

SHA1 hash over the code stored in external flash, the version number defined in the trusted computing platform alliance (“TCPA”) main specification, date information, the text size (e.g., the size in bytes of the flash rodata and flash function table), the image size (e.g., the size in bytes of the flash rodata, flash function table, and flash firmware) and the heap pointer address.

[0106] As will be discussed in more detail below, the secure code descriptor also may include information related to one or more keys that may be used to verify received information that is to be loaded into the flash memory. For example, one field may contain the version of the signature key and another field may contain, for example, a 2048 bit RSA key that is used to verify the received information.

[0107] When a chip implementing a TPM is first soldered to a motherboard by an OEM, the secure code and associated information may not be initialized in the external flash. In this state, a TPM may simply execute functions that are required to boot, to load the secure code information into flash, or to test the TPM during manufacturing.

[0108] As represented by block 418, at some point during the manufacturing process (e.g., during a test or boot procedure) the TPM may generate an internal sequence number and internal keys that may be unique to that specific TPM chip.

[0109] The sequence number may be used to track the number of updates to the flash memory. For example, every time new secure code information is written to flash memory, the sequence number in an internal memory may be incremented and stored in the flash memory.

[0110] The internal keys (or keys based on these keys) may be used, for example, to encrypt and perform authentication operations on information that is stored in the flash memory. For example, the keys stored in the OTP memory may be used to encrypt other keys so that these other keys may be securely stored in external memory.

[0111] The key information may be securely stored within a security boundary associated with the TPM. For example, the TPM may be configured to only store this information in the clear in an on-chip memory. In this case, the TPM would never send this information outside of the chip in the clear. Accordingly, a key or set of keys that is stored in the OTP memory may be used to encrypt other keys (that may be used for the encryption, decryption, verification and authentication operations) so that those other keys may be securely stored in the flash memory.

[0112] The operations represented by blocks 420 and 422 may be performed in the event there is secure code information to be loaded into the flash memory. At block 420 the TPM (e.g., cryptographic processor(s) 330 in FIG. 3) generates encryption and authentication keys that may be used to encrypt and authenticate information stored in the flash memory. These keys may be securely stored in the flash memory (e.g., in the TPM keys/data section 356) by encrypting the keys using the internal keys discussed above.

[0113] These encryption and authentication keys (or related keys) also may be accessed by the instruction cache controller (e.g., the cryptographic processor(s) 334 in FIG. 3) for operations that are discussed below. The instruction

cache controller uses these keys to decrypt and authenticate secure code information that it retrieves from the flash memory.

[0114] The TPM (e.g., cryptographic processor(s) 330) uses the encryption and authentication keys to encrypt and/or authenticate the flash secure code information. This information may include, for example, the flash secure code descriptor (including the public verification and encryption keys), the flash function table, the flash read-only data, and the compiled flash secure code.

[0115] As represented by block 422, the encrypted flash secure code information may then be loaded into the flash memory. During manufacturing, owner authorization may not be required to execute the secure code load commands. This allows an OEM to load the secure code information into flash memory without creating the TPM device keys (e.g., EK and SRK). Once the secure code descriptor disables unauthorized field upgrades (e.g., via a corresponding field entry), any subsequent field upgrade commands may require owner authorization. After the secure code information has been successfully loaded to flash memory, the TPM may be temporarily deactivated until the next system reboot.

[0116] FIG. 5 illustrates one embodiment of boot operations that may be performed in a TPM that has been deployed in the field. As will be discussed in more detail below, a variety of conditions may cause the TPM to be reset.

[0117] As represented by block 502, after a reset the system defaults to executing functions as defined in the internal function table. For example, register 28 (r28) may be set to point to the base address of the on-chip function pointer table.

[0118] As represented by block 504, the system then determines the cause of the reset. For example, when an error caused the reset, the system may store the type of error in a status register before initiating the reset. As a result, when the system commences the boot operation, the system may check the appropriate status registers to determine what caused the reset. In some embodiments reset types may include, for example, a power-on reset (“POR”), a reset initiated via the LPC bus, a reset initiated by the security assurance logic, a reset initiated as a result of an instruction cache authentication failure (e.g., a failure that occurs when verifying data read from the flash).

[0119] As represented by blocks 506 and 508, in some embodiments a flash verification failure reset may cause the system to enter a failure mode. In this case, when the system tried to read code stored in the flash memory, the authentication check over the code may have failed. Since the code in the flash memory cannot be trusted the system may continue to only execute code according to the internal function table (e.g., execute code in the internal ROM).

[0120] In some embodiments similar steps may be taken when the reset was initiated by the security assurance logic. Again, a security assurance logic failure may indicate that the integrity of the system has been compromised. Accordingly, the system may be set to only execute the original set of functions that were loaded into the internal memory.

[0121] Alternatively, if the reset is caused by other types of resets the system may perform the standard boot proce-