Cryptography_AES_Demonstration for KIT_T2G-B-H_LITE

Customer training workshop

Q3 2024

Scope of work

- This code example encrypts and decrypts user input data using the AES algorithm with a 128-bit key. The encrypted and decrypted data are displayed on a UART terminal emulator.

- **Device**
  - The TRAVEO™ T2G CYT4BF8CDS device is used in this code example.

- **Board**
  - The TRAVEO™ T2G KIT_T2G-B-H_LITE board is used for testing.

Introduction

- **The cryptography block has following features**
  - Advanced Encryption Standard (AES) functionality according to FIPS 197:
    The AES component can be used to encrypt/decrypt data blocks of 128-bit length and supports programmable key length (128/192/256-bit key).
  - CHACHA20 functionality according to RFC 7539:
    CHACHA20 is a stream cipher, which produces output consisting of 512-bit random-looking bits. These random bits can be XORed with plain text to produce cipher text.
  - Triple Data Encryption Standard (TDES):
    The TDES component can be used to encrypt/decrypt data blocks of 64-bit length using a 64-bit key.
  - Secure Hash Algorithm (SHA) functionality according to FIPS 180-4/FIPS-202:
    This component can be used to produce a fixed-length hash (also called “message digest”) of up to 512 bits from a variable length input data (called “message”). SHA1, SHA2, and SHA3 hashes are supported.
  - Cyclic Redundancy Check (CRC) functionality:
    This component performs a cyclic redundancy check with a programmable polynomial of up to 32 bits.
  - String (STR) functionality:
    This component can be used to efficiently copy, set, and compare memory data.

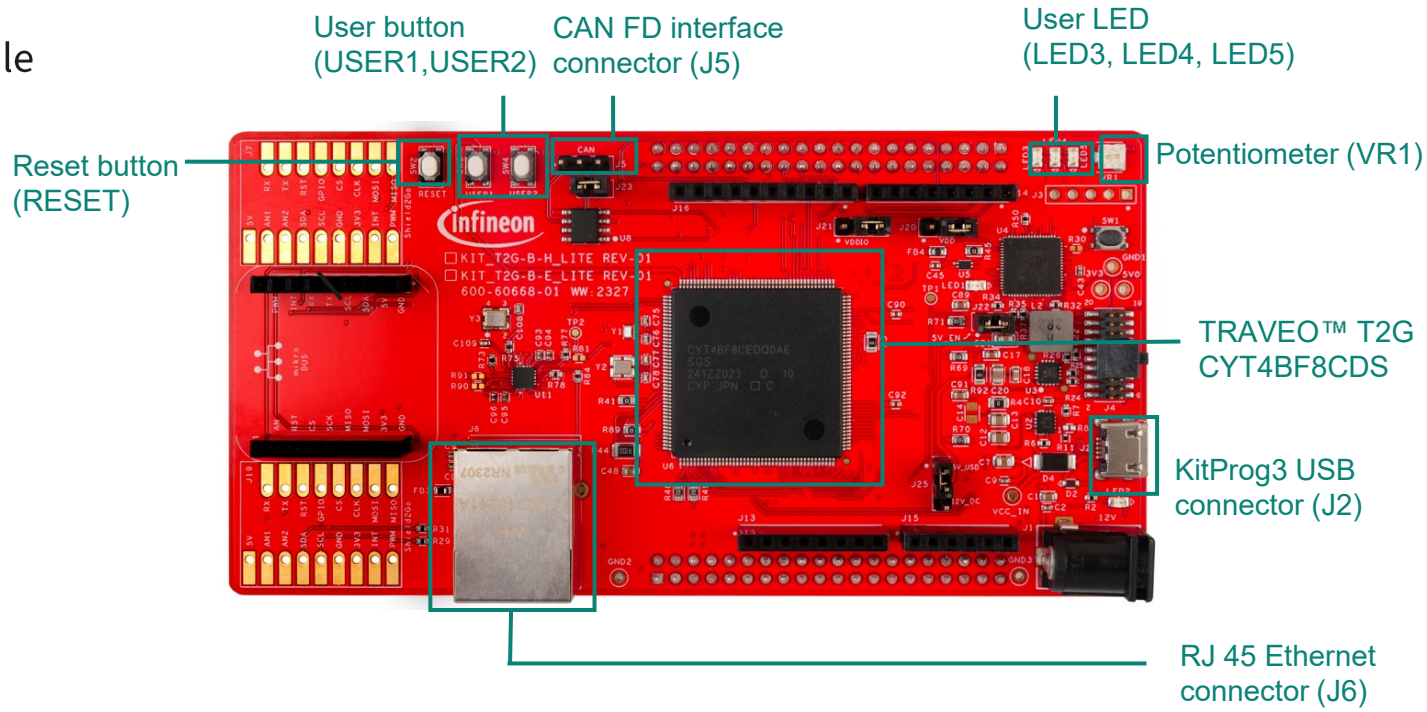
Introduction

- **The cryptography block has the following features**
  - Pseudo Random Number Generator (PRNG):
    This component generates pseudo random numbers in a fixed range. This generator is based on three Linear Feedback Shift Registers (LFSRs).
  - True Random Number Generator (TRNG):
    This component generates true random numbers of up to 32 bits using ring oscillators.
  - Vector Unit (VU):
    This component acts as a coprocessor to offload asymmetric key operations from the main processor.
  - AHB[1] master-interface:
    This allows to fetch operands directly from the system memory.
  - Device key functionality:
    The device key has its usage restricted to specific functionality; they cannot be accessed by the software that implements that functionality.
    Two independent device keys are supported.

[1]:AMBA(advanced microcontroller bus architecture) high-performance bus, Arm data transfer bus

Hardware setup

- This code example has been developed for the KIT_T2G-B-H_LITE board.
- Connect the PC to the board using the provided USB cable through the KitProg3 USB connector (J2).



Implementation

- AES is a symmetric block cipher data encryption algorithm, which means that it uses the same key for encryption and decryption of data. The AES operation works on the 128-bit block size and uses keys of 128 bits, 192 bits, or 256 bits of length. In this example, the user input message is read from the UART terminal and encrypted using the AES algorithm with a key length of 128 bits. The 128-bit encrypted data is displayed on the UART terminal. Then, you can view the decrypted message on the UART terminal and verify that the decryption operation produces the same original encrypted message.
- Press the reset button (RESET) on the kit and enter the message to be encrypted.
  Note that the maximum message size is restricted to 100 characters in this example.

- **Follow these steps to configure this code example:**
  - STDIN / STDOUT setting
  - Display the initial message to terminal
  - Enable the cryptography block
  - Disable data cache
  - Get the message from terminal
  - Encrypt message
  - Decrypt message

Implementation (contd.)

- **STDIN/STDOUT setting**
  - Initialization of the GPIO for UART is done in the *Cy_retarget_io_init()* function
    - Initialize P0.1 as UART TX, P0.0 as UART RX (these pins are connected to the KitProg3 COM port)
    - The serial port parameters change to **8N1** and **115200 baud**

- **Display the initial message to terminal**
  - The terminal can be displayed by ***printf()***
    - Display data is specified as ***CLEAR_SCREEN*** and ***SCREEN_HEADER***

- **Enable the cryptography block**
  - The cryptography block is enabled using the *Cy_Crypto_Core_Enable()* function
    - Enable crypto hardware

- **Disable data cache**
  - Data cache in the CM7 CPU is disabled using the ***SCB_DisableDCache()***[1]
    - This ensures that the cryptograph block can read the string on SRAM entered by the user via STDIN

[1]: The CPU cache operation instructions are provided by Cortex microcontroller software interface standard (CMSIS) with intrinsic functions.

Implementation (contd.)

- **Get the message from the terminal**
  - Getting the character from the terminal is done using the *cyhal_uart_getc()* function
    - This is a blocking call which waits till a character is received or till ***UART_TIMEOUT_MS*** has elapsed
  - UART RX activation is checked by the *cyhal_uart_is_rx_active()* function
    - If RX is active, send the received character to the terminal by calling the *cyhal_uart_putc()* function
      - This is a blocking call, which waits till the character is sent out from the UART completely
    - When the received character is “Enter” key, the message input is completed
    - If the message exceeds 100 characters, the message input is requested again
  - If you want to increase the message size, you can update the ***MAX_MESSAGE_SIZE*** macro in the *main.c* file

Implementation (contd.)

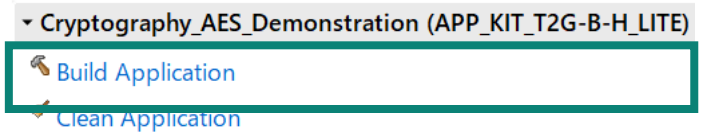
- Encrypt message
  - The message encryption is done using the ***encrypt_message()*** function. This function runs the following:
    - Calculates the number of AES blocks from the message size using ***AES128_ENCRYPTION_LENGTH***
    - Initializes the AES operation using the *Cy_Crypto_Core_Aes_Init()* function
      - Using Key and Key length for initialization
    - Performs AES encryption using the *Cy_Crypto_Core_Aes_Ecb()* function
      - It performs the AES operation on a single block.
    - The *Cy_Crypto_Core_WaitForReady()* function waits for completion of the cryptography operation
      - It waits until all instructions in FIFO are completed
    - Sending a character to the terminal is done using the ***printf()*** and ***print_data()*** function
      - The key used for encryption and the encryption results are displayed on the terminal
    - Change the ***aes_key*** in the *main.c* file to change the key used for encryption
    - After all encryption, *Cy_Crypto_Core_Aes_Free()* clears the AES operation context

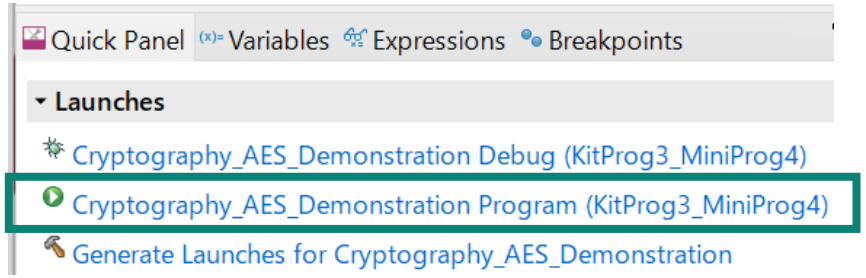
Implementation (contd.)

- Decrypt message
  - The message decryption is done in the ***decrypt_message()*** function. This function runs the following operations:
    - Calculates the number of AES blocks from the message size using ***AES128_ENCRYPTION_LENGTH***
    - Initializes the AES operation using the *Cy_Crypto_Core_Aes_Init()* function
      - Using Key and Key length for initialization
    - Performs AES decryption using the *Cy_Crypto_Core_Aes_Ecb()* function
      - It performs the AES operation on a single block
    - The *Cy_Crypto_Core_WaitForReady()* function waits for the completion of the cryptography operation
      - It waits until all instructions in FIFO are completed
    - Sending the character to the terminal is done using the ***printf()*** function
      - The decryption result is displayed on the terminal
    - After all the decryption, *Cy_Crypto_Core_Aes_Free()* clears the AES operation context

Compiling and programming

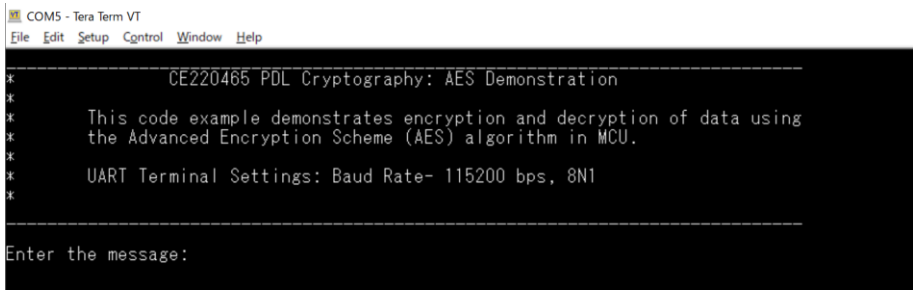
1. Connect to power and USB cable
2. Use Eclipse IDE for ModusToolbox™ software for compiling and programming
3. For compilation:
   a. Select the target application project in the Project Explorer
   b. In the Quick Panel, scroll down, and click **“Build Application”** in the Cryptography_AES_Demonstration (APP_KIT_T2G-B-H_LITE)
4. Open a terminal program (such as Tera Term) and select the KitProg3 COM port. Set the serial port parameters to **8N1** and **115200 baud**.
5. For programming:
   a. Select the target application project in the Project Explorer
   b. In the Quick Panel, scroll down, and click **“Cryptography_AES_Demonstration program (KitProg3_MiniProg4)”** in Launches

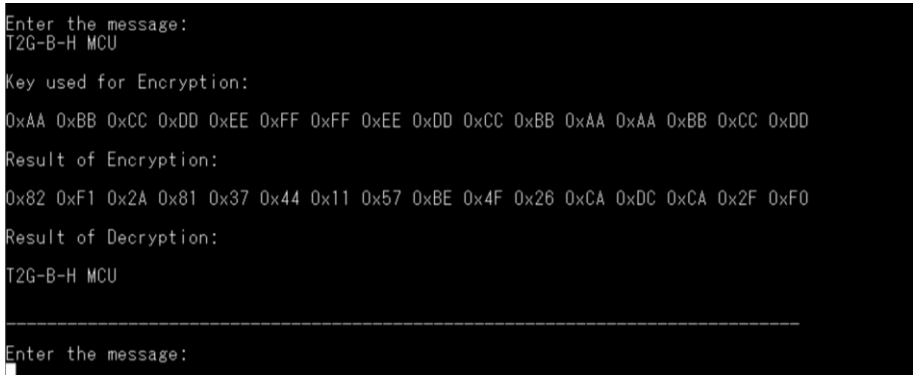




Run and test

1. Once the programming is successfully complete, a message is displayed in the terminal window as shown in the figure.
2. When you input the message for encryption, the “Enter” key is pressed, and displays the key used for encryption, encryption result, and decryption result on the terminal.





References

- Datasheet
  - CYT4BF TRAVEO™ T2G 32-bit Automotive MCU based on Arm® Cortex®- M7 dual
- Architecture reference manual
  - TRAVEO™ T2G Automotive MCU body controller high architecture reference manual
- Registers reference manual
  - TRAVEO™ T2G Automotive MCU: TVII-B-H-8M body controller high registers reference manual
- PDL/HAL
  - Peripheral driver library (PDL)
  - Hardware abstraction layer (HAL)
- Training
  - TRAVEO™ T2G training

Revision History

Revision	ECN	Submission Date	Description of Change
**	7782101	2022/07/05	Initial release
*A	8067284	2024/08/27	Replaced development board from KIT_T2G-B-H_EVK to KIT_T2G-B-H_LITE