0x3db181f8 (Hex)
decrypted_data	uint32_t [4]	0x2000003c <decrypted_data>
decrypted_data[0]	uint32_t	0x6bc1bee2 (Hex)
decrypted_data[1]	uint32_t	0x2e409f96 (Hex)
decrypted_data[2]	uint32_t	0xe93d7e11 (Hex)
decrypted_data[3]	uint32_t	0x7393172a (Hex)

We can see that the encryption and decryption work perfectly because the input data and the decrypted one are the same. As for the encrypted data, we can see that it has totally changed from the input_data.