(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

COURBEVOIE

(11) **N° de publication :** **3 089 655**
(à n'utiliser que pour les commandes de reproduction)

(21) **N° d'enregistrement national :** **19 07796**

(51) Int Cl[8] : ***G 06 F 21/52*** *(2019.01)*

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 11.07.19.

(30) **Priorité :** 06.12.18 FR 1872456.

(43) **Date de mise à la disposition du public de la demande :** 12.06.20 Bulletin 20/24.

(56) **Liste des documents cités dans le rapport de recherche préliminaire :** *Se reporter à la fin du présent fascicule*

(60) **Références à d'autres documents nationaux apparentés :**

( ) **Demande(s) d'extension :**

(71) **Demandeur(s) :** *IDEMIA IDENTITY & SECURITY FRANCE Société par actions simplifiée (SAS)* — FR.

(72) **Inventeur(s) :** BLANCO Fabien et BERNARD Jean-Yves.

(73) **Titulaire(s) :** IDEMIA IDENTITY & SECURITY FRANCE Société par actions simplifiée (SAS).

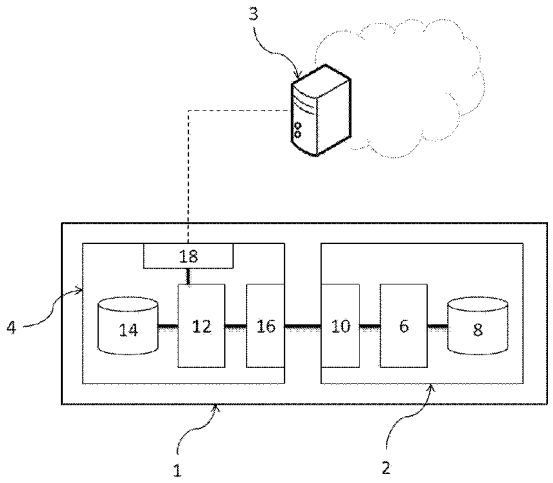
(74) **Mandataire(s) :** REGIMBEAU.

(54) Dispositif tel qu'un objet connecté pourvu de moyens pour contrôler l'exécution d'un programme exécuté par le dispositif.

(57) Dispositif tel qu'un objet connecté pourvu de moyens pour contrôler l'exécution d'un programme exécuté par le dispositif

La présente invention concerne un dispositif (1) tel qu'un objet connecté comprenant un premier circuit électronique (2) comprenant :- une première unité de traitement (6) pour exécuter un programme,- une première mémoire (8) pour mémoriser des données au cours de l'exécution du programme,- un port de debug (10) dédié au contrôle de l'exécution du programme depuis l'extérieur du premier circuit,un deuxième circuit électronique (4) connecté au port de debug (10), comprenant :- une deuxième mémoire (14) mémorisant des données de référence relatives au programme,- une deuxième unité de traitement (12) pour mettre en œuvre les étapes suivantes de manière automatique et autonome via le port de debug (10) : contrôler l'intégrité des données mémorisées par la première mémoire (8) et/ou la conformité de l'exécution du programme par la première unité de traitement (6) à une exécution de référence, à l'aide des données de référence.

Figure pour l'abrégé : figure 1



FR 3 089 655 - A1

Description

Titre de l'invention : Dispositif tel qu'un objet connecté pourvu de moyens pour contrôler l'exécution d'un programme exécuté par le dispositif

Domaine technique

[0001] La présente invention concerne un dispositif comprenant une circuit électronique adapté pour exécuter un programme susceptible d'être corrompu par des attaques.

[0002] La présente invention trouve notamment application dans le domaine des objets connectés.

Technique antérieure

[0003] Un circuit électronique non sécurisé exécutant un programme est typiquement sensible à plusieurs vecteurs d'attaques, que ce soit via des attaques hardware ou software.

[0004] Il est difficile de mettre en œuvre des moyens génériques de sécurisation d'un circuit électronique exécutant un programme microcontrôleur ou d'un microprocesseur sans que ce circuit électronique ne nécessite soit la modification du programme soit la modification du matériel. Dans les deux cas, les modifications à apporter représentent un coût substantiel à cause du savoir-faire ou du matériel nécessaire.

Exposé de l'invention

[0005] Un but de l'invention décrit est de contrôler si un circuit électronique non sécurisé subit une attaque altérant son fonctionnement, sans que l'implémentation de ce contrôle n'engendre un surcout de fabrication du circuit électronique à contrôler.

[0006] Un autre but de l'invention est de faire en sorte que ce contrôle puisse être mis en œuvre facilement pour une grande variété de circuits électroniques existants.

[0007] Il est donc proposé, selon un premier aspect de l'invention, un dispositif tel qu'un objet connecté comprenant un premier circuit électronique comprenant :

- une première unité de traitement configurée pour exécuter un programme,
- une première mémoire configurée pour mémoriser des données du programme ou manipulées par le programme au cours de son exécution,
- un port de debug dédié au contrôle de l'exécution du programme depuis l'extérieur du premier circuit,

le dispositif comprenant en outre un deuxième circuit électronique connecté au port de debug, le deuxième circuit électronique comprenant :

- une deuxième mémoire mémorisant des données de référence relatives au programme,
- une deuxième unité de traitement configurée pour mettre en œuvre les étapes

suivantes de manière automatique et autonome via le port de debug : contrôler l'intégrité des données mémorisées par la première mémoire et/ou la conformité de l'exécution du programme par la première unité de traitement à une exécution de référence, à l'aide des données de référence.

[0008] Un port de debug est d'habitude utilisé pour détecter, reproduire et analyser des bugs présents dans un programme exécuté par un circuit électronique au cours de son développement. L'utilisation conventionnelle d'un port de debug est la suivante : un développeur surveille et contrôle manuellement l'exécution d'un programme buggé d'un circuit depuis une station de travail au moyen d'un programme de débogage, ou debugger et d'un tel port de debug. Le débogage ainsi réalisé par le développeur est une tache empirique qui dépend essentiellement de ses connaissances générales en informatique, sa connaissance du code source du programme contrôlé, de son expérience passée en matière de débogage et de son intuition. Dans le dispositif selon le premier aspect de l'invention, le port de debug est astucieusement détourné de son utilisation habituelle pour mettre en œuvre un contrôle d'intégrité totalement automatique et autonome (par « automatique et autonome », on entend : sans intervention humaine). En se fondant sur les données de référence mémorisées par le deuxième circuit électronique, ce contrôle d'intégrité permet de détecter si l'exécution est conforme à des règles préétablies ou non, auquel cas il est possible que le premier circuit électronique ait été attaqué.

[0009] De plus, comme la plupart des circuits électroniques existants équipés d'une unité de traitement (notamment microcontrôleurs ou processeurs) disposent d'un tel port de debug, l'invention est applicable à une grande variété de circuits.

[0010] Par ailleurs, comme le contrôle d'intégrité est commandé depuis l'extérieur du premier circuit électronique, en l'occurrence par le deuxième circuit électronique, il n'est pas nécessaire de modifier le premier circuit électronique en profondeur.

[0011] Le dispositif selon le premier aspect de l'invention peut également comprendre les caractéristiques suivantes.

[0012] De préférence, les étapes mises en œuvre de manière automatique et autonome par la deuxième unité de traitement comprennent par ailleurs une commande de suspension du programme, l'étape de contrôle d'intégrité et/ou de conformité étant mise en œuvre pendant que le programme est suspendu.

[0013] De préférence, la commande de suspension comprend le placement d'un point d'arrêt en un endroit prédéterminé du programme, de sorte à suspendre le programme à l'endroit prédéterminé, ou le placement d'un point d'observation sur une variable du programme, de sorte à suspendre le programme lorsque la variable est modifiée.

[0014] De préférence, les étapes mises en œuvre de manière automatique et autonome par la deuxième unité de traitement comprennent une étape consistant à vérifier si une

condition indépendante de la manière dont le programme s'exécute est remplie, par exemple vérifier si un délai de durée prédéterminée s'est écoulé depuis un précédent démarrage du programme, une précédente reprise du programme ou une précédente mise sous tension du dispositif, l'étape de commande de suspension étant mise en œuvre lorsque la condition est remplie.

[0015] De préférence, les étapes mises en œuvre de manière automatique et autonome par la deuxième unité de traitement comprennent la commande d'une reprise du programme par la première unité de traitement lorsqu'aucune compromission du programme ou des données manipulées par le programme n'est révélée lors de l'étape de contrôle, ladite reprise n'étant pas mise en œuvre lorsqu'une compromission du programme ou des données manipulées par le programme est révélée lors de l'étape de contrôle.

[0016] De préférence, les étapes mises en œuvre de manière automatique et autonome par la deuxième unité de traitement comprennent une commande d'un arrêt définitif du programme et/ou d'une reconfiguration du premier circuit électronique dans un état empêchant une exécution ultérieure du programme par la première unité de traitement, lorsqu'une compromission du programme ou des données manipulées par le programme est révélée lors de l'étape de contrôle.

[0017] De préférence, les étapes mises en œuvre de manière automatique et autonome par la deuxième unité de traitement comprennent la génération d'un rapport indiquant si une compromission du programme ou des données manipulées par le programme a été révélée lors de l'étape de contrôle.

[0018] De préférence, le dispositif comprend une interface de communication avec l'extérieur du dispositif configurée pour transmettre le rapport généré à un serveur à distance du dispositif.

[0019] De préférence, les étapes mises en œuvre de manière automatique et autonome par la deuxième unité de traitement comprennent la commande la transmission du rapport au serveur via l'interface de communication pendant que le programme est suspendu.

[0020] De préférence, l'interface de communication est configurée pour transmettre le rapport au serveur sans que le rapport ne passe par le premier circuit électronique. En variante, l'interface de communication fait partie du premier circuit électronique.

[0021] De préférence, les étapes mises en œuvre de manière automatique et autonome par la deuxième unité de traitement comprennent la commande d'une écriture, via le port de debug, du rapport dans un zone d'échange allouée dans la première mémoire, de sorte que le rapport soit par la suite relayé au serveur via l'interface de communication.

[0022] Le premier circuit électronique peut comprendre un deuxième port distinct du port de debug, et les étapes mises en œuvre de manière automatique et autonome par la deuxième unité de traitement peuvent comprendre la commande d'une transmission du rapport à l'interface de communication via le deuxième port, de sorte que le message

d'erreur soit par la suite relayé au serveur via l'interface de communication.

[0023] De préférence, les données de référence indiquent une séquence d'évènements survenant dans un ordre prédéterminé au cours de l'exécution de référence du programme, et dans lequel le contrôle de conformité de l'exécution du programme à une exécution de référence comprend une comparaison entre l'ordre prédéterminé et un ordre dans lequel surviennent les évènements au cours de l'exécution du programme par la première unité de traitement.

[0024] De préférence, les étapes mises en œuvre de manière automatique et autonome par la deuxième unité de traitement comprennent une commande de mise à jour du programme lorsqu'une compromission du programme ou des données manipulées par le programme est révélée lors de l'étape de contrôle.

[0025] De préférence, la première mémoire mémorise une première version du programme et une deuxième version du programme avant la commande de mise à jour, la première unité de traitement est configurée pour exécuter sélectivement la première version du programme avant la mise à jour, et dans lequel la commande de mise à jour comprend l'envoi au premier circuit électronique, via le port de debug, d'au moins une commande demandant à la première unité de traitement d'exécuter sélectivement la deuxième version du programme mémorisé dans la première mémoire en lieu et place de la première version du programme.

[0026] Alternativement, la première unité de traitement est configurée pour exécuter une première version du programme avant la mise à jour, et la commande de mise à jour comprend l'envoi au premier circuit électronique, via le port de debug, d'une deuxième version du programme obtenue par le deuxième circuit électronique de sorte que la première unité de traitement exécute la deuxième version du programme en lieu et place de la première version du programme.

[0027] De préférence, la commande de mise à jour comprend un téléchargement de la deuxième version du programme auprès d'un serveur à distance du dispositif, ou une lecture de la deuxième version du programme dans une mémoire du dispositif, telle que la deuxième mémoire.

[0028] De préférence, la deuxième version est lue dans la mémoire du dispositif seulement lorsqu'une une tentative d'établissement d'un canal de communication sécurisé entre la deuxième unité de traitement et le serveur échoue.

[0029] De préférence, la commande de mise à jour comprend une vérification d'une signature numérique de la deuxième version du programme, ou un décryptage de la deuxième version du programme.

[0030] Il est également proposé, selon un deuxième aspect de l'invention, un procédé mis en œuvre dans un dispositif tel qu'un objet connecté, le dispositif comprenant :

- un premier circuit électronique comprenant une première unité de traitement

configurée pour exécuter un programme, une première mémoire configurée pour mémoriser des données du programme ou manipulées par le programme au cours de son exécution, et un port de debug dédié au contrôle de l'exécution du programme depuis l'extérieur du premier circuit,

- un deuxième circuit électronique connecté au port de debug et comprenant une deuxième mémoire mémorisant des données de référence relatives au programme,

le procédé comprenant les étapes suivantes mises en œuvre par une deuxième unité de traitement du deuxième circuit électronique, de manière automatique et autonome via le port de debug : contrôler l'intégrité des données mémorisées par la première mémoire et/ou la conformité de l'exécution du programme par la première unité de traitement à une exécution de référence, à l'aide des données de référence.

[0031] Il est également proposé un produit programme d'ordinateur comprenant des instructions de code de programme pour l'exécution des étapes du procédé selon le deuxième aspect de l'invention, lorsque ce procédé est exécuté par une unité de traitement.

Brève description des dessins

[0032] D'autres caractéristiques, buts et avantages de l'invention ressortiront de la description qui suit, qui est purement illustrative et non limitative, et qui doit être lue en regard des dessins annexés.

[0033] [fig.1]

[0034] [fig.2]

[0035] [fig.3]

[0036] Les figures 1 à 3 illustrent de façon schématique des dispositifs selon trois modes de réalisation différents de l'invention.

[0037] [fig.4]

[0038] La figure 4 est un organigramme d'étape d'un procédé susceptible d'être mis en œuvre par l'un des dispositifs représentés sur les figures 1 à 3.

[0039] Sur l'ensemble des figures, les éléments similaires portent des références identiques.

Description des modes de réalisation

[0040] En référence à la **figure 1,** un dispositif 1 comprend un premier circuit électronique 2 et un deuxième circuit électronique 4, distinct du premier circuit électronique 2.

[0041] Le premier circuit électronique 2 comprend une première unité de traitement 6, une première mémoire 8 et un port de debug 10.

[0042] La première unité de traitement 6 est configurée pour exécuter un programme d'ordinateur, que l'on appellera dans la suite « programme cible ». La première unité de traitement 6 comprend typiquement un ou plusieurs processeurs.

[0043] La première mémoire 8 est configurée pour mémoriser des données du programme

cible ou manipulées par le programme cible au cours de son exécution. La première mémoire 8 comprend typiquement au moins une unité de mémoire non-volatile (flash, EEPROM, NVM, etc.) configurée pour stocker le programme cible et des données de manière persistante, et au moins une unité de mémoire volatile (RAM, registres, etc.), configurée pour mémoriser des instructions de programme à exécuter ou des données manipulées temporairement par le programme cours de son exécution.

[0044] La mémoire 8 est notamment accessible via le port de debug 10.

[0045] Le port de debug 10 est dédié au contrôle de l'exécution du programme cible depuis l'extérieur du premier circuit électronique 2. Le port de debug 10 constitue une interface de communication entre le programme cible et l'extérieur du premier circuit électronique 2.

[0046] Le port de debug 10 par exemple un port TAP ou JTAG défini par l'un des spécifications JTAG connues (par exemple IEEE 1149.1 ou IEEE 1149.9), ou bien un port de type SWD.

[0047] Le premier circuit électronique 2 peut être : un FPGA, un ASIC, un microcontrôleur, etc.

[0048] Le deuxième circuit électronique 4 comprend un port 16, une deuxième unité de traitement 12 et une deuxième mémoire 14.

[0049] Le port 16 est connecté directement au port de debug 10 du premier circuit électronique 2.

[0050] La deuxième unité de traitement 12 est configurée pour exécuter un programme dit « programme de contrôle ». Le programme de contrôle a pour fonction de contrôler l'intégrité de données mémorisées dans la première mémoire 8 au cours de l'exécution du programme cible (en particulier, des portions du programme cible lui-même, des données manipulées par le programme cible au cours de son exécution), ou bien des données de contexte d'exécution, telles que le « program counter » ou les registres de l'unité de traitement, ainsi que vérifier le déroulement du programme de la deuxième unité de traitement 6 (via une chaine valide de points d'arrêt atteints).

[0051] Le programme de contrôle comprend notamment (mais pas seulement) des instructions adaptées pour mettre en œuvre des fonctionnalités que l'on trouve dans un debugger classique, comme par exemple le placement de points d'arrêts dans le programme cible.

[0052] La deuxième mémoire 14 est configurée pour mémoriser des données de référence relatives au programme cible (des exemples de telles données de référence sont donnés dans la suite).

[0053] La deuxième mémoire 14 comprend typiquement au moins une unité de mémoire non-volatile (flash, EEPROM, NVM, etc.) configurée pour stocker le programme de contrôle et/ou les données de référence de manière persistante, et au moins une unité

de mémoire volatile (RAM, registres, etc.), configurée pour mémoriser des instructions du programme de contrôle à exécuter ou des données manipulées temporairement par le programme de contrôle au cours de son exécution.

[0054] Le premier circuit électronique 2 peut être : un FPGA, un ASIC, etc.

[0055] De préférence, le deuxième circuit électronique 4 est davantage sécurisé contre des attaques extérieures que ne l'est le premier circuit électronique 2. Dans certains domaines, le terme « élément sécurisé » (abrégé en SE) est utilisé pour désigner un circuit disposant d'un niveau de sécurité élevé, et le terme de « microcontrôleur » (abrégé en MCU) est utilisé pour désigner un circuit disposant d'un niveau de sécurité comparativement plus faible. Le premier circuit électronique 2 peut donc être considéré comme un SE et le deuxième circuit électronique 4 peut être considéré comme un MCU.

[0056] Le dispositif 1 est par exemple un objet connecté, ou bien lui-même un circuit électronique tel qu'un système sur puce (SoC).

[0057] Le dispositif 1 comprend par ailleurs une interface de communication 18 avec un serveur distant 3, typiquement une interface de communication radio sans fil (Wi-Fi, NFC, Bluetooth, 3/4/5G, etc.) ou bien filaire (de manière directe, par exemple via Ethernet, ou de manière indirecte au moyen d'un bus I^2C contrôlant un émetteur radio agissant comme un relai entre l'interface 18 et le serveur 3).

[0058] Lorsque le dispositif 1 est un objet connecté, le serveur 3 distant est apte à communiquer avec un réseau d'objets connectés auquel le dispositif 1 est susceptible d'adhérer. Le serveur 3 a notamment pour fonction de collecter des informations communiques par différents objets connectés tel que l'objet 1.

[0059] Comme on le verra dans la suite, le deuxième circuit électronique 4 et le serveur 3 peuvent échanger des informations, en passant par des chemins différents.

[0060] Dans le premier mode de réalisation illustré en figure 1, l'interface de communication 18 fait partie du deuxième circuit électronique 4, et le premier circuit électronique ne comprend pas nécessairement d'interface de communication directe avec l'extérieur du dispositif 1 (bien que cela soit possible). Ceci a l'avantage d'éviter au deuxième circuit électronique d'être « à la merci » du premier circuit électronique 2 pour pouvoir communiquer vers l'extérieur : dans le cas où le premier circuit électronique 2 ne fait pas les actions nécessaires pour relayer des messages venant du deuxième circuit électronique 4, ce dernier ne peut pas communiquer avec l'extérieur.

[0061] On a illustré en **figure 2** un deuxième mode de réalisation du dispositif 1, dans lequel l'interface de communication 18 fait partie du premier circuit électronique 2. Ceci a l'avantage d'éviter un surcout, des contraintes ou des limitations inutiles dans un dispositif 1 dont le premier circuit électronique 2 dispose de matériel de communication avec l'extérieur.

[0062] Toutefois, il demeure que le deuxième circuit électronique 4 a besoin de communiquer avec le serveur 3, situé à l'extérieur du dispositif 1. A cette fin, le premier circuit électronique 2 comprend un port 20 distinct du port de debug 10, et le deuxième circuit électronique 4 comprend un port 22 relié au port 20 par une liaison indépendante de la liaison qui relie le port 16 au port de debug 10. Le port 20 est relié à la première unité de traitement 6. La liaison entre les ports 20 et 22 est par exemple un bus I2C.

[0063] Dans ce deuxième mode de réalisation, le programme exécuté sur la première unité de traitement 6 s'assure de recevoir (ou transmettre) des données venant du premier circuit électronique 2 vers le (ou venant du) serveur 3 grâce à sa capacité de communication vers l'extérieur assurée par l'interface de communication 18. Dans le cas présent, le succès de la communication entre le deuxième circuit électronique 4 et le serveur 3 est donc assuré par la coopération du programme exécuté sur le premier circuit électronique 2.

[0064] Cette configuration peut être particulièrement adaptée, mais non limitée, aux cas où le deuxième circuit électronique 4 fournit au premier circuit électronique 2 des fonctions annexes, auquel cas ce dernier peut alors faire des requêtes au premier, qui se comporte alors comme un composant esclave classique sur un bus de communication embarqué (comme présenté avant, par exemple I^2C).

[0065] Est illustré en **figure 3** un troisième mode de réalisation, dans lequel l'interface de communication 18 fait partie du premier circuit électronique 2, tout comme dans le deuxième mode de réalisation. Toutefois, à la différence du deuxième mode de réalisation illustré en figure 2, le deuxième circuit électronique communique avec le serveur 3 par le biais du port de debug 10, et non par le biais d'un port supplémentaire. Ceci permet de simplifier l'architecture du dispositif 1 par rapport au deuxième mode de réalisation. Le port de debug 10 sert en effet non seulement à gérer des communications entre les circuits 2 et 4 (au sein du dispositif 1) mais également à gérer des communications entre le deuxième circuit électronique 4 et le serveur 3.

[0066] Le dispositif 1 est configuré comme suit dans une phase préliminaire de configuration.

[0067] Le code du programme cible est compilé, de sorte à produire un exécutable destiné à être exécuté sur le premier circuit électronique 2.

[0068] Après compilation, des données d'intégrité sont calculées sur des segments prédéterminés du code du programme cible.

[0069] Une liste d'adresses du programme cible sont déterminées dans le but d'y placer des points d'arrêts.

[0070] Le code du programme cible est chargé dans la première mémoire 8, de sorte à pouvoir ensuite être exécuté par la première unité de traitement 6.

[0071] Par ailleurs, la liste d'adresses déterminée et les données d'intégrité sont mémorisées dans la deuxième mémoire 14 en tant que données de référence. Chaque adresse est associée dans la mémoire avec au moins une donnée d'intégrité.

[0072] Le code du programme de contrôle est chargé dans la deuxième mémoire 14, de sorte à pouvoir être exécuté par la première unité de traitement 12.

[0073] La phase préliminaire de configuration du dispositif 1 est alors terminée.

[0074] En référence à la **figure 3**, les étapes suivantes sont mises en œuvre au cours d'une phase ultérieure d'utilisation du dispositif 1, une fois celui-ci mis sous tension.

[0075] La deuxième unité de traitement 12 commande une suspension du programme cible exécuté par la première unité de traitement 6 (étape 100).

[0076] Différentes politiques de suspension du programme cible peuvent être mises en œuvre, prises seules ou en combinaison.

[0077] La deuxième unité de traitement 12 peut par exemple placer des points d'arrêts définis par les données de référence (au sens où les données de référence pointent vers des endroits précis du code du programme en lesquels l'exécution du programme sera suspendue). Ce placement de points d'arrêts est typiquement mis en œuvre lors d'une mise sous tension du premier circuit électronique 2 (ou plus généralement du dispositif 1), via une commande appropriée émise par le premier circuit électronique 2 transitant par le port de debug 10, avant que le programme cible ne démarre ou bien très peu de temps après son démarrage par la première unité de traitement 6.

[0078] L'exécution du programme est suspendue à chaque fois que le pointeur d'exécution atteint l'une des adresses où a été placé un point d'arrêt.

[0079] Un avantage d'une suspension au moyen de points d'arrêts est de pouvoir cibler de manière déterministe des parties critiques du programme cible les plus susceptibles d'être corrompues par des attaques, ou cibler des zones ayant peu d'impact sur les opérations ne devant pas être interrompues (pour une meilleure stabilité du système).

[0080] La deuxième unité de traitement 12 peut également commander la suspension en placement des points d'observations au niveau de variables du programme cible. Dans ce cas, une suspension survient lorsque la variable est modifiée.

[0081] Additionnellement, le circuit 4 peut vérifier de manière répétée (selon une période fixe ou aléatoire par exemple) que les points d'arrêts ou d'observations placés sont toujours présents.

[0082] La deuxième unité de traitement 12 peut également suspendre le programme cible lorsqu'une condition indépendante de la manière dont le programme s'exécute est remplie.

[0083] La condition est par exemple une condition temporelle : la deuxième unité de traitement déclenche la suspension une fois qu'un délai de durée prédéterminée s'est écoulé depuis le démarrage ou une précédente reprise du programme cible. A cet effet,

la deuxième unité de traitement 12 dispose d'un compteur de temps qui est réinitialisé au démarrage ou à chaque reprise du programme cible.

[0084] Certaines attaques peuvent avoir pour conséquence que le pointeur d'exécution ne passe plus jamais par les adresses où ont été placés des points d'arrêts, ou ne modifie plus les variables où ont été placés des points d'observation, ce qui est préjudiciable. L'avantage d'une suspension déclenchée sur respect d'une condition indépendante de la manière dont le programme s'exécute est par conséquent de pouvoir garantir qu'il y aura une suspension, quels que soient les endroits du programme cible où passe le pointeur d'exécution de la première unité de traitement 6 et quelles que soient les variables que le programme cible modifie. Cet avantage est en particulier obtenu lors de l'utilisation d'une condition temporelle de déclenchement de la suspension.

[0085] Pendant que le programme cible est suspendu, la deuxième unité de traitement 12 met en œuvre un contrôle d'intégrité du programme ou de données manipulées par ce dernier et/ou un contrôle de la conformité de l'exécution du programme par la première unité de traitement 6 à une exécution de référence. Pour opérer de tels contrôles la deuxième unité de traitement 12 accède en lecture à la deuxième mémoire 14 en se fondant sur les données de référence qui sont mémorisées dans la deuxième mémoire 14 (étape 102).

[0086] L'étape de contrôle 102 couvre tout traitement mettant en œuvre des vérifications de cohérence de l'exécution et d'intégrité du programme et/ou de ses données de manière à détecter des altérations de ce même programme ou des anomalies d'exécution pouvant être la conséquence d'évènements aléatoires, mais aussi pouvant avoir été générées par un attaquant (dépassement de tampon par exemple).

[0087] Un premier type de contrôle susceptible d'être mis en œuvre au cours de l'étape 102 porte sur l'intégrité du programme ou de données manipulées par ce dernier.

[0088] Typiquement, lorsque le programme cible est suspendu, la deuxième unité de traitement 12 vérifie si une donnée à contrôler présente dans la première mémoire 8 et une donnée d'intégrité faisant partie des données de référence mémorisées dans la deuxième mémoire 14 sont cohérentes. La deuxième unité de traitement 12 peut par exemple comparer la donnée d'intégrité et la donnée à contrôler, auquel cas les deux données sont considérées cohérentes lorsqu'elles sont de valeurs identiques.

[0089] Toutefois, la vérification de cohérence entre une donnée d'intégrité et une donnée à contrôler peut être plus complexe qu'une simple comparaison. L'une des données en entrée de la vérification de cohérence peut par exemple subir une transformation, et le résultat de cette transformation peut être comparé avec l'autre donnée. Les deux données en entrée de la vérification de cohérence peuvent également subir des transformations respectives, les résultats de ces deux transformations être ensuite comparés.

[0090] La vérification de cohérence peut être répétée pour plusieurs données de référence

(lorsque plusieurs données d'intégrité ont été associées à la même adresse, par exemple).

[0091] Par exemple, une donnée de premier circuit électronique 2 à contrôler au cours de de l'étape contrôle d'intégrité est une variable globale.

[0092] Le contrôle d'intégrité 102 produit deux résultats possibles : un résultat positif, en cas de cohérence entre les données d'intégrité et à contrôler, et un résultat négatif dans le cas contraire (incohérence entre les données). Ce résultat négatif révèle une compromission du programme cible ou de données manipulées par le programme cible.

[0093] Un deuxième type de contrôle susceptible d'être mis en œuvre au cours de l'étape 102 porte la question de savoir si le programme s'exécute d'une manière conforme à un schéma préétabli.

[0094] Les données de références renseignent alors sur une exécution de référence du programme. L'exécution de référence du programme est une exécution du programme dans le cas où celui-ci n'a fait l'objet d'aucune modification ou altération par une attaque survenant suite à son installation dans le dispositif 1.

[0095] Au cours de l'étape 102, la deuxième unité de traitement 12 utilise les données de référence pour déterminer si l'exécution du programme par la première unité de traitement 6 est conforme à cette exécution de référence.

[0096] Ce contrôle de conformité produit deux résultats possibles : un résultat positif, en cas de conformité entre l'exécution du programme et l'exécution de référence, et un résultat négatif dans le cas contraire (incohérence entre les exécutions). Ce résultat négatif révèle également une compromission du programme cible ou de données manipulées par le programme cible.

[0097] Une façon de mettre en œuvre un tel contrôle de conformité est de référencer dans les données de référence une séquence d'évènements censés survenir dans un certain ordre, ou plus généralement un graphe orienté d'évènements. Par exemple, ces évènements sont des passages d'un pointeur d'exécution par différents endroits du programme dans un ordre prédéfini. Prenons le cas par exemple de trois instructions de code A, B, C du programme. A est une instruction de code du programme exécutée à son démarrage, B est une instruction de code du programme se trouvant au début d'une fonction censée être exécutée après A, et C est une instruction de code du programme se trouvant à la fin de cette même fonction. Dans le cas où le programme ne subit aucune modification ou altération, le pointeur d'exécution de la première unité de traitement 6 est censé passer par les instructions A, puis B, puis C. Par contre, si ces trois instructions sont exécutées dans un ordre différent, par exemple B, puis A, puis C, ceci est révélateur d'une altération du programme survenue après son installation initiale.

[0098] Aussi, la deuxième unité de traitement 12 détecte l'ordre dans lequel les évènements

référencés dans les données de référence surviennent au cours de l'exécution du programme par la première unité de traitement 6.

[0099] La deuxième unité de traitement compare ensuite l'ordre prédéfini dans les données de référence (A puis B puis C dans l'exemple présenté ci-dessus), et l'ordre qu'elle a détecté.

[0100] Si les deux ordres comparés correspondent, le résultat du contrôle de conformité est positif. Sinon, ce résultat est négatif.

[0101] Bien entendu, les deux types de contrôle précités (intégrité de données / conformité d'exécution du programme) peuvent être combinés au cours de l'étape 102. Dans ce cas, on considère que le résultat général du contrôle mis en œuvre au cours de l'étape 102 est positif seulement si tous les types de contrôles sous-jacents mis en œuvre au cours de cette étape 102 aboutissent à des résultats positifs.

[0102] Le résultat du contrôle effectué au cours de l'étape 102 est vérifié (étape 104).

[0103] Lorsque le résultat du contrôle est positif, la deuxième unité de traitement 12 commande une reprise du programme cible (étape 106), en émettant une commande appropriée via le port de debug 10.

[0104] Lorsque le résultat du contrôle est négatif, la deuxième unité de traitement 12 peut commander un arrêt complet du programme cible (étape 108), ce qui implique, à la différence d'une simple suspension, une libération de données allouées dans la première mémoire 8 au cours de l'exécution du programme cible. Cet arrêt est commandé via une commande d'arrêt transmise via le port de debug 10.

[0105] Lorsque le résultat du contrôle est négatif, la deuxième unité de traitement 12 peut par ailleurs commander une reconfiguration du premier circuit électronique 2 dans un état empêchant une exécution ultérieure du programme par la première unité de traitement 6. Ceci est typiquement mis en œuvre via une commande de reconfiguration transitant par le port de debug 10.

[0106] Par exemple, cette commande de reconfiguration peut altérer le contenu de la première mémoire 8 (restauration des données programme par exemple, on peut aussi envisager d'agir sur la RAM), altérer l'exécution du programme (contexte, état), ou en manipulant des périphériques internes du premier circuit électronique 2.

[0107] Par ailleurs, la deuxième unité de traitement 12 génère un rapport indiquant le résultat du contrôle 102 effectué. Ce résultat peut par exemple se présenter sous la forme d'une information binaire (OK ou erreur), ou bien présenter une forme plus détaillée (renseignant sur le type de contrôle effectué, etc.).

[0108] Le rapport généré est transmis au serveur 3 distant (étape 112), via un canal sécurisé établi entre l'interface de communication 18 du dispositif 1 et le serveur 3.

[0109] Le canal sécurisé est typiquement établi avec à la suite d'une authentification entre le dispositif 1 et le serveur 3 faisant intervenir au moins une clé secrète. Par exemple,

cette authentification est une authentification mutuelle entre le dispositif 1 et le serveur 3, et/ou utilise un mécanisme cryptographique à base de clé publique/clé privée. Les clés utilisées peuvent être stockées dans la mémoire 14 de manière protégée contre des attaques.

[0110] Le rapport généré peut être transmis en réponse à un résultat négatif du contrôle 102 effectué. Toutefois, il convient de garder à l'esprit qu'il est également avantageux de transmettre ce rapport également dans le cas d'un résultat positif du contrôle 102 effectué.

[0111] L'étape 112 de transmission du rapport au serveur 3 peut être mise en œuvre de différentes manières.

[0112] Dans le premier mode de réalisation du dispositif 1 illustré en figure 1, le rapport ne transite pas par le premier circuit électronique 2 car l'interface de communication 18 fait partie du deuxième circuit électronique 4. Le rapport est simplement émis par cette interface de communication 18 sur commande de la deuxième unité de traitement 12.

[0113] Dans les modes de réalisation du dispositif 1 illustrés en figure 2 et 3, le rapport transite par le premier circuit électronique 2 avant de parvenir au serveur 3.

[0114] Dans le deuxième mode de réalisation illustré en figure 2, la deuxième unité de traitement 12 commande une écriture, via les port supplémentaires 22 et 20, du rapport dans une zone d'échange préalablement allouée dans le premier circuit électronique 2, typiquement dans la première mémoire 8. Par exemple, la première unité de traitement 6 peut accéder périodiquement en lecture à la zone d'échange, et le rapport être transmis par le premier circuit électronique 2 en réponse à une telle lecture dans la zone d'échange. En variante, la transmission du rapport n'est déclenchée que sur réception, via l'interface de communication 18, d'une requête émise par le serveur 3 distant.

[0115] Dans le troisième mode de réalisation du dispositif 1 illustré en figure 3, le rapport transite par le premier circuit électronique 2 avant de parvenir au serveur 3, mais d'une manière différente du deuxième mode de réalisation. Une requête émise par le serveur 3 est relayée par le premier circuit électronique 2 au deuxième circuit, via le port de debug 10. Sur réception de cette requête, le deuxième circuit se charge s'y répondre en envoyant au serveur 3 le rapport. Le rapport peut ne pas passer par la première unité de traitement 6 ou la première mémoire 8. Cette zone d'échange peut être interne à l'interface de communication 18 et indépendante de la mémoire 8.

[0116] L'envoi du rapport au serveur peut être mis en œuvre en réponse à la réception par le deuxième circuit 4 d'un message provenant du serveur 3 demandant un rapport.

[0117] Par exemple, le circuit 2 reçoit un tel message du serveur 3 via l'interface de communication 18. Ce message est stocké soit automatiquement grâce l'architecture du circuit 2 dans la mémoire 8 (par exemple DMA), soit via le programme exécuté sur l'unité de

traitement 6 (sans DMA). Une fois ce message stocké, l'unité de traitement 6 fait une série de post-traitement pour en extraire les données utiles, et les formate de manière à pouvoir les envoyer au deuxième circuit 4, et l'unité de traitement 6 exécutera une partie de son programme qui contrôle l'interface de communication 10 pour envoyer ce message formaté, soit automatiquement, soit périodiquement, soit lors de réception d'une requête sur l'interface de communication 10 envoyée via l'interface de communication 16 par l'unité de traitement 12 (requête d'interruption de l'interface de communication 10).

[0118] De préférence, la transmission du rapport est mise en œuvre pendant que le programme cible est dans un état suspendu. Ceci présente l'avantage d'éviter que le premier circuit électronique 2 dans un état corrompu ne tente d'empêcher la transmission de ce message ou d'en altérer le contenu de sorte à faire croire au serveur 1 qu'aucune compromission n'a été détectée par le deuxième circuit électronique 4.

[0119] On distingue deux cas de figures, qui dépendent des capacités matérielles de l'interface de communication 18 lors d'une suspension du programme cible.

[0120] Dans un premier cas, l'interface de communication 18 est fonctionnelle même lorsque le programme cible a été suspendu.

[0121] Dans un deuxième cas, la suspension du programme cible entraîne également une suspension de l'interface de communication 18 ; autrement dit, il est alors nécessaire reprendre l'exécution du programme cible pour que l'interface de communication 18 également fonctionner. Ceci peut être causé par deux raisons :

[0122]
- Une raison matérielle pure, comme par exemple lorsque la suspension via le port de debug 10 entraîne un arrêt d'alimentation en horloge de l'interface de communication 18
- Une raison de performance : passer par le port de debug 10 pour écrire directement dans l'interface de communication 18 n'est pas assez performant pour assurer une communication stable et fiable.

[0123] Dans ce deuxième cas, il peut être prévu ce qui suit dans un mode de réalisation non limitatif dans lequel le port 20 est un bus I^2C.

[0124]
- La deuxième unité de traitement 12 suspend la première unité de traitement 6 via un point d'arrêt placé au moment où la première unité 6 va s'endormir à la fin d'un cycle : à ce moment-là, la première unité de traitement 6 a fini ses opérations critiques et de communication avec l'extérieur via l'interface de communication 18.
- La deuxième unité de traitement 12 formate ses messages à envoyer vers l'extérieur indépendamment de la première unité de traitement 6.
- La deuxième unité de traitement 12 sauvegarde le contenu des registres de configuration d'un périphérique I^2C de la première unité de traitement 6, par

exemple dans la deuxième mémoire 14.

- La deuxième unité de traitement 12 écrit dans les registres de configuration du périphérique I^2C de la première unité de traitement 6 pour préparer la communication vers l'extérieur du dispositif 1.
- La deuxième unité de traitement 12 écrit progressivement dans le registre du buffer d'émission du périphérique I^2C pour transmettre les données formatées vers l'extérieur.
- Le SE restaure le contenu des registres de configuration du périphérique I2C du MCU.
- Le SE rend la main au MCU, qui peut alors réellement partir dans sa période endormie.

[0125] Dans le second cas, il est possible d'ajouter du code dans le premier circuit électronique 2 qui est dédié à cette manipulation faite par la première unité de traitement 6. Par exemple :

[0126]
- Une zone libre est déterminée dans la première mémoire 8.
- Pendant que le programme cible est suspendu, la deuxième unité de traitement 12 écrit dans la zone libre une boucle d'attente de type boucle infinie, cette boucle pouvant éventuellement assister la communication faite par la deuxième unité de traitement 12. Cette boucle peut lire en mémoire (zone échange) une liste de bytes et les envoyer automatiquement dans le buffer I^2 C.La deuxième unité de traitement 12 force un changement de « program counter » et sauvegarde le contexte d'exécution de la première unité de traitement 6 de façon à sauter sur la nouvelle boucle d'attente.
- La deuxième unité de traitement 12 commande les écritures voulues dans le registre de communication de l'interface de communication 18, soit via le port de debug 10 (Boucle infinie vide), soit via la boucle d'attente écrite (boucle infinie avec assistance de communication).
- Une fois la communication finie, la deuxième unité de traitement 12 restaure le contexte et rend la main au programme cible.

[0127] Lorsque le serveur 3 reçoit un rapport contenant un message d'erreur, le serveur 3 peut par exemple commander un arrêt à distance du premier circuit électronique 2 si un tel arrêt n'a pas déjà été mis en œuvre de manière autonome par la deuxième unité de traitement 12 au cours de l'étape 108. Alternativement, ou à titre complémentaire, le serveur 3 révoque le dispositif 1, c'est à dire lui interdit l'accès à certains services (par exemple des services d'un réseau d'objets connectés auquel le serveur 3 appartient).

[0128] Lorsque le résultat du contrôle vérifié à l'étape 104 s'avère négatif, la deuxième unité de traitement 12 peut commander une mise à jour du programme cible jusqu'ici exécuté par la première unité de traitement 6 (étape 114). Cette étape est très

avantageuse, car permet de restaurer le premier circuit 2 dans un état de fonctionnement normal.

[0129] Au cours de la mise à jour, une première version du programme cible, qui était jusqu'ici mémorisée dans la première mémoire 8 et exécutée par la première unité de traitement, est remplacée par une deuxième version du programme.

[0130] La deuxième version de programme peut apporter des nouvelles fonctionnalités et/ou des correctifs par rapport à la première version du programme.

[0131] La commande de mise à jour peut être réalisée de plusieurs manière différentes.

[0132] La deuxième version du programme cible peut par exemple avoir été mémorisée à l'avance dans la première mémoire 8 à un emplacement différent de la première version du programme. Lors de sa mise sous tension, la première unité de traitement 12 se fonde sur une adresse mémoire pour déterminer quelle version du programme cible exécuter. Avant la mise à jour, cette adresse mémoire est l'adresse de début de l'emplacement où la première version du programme est mémorisée dans la première mémoire 14. Pour commander une mise à jour, la deuxième unité de traitement 12 envoie au premier circuit 2, via le port de debug 10, au moins une commande ordonnant à la première unité de traitement 6 de changer de version de programme à exécuter. Sur réception de cette commande, la première unité de traitement modifie l'adresse mémoire précitée à l'adresse de début de l'emplacement où la deuxième version du programme est mémorisée dans la première mémoire 8. Un avantage de ce mode de réalisation est que la commande de mise à jour peut être réalisée très rapidement, sans qu'il soit nécessaire de transmettre une quantité de données au premier circuit 2. Toutefois, un tel mode de réalisation ne peut pas être utilisé si la première mémoire a une capacité de stockage trop réduite.

[0133] Alternativement, La deuxième version du programme est transmise par le deuxième circuit 4 au premier circuit 2 via le port de debug 10, puis est écrite dans la première mémoire 8 de sorte que la première unité de traitement 6 exécute la deuxième version du programme en lieu et place de la première version du programme. Par exemple, la deuxième version du programme est écrite à l'emplacement mémoire jusqu'ici occupé par la première version du programme dans la première mémoire 8.

[0134] Cette deuxième version du programme peut être obtenue par la deuxième unité de traitement 12 par téléchargement auprès d'un serveur distant (on supposera dans la suite que ce serveur est le serveur 3, bien que cela ne soit pas obligatoire). Ceci est avantageux car permet de pouvoir obtenir une version récente du programme.

[0135] Alternativement, cette deuxième version peut être mémorisée dans la deuxième mémoire 14, ou bien dans une mémoire tierce du dispositif 1 à laquelle la deuxième unité de traitement a accès en lecture. Dans ce cas, la deuxième unité de traitement 12 obtient la deuxième version du programme par un simple accès en lecture dans la

mémoire concernée. Cette variante est avantageuse offre la possibilité de mettre à jour le programme même lorsqu'il n'est pas possible de télécharger de version du programme auprès du serveur 3. Un autre avantage de cette variante est de pouvoir garantir l'origine de la deuxième version du programme. Généralement, la version mémorisée dans la mémoire 8 ou la mémoire tierce est une version de sécurité (« failsafe ») du programme, censée fonctionner de manière normale.

[0136] Très préférentiellement, la deuxième unité de traitement tente d'abord d'établir un canal de communication sécurisé avec le serveur 3 via l'interface de communication 18, dans le but de télécharger une nouvelle version du programme cible. Si cette tentative réussit, la deuxième unité traitement 12 reçoit du serveur 3 une nouvelle version du programme cible. La deuxième unité de traitement 12 envoie via le port de debug 10 la version de programme qu'elle a reçu du serveur 3 via ce canal de communication sécurisé. En revanche, si la tentative d'établissement de canal sécurisé échoue, alors la deuxième unité de traitement 12 lit la version de programme mémorisée dans la deuxième mémoire 8 ou la mémoire tierce. La tentative d'établissement du canal de communication peut notamment être mise en échec par le premier circuit 2 dans les modes de réalisations représentés sur les figures 2 et 3, dans lesquels le deuxième circuit 4 doit passer par le premier circuit 2 pour communiquer avec le serveur 3.

[0137] Quelle que soit l'emplacement d'origine de la deuxième version du programme cible utilisée pour mise à jour (première mémoire 8, deuxième mémoire 14, mémoire tierce du dispositif 1, ou serveur externe 3), il est préférable que cette deuxième version soit signée numériquement et/ou dans un état crypté à son emplacement d'origine. Dans ce cas, la deuxième unité de traitement 12 met en œuvre une vérification de signature numérique et/ou un décryptage de cette deuxième version du programme cible, avant de demander à la première unité de traitement 12 de l'utiliser à la place de la première version du programme cible.

[0138] Les étapes qui précèdent sont répétées à chaque fois qu'une nouvelle suspension du programme cible se produit.

[0139] Dans ce qui précède, il a été présenté des modes de réalisation dans lesquels le contrôle de l'étape 102 est mis en œuvre pendant que le programme est suspendu. Cette suspension a pour avantage de simplifier grandement la mise en œuvre du contrôle effectué. Cependant, il convient de noter qu'une telle suspension n'est pas obligatoire, bien qu'avantageuse. En effet, il peut être envisagé que la deuxième unité de traitement 12 injecte via le port de debug 10 des instructions de code spéciales ayant vocation à déclencher, au cours de leur exécution par la première unité de traitement 6, l'envoi vers le deuxième circuit 4 de données à contrôler au cours de l'étape de contrôle 102. L'exécution du code injecté et l'envoi des données qui sont vérifiées peuvent tout à fait être effectuées sans suspension du programme.

[0140] Il est par ailleurs entendu que l'étape 108 de commande d'arrêt, l'étape 110 de génération de rapport, et l'étape de commande de mise à jour 114 constituent des mesures alternatives susceptibles d'être mises en œuvre isolément les unes des autres, mais qu'elles peuvent également être cumulées au sein d'un même mode de réalisation.

Revendications

[Revendication 1] Dispositif (1) tel qu'un objet connecté comprenant un premier circuit électronique (2) comprenant :
- une première unité de traitement (6) configurée pour exécuter un programme,
- une première mémoire (8) configurée pour mémoriser des données du programme ou manipulées par le programme au cours de son exécution,
- un port de debug (10) dédié au contrôle de l'exécution du programme depuis l'extérieur du premier circuit,
le dispositif (1) comprenant en outre un deuxième circuit électronique (4) connecté au port de debug (10), le deuxième circuit électronique (4) comprenant :
- une deuxième mémoire (14) mémorisant des données de référence relatives au programme,
- une deuxième unité de traitement (12) configurée pour mettre en œuvre les étapes suivantes de manière automatique et autonome via le port de debug (10) : contrôler l'intégrité des données mémorisées par la première mémoire (8) et/ou la conformité de l'exécution du programme par la première unité de traitement (6) à une exécution de référence, à l'aide des données de référence.

[Revendication 2] Dispositif (1) selon la revendication précédente, dans lequel les étapes mises en œuvre de manière automatique et autonome par la deuxième unité de traitement comprennent par ailleurs une commande de suspension du programme, l'étape de contrôle d'intégrité et/ou de conformité étant mise en œuvre pendant que le programme est suspendu.

[Revendication 3] Dispositif (1) selon la revendication précédente, dans lequel la commande de suspension comprend le placement d'un point d'arrêt en un endroit prédéterminé du programme, de sorte à suspendre le programme à l'endroit prédéterminé, ou le placement d'un point d'observation sur une variable du programme, de sorte à suspendre le programme lorsque la variable est modifiée.

[Revendication 4] Dispositif selon la revendication précédente, dans lequel les étapes mises en œuvre de manière automatique et autonome par la deuxième unité de traitement (12) comprennent une étape consistant à vérifier si une condition indépendante de la manière dont le programme s'exécute est remplie, par exemple vérifier si un délai de durée prédéterminée s'est

écoulé depuis un précédent démarrage du programme, une précédente reprise du programme ou une précédente mise sous tension du dispositif (1), l'étape de commande de suspension étant mise en œuvre lorsque la condition est remplie.

[Revendication 5] Dispositif (1) selon l'une des revendications précédentes, dans lequel les étapes mises en œuvre de manière automatique et autonome par la deuxième unité de traitement (12) comprennent la commande d'une reprise du programme par la première unité de traitement (6) lorsqu'aucune compromission du programme ou des données manipulées par le programme n'est révélée lors de l'étape de contrôle, ladite reprise n'étant pas mis en œuvre lorsqu'une compromission du programme ou des données manipulées par le programme est révélée lors de l'étape de contrôle.

[Revendication 6] Dispositif selon l'une des revendications précédentes, dans lequel les étapes mises en œuvre de manière automatique et autonome par la deuxième unité de traitement (12) comprennent une commande d'un arrêt définitif du programme et/ou d'une reconfiguration du premier circuit électronique (2) dans un état empêchant une exécution ultérieure du programme par la première unité de traitement (6), lorsqu'une compromission du programme ou des données manipulées par le programme est révélée lors de l'étape de contrôle.

[Revendication 7] Dispositif (1) selon l'une des revendications précédentes, comprenant une interface de communication (18) avec un serveur (3), et dans lequel les étapes mises en œuvre de manière automatique et autonome par la deuxième unité de traitement (12) comprennent la génération d'un rapport indiquant si une compromission du programme ou des données manipulées par le programme a été révélée lors de l'étape de contrôle, et la commande la transmission du rapport au serveur (3) via l'interface de communication (18) pendant que le programme est suspendu.

[Revendication 8] Dispositif (1) selon la revendication précédente, dans lequel l'interface de communication (18) est configurée pour transmettre le rapport au serveur (3) sans que le rapport ne passe par le premier circuit électronique (2), ou fait partie du premier circuit électronique (2).

[Revendication 9] Dispositif (1) selon la revendication précédente, dans lequel :
- les étapes mises en œuvre de manière automatique et autonome par la deuxième unité de traitement (12) comprennent la commande d'une écriture, via le port de debug (10), du rapport dans un zone d'échange allouée dans la première mémoire (8), de sorte que le rapport soit par la

suite relayé au serveur (3) via l'interface de communication (18), ou dans lequel

- le premier circuit électronique (2) comprend un deuxième port (20) distinct du port de debug (10), et dans lequel les étapes mises en œuvre de manière automatique et autonome par la deuxième unité de traitement (12) comprennent la commande d'une transmission du rapport à l'interface de communication (18) via le deuxième port (20), de sorte que le message d'erreur soit par la suite relayé au serveur via l'interface de communication (18).

[Revendication 10] Dispositif (1) selon l'une des revendications précédentes, dans lequel les données de référence indiquent une séquence d'évènements survenant dans un ordre prédéterminé au cours de l'exécution de référence du programme, et dans lequel le contrôle de conformité de l'exécution du programme à une exécution de référence comprend une comparaison entre l'ordre prédéterminé et un ordre dans lequel surviennent les évènements au cours de l'exécution du programme par la première unité de traitement (6).

[Revendication 11] Dispositif (1) selon l'une des revendications précédentes, dans lequel les étapes mises en œuvre de manière automatique et autonome par la deuxième unité de traitement (12) comprennent une commande de mise à jour (114) du programme lorsqu'une compromission du programme ou des données manipulées par le programme est révélée lors de l'étape de contrôle.

[Revendication 12] Dispositif selon la revendication 11, dans lequel la première mémoire (8) mémorise une première version du programme et une deuxième version du programme avant la commande de mise à jour, la première unité de traitement (8) est configurée pour exécuter sélectivement la première version du programme avant la mise à jour, et dans lequel la commande de mise à jour comprend l'envoi au premier circuit électronique (2), via le port de debug (10), d'au moins une commande demandant à la première unité de traitement (6) d'exécuter sélectivement la deuxième version du programme mémorisé dans la première mémoire (8) en lieu et place de la première version du programme.

[Revendication 13] Dispositif selon la revendication 11, dans lequel la première unité de traitement (8) est configurée pour exécuter une première version du programme avant la mise à jour, et la commande de mise à jour comprend l'envoi au premier circuit électronique (2), via le port de debug (10), d'une deuxième version du programme obtenue par le

deuxième circuit électronique (4) de sorte que la première unité de traitement exécute la deuxième version du programme en lieu et place de la première version du programme.

[Revendication 14] Dispositif selon la revendication précédente, dans lequel la commande de mise à jour (114) comprend un téléchargement de la deuxième version du programme auprès d'un serveur (3) à distance du dispositif (1), ou une lecture de la deuxième version du programme dans une mémoire du dispositif (1), telle que la deuxième mémoire (14).

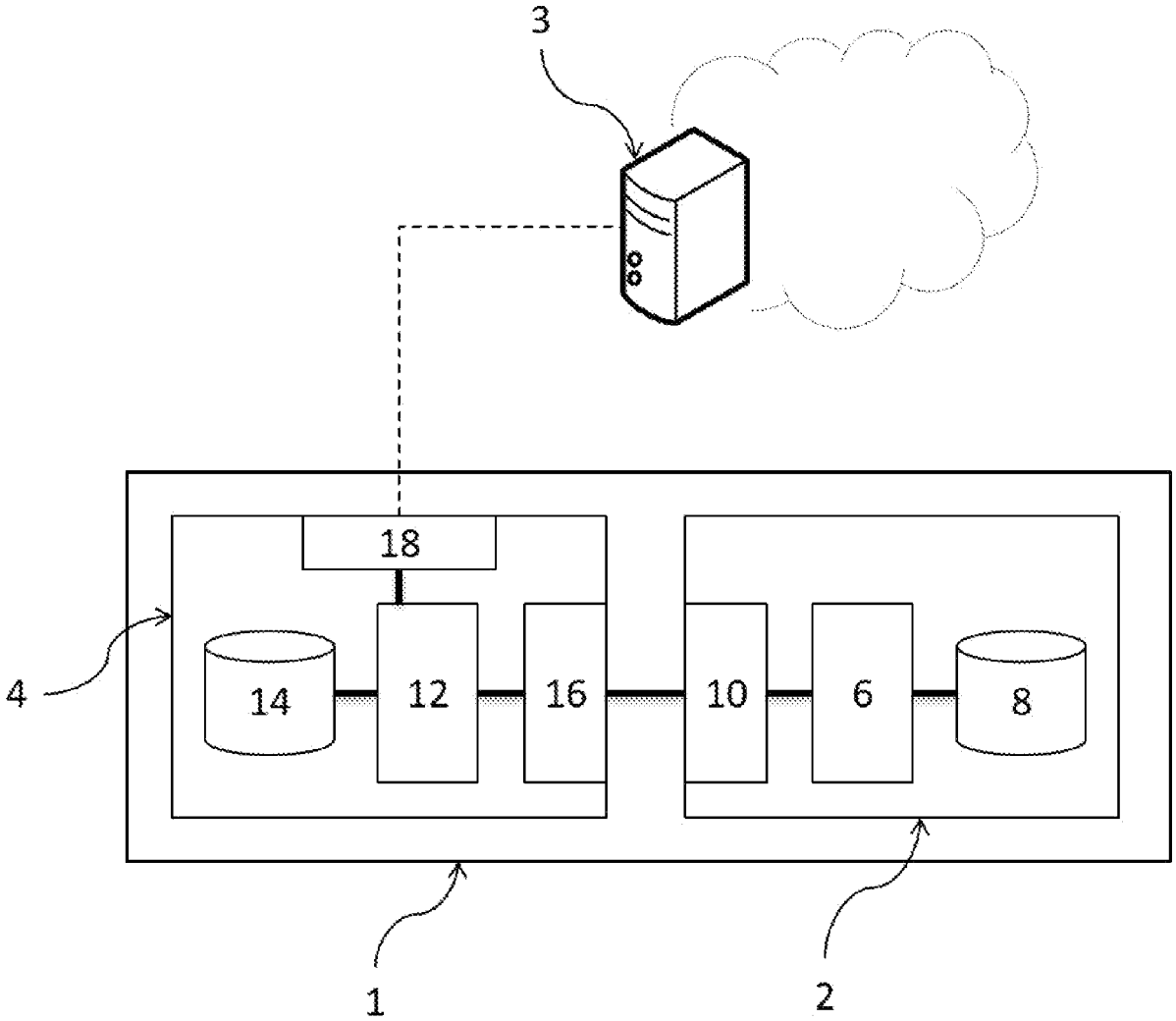
[Revendication 15] Dispositif selon la revendication précédente, dans lequel la deuxième version est lue dans la mémoire du dispositif seulement lorsqu'une tentative d'établissement d'un canal de communication sécurisé entre la deuxième unité de traitement (12) et le serveur (3) échoue.

[Revendication 16] Dispositif selon l'une des revendications 12 à 15, dans lequel la commande de mise à jour (114) comprend une vérification d'une signature numérique de la deuxième version du programme, ou un décryptage de la deuxième version du programme.

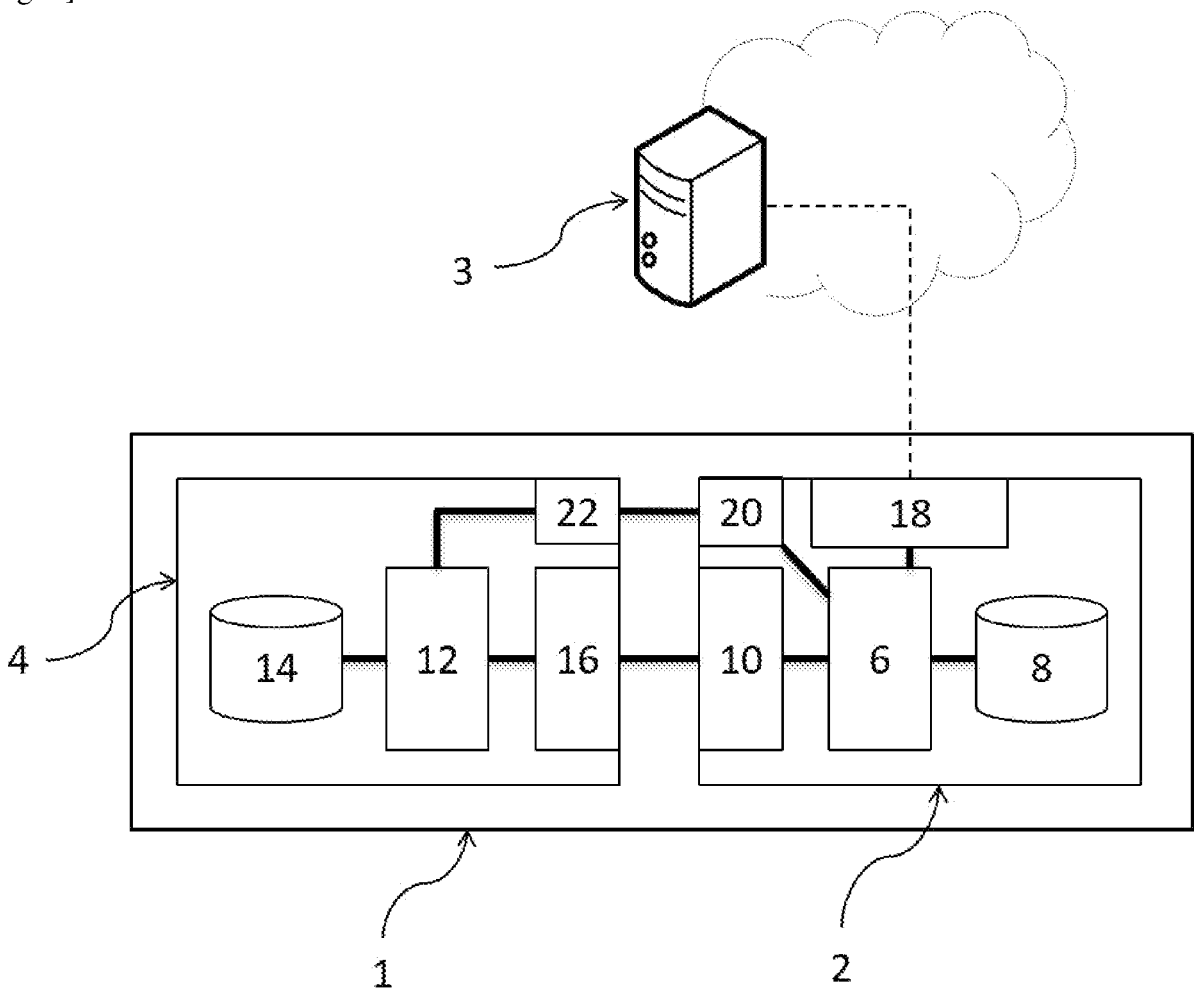
[Revendication 17] Procédé mis en œuvre dans un dispositif (1) tel qu'un objet connecté, le dispositif (1) comprenant :
- un premier circuit électronique (2) comprenant une première unité de traitement (6) configurée pour exécuter un programme, une première mémoire (8) configurée pour mémoriser des données du programme ou manipulées par le programme au cours de son exécution, et un port de debug (10) dédié au contrôle de l'exécution du programme depuis l'extérieur du premier circuit,
- un deuxième circuit électronique (4) connecté au port de debug (10) et comprenant une deuxième mémoire (14) mémorisant des données de référence relatives au programme,
le procédé comprenant les étapes suivantes mises en œuvre par une deuxième unité de traitement (12) du deuxième circuit électronique (4), de manière automatique et autonome via le port de debug (10) :
contrôler l'intégrité des données mémorisées par la première mémoire (8) et/ou la conformité de l'exécution du programme par la première unité de traitement (6) à une exécution de référence, à l'aide des données de référence.

[Revendication 18] Produit programme d'ordinateur comprenant des instructions de code de programme pour l'exécution des étapes du procédé selon la revendication précédente, lorsque ce procédé est exécuté par une unité de traitement (12).

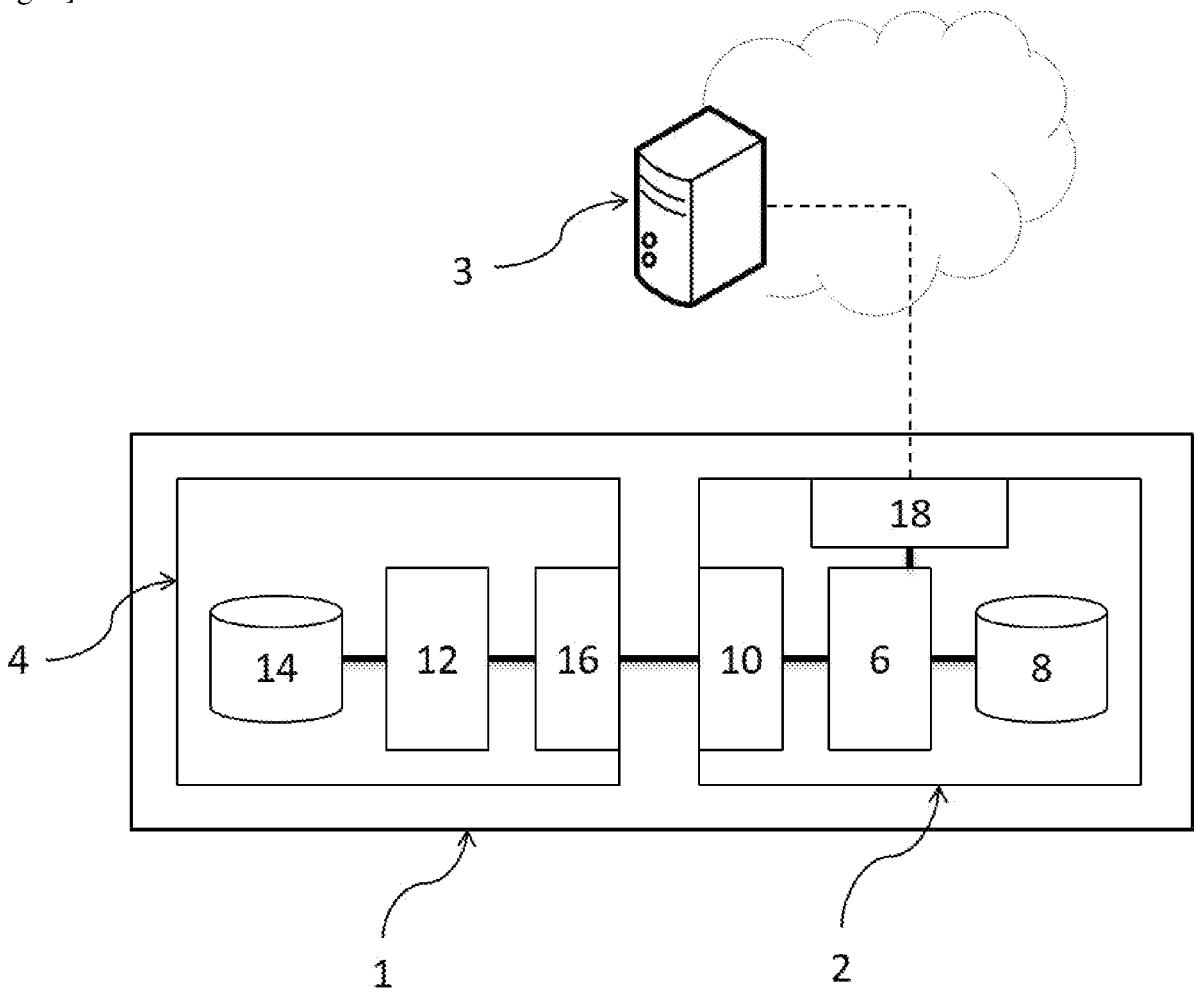
[Fig. 1]



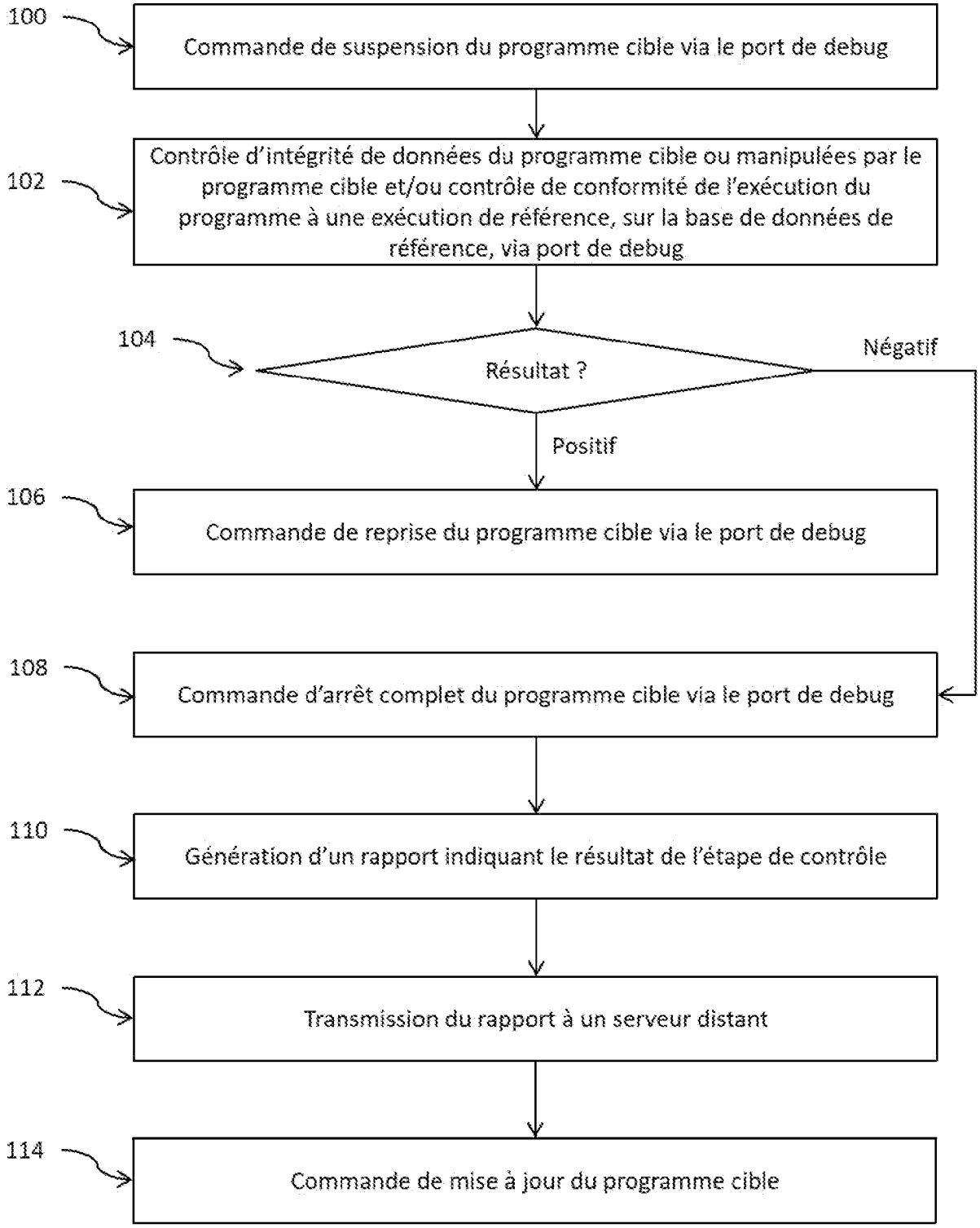
[Fig. 2]



[Fig. 3]



[Fig. 4]


100
Commande de suspension du programme cible via le port de debug
102
Contrôle d'intégrité de données du programme cible ou manipulées par le programme cible et/ou contrôle de conformité de l'exécution du programme à une exécution de référence, sur la base de données de référence, via port de debug
104
Résultat ?
Négatif
Positif
106
Commande de reprise du programme cible via le port de debug
108
Commande d'arrêt complet du programme cible via le port de debug
110
Génération d'un rapport indiquant le résultat de l'étape de contrôle
112
Transmission du rapport à un serveur distant
114
Commande de mise à jour du programme cible

RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE

RAPPORT DE RECHERCHE PRÉLIMINAIRE

établi sur la base des dernières revendications déposées avant le commencement de la recherche

N° d'enregistrement national

FA 871001
FR 1907796

DOCUMENTS CONSIDÉRÉS COMME PERTINENTS		Revendication(s) concernée(s)	Classement attribué à l'invention par l'INPI
Catégorie	Citation du document avec indication, en cas de besoin, des parties pertinentes		
X	WO 2009/055147 A1 (MICROSOFT CORP [US]) 30 avril 2009 (2009-04-30) * abrégé * * alinéa [0015] - alinéa [0020] * * alinéa [0024] - alinéa [0049] * * revendications 3, 4, 10, 12 * * figures 1-4 * -----	1-18	G06F21/52 G06F21/70
A	EP 3 382 588 A1 (IDEMIA IDENTITY & SECURITY FRANCE [FR]) 3 octobre 2018 (2018-10-03) * le document en entier * -----	1-18	
A	EP 1 702 268 A2 (TRUSTED LOGIC [FR]) 20 septembre 2006 (2006-09-20) * le document en entier * -----	1-18	
			DOMAINES TECHNIQUES RECHERCHÉS (IPC)
			G06F

Date d'achèvement de la recherche	Examinateur
23 mars 2020	Bae, Jun-Young

CATÉGORIE DES DOCUMENTS CITÉS

X : particulièrement pertinent à lui seul
Y : particulièrement pertinent en combinaison avec un autre document de la même catégorie
A : arrière-plan technologique
O : divulgation non-écrite
P : document intercalaire

T : théorie ou principe à la base de l'invention
E : document de brevet bénéficiant d'une date antérieure à la date de dépôt et qui n'a été publié qu'à cette date de dépôt ou qu'à une date postérieure.
D : cité dans la demande
L : cité pour d'autres raisons

.....

& : membre de la même famille, document correspondant

EPO FORM 1503 12.99 (P04C14)

ANNEXE AU RAPPORT DE RECHERCHE PRÉLIMINAIRE RELATIF A LA DEMANDE DE BREVET FRANÇAIS NO. FR 1907796 FA 871001

La présente annexe indique les membres de la famille de brevets relatifs aux documents brevets cités dans le rapport de recherche préliminaire visé ci-dessus.
Les dits membres sont contenus au fichier informatique de l'Office européen des brevets à la date du 23-03-2020
Les renseignements fournis sont donnés à titre indicatif et n'engagent pas la responsabilité de l'Office européen des brevets, ni de l'Administration française

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
WO 2009055147 A1	30-04-2009	US 2009113210 A1	30-04-2009
		WO 2009055147 A1	30-04-2009
EP 3382588 A1	03-10-2018	EP 3382588 A1	03-10-2018
		FR 3064781 A1	05-10-2018
		US 2018285189 A1	04-10-2018
EP 1702268 A2	20-09-2006	EP 1702268 A2	20-09-2006
		FR 2864655 A1	01-07-2005
		JP 4777903 B2	21-09-2011
		JP 2007517299 A	28-06-2007
		US 2010070804 A1	18-03-2010
		WO 2005073859 A2	11-08-2005

EPO FORM P0465