Embedded Security Showcase on PSoC64 (ESSOP)

Analysis and Design Decisions

Author: Aadarsh Kumar Singh, Embedded Systems and Microelectronics

Status: Released

Revision	Date	Editor	Reason
1.0	25.08.2020	Aadarsh Kumar Singh	Design Decisions

Contents

1. Hardware Design Decisions 2

1.1 Hardware features essential in a MCU to fulfill the specified security requirements........ 2

1.2 Choosing PSoC64 MCU for student car and remote control 2

2 Software Design Decisions........ 3

2.1 Selecting Mbed OS library........ 3

2.2 Cryptographic Algorithms to be employed for the given use case........ 3

2.2.1 Digital Signatures 4

2.2.2 Encryption/Decryption........ 4

2.2.3 Secure key exchange........ 5

3 Security Library and their version 6

1. Hardware Design Decisions

1.1 Hardware features essential in a MCU to fulfill the specified security requirements

The MCU of the car and remote control should have all the required security capabilities for supporting the software to establish secure communication between them. The required security capabilities are listed below:

- **E-Fuses**:
  - The MCU should support e-fuses for secure storage of data assets like unique identification numbers (UID), manufacturer numbers that are exclusive to the device and never change throughout lifetime of the product.
  - An example use case where e-fuses are required is described below:
    For verifying the firmware authenticity, we use keys to sign the firmware. These keys are unique for the student car/remote and it has to remain the same throughout the lifetime. Unauthorized actors must not be allowed to overwrite them. This can be ensured by using e-fuses. Once the keys are written to the secure key storage section e-fuses are blown off, nobody can now change the keys that are flashed into this memory region.

- **Hardware Accelerator for cryptographic algorithms:**
  - The cryptographic algorithms are extensive and require complex mathematical computation. MCU must have hardware accelerator to run these algorithms for high performance in terms of speed.

- **Isolated execution environment for trusted applications:**
  - It is necessary that the MCU offers a strong hardware based isolation between unsecure and secure environment to protect the crucial data assets like protocol data, cryptographic keys, and firmware credentials from being tampered. They must be stored in a secure manner such that it is only accessible to the authentic actors.
  - The best way to isolate the secure environment from non-secure environment is to use multicore MCU with a dedicated core for performing all the cryptographic procedures, storing cryptographic keys and the other core used for performing other non-secure operations. It has to have a secure inter core communication. Only authorized actors would have permission to execute operations/functionalities on the secure core.

1.2 Choosing PSoC64 MCU for student car and remote control

The student car and the remote control uses a PSOC5 MCU. The PSoC5 lacks the following security capabilities:

- PSoC5 is a single core MCU that does not have dedicated hardware accelerators necessary to run cryptographic algorithms like ECDSA, AES etc.
- There is no isolation between the secure and non-secure environment. Hence, it does not have hardware support to allow only the authorized actors to execute processes on the device.
- E-fuses are not present, hence immutable data assets that are uniquely associated with the device can be overwritten by unauthorized sources.

Hence, we need to change the MCU so that it has the required security capabilities. We have chosen PSoC64 MCU as it has:

- **Isolated execution environment:**
  - PSoC64 is a dual core controller with a dedicated core (HSM) for performing all secure /cryptographic operations and cortex-M4 for performing unsecure/high performance operations.
  - Inter-Processor Communication (IPC) channels between the Arm Cortex-M4 and Cortex-M0+ cores are provided to support isolated API-based interaction.

    - An example is secure boot, once the firmware is flashed, cortex M0+ (secure HSM core) verifies the authenticity of the firmware then allows it to boot. If the attacker flashes a false firmware into the device, the HSM core will not allow it to boot. Hence, only authorized actors will be able to run the applications on secure HSM core. Only when the secure core allows it can run applications on the cortex-M4 core.

- **Integrated secure element functionality:**
    - It has Memory protection units (MPU), peripheral protection units (PPU) that allows cores to access only the permitted memory regions/peripherals. In addition, PSoC64 provides a framework to add secure services such as key storage, cryptography, etc.

- **Hardware accelerator for cryptographic algorithms**
    - The hardware accelerator are used for running cryptographic algorithms like AES, 3DES, RSA, ECC, SHA-256, SHA-512 and True Random Number Generator (TNRG) to improve the performance of the system in terms of speed.

- **E-fuses**
    - It has e-fuses support for secure storage of immutable data assets.

2 Software Design Decisions

2.1 Selecting Mbed OS library

As per the hardware design analysis, the car and the remote will use PSoC64 as the MCU. To develop an application for establishing secure communication between, we require a software library that contains all the necessary device drivers of PSoC64. In addition, it should contain a security library which is compatible with the hardware accelerators present in PSoC64 for running cryptographic algorithm and provide device driver for BLE communication.

- Mbed-OS is an open-source embedded operating system, which supports PSoC64 MCU. As mbed –OS is suitable for PSoC64 , we had evaluated the software library based on our requirements:
    - Modular:
        - Device driver for all the supported PSoC64 peripherals such as digital and analog IO, interrupts, port and bus IO, PWM, I2C, SPI,DMA and serial are present. It is compatible with wide range of software libraries for RTOS, sensors and thread safety support.
        - Mbed OS contains all the supported toolchain libraries of PSoC64. It is easy to integrate Mbed-OS into a C/C++ project running in eclipse IDE.
    - Secure :
        - Supports mbed-TLS security library that is compatible with hardware accelerators present in the PSoC64 for running the cryptographic algorithms.
        - Supports ARM Platform Security Architecture (PSA) framework.
    - Connected:
        - Driver support for Bluetooth Low Energy (BLE) present, since the communication between the car and the remote control takes place via BLE.

2.2 Cryptographic Algorithms to be employed for the given use case

- The key used for signing the firmware (secure boot to avoid firmware abuse), the keys used for the signing the credentials (to avoid impersonation) and keys used for encryption must be different. This is necessary because if same key is used to avoid the threats, compromising the confidentiality of the keys can lead to multiple/all security threats.

- Asymmetric cryptographic algorithms must be used for generating digital signatures whereas symmetric cryptographic algorithm for encryption/decryption. The reason is:
  - The cryptographic algorithm used for generating signature must be asymmetric to ensure non-repudiation, i.e. the actors (OEMs) signing the firmware (or a message) are held credible for the firmware that is flashed.
  - Encryption can be done using symmetric cryptographic algorithm, since the credibility of the actor/source is verified during authentication. It is only required to encode the original protocol such that only the authorized device is able to decode it.

2.2.1 Digital Signatures

- Digital signatures has to be used for validating the authenticity and integrity of the
  - firmware prior to boot/upgrade
  - protocol data in- transit from the sender to the receiver

- Elliptic Curve Digital Signature Algorithm (ECDSA) algorithm is used for generating digital signature. Before generating the digital signature of a data (firmware/protocol), it is hashed to reduce the size of the generated digital signatures. SHA-256 is used as hashing algorithm in the application.

- The size of the cryptographic keys are 32 bytes and the size of generated signatures are of 64 bytes.

- ECDSA algorithm was employed because :
  - The crypto hardware accelerator present in the PSoC64 MCU supports it.
  - It is asymmetric and offers same level of security in comparison with other algorithms like RSA but with smaller key lengths. The algorithm for generating the signature becomes faster as the computations involves smaller keys. Small public keys require less data to pass around to establish secure connections. This implies faster connection rate.

2.2.2 Encryption/Decryption

- Application will use encryption to avoid man in middle attack aimed at tampering the in-transit data.

- To encrypt the protocol data we need to use AES encryption algorithm.
  - The main advantage is its speed. AES is symmetric key algorithm hence it requires less computational power than asymmetric one making it faster and more efficient to run.
  - The crypto hardware accelerator present in the PSoC64 MCU supports it.

- AES algorithm has three types: AES-128, AES-256, AES-192 based on the key size. We use AES-128 for encryption. This implies the size of the key will be 16 bytes. The reason to use AES-128 is that larger key size results in more computations which makes encryption process slower. AES-128 has the smallest key size and it offers the optimal security level for our use case.

- There are two cipher types namely block ciphers and stream ciphers. In case of block ciphers, encryption is performed block by block, whereas stream ciphers encrypt one bit/byte at a time. The advantages and disadvantage of each cipher mode types are tabulated below:

Cipher Mode Types	**Advantages**	**Disadvantages**
Stream Cipher	Faster than block ciphers	It offers low diffusion since all information of a plaintext symbol is contained in a single cipher text symbol.
	Low memory requirement than block ciphers.	It is vulnerable to modifications as any interceptor who breaks the algorithm might insert spurious text that looks authentic.
Block Cipher	It offers high diffusion since information from one plaintext symbol is diffused into several cipher text symbols.	Slower than stream ciphers as computations are more complex
	It has stronger immunity to tampering.	Requires more memory

For the application, we will use AES in CBC block cipher mode for performing encryption/decryption with block size of 16 bytes for AES-128/ AES-192/ AES-256 encryption. Block cipher were chosen as it had high diffusion and strong immunity to tampering as compared to stream ciphers. To speed up the rate of encryption we will use hardware accelerator present in PSoC64.

- Main advantage of CBC mode is that decryption can be performed in parallel. It uses Initialization vector (IV) to hide data patterns while encrypting the message. IV is randomly generated number having same size as the block size, its role is to make each block of encrypted message unique.

2.2.3 Secure key exchange

- As we use symmetric encryption, the receiver has to use the same key which was used for encryption to decrypt the ciphered text to plain text. In addition, Initialization vector has to be sent to the receiver in order to decrypt each block without any mismatch of original plain text and decrypted text.

- The key and the IV has to have a cryptographic basis for exchange between the car and remote control. Hence, we require a secure key exchange algorithm. This application uses elliptic-curve Deffie–Hellman (ECDH) key exchange algorithm.

- The crypto hardware accelerator present in the PSoC64 MCU is compatible with Elliptic-curve Deffie–Hellman (ECDH).

- The car and the remote generates a 32-byte Deffie–Hellman key using Elliptic Curve 25519. The generated key has to be exchanged with the corresponding communication partner. The partner will use the exchanged key to compute a secret key of 32 bytes. First 16 bytes will be used as encryption keys and the next 16 bytes could be used as Initialization vector.

3 Security Library and their version

- We use Mbed-OS 5 software library with version Mbed-OS 5.14.
- It has mbed-TLS security library with the version mbedtls-2.21.0.
- The architecture of PSoC64 MCU is complaint to Arm Platform Security Architecture (PSA). The mbed OS uses PSA Cryptography library, which internally uses the mbed-TLS library to builds services such as secure boot, secure storage and secure inter- core communication compliant to Arm Platform Security Architecture (PSA).
- The PSA library 1.0.0 will be used for developing services for encryption, decryption, diffie-hellman key exchange signature generation and verification.