

secure code descriptor, the flash function table, any flash read-only data, and the compiled secure code.

[0169] In the event a new verification key has been created at block 714, the above information may still be signed using the prior version of the verification key. In this case the TPM will continue to use the prior version of the verification key until the new verification key has been successfully loaded into the TPM. In some embodiments signing is performed using the RSASSA-PKCS1-v1.5 algorithm.

[0170] As represented by block 718, after the secure environment returns the signed (and optionally encrypted) secure code information, the manufacturer sends this secure code information to the TPM. In some embodiments the process of sending code, etc. to the TPM involves invoking one or more commands. For example, the TPM field upgrade process may be divided into several parts. The first part (information request) may relate to obtaining all necessary information to do an appropriate update. Other parts (upgrade start and update) may relate to commencing and performing the transfer of the secure code information to the TPM. Another part (upgrade complete) may relate to completing the transfer.

[0171] These commands may be invoked, for example, by the manufacturer's processing system. The manufacturer's system may communicate with the TPM via an external interface 210 as shown in FIG. 2. In the illustrated embodiment, the commands are received by the host processing component 208 which sends the commands to the TPM via the bus 214 (e.g., an LPC bus). One embodiment of these commands and a process for updating code, etc., in response to these commands will be discussed in conjunction with FIG. 8.

[0172] Before the manufacturer initiates the transfer of the new code, the manufacturer's system may invoke a command to request field upgrade information that may be needed to invoke and complete the transfer. This capability may be used to obtain all relevant information about the TPM chip so that the caller can start the upgrade process. For example, this command may provide information about the chip revision, the current version of the verification key and any currently loaded firmware. Also, this command may be used to determine whether new secure code information needs to be loaded into the TPM. This command may be called under normal operating conditions and may not require owner authorization.

[0173] As represented by block 802, the TPM thus receives an upgrade information request. At block 804, the TPM returns selected information in response to this request. This information may include, for example, the version of the upgrade information structure and a firmware valid field that indicates that the firmware is valid.

[0174] In some embodiments this information may include a flag that indicates the TPM currently has an owner and an ownerAuthReq field that indicates that owner authorization is required. Owner authorization may be required when the TPM encryption key (EK) has been programmed or when the ownerAuthReq bit in the secure code descriptors is set.

[0175] The returned information also may include version numbers (e.g., relating to the TCPA main specification, the LPC bus and the signature key), the most-significant 4 bytes

of the SHA1 hash over the firmware stored in external flash, and the maximum data size which may be sent to the TPM by the upgrade update and upgrade complete commands.

[0176] After the manufacturer receives the necessary information to initiate the transfer of the secure code information, it sends an upgrade start command to the TPM. This command starts the upgrade process by providing a new secure code descriptor to the TPM. Owner authorization may be required if the EK has been generated.

[0177] The upgrade start command may include, for example, the size of the secure code descriptor, the secure code descriptor and a signature (e.g., a RSASSA-PKCS1-v1.5 2048 b signature) calculated over the secure code descriptor and the authorization digest for returned parameters.

[0178] As represented by block 806 once the TPM receives this command, it may verify various aspects of the command. For example, the TPM may verify that authorization is required when the EK has been generated. If authorization is required, the TPM may use the authorization protocol to verify that the command was sent by the owner.

[0179] In some embodiments the TPM may verify a signature over any received data (e.g., the new functions and function table). By using the cryptographic public key obtained from a trusted source as discussed above, the system may be assured that the data came from the trusted source (e.g., the holder of the corresponding private key) and that the data has not been compromised.

[0180] Initially, the TPM verifies the signature of the new secure code descriptor (block 808). If a valid secure code descriptor was read from the flash memory at boot the TPM uses the public verification key stored in flash memory. If a valid secure code descriptor was not read from the flash memory at boot the TPM uses the public verification key stored in the internal ROM.

[0181] As represented by block 810 the TPM temporarily deactivates the TPM. The system may then ensure that the device only executes code from the internal memory. For example, the system may set the processor register r28 to point to the on-chip function pointer table.

[0182] The system may then update and save the old secure code descriptor in preparation for writing the new secure code descriptor to the flash memory. The system sets the volatile secureCode field to false. The system also copies the new secure code descriptor to non-volatile memory. To update the old secure code descriptor, the system may clear the firmwareLoaded field, generate a new IV and encrypt and authenticate the descriptor with the flash 3DES and HMAC-SHA1 keys. The system then writes the old secure code descriptor back to the flash memory.

[0183] The secure code descriptor may be double-buffered in the external flash. This may guarantee that the flash secure code descriptor is not corrupted on a failing write. One embodiment of such double-buffering is discussed below.

[0184] As represented by block 812, the TPM opens a thread that calculates a SHA-1 digest over the received secure code information. Accordingly, the TPM initially incorporates the new secure code descriptor into the digest. The TPM also may clear the counter that keeps track of the number of 256 bytes secure code blocks.