

dures. For example, some resets such as a power-on reset or an LPC reset may not have been caused by a condition that compromised the integrity of the system.

[0122] As represented by block 510, the system may configure certain information relating to the internal keys (e.g., those stored in the OTP memory). For example, the system may read the OTP memory and initialize any corresponding data structures in the data buffer. In addition, when the OTP has been programmed (e.g., the system is in a normal operating mode) the system may enable various operations (e.g., frequency and/or reset monitoring) relating to the security assurance logic.

[0123] As represented by block 512, during the normal boot sequence the system may initialize the secure code hardware (e.g., data structures and data buffers). Here the system may set a secure code flag to FALSE. In this mode, only commands that are used to load secure code are allowed. This may be enforced in the command switch. In some embodiments any functions that initialize secure code hardware may not reside in flash memory since they may be used to change the pointer to the function table. The system also may verify that the type, size, and page size of the flash memory are supported by the system.

[0124] As represented by block 514, the system also may initialize keys for the instruction cache. For example, in an embodiment that supports 3DES and HMAC the system may initialize a 3DES key for decryption operations and an HMAC-SHA1 key for authentication operations. In some embodiments these keys may be subsequently encrypted using an internal key and stored in the flash memory (e.g., TPM keys/data 356 in FIG. 3). It should be appreciated that these operations are provided as examples only and that the system may use other cryptographic algorithms and perform other operations.

[0125] As represented by block 516, the system determines whether any new and/or modified functions have been loaded into the flash memory. In some embodiments this is accomplished by checking to see whether there is a valid secure code descriptor stored in the flash memory. If the flash access fails due to a flash controller reset, the system may retry the flash access.

[0126] If new and/or modified functions have not been loaded into the flash memory (a "NO" at block 516), the system continues to use the on-chip function table (e.g., stored in the on-chip ROM) to determine where to retrieve program code. In this case, all functions may be retrieved from the on-chip ROM. The branch of the process will then proceed to block 524.

[0127] As represented by block 518, if new functions have been loaded into the flash memory (e.g., the flash memory contains a valid secure code descriptor), the system may be configured to use the off-chip (flash) function table to determine where to retrieve program code. Initially, the system decrypts and authenticates the secure code descriptor using, for example, the 3DES and HMAC-SHA1 keys. In some embodiments the secure code descriptor may be double-buffered as discussed herein. If the authentication is not successful, the system will set appropriate field in the secure code descriptor to indicate that the descriptor is invalid.

[0128] If the authentication is successful the system copies the secure code descriptor to the data buffer. The system then

checks the field that indicates whether the secure code descriptor is valid. In addition, the system checks the field that indicates that the secure code stored in the flash memory is valid.

[0129] If both of these are valid, the system may switch to the external function table. For example, register 28 may be set to point to the base address of the flash function table.

[0130] In this case, the new and/or modified functions will be retrieved from the flash memory and any original and unmodified functions will be retrieved from the on-chip ROM. The system may then set the secure code flag to TRUE. In this mode, all TPM commands may be executed in accordance with the TPM specification.

[0131] As represented by block 520, the system initializes the instruction cache. In some embodiments this may involve updating registers that hold the HMAC-SHA1 key, 3DES key, the secure code base address (the address of the first secure code block), the HMAC base address (the address of the first HMAC), the flash page size and the firmware hash from the secure code descriptor. In some embodiments these registers are only reset by a reset such as a POR or an LPC reset.

[0132] As represented by block 522, the system initializes the instruction multiplexer. In some embodiments this may involve updating the data bus address range to which the instruction multiplexer responds. For example, the instruction multiplexer may be initialized to respond to reads of the flash function table and flash rodata.

[0133] As represented by block 524, as the secure code operations complete, the system initializes the TPM. This may involve, for example, reading TPM-related data from the flash memory (e.g., TPM non-volatile ram 358 in FIG. 3) into the persistent memory.

[0134] The system also may initialize the secure buffer logic. After reset the data buffer may only be accessed when the TPM is in secure execution mode. The system thus relaxes this lock and allows the IO memory and a portion of the stack to be accessed when the TPM is not in secure execution mode.

[0135] As represented by block 526, after the boot sequence is successfully completed, the system commences normal program execution flow. In some embodiments this may involve using a command switch structure to handle command requests.

[0136] Referring now to FIG. 6, one embodiment of operations that may be performed when a function call is invoked (block 602) will be treated. As discussed above, a function call may involve a first instruction to retrieve the table offset value for the function from the function table.

[0137] Accordingly, as represented by block 604, the function call may result in a request for the function table. The processor submits this request by issuing a read (e.g., a load word operation) on the data bus.

[0138] As discussed above, the instruction multiplexer may be configured to handle read requests on the data bus for a function table. If the requested function table is stored in the internal ROM (blocks 606-610 are inapplicable to this case), the instruction multiplexer retrieves the function table and returns it to the processor via the data bus (block 612).