

Secure boot can greatly enhance the security of an embedded system by cryptographically verifying that the code being loaded and executed is authentic and has not been unknowingly or maliciously modified. Once the domain of high security applications, the techniques are now being adopted in a broad range of products such as mobile phones, set-top boxes and networking systems such as base stations.

The ESS-04 offers a software toolkit which facilitates the implementation of secure boot. It allows developers to implement secure boot systems using software only constructs or to use hardware offload engines to simplify and accelerate the required verification or decryption operations in the target system. The toolkit is licensed as C source code to allow it to be ported to the target system.

### Key Features:

- Multi-phase loader cryptographically validates a phase before loading it
- Supports hardware assist to speed boot time if security cores are available
- Secure access to engineering mode bypasses secure boot for lab debugging purposes
- C language source code to facilitate adaptation to the end system

### Overview of Secure Boot

Secure boot systems rely on well proven cryptographic algorithms for signing and optionally decrypting code to ensure that the processor is initialized into a known state and is running code from a trusted source. Multiple phases are used to segment the bootstrap function, which allows each phase to release system resources to use if that facility exists in the SoC or system architecture. The multi-phase hierarchy also provides support for multiple, independent trusted code sources which is required should several companies be contributing to a code build..

The phases are outlined below:

**Phase 0:** This is a minimal system loader which self-tests the required cryptographic hardware components (if available), then loads and verifies Phase 1. The loader can do either a complete cryptographic verification and optional decryption of the Phase 1 loader. Elliptic offers three options for Phase 0 – an all-hardware solution in a finite-state machine, a firmware-only solution or a combination of both. The choice is based upon the security required and the system cost objectives. Elliptic recommends that the firmware-only solution be implemented with the Phase 0 code stored in on-chip ROM to enhance the protection of the embedded system.

**Phase 1:** Phase 1 is full-featured loader with cryptographic signature verification and optional code decryption of the final operating system and application code.

It is a software loader which resides in external Flash and is loaded into on-chip SRAM for execution after the Phase 0 loader cryptographically verifies it. If desired, the Phase 1 loader can be decrypted as well to hinder attackers' efforts to reverse-engineer the system. The software loader may also leverage security offload resources if available.

**Phase 2:** Phase 2 is the target Operating System and Application. In sophisticated systems such as mobile phones, the Application can consist of code from many different developers including the IC vendor, the handset vendor, network provider and even downloaded applications from third parties. It is also possible that multiple feature sets can be implemented in an SoC and selectively made available during the secure boot process based on their contracted service level through a field configuration update. Elliptic can support many key hierarchy permutations to permit a flexible software development scenario.

The ESS-04 also supports secure update. As an update changes the length and signature of the Phase 2 code, the secure update utility must modify and re-write the credentials in a secure manner.

Secure boot relies on the availability of on-chip secrets. Elliptic supports a variety of on-chip secrets such as eFuse, RTL, OTP and MTP. Each NVM choice has its merits and costs which can be discussed during the architecture stage.

In the situation where a new SoC design is being created, it is possible to enhance the robustness of secure boot by limiting the availability and visibility of secrets and other cryptographic capabilities available on-chip. This can include allowing visibility of on-chip security credentials during the boot phase then removing them from the processor bus for run time. As such, the highly robust secure boot solution is best created when they are developed from the architecture stage forward.

## Sample Implementation

Figure 1 illustrates an example implementation of secure boot using the ESS-04. This architecture uses the following hardware and software resources:

- NVM that contains the RSA Root public key Hash value and Silicon ID for the device
- Flash Memory contains the boot images to be verified by the SecureBoot SDK authentication functions.
- The Boot Application which is the boot reference application that implements 3-phase boot process.
- The SecureBoot SDK implements boot authentication functions.

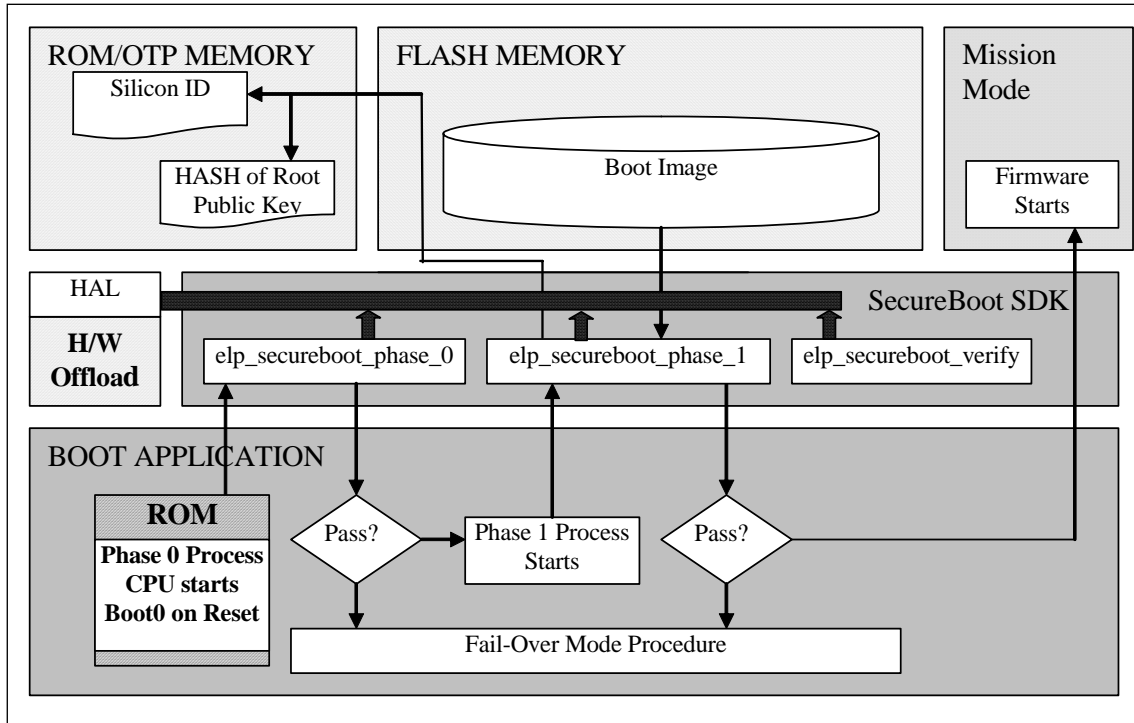


Figure 1 Secure Boot Toolkit Architecture

- Phase 2 or Mission Mode is the final operating system and applications code to run on the target device.

As the tool kit is highly configurable, it will support many permutations in secure boot. Further technical details on the options available to developers can be provided under NDA. Please contact Elliptic directly to conclude an NDA and then receive technical information on the Secure Boot.

## Contact Information

Elliptic Semiconductor, Inc.  
62 Steacie Dr., Suite 201  
Kanata, ON, K2K 2A9

Phone: +1 613 254-5456  
Fax: +1 613 254-7260  
Email: [info@ellipticsemi.com](mailto:info@ellipticsemi.com)