(51) International Patent Classification:
G06F 21/57 (2013.01)  G06F 11/22 (2006.01)

(21) International Application Number:
PCT/IB2012/0055 148

(22) International Filing Date:
27 September 2012 (27.09.2012)

(25) Filing Language:
English

(26) Publication Language:
English

(30) Priority Data:
61/634,430 29 February 2012 (29.02.2012) US

(71) Applicant (for all designated States except US): NDS LIMITED [GB/GB]; One London Road, Staines Middlesex TW18 4EX (GB).

(72) Inventors; and

(71) Applicants (for US only): SELLA, Yaaron [IL/IL]; 223 Moshav Beit Nekofa, P.O.Box 115, 90830 Beit Nekofa (IL). FRAENKEL, Ittai [IL/IL]; 2 Leshem Street, 99785 Nof Ayalon (IL). MANTIN, Itzik [IL/IL]; 6 Hamizpe Street, 6085000 Shoham (IL).

(74) Agents: KATZ, Samuel, M. et al; NDS Legal (Patents), One London Road, Staines Middlesex TW18 4EX (GB).


Published:
— with international search report (Art. 21(3))

(54) Title: CODE CHECKING SYSTEM

FIG. 4

(57) Abstract: A system including a counter having a value, a processor to perform, as part of a boot-loading process, a code check including performing a partial check or a full check of the program code dependent on the value of the counter, the program code having a plurality of sections, adjust the counter value in a first direction if the full check is successful, and adjust the counter value in a second direction, if the partial check fails, and continue with the boot-loading process if the code check is successful, wherein the full check includes authenticating all of the program code against a first digital signature, the partial check includes authenticating one of the sections of the program code against a second digital signature, the first signature being the same as, or different from, the second signature. Related apparatus and methods are also described.
The present invention relates to checking program code authenticity.

For security reasons, devices are often designed to perform, upon boot, a public key signature check of their own code prior to executing the code. The motivation is to ensure the device executes authorized code only. The problem is that as the device's code size gets larger, the above signature check takes longer, and as a result the device boot time increases. This annoys the user, and in some cases, it is simply unacceptable. Generally, the most time-consuming part in a public key signature check is cryptographic hashing of the code.

The following references are also believed to represent the state of the art:


US Patent 6,718,461 to Ewertz; and

SUMMARY OF THE INVENTION

The present invention, in certain embodiments thereof, seeks to provide an improved code checking system.

There is thus provided in accordance with an embodiment of the present invention, a system including a counter having a value, a first processor to perform, as part of a boot-loading process, a code check including performing a partial check of program code or a full check of the program code dependent on the value of the counter, the program code having a plurality of sections, adjust the counter value in a first direction if the full check is successful, and adjust the counter value in a second direction, opposite to the first direction, if the partial check fails, and continue with the boot-loading process if the code check is successful, wherein the full check includes authenticating all of the program code against a first digital signature, the partial check includes authenticating one of the sections of the program code against a second digital signature, and the partial check excludes authenticating all the sections of the program code, the first signature being the same as, or different from, the second signature.

Further in accordance with an embodiment of the present invention, the partial check includes randomly or pseudo-randomly selecting the one section from the plurality of sections of the program code.

Still further in accordance with an embodiment of the present invention, the system includes a second processor operative to execute the program code, wherein the first processor is operative only to run the boot-loading process and the counter is protected so that the counter can be adjusted or reset by the first processor but not by the second processor.

Additionally in accordance with an embodiment of the present invention, the counter is protected from being reset or adjusted manually.

Moreover in accordance with an embodiment of the present invention, the first processor operative is also operative to execute the program code, wherein the counter is protected so that the counter can only be adjusted or reset by the first processor during the boot-loading process.
Further in accordance with an embodiment of the present invention, the first digital signature is different from the second digital signature which is different for each one of the sections of the program code so that there are a plurality of second digital signatures.

Still further in accordance with an embodiment of the present invention, the second digital signatures are compressed together into a single physical structure.

Additionally in accordance with an embodiment of the present invention, the first digital signature is different from the second digital signature.

Moreover in accordance with an embodiment of the present invention, the first digital signature is the same as the second digital signature.

Further in accordance with an embodiment of the present invention, the first processor is operative to decrement the counter value if the full check is successful, increment the counter value if the partial check fails, and perform the full check of the program code when the value of the counter is greater than, or equal to, a threshold value.

Still further in accordance with an embodiment of the present invention, the first processor is operative to increment the counter value if the full check is successful, decrement the counter value if the partial check fails, and perform the full check of the program code when the value of the counter is less than, or equal to, a threshold value.

There is also provided in accordance with still another embodiment of the present invention, a method including performing, as part of a boot-loading process, a code check including performing a partial check of program code or a full check of the program code dependent on the value of the counter, the program code having a plurality of sections, adjusting the counter value in a first direction if the full check is successful, adjusting the counter value in a second direction, opposite to the first direction, if the partial check fails, continuing with the boot-loading process if the code check is successful, wherein the full check includes authenticating all of the program code against a first digital signature, the partial check includes authenticating one of the sections of the program code against a
second digital signature, and the partial check excludes authenticating all the
sections of the program code, the first signature being the same as, or different
from, the second signature.
BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be understood and appreciated more fully from the following detailed description, taken in conjunction with the drawings in which:

Fig. 1 is a partly pictorial, partly block diagram view of a system constructed and operative in accordance with an embodiment of the present invention;

Fig. 2 is a partly pictorial, partly block diagram view of a system constructed and operative in accordance with an alternative embodiment of the present invention;

Fig. 3 is a partly pictorial, partly block diagram view of the system of Fig. 1 authenticating code;

Fig. 4 is a partly pictorial, partly block diagram view of the code signed in accordance with an alternative method for use in the system of Fig. 1;

Fig. 5 is a flow chart showing a first mode of operation of the code checking system of Fig. 1; and

Fig. 6 is a flow chart showing a second mode of operation of the code checking system of Fig. 1.
DETAILED DESCRIPTION OF AN EMBODIMENT

Reference is now made to Fig. 1, which is a partly pictorial, partly block diagram view of a system 10 constructed and operative in accordance with an embodiment of the present invention.

The system 10 typically includes a counter 12 and a processor 14. The processor 14 is operative to check program code 16.

In addition to performing the boot-load, the processor 14 is operative to execute the program code 16 after a successful boot-load procedure.

As part of the code checking procedure, the processor 14 uses the counter 12 as a private sanction counter which is stored in non-volatile memory. The use of the counter will be described in more detail below with reference to Figs. 3 to 5.

The counter 12 is typically protected from being reset or adjusted manually (by human intervention) and is accessible for writing (adjusting or resetting) by the processor 14 only during the boot-loading process and not during execution of the program code 16. One mechanism for implementing the private sanction counter 12 can be a one-time-write random access memory (RAM) bit, which allows writing to the sanction counter 12 only when the RAM bit is zero. The processor 14 would typically set the RAM bit to one before it terminates and transfers control to the main image of the program code 16. The one-time write RAM bit is typically reset when the CPU is reset.

Reference is now made to Fig. 2, which is a partly pictorial, partly block diagram view of a system 18 constructed and operative in accordance with an alternative embodiment of the present invention.

The system 18 is substantially the same as the system 10 of Fig. 1, except that in the system 18, the processor 14 is operative to only run the boot-loading process which checks at least part of the program code 16 and another processor 20 is operative to execute the program code 16 after successful completion of the boot-loading.
The counter 12 is typically a private sanction counter protected so that the counter 12 can be adjusted or reset by the processor 14 but not by the processor 20. Additionally, the counter 12 is protected from being reset or adjusted manually. The private sanction counter 12 can secured for "private use" by a hardware memory protection unit ( MPU) or memory management unit ( MMU) mechanism, by way of example only.

For the sake of simplicity, the system for checking the program code 16 will now be described with reference to the system 10 of Fig. 1. However, it will be appreciated by those ordinarily skilled in the art that the code checking described herein may also be implemented with any suitable code checking system, for example, but not limited to, the system 18 of Fig. 2.

Reference is now made to Fig. 3, which is a partly pictorial, partly block diagram view of the system 10 of Fig. 1 authenticating the code 16.

The program code 16 includes n sections 22.

The value of n is greater than 1 and typically less than 20, for example, but not limited to, n = 5. It will be appreciated by those ordinarily skilled in the art that n may have any suitable value and even a value greater than 20 may be suitable in certain applications.

For authentication purposes, the program code 16 is signed with an entire code signature 26 and each of the sections 22 is individually signed with a different signature 24. The signature 26 is typically different from any of the signatures 24. Fig. 4 describes the program code 16 and the sections 22 being signed in accordance with a different scheme.

As the device running the program code 16 needs to store more signatures (the signatures 24 of each section 22 and the entire code signature 26), the signatures 24 and the entire code signature 26 may be compressed if possible. By way of example, if an RSA signature scheme is used, then the signatures 24 and the entire code signature 26 can potentially be compressed into a single, physical signature.
In overview, the processor 14 randomly or pseudo-randomly selects one section 22 of the program code 16. The processor 14 then calculates a hash, using a cryptographic hash function (e.g. SHA-1 or SHA-256) on the selected section 22, instead of on the entire program code 16 yielding a hash value 28 which is authenticated against the signature 24 of the selected section 22 typically using a public key 32 as is known in the art of digital signature authentication (block 30). Alternatively, non-asymmetric digital signature authentication may be performed using a suitable secret.

As the hash is calculated for the selected section 22, instead of on the entire program code 16, the partial check typically cuts the signature check time to roughly 1/n of the original time, which represents a very significant saving. However, checking only 1/n of the program code 16 introduces some issues, complications and security problems that need to be addressed.

One issue is how to arrange the code in n sections. One approach is to allocate the first section 22 to bytes 0, n, 2*n, ..., the second section 22 to bytes 1, n+1, 2*n+1, ... etc. Another approach is to have the first section 22 be the first 1/n of the code 16 and the second section 22 be the second 1/n of the code 16. It will be appreciated by those ordinarily skilled in the art that there are many suitable ways in which to arrange the code 16 in n sections.

Yet another issue, is that only checking part of the code 16 is insecure. A carefully crafted rogue image would only be discovered by the processor 14 with a probability of 1/n and a signature check failure could be overcome with a simple reboot. In order to address the security problem identified above, the processor 14 uses the counter 12 as a private sanction counter which is stored in non-volatile memory.

If the partial security check fails, the value of the counter 12 is adjusted. When the counter 12 reaches a certain value, a full signature check on the entire program code 16 is required at boot. The full check takes more time, but due to security concerns the time constraint is overridden.

The full check is typically performed by the processor 14 calculating a cryptographic hash of the entire program code 16 yielding a hash
value which is authenticated against the entire code signature 26 using a public key or secret as appropriate.

How and when the counter 12 is updated is described in more detail with reference to Figs. 5 and 6.

Reference is now made to Fig. 4, which is a partly pictorial, partly block diagram view of the code 16 signed in accordance with an alternative method for use in the system 10 of Fig. 1.

For authentication purposes, the program code 16 and all the sections 22 are signed with a single signature 34 and each of the sections 22 has a hash value 36 and the entire program code 16 has a entire code hash 38. When all the hash values 36 and the entire code hash 38 are hashed together, using a cryptographic hash function, they yield another hash (a hash of the hash values 36, 38) effectively creating a two-level hash tree. The hash of the hash values 36, 38 is authenticated using the signature 34 so that any of the sections 22, or the entire program code 16, may be authenticated with the signature 34 in conjunction with the other hash values and an appropriate public key or secret, as appropriate.

So for example, in order to check one of the sections 22 (a partial check), the processor 14 (Fig. 3) calculates a hash, using a cryptographic hash function (e.g. SHA-1 or SHA-256) on the selected section 22, instead of on the entire program code 16, yielding a hash value which is compared with the existing hash value 36 of the selected section 22. If the calculated hash of the selected section 22 is not equal to the existing hash value 36 for the selected section 22, then the check of the selected section 22 has failed. If the calculated hash of the selected section 22 is equal to the existing hash value 36 for the selected section 22, then the existing hash value 36 for the selected section 22 needs to be authenticated against the signature 34. The authentication of the existing hash value 36 is performed as follows. First, the existing hash value 36 of the selected section 22 and the existing hash values 36 of all the other sections 22 and the entire code hash 38 of the entire program code 16 are hashed together, using a cryptographic hash function, yielding another hash which is authenticated against
the signature typically using a public key or secret depending on the mode of
digital signature authentication employed.

Reference is now made to Fig. 5, which is a flow chart showing a
first mode of operation of the system 10 of Fig. 1.

The value of the counter 12 (Fig. 3) typically starts at zero and the
counter value is adjusted by failures of the partial check and successes of the full
check during successive boot attempts, as will now be described in more detail.

The boot process starts at step 40.

In step 42 the counter value is compared to a threshold value. If the
counter value is less than, or equal, to a threshold value Z, a partial check is
performed (steps 44, 46). If the counter value is greater than the threshold value Z,
then a full check is performed (step 48).

Therefore, for each boot attempt either a partial check or a full
check is performed. If the check (partial or full) fails, a new boot attempt is
performed, as will be described below.

The partial checking route is now described in more detail.

In step 44, one of the sections 22 (Fig. 3) of the program code 16 is
selected (typically randomly or pseudo-randomly). In step 46, the selected
section 22 is checked for authenticity.

In step 50, if the selected section 22 is authentic the boot process
continues in step 52. If the selected section 22 is not authentic, the counter value is
incremented by X in step 54 and the process returns to the start of the process
(step 40).

The authentication of the selected section 22 may fail for various
reasons, for example, but not limited to, a processing error, an error in the
signature authenticating the selected section 22, or due to the selected section 22
having been tampered with. If the selected section 22 has been tampered with,
authentication of the entire program code 16 will fail as well. However, if the
selected section 22 has not been tampered with, then authenticating the entire
program code 16 against the entire code signature 26 (Fig. 3) may prove to be an effective authentication step without ignoring potential security risks.

In step 48, the entire program code 16 is checked for authenticity (for example against the entire code signature 26). In step 56, if the entire program code 16 is authentic, the counter value is decremented by Y in step 58, and the boot process is continued in step 52. If the authentication of the entire program code 16 fails, the boot-loading process is restarted (step 40).

If the authentication of the program code 16 fails, then depending on the values of X, Y and Z the process with continue to loop around steps 42, 48, 56 and 40 until a new version of the program code 16 is installed at which time the counter value is reset to zero.

It will be appreciate that X, Y and Z can be set to any suitable values, for example, Y could be equal to 1, X greater than or equal to one (e.g. 3), and Z less than or equal to X (e.g. 1 or 2).

Reference is now made to Fig. 6, which is a flow chart showing a second mode of operation of the system 10 of Fig. 1. The flow chart of Fig. 6 is substantially the same as the flow chart of Fig. 4 except that in step 58, instead of decrementing the counter by Y, the counter is reset to the initial value, (e.g. zero).

The process described with reference to Figs. 5 and 6 may be generalized as follows.

The counter value starts at a certain initial value (e.g. zero or could be any other positive or even negative value) and increases (by fixed value or as a function (e.g. a fraction or exponential function) of the counter value) on failure of the partial check. The flowchart in Fig. 5 checks for the counter value being greater than the threshold value. However, the check for performing a full check could be when the counter value equals (or hits) the threshold value. Therefore, the partial check is performed unless the counter value hits or exceeds (depending on the criteria) a threshold value (could be a negative value, a positive value or zero) thereby requiring that a full check be performed. If the full check is successful, the counter value is reset to the initial value or reduced by a certain value (by a fixed value or as a function (e.g. a fraction or exponential function) of the counter value).
value). When a new download is received the download is checked in full and the counter value is reset to the initial value.

Other options are now described below.

The counter value starts at a certain initial value (e.g. zero or it could be any other positive or even negative value) and is reduced (by a fixed value or as a function (e.g. a fraction or exponential function) of the counter value) on failure of the partial check. The partial check is performed unless the counter value hits or falls below a certain threshold value (could be a negative value, a positive value or zero) thereby requiring that a full check be performed. If the full check is successful, the counter value is reset to the initial value or increased by a certain value (by a fixed value or as a function (e.g. a fraction or exponential function) of the counter value). When a new download is received, the download is checked in full and the counter value is reset to the initial value.

Another option is as follows. The counter value starts at a certain initial value (e.g. zero or could be any other positive or even negative value) and is increased (by a fixed value or as a function (e.g. a fraction or exponential function) of the counter value) on success of the full check. The full check is performed unless the counter value hits or exceeds a certain threshold value (could be a negative value, a positive value or zero) thereby allowing a partial check to be performed. If the partial check fails, the counter value is reset to the initial value or reduced by a certain value (by a fixed value or as a function (e.g. a fraction or exponential function) of the counter value). When a new download is received, the counter value is reset to the initial value.

Yet another option is as follows. The counter value starts at a certain initial value (e.g. zero or could be any other positive or even negative value) and is decreased (by a fixed value or as a function (e.g. a fraction or exponential function) of the counter value) on success of the full check. The full check is performed unless the counter value hits or falls below a certain threshold value (could be a negative value, a positive value or zero) thereby allowing a partial check to be performed. If the partial check fails, the counter value is reset to the initial value or increased by a certain value (by a fixed value or as a function (e.g.
a fraction or exponential function) of the counter value). When a new download is received, the counter value is reset to the initial value.

Reference is again made to Fig. 3.

The system 10 is now described in more detail.

The processor 14 is typically a processor to perform, as part of a boot-loading process, a code check including performing a partial check of the program code or a full check of the program code dependent on the value of the counter 12.

The full check typically includes authenticating all of the program code 16 against a digital signature (for example, the entire code signature 26 or the signature 34 according to the authentication method of Fig. 4).

The partial check typically includes randomly or pseudo-randomly selecting one of the sections 22 from the plurality of sections 22 of the program code 16. The partial check includes authenticating the selected section 22 of the program code 16 against a digital signature (for example, one of the digital signatures 24 or the signature 34 according to the authentication method of Fig. 4). Typically only one of the sections 22 is checked as part of the partial check during the boot-loading process. However, the partial check may include checking two or more sections 22 of the program code 16. However, it should be noted that the partial check excludes performing a signature check on all of the sections 22 of the program code 16.

The full check or the partial check may be based on asymmetric cryptography using a public key as described herein above. Alternatively, the full check or the partial check may be based on symmetric cryptography, for example, by authenticating the program code 16 or the selected section 22 against a message authentication code (MAC) based on a secret shared between the processor 14 and the supplier/distributor of the program code.

The processor 14 is typically operative to adjust the counter value in a first direction if the full check is successful and adjust the counter value in a second direction, opposite to the first direction, if the partial check fails.
In accordance with one embodiment of the present invention, the processor 14 is operative to: decrement the counter value if the full check is successful; increment the counter value if the partial check fails; and perform the full check of the program code 16 when the value of the counter 12 is greater than, or equal to, a threshold value.

In accordance with an alternative embodiment of the present invention, the processor 14 is operative to: increment the counter value if the full check is successful; decrement the counter value if the partial check fails; and perform the full check of the program code 16 when the value of the counter 12 is less than, or equal to, a threshold value.

The processor is operative to continue with the boot-loading process if the code check yields a positive validation.

It will be appreciated the system 10 may be implemented in any suitable environment for example, but not limited to, in a car, in a router, in a TV or in a set-top box.

It is appreciated that software components of the present invention may, if desired, be implemented in ROM (read only memory) form. The software components may, generally, be implemented in hardware, if desired, using conventional techniques. It is further appreciated that the software components may be instantiated, for example, as a computer program product; on a tangible medium; or as a signal interpretable by an appropriate computer.

It will be appreciated that various features of the invention which are, for clarity, described in the contexts of separate embodiments may also be provided in combination in a single embodiment. Conversely, various features of the invention which are, for brevity, described in the context of a single embodiment may also be provided separately or in any suitable sub-combination.

It will be appreciated by persons skilled in the art that the present invention is not limited by what has been particularly shown and described hereinabove. Rather the scope of the invention is defined by the appended claims and equivalents thereof.
What is claimed is:

CLAIMS

1. A system comprising:
   a counter having a value;
   a first processor to:
   perform, as part of a boot-loading process, a code check
   including performing a partial check of program code or a full check of the
   program code dependent on the value of the counter, the program code having a
   plurality of sections;
   adjust the counter value in a first direction if the full check is
   successful; and
   adjust the counter value in a second direction, opposite to the
   first direction, if the partial check fails; and
   continue with the boot-loading process if the code check is
   successful, wherein: the full check includes authenticating all of the program code
   against a first digital signature; the partial check includes authenticating one of the
   sections of the program code against a second digital signature; and the partial
   check excludes authenticating all the sections of the program code, the first
   signature being the same as, or different from, the second signature.

2. The system according to claim 1, wherein the partial check includes
   randomly or pseudo-randomly selecting the one section from the plurality of
   sections of the program code.

3. The system according to claim 1 or claim 2, further comprising a
   second processor operative to execute the program code, wherein: the first
   processor is operative only to run the boot-loading process and the counter is
   protected so that the counter can be adjusted or reset by the first processor but not
   by the second processor.
4. The system according to any of claims 1-3, wherein the counter is protected from being reset or adjusted manually.

5. The system according to claim 1 or claim 2, wherein the first processor operative is also operative to execute the program code, wherein the counter is protected so that the counter can only be adjusted or reset by the first processor during the boot-loading process.

6. The system according to any of claims 1-5, wherein the first digital signature is different from the second digital signature which is different for each one of the sections of the program code so that there are a plurality of second digital signatures.

7. The system according to claim 5, wherein the second digital signatures are compressed together into a single physical structure.

8. The system according to any of claims 1-5, wherein the first digital signature is different from the second digital signature.

9. The system according to any of claims 1-5, wherein the first digital signature is the same as the second digital signature.

10. The system according to any of claims 1-9, wherein the first processor is operative to:
    
    decrement the counter value if the full check is successful;
    
    increment the counter value if the partial check fails; and
    
    perform the full check of the program code when the value of the counter is greater than, or equal to, a threshold value.

11. The system according to any of claims 1-9, wherein the first processor is operative to:
increment the counter value if the full check is successful;
decrement the counter value if the partial check fails; and
perform the full check of the program code when the value of the
counter is less than, or equal to, a threshold value.

5    12. A method comprising:
performing, as part of a boot-loading process, a code check
including performing a partial check of program code or a full check of the
program code dependent on the value of the counter, the program code having a
plurality of sections;

10   adjusting the counter value in a first direction if the full check is
successful;

   adjusting the counter value in a second direction, opposite to the
first direction, if the partial check fails;
   continuing with the boot-loading process if the code check is

15   successful, wherein: the full check includes authenticating all of the program code
against a first digital signature; the partial check includes authenticating one of the
sections of the program code against a second digital signature; and the partial
check excludes authenticating all the sections of the program code, the first
signature being the same as, or different from, the second signature.
FIG. 5

START

42

COUNTER VALUE > THRESHOLD Z

YES

NO

44

48

PERFORM FULL CHECK

46

PICK A SECTION TO CHECK

54

INCREMENT COUNTER BY X

40

50

SECTION AUTHENTIC?

YES

NO

52

CONTINUE BOOT

56

ENTIRE CODE AUTHENTIC?

YES

NO

58

DECREMENT COUNTER BY Y
**INTERNATIONAL SEARCH REPORT**

International application No
PCT/IB2012/055148

A. CLASSIFICATION OF SUBJECT MATTER

INV. G06F21/57 G06F11/22

ADD.

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 G07F

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 practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>US 6 725 368 B1 (LIEBENOW FRANK W [US]) 20 April 2004 (2004-04-20) column 1, line 46 - column 2, line 12 column 4, line 24 - line 62 figure 1</td>
<td>1-12</td>
</tr>
</tbody>
</table>

Further documents are listed in the continuation of Box C. 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 application or patent 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
- **Z**: document member of the same patent family

Date of the actual completion of the international search
13 March 2013

Date of mailing of the international search report
25/03/2013

Name and mailing address of the ISA/ Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2
NL-2280 HV Rijswijk
Tel. (+31-70) 340-2040, Fax. (+31-70) 340-3016

Cartrysse, Kathy
<table>
<thead>
<tr>
<th>Patent document cited in search report</th>
<th>Publication date</th>
<th>Patent family member(s)</th>
<th>Publication date</th>
</tr>
</thead>
<tbody>
<tr>
<td>EP 1550988 A2</td>
<td>06-07-2005</td>
<td>AI 2004237915</td>
<td>14-07-2005</td>
</tr>
<tr>
<td>CA 2491157 A</td>
<td>30-06-2005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EP 1550988 A2</td>
<td>06-07-2005</td>
<td>AI 200410372 A</td>
<td>27-09-2006</td>
</tr>
<tr>
<td>US 2005143171 A</td>
<td>30-06-2005</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZA 200410372 A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>US 6725368 BI</td>
<td>20-04-2004</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>US 2008229055 AI</td>
<td>18-09-2008</td>
<td>NONE</td>
<td></td>
</tr>
</tbody>
</table>