Soluzioni Hardware e Software per la Cybersecurity in applicazioni embedded basate su Microcontrollori

Il Contesto: Cyber Resilience Act e MDR 2017/745

Negli ultimi anni, la discussione riguardo il tema della *cybersecurity* ha assunto un ruolo centrale durante l'intero ciclo di vita dei dispositivi elettronici. In particolare, in seguito alla pubblicazione del Regolamento EU 2024/2847, meglio noto come *Cyber Resilience Act* [1], tutti i prodotti a marchio CE dovranno soddisfare i requisiti di cyber-sicurezza a partire dall'11 Dicembre 2027. Per quanto riguarda nello specifico il settore medicale, requisiti più stringenti di sicurezza cyber [2] sono posti nel *Medical Device Regulation* (MDR) 2017/745 [3].

In questo contesto, soluzioni sia hardware che software devono essere adottate fin dalle prime fasi della progettazione, tenendo a mente lo stato dell'arte nel contesto della cybersecurity ed evitando il più possibile soluzioni non standard.

Soluzioni Hardware

A livello hardware, si possono individuare numerosi componenti integrati, che rispondono alle più diverse esigenze in ambito security.

1. Acceleratori crittografici: questi componenti permettono di accelerare in hardware le operazioni di crittografia (AES-128, AES-256, SHA, TRNG, ecc.), in modo da ridurre il carico computazionale della CPU host. La presenza di questo tipo di componente è imprescindibile per abilitare l'esecuzione degli algoritmi crittografici con tempistiche ridotte.

2. Hardware Security Module (HSM): dispositivo che assolve diverse funzioni, tra cui:
   - custodire informazioni segrete, quali ad esempio le chiavi crittografiche;
   - eseguire algoritmi di crittografia, sia simmetrici che asimmetrici;
   - generare chiavi crittografiche sicure.

   Questo modulo può avere ha una sua CPU dedicata, oppure condividere le risorse con la CPU host. In particolare, nell'ambito delle architetture ARM®, si possono individuare due diverse tipologie di HSM:

a. *ARM® Trust Zone®*. Tecnologia utilizzata sia in application processors basati su architettura *Cortex-A* [4], che processori *Arm Cortex-M* [5], allo scopo di creare un *Trusted Execution Environment* (TEE): la Trust Zone permette di isolare l'esecuzione del firmware sicuro da quello non sicuro, all'interno dello stesso processore, tramite una separazione hardware dei due mondi.

b. *Secure Enclave*. CPU dedicata alla gestione dei segreti e delle operazioni crittografiche, diversa dal processore host e per questo più sicura, in quanto fisicamente isolata. Questa tecnologia è disponibile in architetture dual/multi-core: alcuni esempi sono *NXP's EdgeLock Secure Enclave* [6] e *Microchip's PIC32CK* [7] e *PIC32CZ* MCUs [8].

3. Secure Element (SE): dispositivo integrato esterno che non condivide le risorse della CPU host. Il SE garantisce la massima sicurezza in termini di attacchi, in quanto è fisicamente distaccato dalla CPU host e implementa uno salvataggio sicuro delle informazioni critiche, ad esempio le chiavi crittografiche e i certificati, a livello hardware. Il SE può integrare sia un acceleratore crittografico che un HSM, fornendo quindi prestazioni ottimizzate in un unico componente. Inoltre, il SE è un dispositivo *tamper-proof*: un'eventuale manomissione viene rilevata dal sistema stesso, garantendone la massima sicurezza.

   Esistono, quindi, numerose soluzioni hardware capaci di soddisfare i bisogni più variegati in termini di cybersecurity. Tuttavia, queste sono da integrare con apposite implementazioni software, per una protezione a tutto tondo del sistema integrato.

Soluzioni Software

In ambito firmware e software, si possono identificare diversi processi in grado di implementare le strategie di cybersicurezza:

- Secure Boot: il *Secure Boot* è un algoritmo che assicura l'esecuzione di un firmware autentico, solitamente eseguito durante la fase di boot del sistema embedded, quindi dal bootloader. Quest'ultimo controlla sia l'autenticità che l'integrità del codice applicativo, ad esempio tramite un algoritmo asimmetrico come *Elliptic Curve Digital Signature Algorithm* (ECDSA), che autentica il firmware conoscendo la chiave pubblica associata alla chiave privata con cui è stata generata la firma digitale.

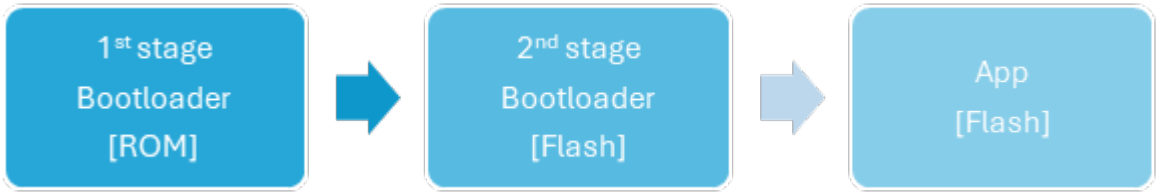

Figura 1: Secure Boot

Data l'importanza critica del bootloader durante il processo di *Secure Boot*, è fondamentale verificare l'autenticità del bootloader stesso: nasce quindi la necessità di un bootloader primario e immutabile (*first stage* bootloader), che solitamente risiede nella ROM, che si occupa di autenticare il bootloader secondario (*second stage* bootloader), che a sua volta autentica l'applicativo e può essere aggiornato al bisogno.

- Secure Update: il sistema embedded deve essere protetto non solo in fase di boot, ma anche in fase di software update, sia nel caso di aggiornamento tramite una connessione fisica che *Over-the-Air (OTA) Update*. Il sistema deve quindi accettare in ingresso esclusivamente un software autentico, tramite verifica della firma digitale.

Un aspetto cruciale in questa fase è la cosiddetta *rollback protection*: è buona norma inibire la programmazione di una versione software precedente a quella attuale, per garantire che la versione installata sia quella più aggiornata in termini di sicurezza. Un attacco cyber potrebbe consistere nell'installare una versione datata, con falle nella sicurezza, sfruttando quindi una vulnerabilità del sistema con intento malevolo.

- Data Encryption: la cifratura dei dati sensibili è fondamentale per garantire la segretezza delle informazioni sensibili, quali ad esempio i segreti crittografici e le password, e può riguardare sia dati in transito, che statici.

L'Agenzia per la cybersicurezza nazionale (ACN) mantiene in aggiornamento le Linee Guida per le Funzioni Crittografiche [9], mettendo in primo piano, ad esempio, la conservazione delle password tramite *hashing*. Le funzioni di hash crittografiche non sono invertibili: si fornisce la password in input alla funzione di hash, che ne calcola il digest, una stringa di lunghezza fissa, dal quale non si può risalire alla password in chiaro, assicurandone la segretezza.

Un altro esempio di crittografia si applica alla conservazione sicura del firmware in memoria, rendendolo illeggibile ad eventuali attacchi cyber. Il firmware può essere cifrato nella memoria flash e letto tramite *On-The-Fly Decryption* (OTFDEC), protocollo che consiste nel decifrare in tempo reale i dati e il codice salvati in memoria, con latenza minima.

- Secure Communication: come già evidenziato nel punto precedente, anche le informazioni in transito devono essere protette da eventuali attacchi che ne comprometterebbero l'integrità e la confidenzialità. Per fare ciò, si possono utilizzare protocolli di comunicazioni sicuri.

Un esempio è il *Transport Layer Security* (TLS), un protocollo standard di comunicazione che garantisce privacy, autenticità e integrità dei dati trasmessi, operando al di sopra del livello di trasporto in reti TCP/IP. La sicurezza risiede nella fase di *handshake*, o negoziazione, che crea un canale sicuro tra i due attori della comunicazione: si utilizza un algoritmo asimmetrico, basato su chiave pubblica e chiave privata, per condividere una chiave simmetrica da utilizzare per cifrare i messaggi scambiati durante la sessione di comunicazione. L'autenticazione assume un aspetto centrale nel TLS, infatti il server deve fornire al client un certificato che ne garantisca l'autenticità. Nel *mutual TLS*, anche il client deve essere autenticato dal server.

Si può notare come la gestione delle chiavi e dei certificati sia centrale nell'implementazione delle strategie di cybersecurity. Al fine di facilitare il provisioning dei dispositivi IoT connessi, esistono dei servizi a pagamento, come il Microchip TrustMANAGER [10], che implementano la Public Key Infastructure (PKI), permettono di generare e aggiornare i certificati, e infine abilitano il firmware update tramite OTA.

Un esempio: la cybersecurity in un'applicazione embedded

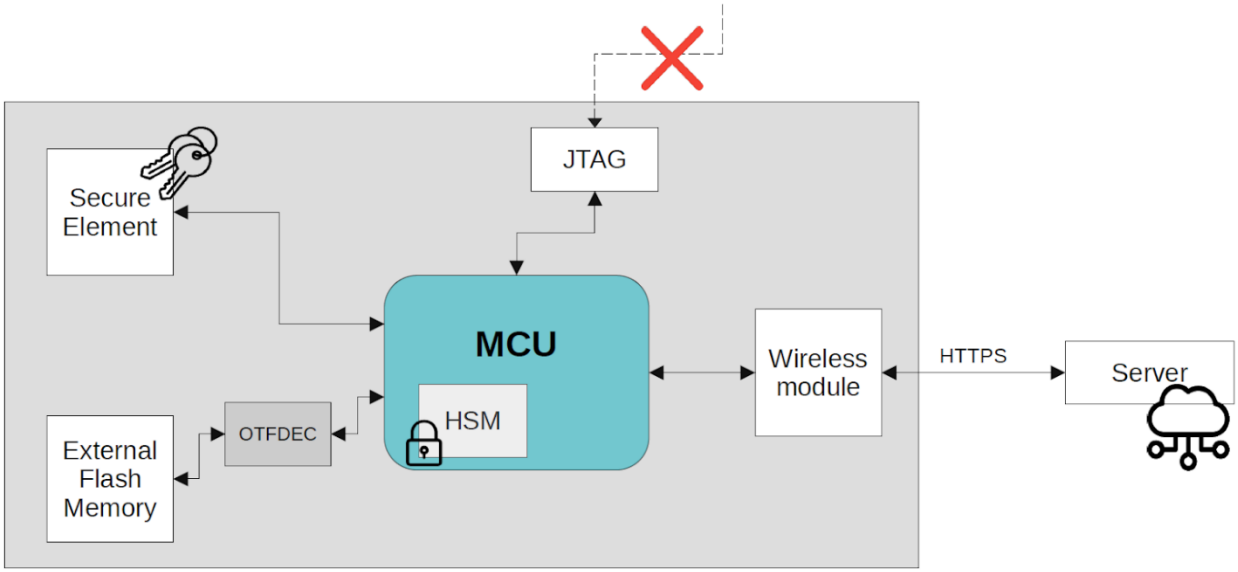

Figura 2: esempio di applicazione embedded

A conclusione, si riporta un esempio di un'applicazione degli standard di cybersecurity in un sistema embedded.

- MCU con HSM: gestisce il secure boot e il secure update;
- On-the-fly Decription: abilita la cifratura e decifrature in tempo reale delle istruzioni dalla memoria flash, proteggendo la segretezza del firmware;
- JTAG port: accesso disabilitato tramite rimozione del canale JTAG, per evitare la lettura o la cancellazione del codice;
- Secure Element: salvataggio sicuro e tamper-proof dei segreti crittografici, accelerazione hardware degli algoritmi di crittografia;
- HTTPS: connessione sicura ad Internet con HTTP over TLS, provisioning di chiavi e certificati, FOTA.

Fonti:

1. Regulation (EU) 2024/2847 of the European Parliament and of the Council of 23 October 2024 on horizontal cybersecurity requirements for products with digital elements and amending Regulations (EU) No 168/2013 and (EU) 2019/1020 and Directive (EU) 2020/1828 (Cyber Resilience Act).
2. Cybersecurity nei dispositivi medici: cosa richiede il MDR 2017/745- Teoresi Group
3. Regulation (EU) 2017/745 of the European Parliament and of the Council
4. TrustZone for Cortex-A – Arm®
5. TrustZone for Cortex-M – Arm®
6. EdgeLock Secure Enclave | NXP Semiconductors
7. PIC32CK SG/GC Arm® Cortex®-M33-Based Microcontrollers (MCUs) | Microchip Technology
8. PIC32CZ CA Arm® Cortex®-M7-Based Microcontrollers (MCUs) | Microchip Technology
9. Crittografia- ACN
10. TrustMANAGER | Microchip Technology

Autore: Gaia Di Federico, Progettista Elettronico R&D