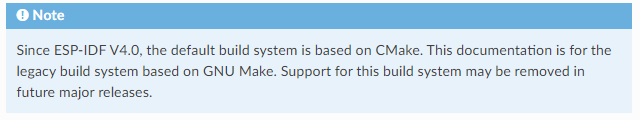
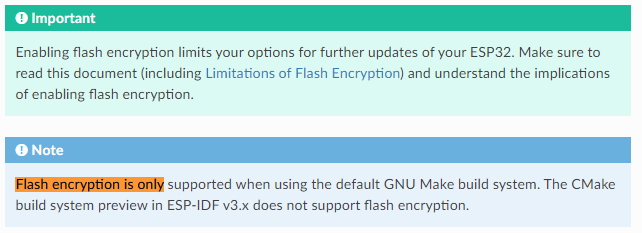
**ESP32 SECURITY REFERENCE MANUAL**

|  |  |  |  |
| --- | --- | --- | --- |
| Rev. | Rev. date | Author | Note |
| *0.1* | *17/06/2020* | *A.Chiebao* | 1st draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Premesse flash Encryption in ESP32**



<https://docs.espressif.com/projects/esp-idf/en/release-v3.3/security/flash-encryption.html>



**Signed App (sicurezza base)**

Esistono due modalità relative alla marcatura dell’app attraverso una chiave:

1. La prima modalità indicata da espressif, chiamata “Remote Signing of Images”, è strettamente legata alla produzione in un eventuale plant. Oltre a essere composta da due chiavi (una privata e una pubblica) richiede la presenza di un server remoto contenente la chiave di firma anziché averla all’interno del pc di compilazione.

Questa modalità non è stata provata.

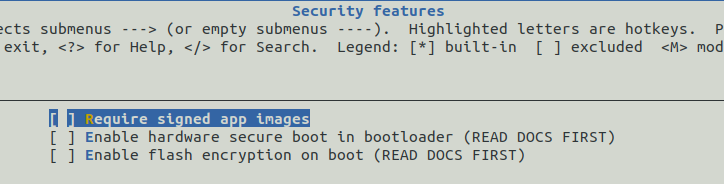
2) La seconda modalità, più semplice, permette di ”segnare” il binario generato con una chiave privata aggiunta in coda al file stesso.

Nella cartella Build del progetto saranno dunque presenti due file. Nel caso del Gateway

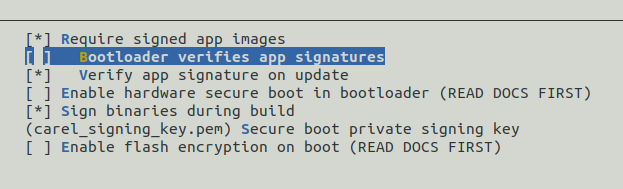
Troveremo “gme\_carel.bin” (binario segnato) e “gme\_carel-unsigned.bin”

I passi per abilitare tale sicurezza vengono di seguito riportati

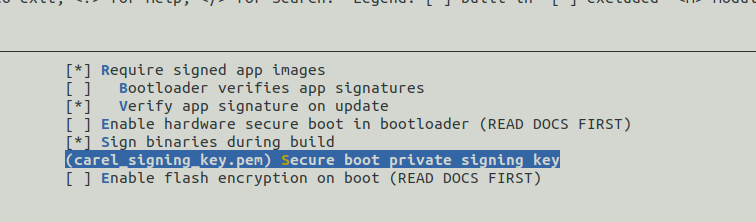
1. Da make menuconfig 🡪 Security features 🡪 abilitare “Required signed app image”



1. Con il menù espanso saranno presenti varie opzioni, bisogna abilitare “Bootloader verifies app signatures”



1. In “Secure boot private signing key” indicare la chiave con la quale si vuole marcare il file



Il comando per generare una chiave randomica è il seguente:

*python ~/esp/esp-idf/components/esptool\_py/esptool/espsecure.py generate\_signing\_key carel\_signing\_key.pem*

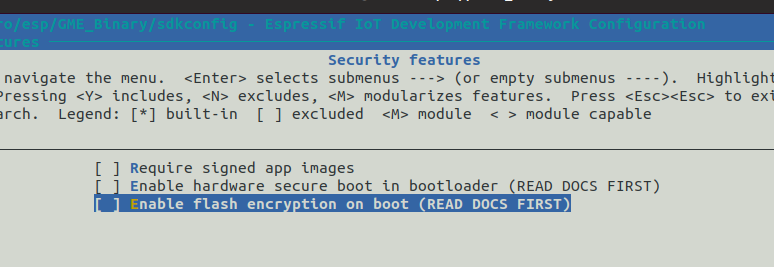
***Verifica durante l’update*: quando questa opzione è abilitata la firma del binario viene verificata in fase di update ota (all’interno del file esp\_ota\_ops.h). Questa opzione aggiunge sicurezza nei confronti degli attacchi via OTA.**

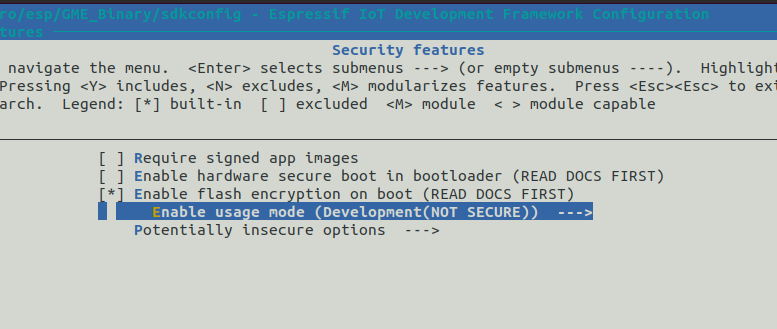
**Verifica durante il boot: quando questa opzione viene abilitata il bootloader dopo la compilazione conterrà il codice di verifica per controllare il binario prima di flashare.**

**(Entrambe le opzioni sono abilitate di default se si usa il secure boot, mentre la verifica della firma nel caso il secure boot non sia abilitato non viene ritenuta significativa ai fini della sicurezza)**

**Enable flash encryption on boot**

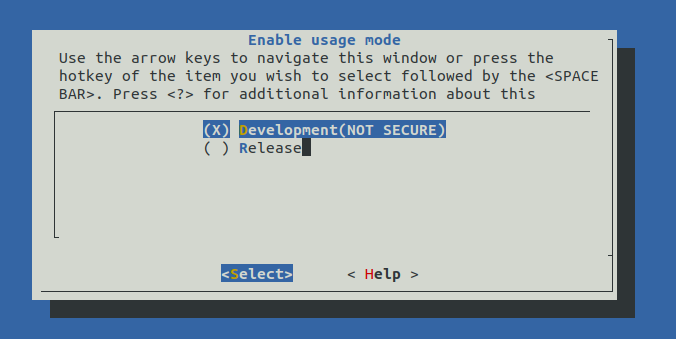
Nel caso del GME si utilizzerà l’encryption della flash durante il boot ( ma senza secure boot!)





In enable usage Mode, in fase di produzione si andrà a selezionare “release”

Durante lo sviluppo rimane selezionato “Devbelopment”



Una volta settato il sistema via make menuconfig si può procedere in due modi:

1. Flashare tutto e lasciare che sia il boot durante il primo avvio a generare la chiave con la quale verranno cifrate tulle le parti della memoria che devono essere cifrate.

(ogni chip genera la sua chiave univoca)

1. Generare tramite il seguente comando una chiave

*espsecure.py generate\_flash\_encryption\_key carel\_encryption\_key.bin*

e poi scriverla nei fuse bit prima di flashare il codice (boot + app + etc..)

*python esp-idf/components/esptool\_py/esptool/espefuse.py --port /dev/ttyUSB0 burn\_key flash\_encryption carel\_encryption\_key.bin*

All’avvio il boot userà questa come chiave e non ne genererà una univoca per ogni chip.

Attualmente per rendere sicuro e cifrare il fw del GME si procede con il seguente modo:

* Iniettare la key all’interno del modulo (one shot…una volta inserita non si torna indietro)
* Utilizzare CMAKE per flashare il boot (l’unico esempio per ora provato e funzionante è

*esp/esp-idf/examples/security/flash\_encryption)*

* Successivamente cifro con la chiave CAREL il binario generato dal build e ne creo uno cifrato

*python ~/esp/esp-idf/components/esptool\_py/esptool/espsecure.py encrypt\_flash\_data --keyfile carel\_encryption\_key.bin --address 0x10000 -o build/gme-enc-app.bin build/gme\_carel.bin*

* Si scrive in flash il binario cifrato generato precedentemente

*python ~/esp/esp-idf/components/esptool\_py/esptool/esptool.py --chip esp32 --port /dev/ttyUSB0 --baud 921600 write\_flash -z 0x10000 gme-enc-app.bin*

* la stessa cosa fatta sopra la si fà con la partition table (cambia l'indirizzo)

*python ~/esp/esp-idf/components/esptool\_py/esptool/espsecure.py encrypt\_flash\_data --keyfile carel\_encryption\_key.bin --address 0x8000 -o gme-enc-tab.bin gme\_part\_tab.bin*

*python ~/esp/esp-idf/components/esptool\_py/esptool/esptool.py --chip esp32 --port /dev/ttyUSB0 --baud 921600 write\_flash -z 0x8000 gme\_part\_tab.bin*

* Lo spiffs non rientra tra le partizioni che vengono cifrate, va dunque flashato come si è sempre fatto.

NOTA BENE

Portare il livello di informazioni del LOG del BOOTLOADER a livello ERROR in modo da ridurre le dimensioni de bootloader stesso.

