Hardware Security Modules for Protecting Embedded Systems

Marko Wolf, ESCRYPト GmbH – Embedded Security, Munich, Germany
André Weimerskirch, ESCRYPト Inc. – Embedded Security, Ann Arbor, USA

1 Introduction & Motivation

Being able to trust another IT system that “it always acts as expected” requires consecutive trust into all layers, which are involved in creating a trustworthy IT system. Figure 1 depicts the “pyramid of trust” for a typical IT system, where trust into the whole IT system requires that each layer can rely on the effective security of its underlying layer without being able to verify it directly. This means for instance that a perfect software and hardware security solution could be rendered useless by a weak underlying security system design. Moreover, potential weakness in the system design cannot be detected nor prevented by the upper hardware and software layers.

In contrast to typical backend IT systems, the hardware layer of embedded systems is often directly exposed to physical attacks, which manipulate hardware or software functions by physical means (e.g., manipulate flash memory or deactivate alarm functions). One approach to make such physical attacks more difficult is to apply especially tamper-protected hardware security modules (HSM) as shown in Figure 2. Such an HSM protects critical information (e.g., personal identification number = PIN, secret cryptographic key) and critical operations (e.g., PIN verification, data encryption), for instance by a strong physical shielding.

How a typical HSM looks like, and what else such an HSM can do to improve the security of an embedded system, will be presented in the following. This whitepaper further clarifies in which situations HSMs cannot help and it gives a basic market overview.

Figure 1: Hardware security layer as part of the trust pyramid

Figure 2: Hardware security module
2 How a Typical Hardware Security Modules Looks Like

Figure 2 displays the core features of a typical hardware security module (HSM).

**Figure 2: Hardware-security-enabled embedded systems architecture**

**Secure memory** – little non-volatile data storage (i.e., some kB) inside the tamper protected HSM to prevent unauthorized readout, manipulation, or deletion of critical information such as cryptographic keys, cryptographic certificates, or authentication data (e.g., PINs or passwords). The secure memory portion of the HSM further contains all HSM configuration information, for instance, information about HSM ownership or access authorizations to secured internals.

**Secure cryptography** – cryptographic algorithms used for data encryption and decryption (e.g., AES or 3DES), data integrity enforcement (e.g., MAC or HMAC) or data origin verification (e.g., by using digital signature algorithms such as RSA or ECC), and all related cryptographic activities (e.g., key generation, key verification).

**Secure functions** – comprise all shielded functions, which are not directly related to cryptography, where the HSM serves as physically protected “trust anchor”. This could be, for instance, a physically protected clock signal, an internal random number generator, a bootstrap protection mechanism, or any critical application function (e.g., to realize a secure dongle).

**Interface and control** – finally refers to the internal HSM logic, which implements the HSM communication with the outside world and which manages the operation of all the HSM-internal building blocks as aforementioned.

**Tamper-protection** – All functional building blocks of a hardware security module as described above are enclosed by a continuous physical (or logical) boundary, which prevents that internal data and processes can be intercepted, copied/cloned, or manipulated yielding to non-authorized use or compromise of internal secrets. This *cryptographic boundary* is usually implemented with algorithmic and physical side-channel countermeasures and with dedicated tamper-protection measures (e.g., special shielding or coatings) to enable side-channel-resistance, tamper-evidence, tamper-resistance, or tamper-response.
3 When Hardware Security Modules Can Help

This section provides a short overview how an HSM can improve the security of an embedded product solution.

- **Shield security assets against software vulnerabilities** – Hardware security modules protect critical information (e.g., identities, signing keys, or encryption keys) by the physical shielding, which cannot be circumvented by any software vulnerability.
- **Deter, detect, or hinder physical, offline and insider (POI) attacker** – Hardware security modules can help to deter, detect, or hinder powerful POI attackers by implementing effective side-channel-resistance and tamper-protection barriers, which (amongst others) have strong access restrictions even for authorized users (e.g., some information will always remain exclusively inside the HSM).
- **Accelerate computationally intense security** – Hardware security modules can accelerate security mechanisms by applying dedicated acceleration circuitry.
- **Reduce security costs** – Hardware security modules might reduce security costs by adding some highly optimized special circuitry (e.g., for standardized cryptography) instead of costly overall upgrade of general-purpose hardware and by avoiding costly physical sealing of a complete ECU.

4 When Hardware Security Cannot Help

As already mentioned during the introduction, even a perfect HSM cannot serve as the only necessary panacea since the security of any embedded system depends just as well on the effective security of the organizational security layer or on the system security layer. Hence, this section gives a short overview which security threats an HSM cannot prevent.

- **Prevent software security vulnerabilities** – An HSM can check the integrity and authenticity of the software layers above (e.g., via secure boot) and provide low-level security functionality as a trusted anchor for more complex software-based security functionality. However, an HSM cannot prevent that the software layer contains security vulnerabilities (e.g., overseen buffer overflows), which can be exploited by hackers or malware at runtime.
- **Prevent inherent system design or organizational security weaknesses** – In order to serve as trusted base for the software security layer above, an HSM in turn has to rely on its below layers, that is, a proper security design and an effective organizational security. Thus, an HSM cannot help if the security can be broken at system level (e.g., a forgotten, but mandatory security measure) or at organizational level (e.g., a too simple password to access, for instance, the root sign key).
- **Prevent POI attacks** – Assuming a sufficient large attack budget, all hardware security modules can be broken. In the end, it is a matter of finding a good balance between the attack costs required to successfully compromise the HSM and the value of the secrets that are protected by the HSM.
5 Hardware Security Modules for Embedded Systems

Hardware security modules are already variously deployed in today’s embedded systems and the fields of application will continue to grow rapidly. In the following application examples, a short market overview, HSM evaluations, and certifications are presented.

Application examples – A well-known application example for HSMs are smart meters (e.g., electricity meters) to prevent manipulation of the digital measurement and of the meter reporting. HSMs are also already used to protect vehicular components (e.g., head unit, V2X communication, engine control, anti-theft, tachograph) to prevent unauthorized modifications, theft or exchange, counterfeits, or espionage. Liability and safety concerns drive the application of HSMs for medical devices (e.g., medical equipment or medical implants). Another prominent application example for HSMs in embedded systems are consumer devices to prevent counterfeits or to protect business models that, for instance, are based on subscription models as known for mobile phones or TV set-top boxes.

Solutions overview – Today’s available solutions of dedicated hardware security modules for embedded systems are rather limited. Well-known solutions are SHE (HIS: Secure Hardware Extension (SHE) – Functional Specification v1.1), EVITA concept (EVITA: D3.2 – Secure On-board Architecture Specification v1.3) and (with some limitations regarding embedded systems) the TPM (TCG: Trusted Platform Module Main Specification v1.2). Another important approach to realize HSMs are so-called security controllers. These are standard embedded systems processors with security enhanced memory management and processor extensions to provide an isolated runtime environment (e.g., ARM TrustZone) and/or small separated internal memory (e.g., FreeScale i.MX series). Sometimes such security controllers also provide integrated cryptographic engines (e.g., AES encryption) or dedicated tamper-protection measures (e.g., special coatings). Finally, up to a certain degree, also common cryptographic co-processors (e.g., IBM CryptoCards) or smartcards can be used as HSMs. However, they are often limited in terms of security functionality, cryptographic performance, or level of security (e.g., they do not provide secure memory or apply only insecure communication to the main processor).

Security evaluation and certification – To evaluate, validate and compare the effectiveness and correctness of a certain HSM realization, public authorities and industry consortiums have established security evaluation and certification standards. The dominant standards are the US-driven NIST FIPS 140 standard “Security Requirements for Cryptographic Modules” and the more recent and globally accepted “Common Criteria for Information Technology Security Evaluation” (ISO 15408).
6 Conclusion and Outlook

Hardware security modules are a strong component to improve the protection and trust of embedded systems. They help to protect central security-critical assets and functionalities against software vulnerabilities. HSMs might also reduce security cost and they can help to deter, detect, or hinder powerful POI attackers. Furthermore, an HSM is able to accelerate computationally intense security algorithms. Even though adding an HSM alone cannot counteract all security threats to an embedded system, HSMs are already deployed in various embedded applications while the potential fields of application and corresponding markets will continue to grow rapidly.