(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 August 2007 (23.08.2007)

PCT

(10) International Publication Number
WO 2007/094857 A1

(51) International Patent Classification:
G06F 21/00 (2006.01) *G06F 21/24* (2006.01)

(21) International Application Number: PCT/US2006/048218

(22) International Filing Date: 18 December 2006 (18.12.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/771,692 9 February 2006 (09.02.2006) US

(71) Applicant *(for all designated States except US)*: **THOMSON LICENSING** [FR/FR]; 46, Quai A. Le Gallo, F-92100 Boulogne-billancourt (FR).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **DUFFIELD, David, Jay** [US/US]; 5459 Fall Creek Rd., Indianapolis, Indiana 46220 (US).

(74) Agents: **LAKS, Joseph, J.** et al.; Thomson Licensing Inc., Two Independence Way, Suite #200, Princeton, New Jersey 08540 (US).

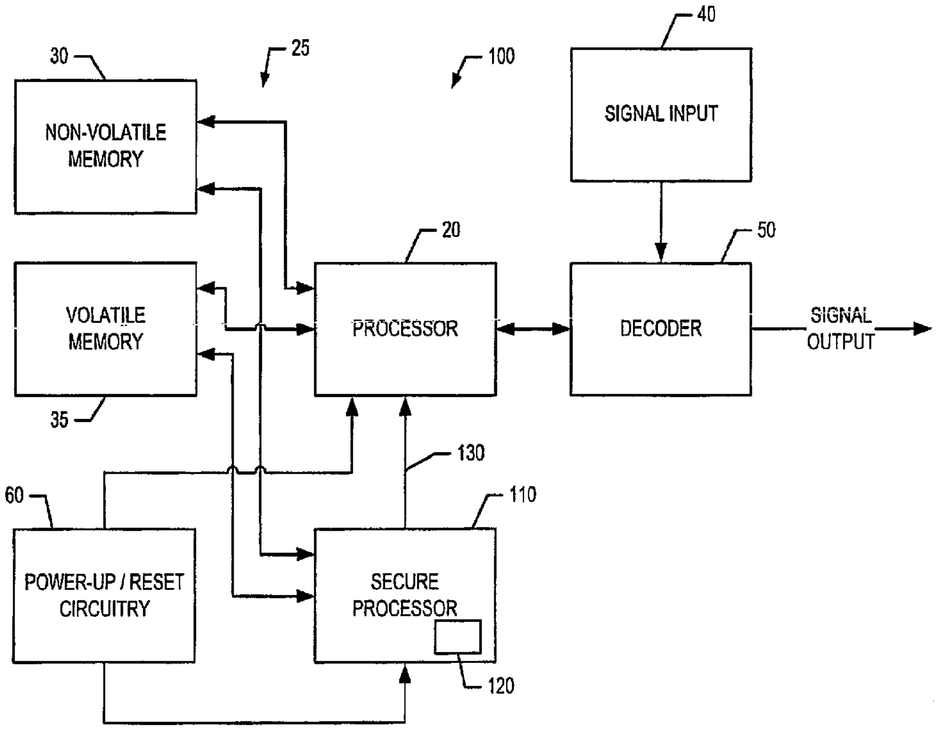
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR SECURING DIGITAL CONTENT



(57) **Abstract:** A method for continuously checking data integrity comprises generating a random number; retrieving data; generating an authentication value in response to the random number and the retrieved data; storing the data and the authentication value; generating a subsequent authentication value from the stored data and the random number; and comparing the stored authentication value with the subsequent authentication value to check data integrity.

WO 2007/094857 A1

METHOD AND APPARATUS FOR SECURING DIGITAL CONTENT

Cross Reference To Related Application

This application claims priority to and all benefits accruing from a provisional application filed in the United States Patent and Trademark Office on February 9, 2006, and there assigned serial number 60/771,692.

Field of the Invention

[01] The present invention relates generally to digital content delivery systems, and more particularly to digital content receivers.

Background of the Invention

[02] Fig. 1 shows a conventional digital set-top box (STB) architecture 10. Architecture 10 includes a processor 20, along with non-volatile memory 30 and volatile memory 35. "Processor", as used herein, refers generally to a computing device including a Central Processing Unit (CPU), such as a microprocessor. A CPU generally includes an arithmetic logic unit (ALU), which performs arithmetic and logical operations, and a control unit, which extracts instructions (e.g., a computer program incorporating code) from memory and decodes and executes the instructions, calling on the ALU when necessary. "Memory", as used herein, refers generally to one or more devices capable of storing data, such as in the form of chips, tapes, disks or drives. Memory may take the form of one or more random-access memory (RAM), read-only memory (ROM), programmable read-only memory (PROM), erasable programmable read-only memory (EPROM), or electrically erasable programmable read-only memory (EEPROM) chips, by way of example only. Memory may be internal or external to an integrated unit, e.g. an integrated circuit (IC), including a processor.

[03] In normal operation, digital content is received using input 40.. Input 40 may take the form of a satellite receiver, Internet Protocol (IP) receiver or digital cable television receiver, for example. The received content is decoded using decoder 50 responsively to processor 20 executing software instructions, *e.g.*, processor executable code, accessed via memory bus 25. Power-up and reset circuitry 60 is used to operate, boot and/or re-boot architecture 10 in a conventional manner. Such architecture is well understood by those possessing an ordinary skill in the pertinent arts.

[04] One drawback of architecture 10 of FIG.1 is its susceptibility to hacking. For example, a hacker can replace the original equipment manufacturer's (OEMs) or other authorized software, such as processor executable code being stored in memory 30 and/or 35, with unauthorized, or modified software, for the purposes of copying or stealing digital content or for other illegal or unauthorized purposes.

[05] Accordingly, it is desirable to provide a method and apparatus that can operate to prevent hackers or pirates from replacing a set-top box's core software with their own or modified software, to prevent or impede unauthorized capture or viewing of digital content.

Summary of the Invention

[06] A method for continuously checking data integrity, including: generating a random number; retrieving data; generating an authentication value in response to the random number and the retrieved data; storing the data and the authentication value; generating a subsequent authentication value from the stored data and the random number; comparing the stored authentication value with the subsequent authentication value to check data integrity.

Brief Description of the Figures

[07] Understanding of the present invention will be facilitated by consideration of the following detailed description of the preferred embodiments of the present invention taken in conjunction with the accompanying drawings, in which like numerals refer to like parts and in which:

[08] FIG. 1 illustrates a block diagram of a conventional digital set-top box (STB) architecture;

[09] FIG. 2 illustrates a block diagram of a digital set-top box (STB) architecture according to an embodiment of the present invention; and

[10] FIGS. 3-6 illustrate flow diagrams depicting process flows associated with the secure processor, main processor and memory in accordance with an embodiment of the invention.

Detailed Description of the Invention

[11] It is to be understood that the figures and descriptions of the present invention have been simplified to illustrate elements that are relevant for a clear understanding of the present invention, while eliminating, for purposes of clarity, many other elements found in typical decoding methods and systems. However, because such elements are well known in the art, and because they do not facilitate a better understanding of the present invention, a discussion of such elements is not provided herein. The disclosure herein is directed to all such variations and modifications known to those skilled in the art.

[12] Fig. 2 shows a block diagram of a digital content receiver architecture 100 according to an embodiment of the present invention. Architecture 100 may be embodied as a set-top box analogous to that of Fig. 1. Alternatively, architecture

100 may be included in another device, such as a personal video recorder (PVR) or a digital television, for example. Like elements in architectures 10 and 100 have been labeled using like references. Architecture 100 additionally includes secure processor 110 with embedded memory 120. Memory 120 may include both volatile and non-volatile memory. Volatile memory used as part of memory 120 and/or 35 may take the form of DDR RAM memory. Non-volatile memory used as part of memory 120 and/or 30 may take the form of boot ROM and/or flash memory. Secure processor 110 may take the form of a conventional secure microprocessor, or integrated circuit (IC) incorporating a microprocessor, for example. Processors 20, 110 may be embedded within a common integrated circuit, for example. In one embodiment, processor 20, 110 may be embedded within a common integrated circuit with decoder 50, for example.

[13] Architecture 100 may support direct memory access (DMA), and allow for DMA data transfers. A DMA data transfer essentially copies a block of memory from one device to another. The transfer may be performed by a DMA controller, which is incorporated into the secure processor 110. Alternatively, bus mastering DMA, where the device takes control of the bus and performs the transfer itself, may be utilized.

[14] Referring now also to Fig. 3, there is shown a flow diagram of a process according to an embodiment of the present invention, and suitable for use with architecture 100 of Fig. 2. Upon boot-up, or reset, secure processor 110 unpacks processor 20 executable code from non-volatile memory 120 at block 310. "Unpacking", as used herein, generally refers to un-compressing. According to an embodiment of the present invention, the secure processor 110 transfer of processor 20 executable code at block 310 may include decrypting, where the processor code is stored in memory 120 in an encrypted form. The code may be unpacked to memory 120, for example.

[15] The packed and/or unpacked code is initially checked for validity by secure processor 110, such as by verifying the integrity of an image or at least a portion thereof, at block 320. Here the term image may refer to the code or a portion of the code. If validated, secure processor 110 stores the processor 20 executable code in volatile memory 35, and the main processor 20 is permitted to operate, *e.g.*, boot, using the secure processor 110 volatile memory 35 loaded boot data, at block 330. Thereafter, the code stored in memory 35 is rechecked for validity, at block 340. Such a re-check may occur periodically, or aperiodically, such as upon the occurrence of a predetermined event, or substantially randomly. If the code stored in volatile memory 35 is determined to be valid by a re-check, architecture 100 continues to operate in a normal fashion, *e.g.*, analogous to operation of STB architecture 10. If the stored code is determined to be invalid at either block 320 or block 340, architecture 100 may be reset, such that processing returns to block 310.

[16] In one embodiment of the present invention, a secure boot is performed. Referring now also to Fig. 4, there is shown a flow diagram of one embodiment of secure boot actions taken at blocks 310, 320. When the architecture 100 is booted or re-booted (*e.g.*, reset), secure processor 110 identifies non-volatile memory 30, such as by checking a jumper configuration, at block 311. At block 312, an initial security state may be established by setting one or more registers to a default condition. One such register may be incorporated within secure processor memory 120 or within memory 35, for example, such that the register is coupled to a reset pin of processor 20 and set to a default value that inhibits operation of processor 20.

[17] At block 313, secure processor 110 boots, such as by using code stored within a boot sector of memory 120. At block 314, secure processor 110 selects a key to authenticate processor 20 executable code stored in non-volatile memory 30. The key may take the form of any data suitable for authenticating code stored in memory 30. For example, the selected key may take the form of a Rivest,

Shamir, and Adelman (RSA) algorithm compatible public key. The key may be stored in memory 120. Optionally, more than one key may be stored in memory 120, and one of the stored keys selected in any conventional manner; as long as the selected key is a public key that corresponds to a private key used to digitally sign the memory 30 boot block stored data, or a portion thereof.

[18] At block 315, secure processor 110 reads the boot block from memory 30. Block 315 may include the operation of identifying the boot block location in non-volatile memory 30 using a pointer, and loading the boot block stored data (*e.g.*, a 7 kilobyte data portion) into memory 120.

[19] At block 316, secure processor 110 authenticates the memory 30 read boot block using the selected key. By way of further example, the read boot block may be verified as being created by a party in possession of the private key corresponding to the public key selected at block 314, by verifying the authenticity of a digital signature incorporated with the boot block data using the selected key.

[20] At block 317, once the boot block is authenticated at block 316, processor 20 executable code stored in non-volatile memory 120 is transferred to volatile memory 35, for subsequent use by processor 20. According to an embodiment of the present invention, processor 20 executable code stored in memory 30 may be encrypted prior to unpacking, to prevent or frustrate unauthorized attempts to operate processor 20. In such a case, block 317 may involve decrypting the processor executable code stored in non-volatile memory 30 and transferring it to volatile memory 120. Where the same RSA private key used to digitally sign the boot block is used to encrypt the processor executable code, the same selected key may be used to decrypt the processor executable code. Alternatively, a different key pair (or even a symmetric key) stored in memory 120 may be used to decrypt the processor 20 executable code. An analogous decrypting can be used as part of block 310.

[21] Referring again to Fig. 3, processor 20 is booted at block 330. According to an embodiment of the present invention, block 330 involves secure processor 110 changing the default security configuration (*e.g.*, setting or resetting the register coupled to the processor 20 reset pin), to enable processor 20 to boot.

[22] Alternatively, any suitable secure boot methodology may be utilized.

[23] Referring still to Fig. 3, the software is again validated at block 340. In one configuration, as part of block 320, a random number is generated and stored. An authentication value indicative of the software (using the random number as a seed value) is also generated and stored in memory. After some temporal period, such as periodically, or upon the occurrence of a triggering event, or pseudo-randomly, the authentication value is again calculated (*i.e.*, recalculated) at block 340. If the software has not been tampered with, the re-calculated authentication value will match the stored authentication value. If the software has been tampered with or replaced, these authentication values will not match. Thus, by comparing each re-calculated authentication value with the stored authentication value, the software may continue to be validated. If the authentication values do not match, architecture 100 (Fig. 2) may be reset, so as to re-initialize the software to a valid condition, such as by secure processor 110 setting or resetting a register coupled to the reset pin of processor 20.

[24] According to an embodiment of the present invention, primitive cryptographic functions can be used to provide a suitable authentication value or code for the processor 20 executable software. One such example may be implemented utilizing a cryptographic hash function. Another such example is implemented utilizing a message authentication code generating function, such as cipher block chaining, where a cryptographic block cipher is used, like DES, 3DES, for example. Referring now also to Fig. 5, there is shown a flow diagram of a process 500 that may be used at blocks 320, 340 (Fig. 3). According to an embodiment of the present invention, either during or after processor 20 executable code initial validation at block 320, a random number may be determined at block 510. Thereafter, at block 520, an authentication value of at least a portion of the processor 20 executable code and the random number is generated. "Authentication value", as used herein, generally refers to a small digital value akin to a "fingerprint" of data. An authentication value is commonly represented as a short string of random-looking letters and numbers. Where an authentication value implementation is utilized, a variety of hash functions may be used to generate the authentication value. Examples of conventional hash functions include: HAVAL, MD2, MD4, MD5, PANAMA, RIPEMD, SHA-x, Tiger(2) and WHIRLPOOL.

[25] Referring still to Fig. 5, once the authentication value is determined (at block 520), the authentication value may be stored at block 530. The authentication value may be stored in memory 120 of secure processor 110 (Fig. 2), to frustrate unauthorized efforts to alter it, for example. The random number determined at block 510 may also be stored in memory 120. At block 540, it is determined whether the processor 20 executable code stored in memory 35 (Fig. 2) should be re-validated. This may occur at pre-determined intervals, such as once every x seconds, minutes or hours, or upon the occurrence of a pre-determined event, such as a channel change or a digital content event occurring (*e.g.*, a user selected program commencing). Alternatively, it may occur pseudo-randomly, such as by determining another pseudorandom number, and using it in combination with a counter as a pseudorandom timer.

[26] When it is determined the processor 20 executable code should be re-validated at block 540, a subsequent authentication value is determined at block 550. To determine a subsequent authentication value, the random number determined at block 510 is recovered from memory 120 (Fig. 2) and used in combination with the processor 110 executable code then stored in memory 120 to determine a subsequent authentication value using the same methodology as was used at block 520.

[27] The authentication value stored at block 530 and subsequent authentication value determined at block 550 are then compared at block 560. If the processor 20 executable code has not been tampered with, the subsequently determined and stored authentication values will match. If the processor 20 executable code has been tampered with or replaced, the authentication values will not match. Thus, by comparing the subsequently determined authentication value with the stored authentication value at block 560, the processor 20 executable code may continue to be validated. If the authentication values do not match, processor 20 may be reset at block 570, so as to re-initialize the software to a valid condition. If the authentication values do match, processing may return to block 540.

[28] Referring now to Fig. 6, there is shown a flow diagram of an authentication process 600 suitable for use at blocks 520, 550 of the process of Fig. 5. At block 620, a memory address is determined using the random number determined at block 510 (Fig. 5). The random number is converted to a physical address in volatile memory 35. This may be accomplished in any suitable manner, such as by truncating a number of most significant bits of the random number (if necessary), and adding the remaining least significant bits portion to a given physical address, such as the lowest physical address, in memory 35. Of course, another physical address may alternatively be used as a starting point.

[29] At block 630, an authentication value of the data content of the memory address determined at block 620 is calculated. According to an embodiment of the present invention, an encrypting DMA transfer of the data content of the memory address determined at block 620 to a register (such as a register included in memory 120 or memory 35, Fig. 2) may be used. The key used in the encrypting DMA transfer may be the same public key that was used to initially authenticate the processor executable code, or may take the form of a different key. Such an encrypting DMA transfer may be accomplished by the DMA engine itself, for example.

[30] At block 640, it is determined if more memory locations are to be considered in the authentication value. This may be accomplished by determining what percentage of the stored processor code is statistically significant for purposes of confirming the processor executable code stored in memory 35 has not changed. The address and authentication value determination is then repeated on at least a number of addresses in memory 35 that corresponds to that percentage. Alternatively, a given number of memory locations, such as 2^{16}, may be considered in the authentication value determination.

[31] When it is determined the data content of additional memory location(s) is to be considered in the authentication value, another address in memory 35 is determined at block 620. According to an embodiment of the present invention, the subsequent address locations in memory 35 may be determined in a pseudorandom fashion. This may be accomplished by performing an encrypting DMA transfer in place using the previous address and a key. Again, the key used may be the same public key that was used to initially authenticate the processor executable code, or may take the form of a different key. Such an encrypting DMA transfer may be accomplished by the DMA engine itself, for example.

[32] Thereafter, a new authentication value is determined at block 630. This may be accomplished by exclusive OR'ing the data contents of the currently determined memory address with the prior determined authentication value, and again performing an encrypting DMA transfer of the resultant in place in an authentication value storing register.

[33] Processing again returns to block 640, such that the memory address and authentication value determining are repeated until a final authentication value is determined. Thereafter, processing ends at block 650.

[34] It will be apparent to those skilled in the art that modifications and variations may be made in the apparatus and process of the present invention without departing from the spirit or scope of the invention. It is intended that the present invention cover the modification and variations of this invention provided they come within the scope of the appended claims and their equivalents.

CLAIMS

1. A method for continuously checking data integrity, comprising:

generating a random number;

retrieving data;

generating an authentication value in response to the random number and the retrieved data;

storing the data and the authentication value;

generating a subsequent authentication value from the stored data and the random number; and

comparing the stored authentication value with the subsequent authentication value to check data integrity.

2. The method of Claim 1, wherein the data is retrieved from a first memory, and the authentication value and data are stored in at least one memory distinct from the first memory.

3. The method of Claim 2, wherein the first memory is a read-only memory.

4. The method of Claim 2, wherein the authentication value and random number are stored in a second memory, and the data is stored in a third memory distinct from the second memory.

5. The method of Claim 1, further comprising:

periodically generating subsequent authentication values from the stored data and the random number; and,

comparing the stored authentication value with the subsequent authentication values to check data integrity.

6. The method of Claim 1, further comprising detecting an event, wherein the generating a subsequent authentication value from the stored data and the random number and comparing the stored authentication value with the subsequent authentication values to check data integrity are performed responsively to the detecting.

7. The method of Claim 1, further comprising setting a timer, wherein the generating a subsequent authentication value from the stored data and the random number and comparing the stored authentication value with the subsequent authentication values to check data integrity are performed responsively to the timer.

8. The method of Claim 1, further comprising:

generating subsequent authentication values from the stored data and the random number; and,

comparing the stored authentication value with the subsequent authentication values to check data integrity;

wherein, the subsequent authentication values are generated and compared at substantially random times.

9. A method for checking the integrity of processor executable code stored in a first memory, said method comprising:

generating an authentication value using at least a portion of the stored processor executable code;

storing the generated authentication value in a second memory;

generating subsequent authentication values using the at least said portion of processor executable code; and

comparing the stored authentication value with the subsequent generated authentication values.

10. The method of Claim 1, further comprising operating a processor responsively to the stored processor executable code.

11. The method of Claim 10, wherein the processor operates over a time period that commences prior to generating said subsequent authentication values.

12. The method of Claim 11, further comprising resetting the processor when one of the subsequent authentication values does not match the stored authentication value.

13. The method of Claim 12, wherein the resetting comprises retrieving the processor executable code from a third memory and storing said retrieved processor executable code in the first memory.

14. The method of Claim 9, wherein said generating the stored authentication value comprises: determining a substantially random number.

15. The method of Claim 14, wherein generating the stored authentication value further comprises: determining a first memory address using the determined substantially random number.

16. The method of Claim 15, wherein the determining a first memory address comprises truncating a value.

17. A device comprising:

a processor;

a memory;

first processor executable code being stored in the memory;

data indicative of a random number being stored in the memory;

data indicative of an authentication value being stored in the memory; and,

second processor executable code being stored in the memory;

wherein, when executed, the second processor executable code causes:

a subsequent authentication value to be determined dependently upon the first processor executable code being stored in the memory and the data indicative of a random number being stored in the memory; and,

the subsequent authentication value to be compared with the data indicative of an authentication value being stored in the memory.

18. The device of Claim 17, wherein the processor and memory are embedded within a common integrated circuit.

19. The device of Claim 17, further comprising a data bus coupled to the processor and memory.

20. The device of Claim 17, further comprising:

third processor executable code being stored in the memory;

wherein, when executed, the third processor executable code resets the processor.

Fig. 1
"Prior Art"

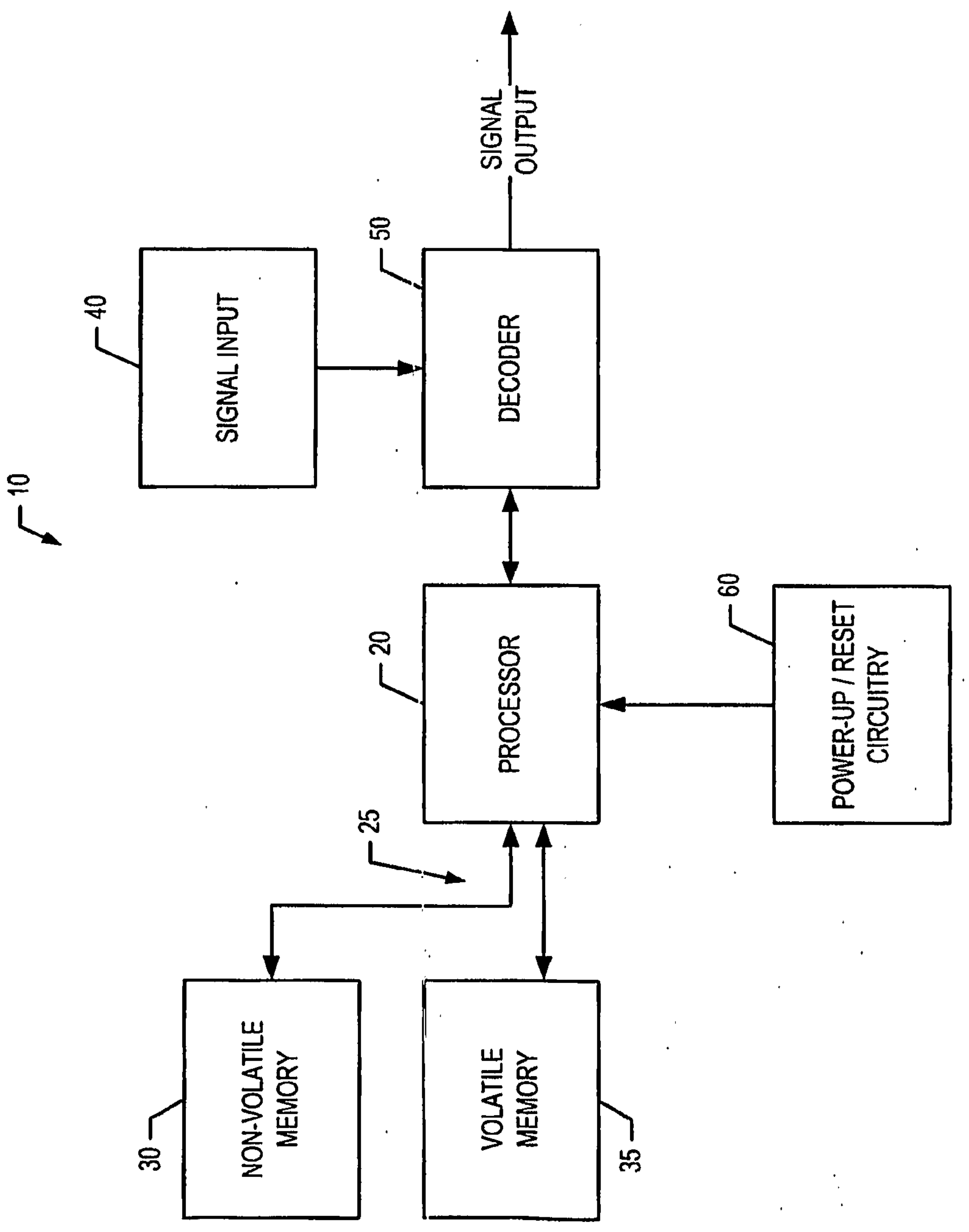


Fig. 2

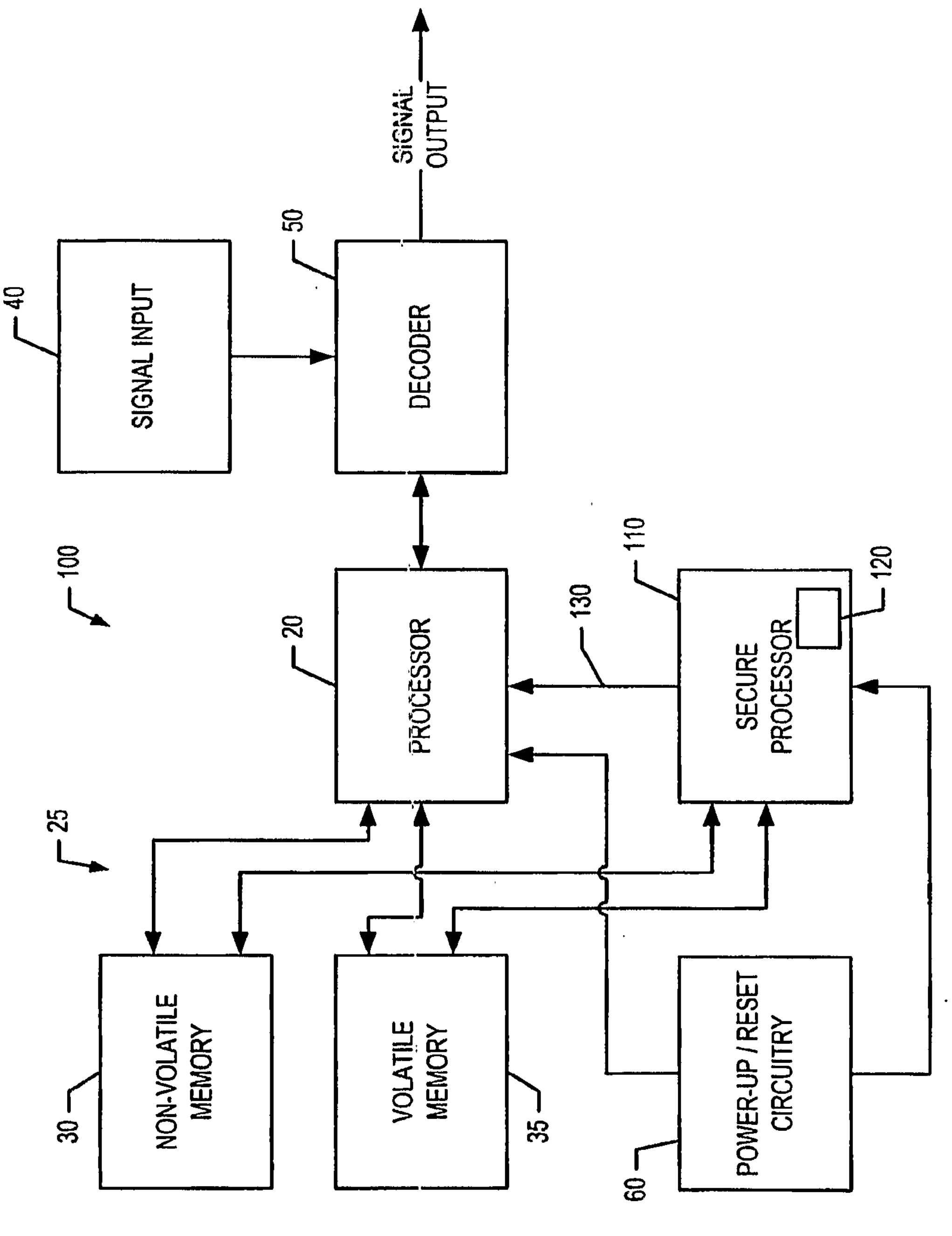

100
25
SIGNAL INPUT
40
DECODER
50
SIGNAL OUTPUT
PROCESSOR
20
130
SECURE PROCESSOR
110
120
NON-VOLATILE MEMORY
30
VOLATILE MEMORY
35
POWER-UP / RESET CIRCUITRY
60

Fig. 3

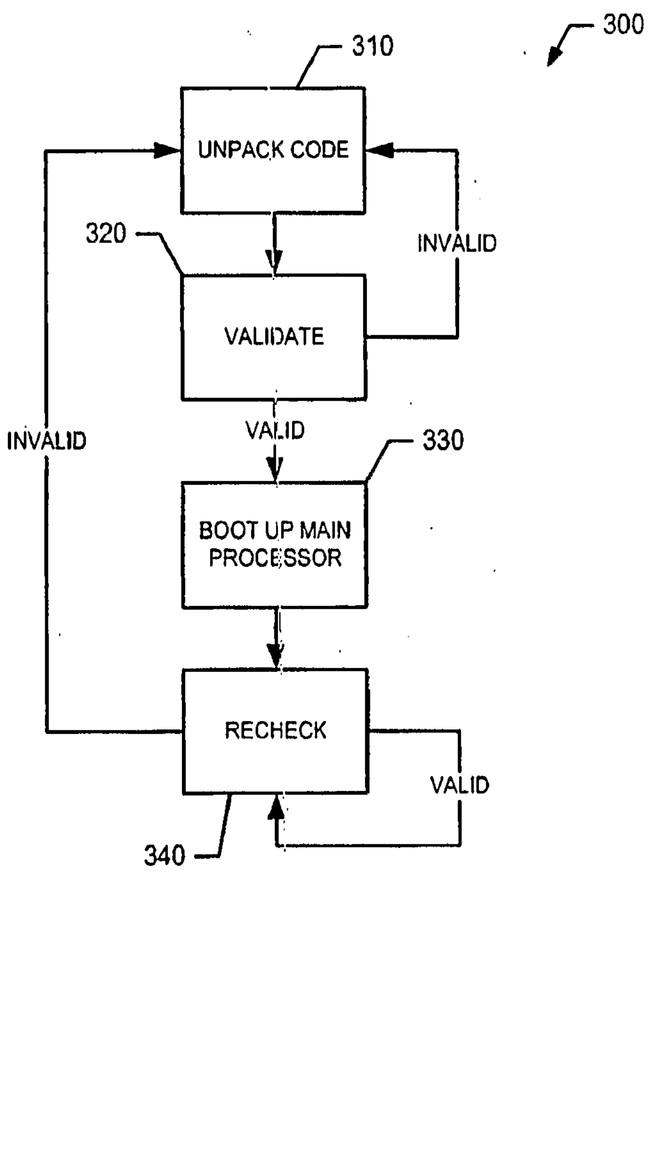
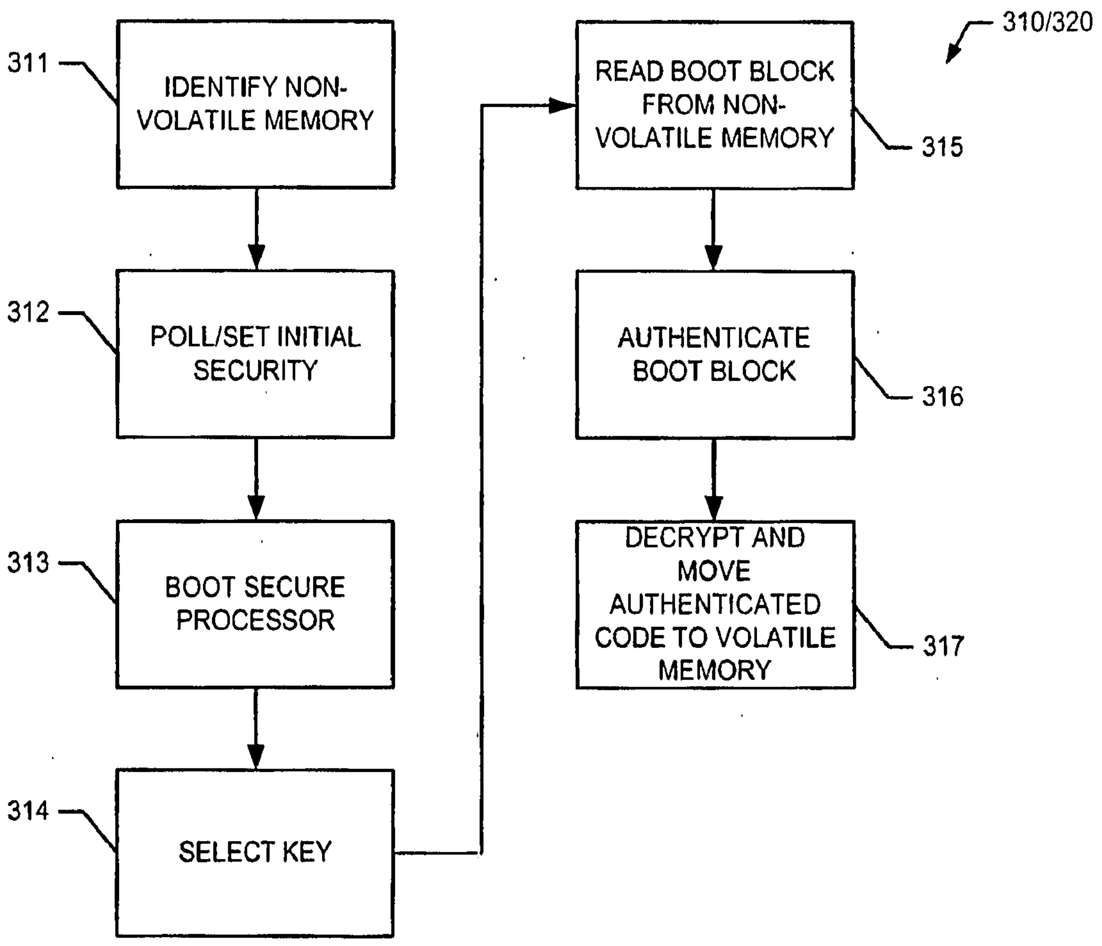

300
310
UNPACK CODE
320
VALIDATE
INVALID
VALID
330
BOOT UP MAIN PROCESSOR
INVALID
RECHECK
VALID
340

Fig. 4


310/320
311
IDENTIFY NON-VOLATILE MEMORY
312
POLL/SET INITIAL SECURITY
313
BOOT SECURE PROCESSOR
314
SELECT KEY
READ BOOT BLOCK FROM NON-VOLATILE MEMORY
315
AUTHENTICATE BOOT BLOCK
316
DECRYPT AND MOVE AUTHENTICATED CODE TO VOLATILE MEMORY
317

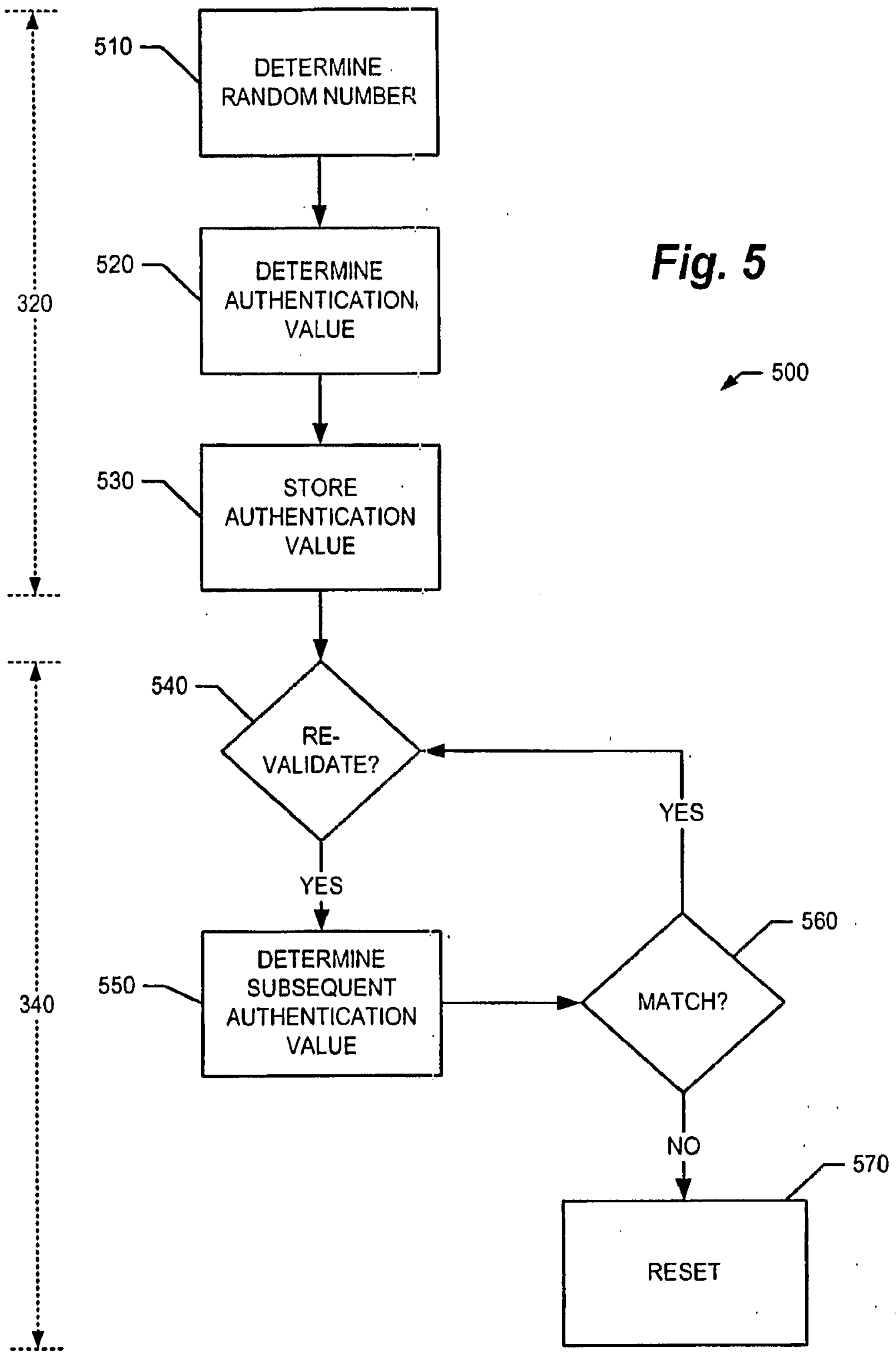
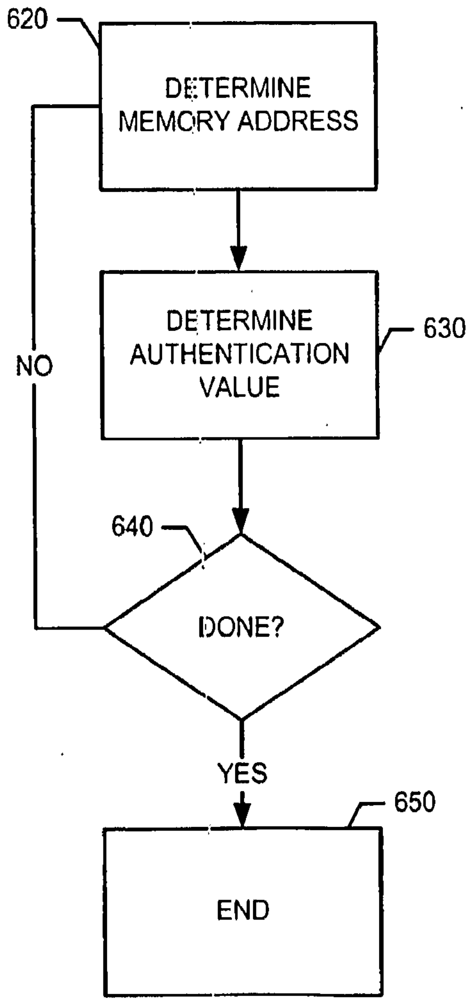

510
DETERMINE RANDOM NUMBER
520
DETERMINE AUTHENTICATION VALUE
530
STORE AUTHENTICATION VALUE
320
540
RE-VALIDATE?
YES
550
DETERMINE SUBSEQUENT AUTHENTICATION VALUE
560
MATCH?
YES
NO
570
RESET
340
500

Fig. 5

Fig. 6


620
DETERMINE MEMORY ADDRESS
DETERMINE AUTHENTICATION VALUE
630
NO
640
DONE?
YES
650
END

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/048218

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/00 G06F21/24

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 757 919 A (HERBERT HOWARD C [US] ET AL) 26 May 1998 (1998-05-26) column 2, line 39 - column 3, line 3 column 4, line 15 - column 5, line 25 figures 3,5	1-4,6,7, 9-20
X	WO 2005/091108 A (NOKIA CORP [FI]; PAATERO LAURI [FI]) 29 September 2005 (2005-09-29) page 4, line 24 - page 5, line 2 page 6, line 13 - line 18 page 8, line 12 - line 19 page 8, line 30 - page 9, line 30 page 15, line 21 - page 16, line 23 page 17, line 8 - page 18, line 4	1-20
	-/--	

[X] Further documents are listed in the continuation of Box C.   [X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search 29 June 2007	Date of mailing of the international search report 05/07/2007
Name and mailing address of the ISA/ European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Alecu, Mihail

2

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2006/048218

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	TCG: "TCG Specification Architecture Overview, Specification Revision 1.2" TCG SPECIFICATION ARCHITECTURE OVERVIEW, TRUSTED COMPUTING GROUP, US, 28 April 2004 (2004-04-28), pages 1-54, XP002413737 the whole document -----	1-20
A	MENEZES A J ET AL: "HASH FUNCTIONS AND DATA INTEGRITY" HANDBOOK OF APPLIED CRYPTOGRAPHY, CRC PRESS SERIES ON DISCRETE MATHEMATICES AND ITS APPLICATIONS, BOCA RATON, FL, CRC PRESS, US, 1997, pages 321-383, XP002275660 ISBN: 0-8493-8523-7 the whole document -----	1-20

2

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2006/048218

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5757919 A	26-05-1998	AU 5688998 A	03-07-1998
		DE 19782169 C2	06-09-2001
		DE 19782169 T0	28-10-1999
		GB 2334866 A	01-09-1999
		HK 1022797 A1	22-03-2002
		JP 2001508893 T	03-07-2001
		WO 9826535 A1	18-06-1998
WO 2005091108 A	29-09-2005	EP 1725923 A1	29-11-2006
		KR 20060127206 A	11-12-2006
		US 2005210287 A1	22-09-2005