Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

AN230415

Setting up Semper Secure Flash for Fast Secure Boot

Author: Zhi Feng

Associated Part Family: S35HL-T/S35HS-T
S36HL-T/S36HS-T
S38HL-T/S38HS-T

This application note describes the steps for setting up Semper™ Secure NOR Flash memories to perform Fast Secure Boot. It also provides guidelines and suggestions to implement host application software for such purpose.

1 Introduction

Some applications, especially in automotive segment, require very fast boot-up time to respond quickly to timing critical messages. For example, systems inside a car listening to the Controller Area Network (CAN) bus are required to boot within 100 ms or faster to handle CAN bus messages. Traditional systems may satisfy this requirement easily; however, systems that require a secure boot find it challenging to meet such time requirements, as by the nature of the secure boot, firmware on different boot-up stages must be validated before they can run, which takes extra time.

Semper Secure NOR flash devices are designed to help such applications to satisfy the secure boot-up time requirement. This application note describes how the flash can be set up for fast secure boot purpose and the actual boot flow. You can follow this application note to implement the flow. Moreover, the Semper Solution Development Kit (S-SDK) provides the fast-secure boot user example as described in this document. You can use the source code example directly if you use the S-SDK.

It is assumed that you are familiar with Semper Secure datasheets and standard operations. For details of the operations, see the corresponding datasheets and application notes.

This document uses “device” to refer to the Semper Secure flash device and “host” to refer to the paired host MCU if not otherwise specified.

2 Fast Secure Boot Requirements

The process of secure boot requires authentication of the boot code itself before the host MCU can execute it. The authentication includes verifying if the code storage hardware is the original hardware, and the boot code has not been tampered with. While there are many ways to implement the secure boot process, this application note shows one of the ways to perform fast secure boot.

The objective of fast secure boot is to complete the secure boot process in a relatively fast manner, preferably within 100 ms. This duration is measured from the system power on to the system is up and running with the boot code. The main steps for the fast secure boot include:

1. Preparing the device. This is done only once during provisioning.
2. Authenticating the flash device.
3. Authenticating the boot code. This is an optional step with Semper Secure flash. As Semper Secure provides secure storage for the boot code, the code cannot be tampered with by unauthorized parties after it is programmed. Therefore, once the flash device is authorized. It is considered the code inside the secure storage is also intact.
4. Reading boot code from the device or executing in place from the flash.

3 Preparing the Device for Fast Secure Boot

3.1 Choosing the Correct Ordering Option

Semper Secure has two main ordering part categories: Symmetric devices and Asymmetric devices. Due to the nature of asymmetric keys, it would take a much longer time for the host MCU and the flash device to complete the mutual authentication process. Therefore, asymmetric devices are not designed to perform fast secure boot. Symmetric devices use a shared secret scheme, so the secure boot process is much faster. The fast secure boot process described in this document applies to symmetric devices only.

3.2 Initial Pairing with the host

Here are the typical initial provisioning steps for pairing up a host MCU with a symmetric Semper Secure device:

1. Validate device firmware.
2. Use the `SetInitialConfig` transaction to set the initial configuration of the device.
3. Install the Master Key (Shared Secret) on the device.
4. Set up Region Configurations.
5. Program Region Secret Keys.
6. Freeze all configuration settings using the `FreezeConfig` transaction.

For detailed explanations of each step, see AN228332, available with Semper Secure Early Access Program.

Step 1 validates the device firmware: Layer 0 (L0) and Layer 1 (L1). The host can store the hash value of the L0 and L1 and verify the values during the fast secure boot sequence.

Step 3 installs the shared secret onto the flash device. This shared secret is also the master key for the device. It is the basis of the mutual authentication between the host and the device. The shared secret is kept on the flash in a safe key storage, encrypted with the Composited Device Identifier (CDI). The CDI itself, is derived from the Unique Device Secret (UDS) and the hash value of the immutable Layer 0 firmware. In another word, the CDI is unique on each device, and it is not exposed to the outside world. If the shared secret can be correctly recovered from using the CDI to decrypt the stored value, that will prove that the flash UDS, Layer 0 Firmware are still intact. Therefore, the flash can be authenticated by the host.

After the shared secret is successfully installed onto the flash, the device is ready for performing a fast secure boot process, assuming the boot code has already been programmed into the secure storage.

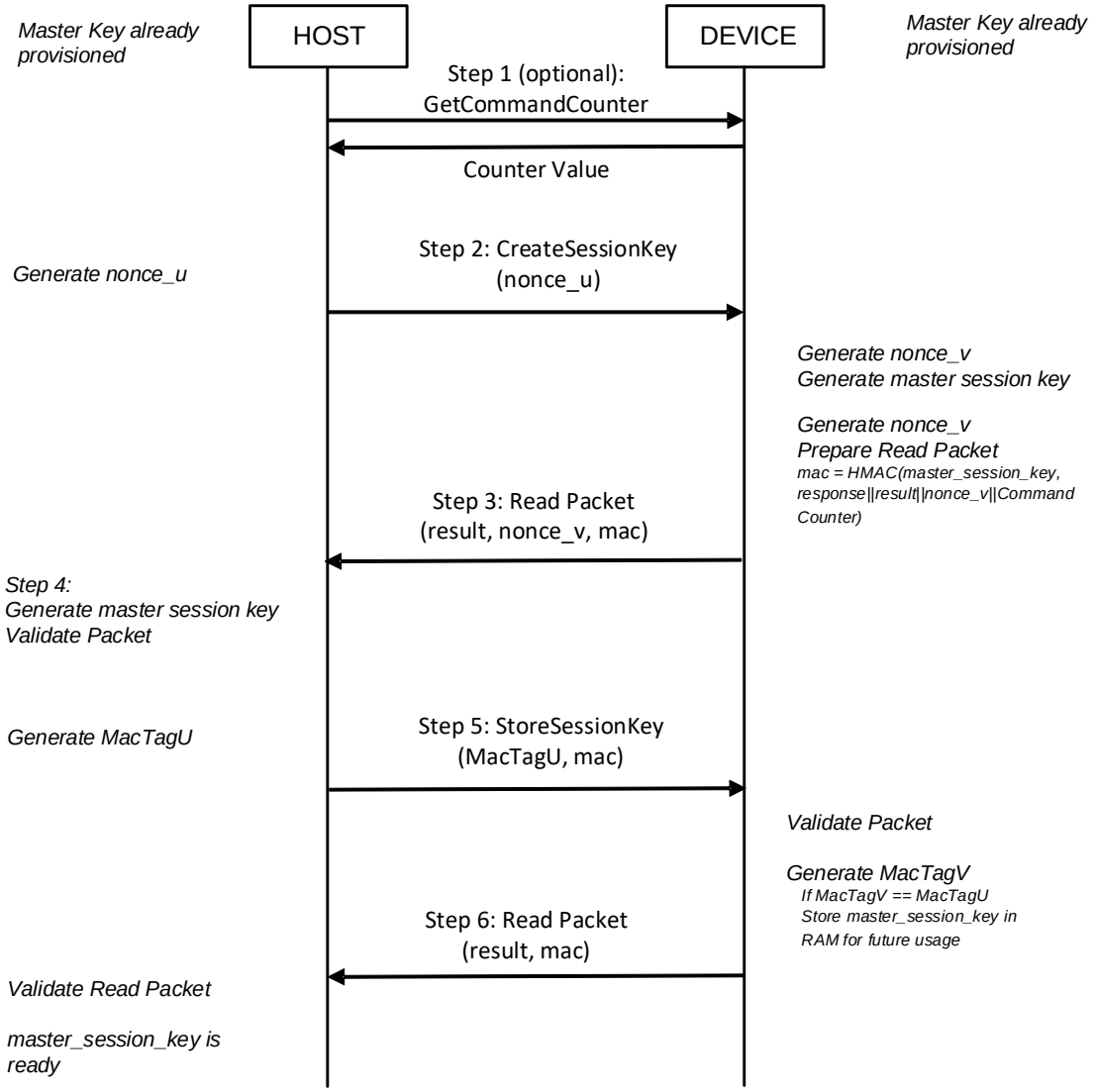
4 Example of Fast Secure Boot Procedures

4.1 Setting up Master Session Key

Upon power on, the host MCU establishes the master session key with the flash device. If the master session key gets generated successfully, it ensures that the device CDI and shared secret are all intact.

Figure 1 shows the steps to generate Master Session Key from the host point of view.

Figure 1. Master Session Key Generation Flow



Here is a detailed explanation of the flow:

1. **Issue the `GetCommandCounter` transaction**. This is an optional step. If the host has already synchronized the Command Counter value with the flash, this step can be omitted. If both the host MCU and the flash went through a Power on Reset (POR), it is necessary for the host to get the Command Counter value for the first time to start the secure transaction process. This is a public transaction that does not require any security parameters.
2. **Issue the `CreateSessionKey` transaction.** The host starts the master session key generation sequence with the `CreateSessionKey` transaction. It should be sent with the type 00h, that is, creating master_session_key. This transaction write packet contains a 16-byte nonce_u value, 20 bytes of security parameter, and the CRC-16 checksum to ensure packet integrity.

   After sending the write packet, the host should monitor the interrupts or poll the Status Register to check when the device has completed the operation.

3. **Read response from the device**. After the device is ready, the host issues the read packet transaction to retrieve the result code from the device. The read packet contains nonce_v value generated by the flash. The packet also has the HMAC value calculated by the master session key newly generated on the flash. Before the host derives the master session key, it cannot validate the packet yet.

4. **Generate the master_session_key on the host**. After getting the nonce_v value from the device, the host should calculate the master_session_key based on following formulas:

```
salt (256 bits) = 0xDEADBEEF || last 28 bytes of Device Configuration Data;

Z (256 bits) = Master_Key (i.e., Shared_Secret);

Kdk (256 bits) = HMAC(salt, Z);

L = 0x100 (HMAC-256) or 0x140 (AES-GCM);

Label (64 bits) = 0x6DE8BC2177D879B2 (HMAC-256) or 0x921743DE8827864D (AES-GCM);

Context (512 bits) = Life_cycle (16b) || 0000h || ++CmdCounter (64b) || security_parameters
(160b) || nonce_u (128b) || nonce_v (128b);

master_session_key = KDF (Kdk, L, Label || Context);
```

   After deriving the master session key, which should be the same as the one on the flash, the host can now validate the read packet by HMAC to ensure its authenticity.

5. **Issue StoreSessoinKey transaction**. The host generates a MacTagU value from the new master session key according to this formula, then includes it in the write packet with type 0000h for master_session_key.

```
MacTagU = HMAC(master_session_key, nonce_v||nonce_u)
```

   After sending the write packet, the host should monitor the interrupts or poll the Status Register to check when the device has completed the operation.

6. **Read response from the device**. After the device is ready, the host issues the read packet transaction to retrieve the result code from the device. The read packet contains the MacTagV value generated by the flash. After validating the packet with the HMAC value, the host should compare the MacTagV with the MagTagU value. If the comparison passes, that means the master session key is successfully validated by both the host and the device.

4.2 Validating Device Firmware

After setting up the master session key, the host can issue the `ValidateFW` transaction to the device. This transaction returns an FMAC value, the calculation of which contains the hash values of L0 and L1 firmware. The host then can validate the stored L0 and L1 hash values to make sure they have not been tampered with.

After these steps, the host and the device are mutually authenticated.

4.3 Setting up Region Session Key

Using the same steps for setting up the master session key, the host can set up the region session key with the flash device for the region that contains the boot code. All secure transactions between the host and the device for this region will be using the region session key.

4.4 Unlocking Region for Reads

This example shows an Execute-In-Place (XiP) from the boot code region. The secure region is set up with the access level as Authenticated Lock region. To perform read operations, the host must first use the region session key to unlock the region. It is done by the `AuthenticatedUnlock` transaction. After this transaction, the host can start performing legacy SPI or Quad SPI reads from the region for XiP.

4.5 Authenticating Boot Code before Reading

Furthermore, before running the boot code, the host can validate the entire boot code by issuing the `AuthenticateMemory` transaction. This transaction uses the region session key to calculate a hash value over the specified address range and returns the hash value to the host. The host then validates the hash value before reading the boot code.

5 Using Semper Solution SDK

Semper Solution Development Kit (S-SDK) is a software package that is designed to help customers develop their own driver or directly use the provided code examples. Fast Secure Boot is one of the examples provided in the S-SDK. You can follow the S-SDK code example or use the platform-independent C code to perform the steps mentioned in this document.

6 Conclusion

By providing secure storage for boot code in secure regions, Semper Secure flash helps the host MCU to perform a fast secure boot to meet the system requirements. The process validates a pre-provisioned shared secret on both the host and the device sides. It can validate both the device and software integrity before the boot code is used.

7 References

002-26101 S35HS-T, S35HL-T Semper Secure Flash with Quad SPI Datasheets

002-28332 AN228332 – Initial Provisioning in Cypress Semper Secure NOR Flash

Note: These documents are available with Semper Secure Early Access Program.

Document History

Document Title: AN230415 – Setting up Semper Secure Flash for Fast Secure Boot

Document Number: 002-30415

Revision	ECN	Submission Date	Description of Change
**	6905239	06/25/2020	Initial release

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Code Examples | Projects | Videos | Blogs | Training| Components

Technical Support

cypress.com/support

Cypress Semiconductor
An Infineon Technologies Company
198 Champion Court
San Jose, CA 95134-1709