Embedded Security Showcase on PSoC64(ESSOP)

PSoC64_SecureBoot

Author: Aadarsh Kumar Singh, Embedded Systems and Microelectronics

Status: released

Revision	Date	Editor	Reason
1.0	05.01.2020	Aadarsh Kumar Singh	Created for understanding the security of PSoC64

Contents

1. Basics of Embedded Security

1.1 Security Services for Embedded Systems

Objective of the security systems are called security services. There are 4 major services for Embedded Security:

- Confidentiality
- Authenticity
- Integrity
- Non-repudiation

1.2 Usage of the services in Embedded World

1.2.1 Confidentiality:

- To establish secure communication with two entities via an unsecure channel without any unauthorized source being able to hack it.
- Implemented using Encryption/Decryption.

1.2.2 Authenticity:

- To authenticate the sender is the original owner of the product and allows others to verify that the product is original.
- Implemented via Digital Signatures.

1.2.3 Message Integrity

- Message has not been modified/tampered.
- Digital Signature implementation ensures integrity.

1.2.4 Non- repudiation

- The sender of the message can't deny the creation of the message.
- This service ensures that the sender is held credible for the message which is sent.
- Ensured by using asymmetric keys.

1.3 Threats based Analysis for Embedded Security Use Case – CIA Model

- The CIA Model stands for confidentiality, integrity and authenticity.
- We identify the data and system assets of the product which we want to secure.
- We identify all sources of threats to these data and system assets
- Each threat is assigned a data security objective (confidentiality, integrity and authenticity).

1.3.1 Use case Analysis: The marksheet of a student.

- The principal needs to authenticate that the certificate provided to the student, is by the college and is original.
- The student should not be able to tamper the document.
- The student should be capable to prove to recruiters that he has got the authentic certificate from the college.

We perform the CIA Analysis for the above use case:

- we require to provide authenticity, integrity and Non-repudiation and not confidentiality.

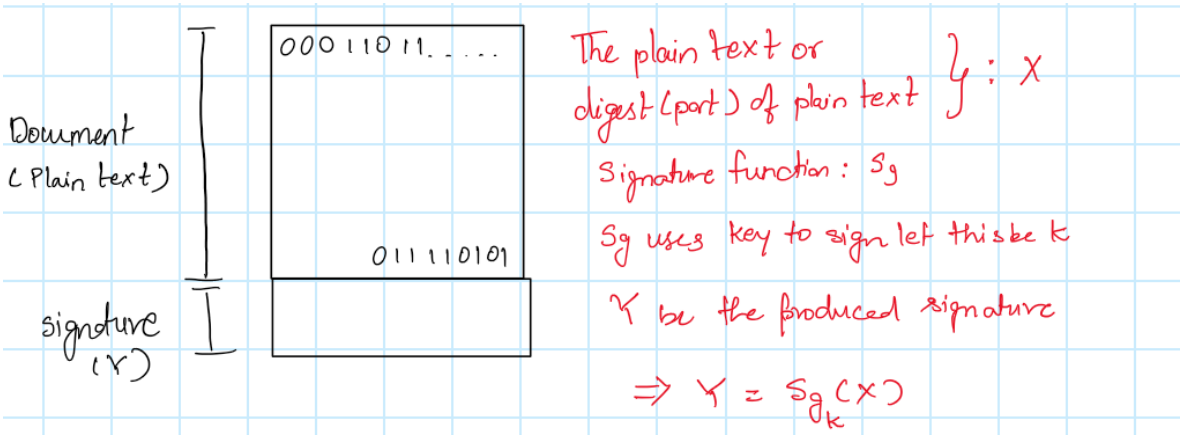
- Since all recruiters should be allowed to view the marks of the student on the marksheet, hence it should not be confidential but the recruiters should be able to verify that the certificate is original and is not tampered.

Threats:

- The manual signature can be copied and student can even tamper the certificate.
- From the services mentioned in the section 1.2, we require to digitally sign the marksheet for authenticity and integrity.

1.4 How Digital Signature Ensures Authenticity and Message Integrity?

- A document in digital world is a binary file, we call the binary file as plain text.
- We take the plain text (usually digest of the plain text) and pass it to the cryptographic function (e.g. RSA).
- This function uses a given key to create a unique signature for the binary file.

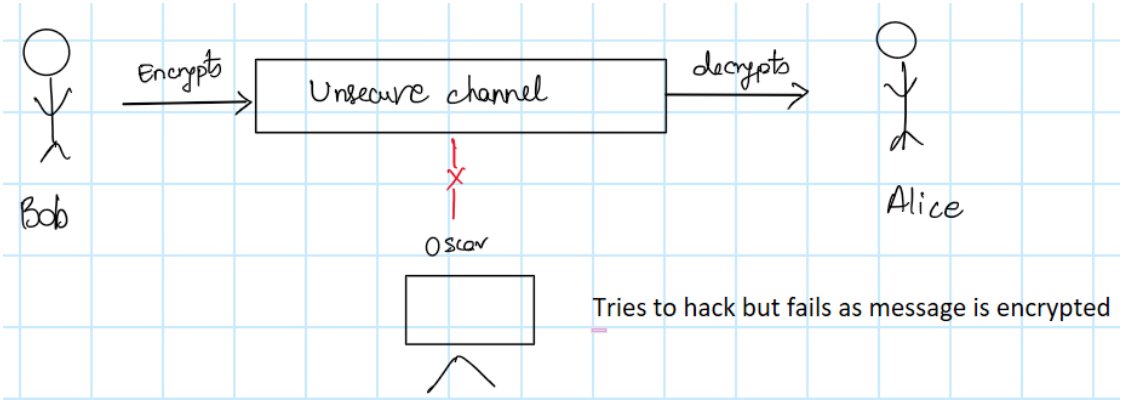


The sender sends the document with the signature to the receiver, the receiver uses the same key which is used for signing the binary and verifies the document. The result of the verification is single bit either Pass or Fail.

How does this ensure authenticity and integrity?

- The owner of the message is the only person who has the key, by using this key he can generate a document with a unique signature. The user can get the key from the owner and verify the signature. If the owner is authentic only then the signature will pass or it will fail.
- The message for a given key will have a unique signature, hence if somebody tries to modify it the signature wont match while verifying, hence integrity comes along with digital signature.

1.5 Encryption/Decryption



For encryption of the message also we use Cryptographic function which require keys. We use a key to encrypt the message using algorithms (e.g. AES-256) and the receiver decrypts the message using the same key.

1.6 Asymmetric keys for Non-Repudiation

- If a cryptographic function uses a single key for signature generation/verification or Encryption/Decryption it is called Symmetric keys.
- If a cryptographic function uses a pair of keys for signature generation/verification or Encryption/Decryption it is called asymmetric keys.
    - Each pair of key set has a public key and a private key.

1.7 Why asymmetric keys are required?

- Asymmetric keys are required for the ability to ensure that a party to a contract or a communication cannot deny the authenticity of their signature on a document or the sending of a message that they originated.

1.7.1 Illustration why symmetric key fails?

- If a customer gives a request to a vendor for producing a customized car. Let's say the customer signs the request document using a symmetric key. The vendor verifies the request and validates that the customer is authentic by using the same key as the customer.
- After production of the vehicle the customer says he doesn't want the car. The vendor can legally sue the customer by showing the digitally signed certificate of the customer, but the customer can win this argument by saying that the vendor also has the same key hence he would have created his own requirement which the customer never asked , this shows the weakness in the symmetric key model. Hence generally asymmetric keys are used.

2.Rule of Thumb for using asymmetric keys

Both Encryption/Decryption and signature generation/verification requires a set of keys: public key and private key. The private key is kept secret and public key can be shared with others.

2.1 Encryption/Decryption:

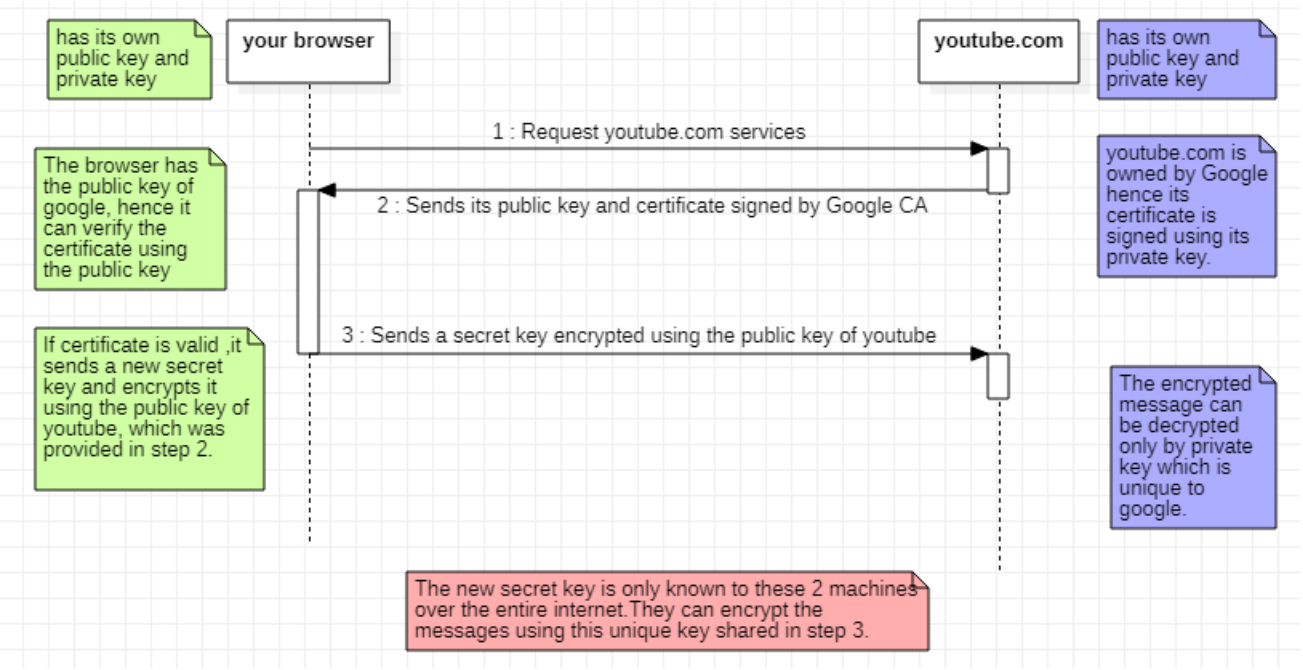
- Encryption can be done only using the public key.
- Decryption can be done only using the private key.
  Analogy: Anybody can put the mail in the postbox only the postman can perform the decryption.
  Meaning: Anybody having the public key can encrypt, but only the person having the private key can decrypt the message.

2.2 Signature / Verification:

- Signature generation requires private key.
- Signature verification can be done using public key.
  Analogy:
- When documents are signed by hand, the signature of the creditor unique to him is used, likewise for digital signature private key unique to the owner is used for signing the data.
- Anybody (Public) who has seen the hand signature of the person can verify the document, likewise the public key can be used to verify the document

3. HTTPS – Security Showcase Illustration for understand basics

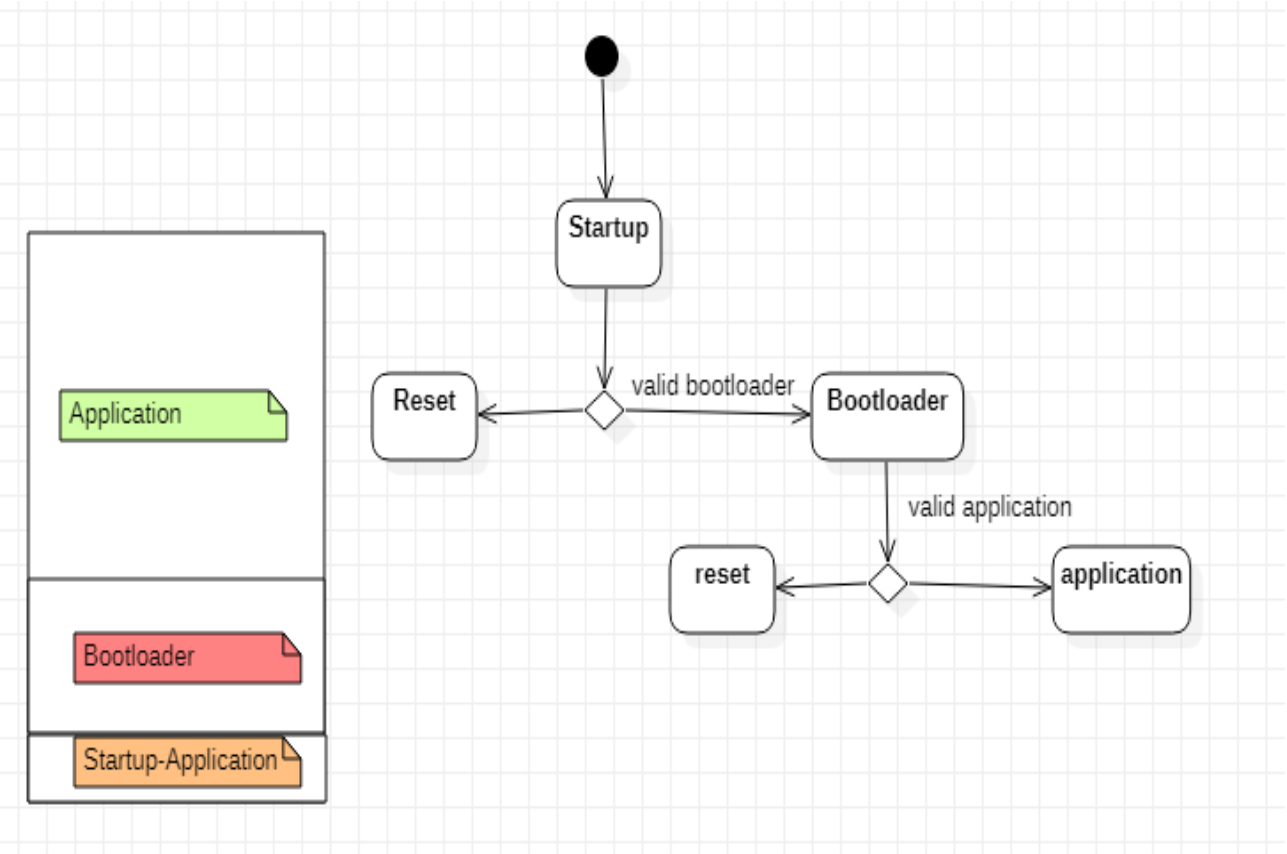
Note: Knowing the Rule of thumb (Section 2) is a must



4. PSOC64 – Security Line (Typical secure boot vs PSoC64 Secure boot)

4.1 Typical secure boot

A typical hardware design of security can be seen below:



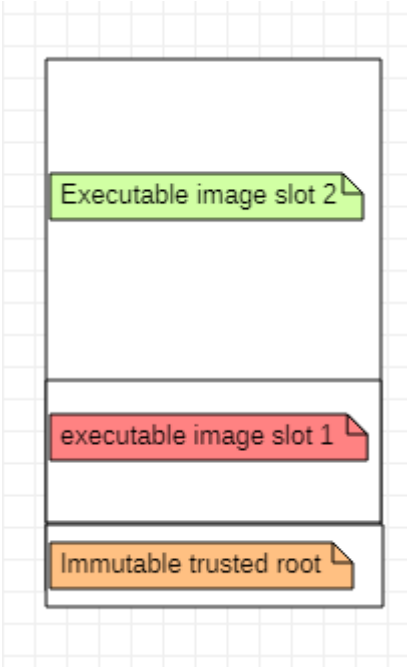
- The startup application validates the Bootloader, the bootloader validates the application.
- The authenticity of the bootloader is checked using a hash value calculated over the bootloader and comparing it with the hash stored in one time write memory location(**eFuse**).
- The bootloader then verifies the application binary can be checked only using digital signature.
- We create a set of key pair: public key and private key.
- We can sign the application using private key and flash it, but for the bootloader to verify the validity of the application, public key corresponding to the private key used for signing has to be pre-flashed.
- Only if the public key is pre-flashed, we can use a software to run some cryptographic function and use the pre-flashed public key to verify the application/bootloader which is going to be flashed.
- Hence, the public key has to be flashed during the production of the product.
- **Threat**: We hand over your public key, as well as a signed image, over to the concerned Contract Manufacturer (CM) and ask them to program both. In this process, there is no cryptographic basis by which you can check that your key got into the device.
- It is essentially "uncontrolled" where you trust your CM not to change this key out and put a root-kit before launching your image.
- This was traditional way of implementation.

4.2 PSoC64 Secure boot

The PSoC64 ensures security using 2 key steps:

4.2.1 Immutable Hardware Root-of-Trust and Services

- The PSA approach defines that an MCU must start up in a trusted manner based on the root-of-trust. Hence, we require an immutable HW based Root of Trust.
- A section of ROM code(Immutable trust root) will calculate a hash over the next section of flash executable code(image slot 1) and compare that hash with one that is stored in a write-once memory location (eFUSE).
- This is same in PSoC 62.



4.2.2 Secure operating Environment

- The Secure operating Environment has different approach, the fundamental difference comes in how the OEM root public key pair gets injected into the device, which is done by provisioning discussed below.

5 Provisioning of PSoC64

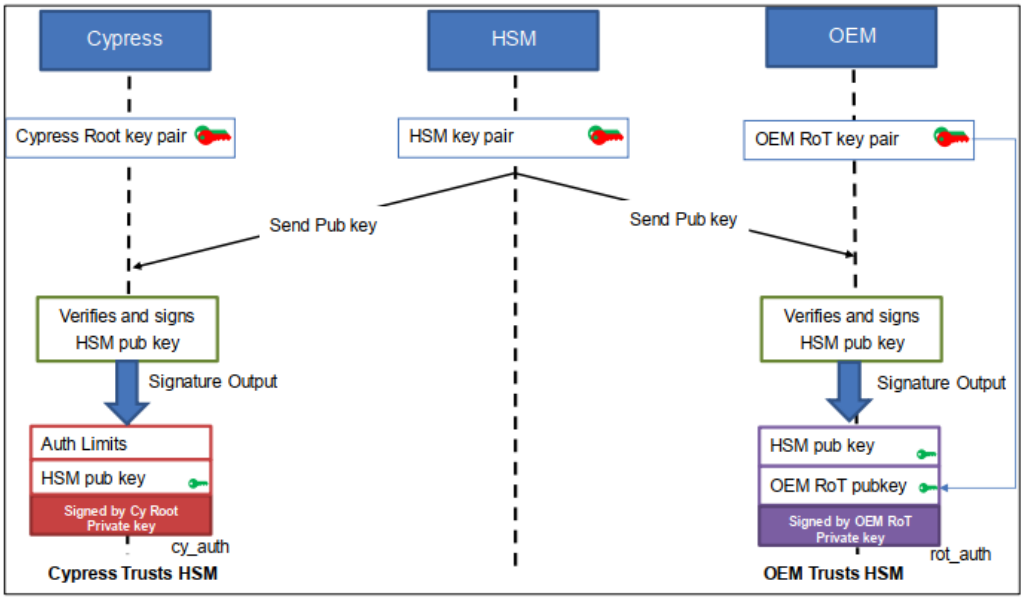
Provisioning is a process whereby secure assets like keys and security policies are injected into the device. This step typically occurs in a secure manufacturing environment that has a Hardware Security Module (HSM). This Involves Two steps:

5.1 Transferring the RoT from Cypress to the development use

The RoT transfer process can be represented as exchange between the following entities:

- Cypress – The owner of the Cypress Root private key.
- Secure Manufacturing environment HSM – **entity authorized to provision and program PSoC 64**.
- OEM/Developer – The user/code developer of the part.
- PSoC 64 Secure MCU – The holder of the Cypress Root public key

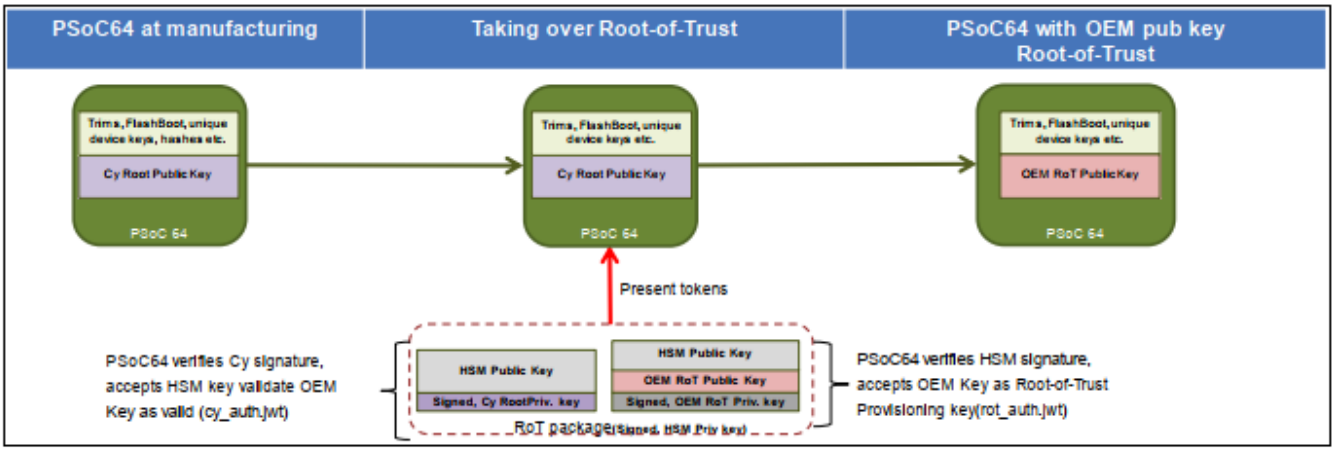
5.1.1 Step 1: Both Cypress and OEM authorizes the HSM



Result of this step is:

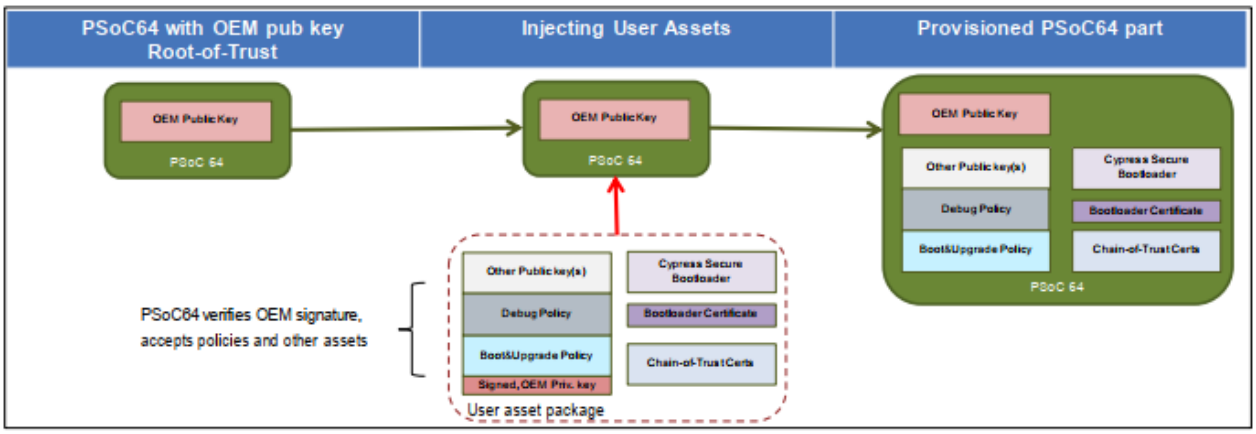
- cy_auth JWT: Contains the public key of the HSM to be trusted. Additional fields such as an expiration date can be specified to limit this token's usage.
- rot_auth JWT: Contains the public key of the HSM to be trusted as well as the OEM RoT public key to which the RoT must be transferred

5.1.2 Provisioning the PSoC64



5.2 Injecting user assets

User assets such as image-signing keys, device security policies, and certificates into the device required are injected after the provisioning is done.



6. References:

- https://www.cypress.com/file/495821/download
- https://www.cypress.com/file/480976/download
- https://www.youtube.com/watch?v=T4Df5_cojAs