(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
20 décembre 2012 (20.12.2012)

WIPO | PCT

(10) Numéro de publication internationale
WO 2012/172245 A1

(51) Classification internationale des brevets :
G06F 21/24 (2006.01)

(21) Numéro de la demande internationale :
PCT/FR2012/051304

(22) Date de dépôt international :
11 juin 2012 (11.06.2012)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
11 55354 17 juin 2011 (17.06.2011) FR

(71) Déposant *(pour tous les États désignés sauf US)* : **MORPHO** [FR/FR]; 27 rue Leblanc, F-75015 Paris (FR).

(72) Inventeurs; et

(75) Inventeurs/Déposants *(pour US seulement)* : **BOULET, Frédéric** [FR/FR]; c/o Morpho, 27 rue Leblanc, F-75015 Paris (FR). **BARTHE, Michaël** [FR/FR]; c/o Morpho, 27 rue Leblanc, F-75015 Paris (FR). **LE, Thanh Ha** [VN/FR]; c/o Morpho, 27 rue Leblanc, F-75015 Paris (FR).

(74) Mandataires : **WLODARCZYK, Lukasz** et al.; Cabinet Plasseraud, 52 rue de la Victoire, F-75440 Paris Cedex 09 (FR).

(81) États désignés *(sauf indication contraire, pour tout titre de protection nationale disponible)* : AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

[Suite sur la page suivante]

(54) Title : SECURE TRANSFER BETWEEN NON-VOLATILE MEMORY AND VOLATILE MEMORY

(54) Titre : TRANSFERT SECURISE ENTRE MEMOIRE NON-VOLATILE ET MEMOIRE VOLATILE

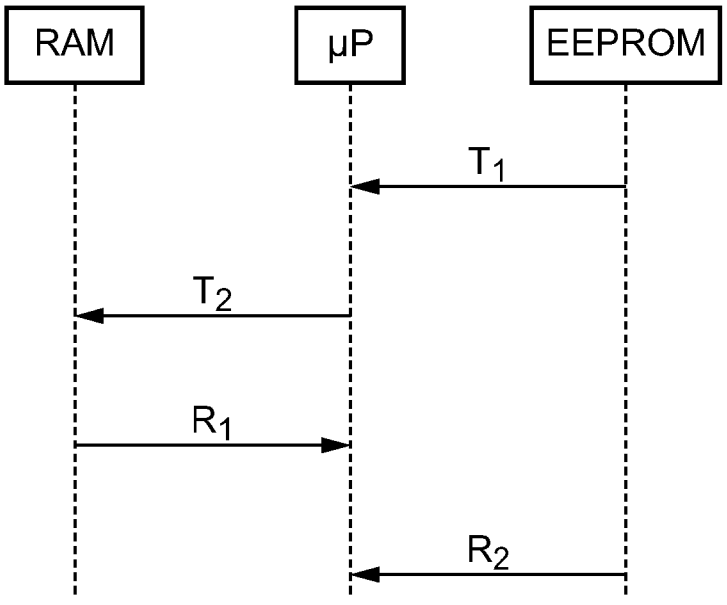

FIG. 2

(57) Abstract : The description particularly relates to a method for securing data transfer (T1, T2) between a non-volatile memory (EEPROM) and a volatile memory (RAM). The description also relates to an electronic device and computer program capable of implementing such a method, as well as to a storage medium including such a computer program.

(57) Abrégé : La description concerne notamment un procédé de sécurisation de transfert de données (T1, T2) entre une mémoire non volatile (EEPROM) et une mémoire volatile (RAM). La description concerne également un dispositif électronique et un programme d'ordinateur aptes à mettre en œuvre un tel procédé, ainsi qu'un support de stockage comprenant un tel programme d'ordinateur.

WO 2012/172245 A1

(84) États désignés *(sauf indication contraire, pour tout titre de protection régionale disponible)* : ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), eurasien (AM, AZ, BY, KG, KZ, RU, TJ, TM), européen (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Déclarations en vertu de la règle 4.17 :

— *relative à la qualité d'inventeur (règle 4.17.iv))*

Publiée :

— *avec rapport de recherche internationale (Art. 21(3))*

TRANSFERT SECURISE
ENTRE MEMOIRE NON-VOLATILE ET MEMOIRE VOLATILE

L'invention concerne la sécurisation de transferts de données entre une mémoire non volatile et une mémoire volatile.

Les mémoires dites "non volatiles" sont des mémoires qui peuvent conserver des données sans nécessiter d'alimentation externe. Les mémoires non volatiles peuvent être programmables (par exemple les mémoires de type EEPROM, Flash, etc.) ou non programmable (comme par exemple les mémoires de type ROM, dont le contenu est défini une fois pour toutes lors de la fabrication).

Les mémoires dites "volatiles" (par exemple les mémoires de type RAM) sont des mémoires qui perdent leur contenu lorsque l'alimentation externe qui leur fournit de l'énergie est interrompue. Les mémoires volatiles sont typiquement beaucoup plus rapides que les mémoires non volatiles en particulier pour les opérations d'écriture. Par exemple une écriture EEPROM ou FLASH est généralement beaucoup plus lente qu'une écriture RAM.

Les dispositifs électroniques comprennent souvent à la fois de la mémoire non volatile (par exemple pour stocker un système d'exploitation, des logiciels, des données, etc.) et de la mémoire volatile permettant par exemple de manipuler des données temporaires (variables d'état, résultats de calculs, etc.) lors de l'exécution d'un logiciel.

Il est fréquemment nécessaire de transférer des données localisées en mémoire non-volatile vers une mémoire volatile (par exemple afin de les manipuler). Inversement, il peut être nécessaire de transférer des données de la mémoire volatile vers la mémoire non volatile. Ceci peut correspondre notamment à la mise à jour de la mémoire non-volatile en fonction de résultats d'un calcul, ou encore à une opération de « SWAP » lorsqu'une quantité de mémoire volatile requise est supérieure à la quantité disponible (en transfère alors temporairement une partie de la mémoire volatile non indispensable à

l'instant donné vers de la mémoire non-volatile).

Cette phase de transfert est très sensible à certaines attaques et perturbations pouvant altérer les données. Les conséquences d'une telle altération (délibérée ou accidentelle) peuvent être catastrophiques notamment pour un dispositif électronique de sécurité (carte à puce, pare-feu, etc.) ou pour les dispositifs électroniques remplissant des fonctions pouvant affecter la vie humaine (serveurs informatiques pour contrôle aérien, serveurs de supervision de centrales nucléaires, etc.).

Dans le premier cas (dispositifs de sécurité), une attaque peut éventuellement permettre l'utilisation non autorisée d'une clé ou d'un code secret bloqué, la modification de paramètres de configuration internes ou de conditions d'accès, la divulgation de clé secrète, etc. Dans le second cas, on peut être confronté à de graves accidents.

Différentes méthodes ont été proposées pour protéger l'intégrité d'une donnée localisée en mémoire non-volatile pendant le transfert de cette donnée vers une mémoire volatile. On connaît par exemple WO 01/98872 (Microchip), WO 02/45035 (Gemplus), ou WO 2009/106757 (Sagem). Ces trois dernières méthodes ont en commun de stocker dans la mémoire non volatile, avec une donnée à protéger, une valeur de contrôle d'intégrité de cette donnée à protéger (par exemple un CRC). Après transfert de la donnée protégée (et de son CRC), on peut ainsi recalculer le CRC et contrôler l'intégrité de la donnée par comparaison des deux CRCs.

Ces méthodes sont susceptibles de poser un certain nombre de problèmes.

En particulier, dans le contexte de Javacard (qui est une plateforme s'appuyant sur le langage Java, et qui est particulièrement adaptée pour les cartes à puces) il est certes possible d'ajouter une valeur de contrôle d'intégrité sur des objets Java à protéger. Cependant, cette possibilité est typiquement gérée par la plateforme Javacard et non pas par l'application Java ayant créé l'objet considéré. Le choix d'activer ou non le contrôle d'intégrité est généralement aisé pour certains objets (par exemple un objet

"code PIN", des clés, etc.) mais il est difficile à mettre en œuvre pour d'autres objets, par exemple pour un buffer (dont le contenu est a priori inconnu). Par conséquent, en pratique, on peut être conduit à activer le contrôle d'intégrité en mode tout ou rien (toujours ou jamais activé). Dans le premier cas (contrôle d'intégrité activé) l'impact en termes de performance est très important. Par exemple, un buffer sera sécurisé en permanence, même quand il se trouve contenir des données sans caractère sensible. Dans le deuxième cas (contrôle d'intégrité désactivé) l'intégrité des données est moins bien assurée. Par exemple, un buffer contenant à un instant donné une donnée sensible peut faire l'objet d'une attaque.

D'autre part, si un objet associé à une valeur de contrôle d'intégrité n'est que partiellement transféré, la plateforme doit recalculer la valeur de contrôle d'intégrité sur la totalité de l'objet et non seulement sur la partie de donnée à transférer. Ceci peut conduire soit à transférer la totalité de l'objet alors que seule une partie est requise (réduction de performance), soit à abandonner la vérification d'intégrité (réduction de sécurité).

Enfin, ces méthodes ne protègent qu'un transfert de la mémoire non volatile vers la mémoire volatile (et non l'inverse).

Quelques composants (par exemple certaines puces pour cartes à puce) disposent d'un mode sécurisé pour le transfert d'une donnée localisée en mémoire non-volatile vers la mémoire volatile (par exemple un bus de données chiffré). Cependant, face à des techniques d'attaque sophistiquées, ce mode de protection est parfois inefficace et peut laisser passer des perturbations pendant le transfert sans qu'aucune alerte ne soit déclenchée. De telles perturbations sont parfois la conséquence d'attaques se basant notamment sur une possibilité de synchronisation pendant le transfert de données.

L'invention vise à améliorer la situation.

Un aspect de l'invention concerne un procédé de sécurisation de transfert de données. Le procédé comprend un transfert de données entre une

mémoire non volatile et une mémoire volatile. Le procédé comprend un calcul, postérieurement à l'initiation dudit transfert, d'une première empreinte desdites données. Le procédé comprend une relecture desdites données à partir de la mémoire source du transfert. Le procédé comprend un calcul, postérieurement à l'initiation de ladite relecture, d'une deuxième empreinte des données relues. Le procédé comprend enfin une validation du transfert si les deux empreintes correspondent.

Ce procédé est avantageux notamment en ce qu'il économise de la mémoire (il n'est pas nécessaire de stocker un CRC pour chaque objet). De surcroît, ce procédé offre une plus grande flexibilité que les procédés de l'art antérieur. En particulier, il n'est pas nécessaire de définir à l'avance les objets à protéger. Le procédé n'est pas donc contraint par une taille prédéfinie d'objet, et peut sans difficulté transférer un sous ensemble d'un objet tout en vérifiant l'intégrité du seul sous ensemble.

Ce procédé améliore particulièrement la protection contre les attaques visant la mémoire cible du transfert, mais également, notamment, contre les attaques visant les bus de données reliant éventuellement les mémoires volatiles et non volatiles.

Un autre aspect de l'invention concerne un programme d'ordinateur comprenant une suite d'instructions mettant en œuvre les étapes du procédé selon l'une des revendications précédentes lorsqu'elles sont exécutées par un processeur.

Un autre aspect de l'invention concerne un support de stockage non transitoire lisible par ordinateur comprenant un programme d'ordinateur selon le précédent aspect de l'invention.

Un autre aspect de l'invention concerne un dispositif électronique sécurisé comprenant une mémoire non volatile ainsi qu'une mémoire volatile. Le dispositif électronique sécurisé comprend un circuit de transfert de données entre la mémoire non volatile et la mémoire volatile. Le dispositif électronique sécurisé comprend un calculateur d'empreintes de données. Le dispositif électronique sécurisé comprend un circuit de protection agencé pour valider

un transfert de données du circuit de transfert de données. Le circuit de protection valide le transfert lorsqu'une première empreinte, calculée (postérieurement à l'initiation du transfert des données) sur les données transférées, correspond à une deuxième empreinte calculée (postérieurement à l'initiation d'une relecture desdites données de la mémoire source du transfert) sur les données relues.

D'autres aspects, buts et avantages de l'invention apparaîtront de manière non limitative à la lecture de la description de quelques uns de ses modes de réalisation.

L'invention sera également mieux comprise à l'aide des dessins, sur lesquels :

- la Figure 1 illustre différentes étapes d'un procédé selon un mode de réalisation de l'invention ;
- la Figure 2 illustre différentes étapes d'un procédé selon un autre mode de réalisation de l'invention ;
- la Figure 3 illustre l'architecture d'un dispositif selon un mode de réalisation de l'invention.
- la Figure 4 illustre l'architecture d'un dispositif selon un autre mode de réalisation de l'invention.

Un mode de réalisation possible concerne un procédé de sécurisation de transfert de données.

Le procédé comprend un transfert de données entre une mémoire non volatile (par exemple de l'EEPROM, de la Flash ou de la ROM) et une mémoire volatile (par exemple de la RAM). Le transfert de données peut avoir lieu indifféremment de la mémoire non volatile vers la mémoire volatile (la mémoire non volatile étant alors la mémoire source du transfert et la mémoire volatile étant la mémoire cible du transfert) ou de la mémoire volatile vers la mémoire non volatile (la mémoire volatile étant alors la mémoire source et la mémoire non volatile étant la mémoire cible), à condition dans ce dernier cas que la mémoire non volatile soit une mémoire réinscriptible (cela ne marcherait

évidemment pas avec de la ROM). Le transfert peut être mis en œuvre à l'aide d'un processeur. Par exemple, le procédé peut comprendre une étape de lecture de la mémoire source par le processeur, le processeur recevant la donnée lue dans un de ses registres, puis une étape d'écriture de la donnée lue (c'est-à-dire du contenu du registre du processeur précédemment rempli) dans la mémoire cible par le processeur. Les termes « lecture » et « écriture » sont utilisés en adoptant le point de vue du processeur.

Si les données à transférer sont de taille plus importante que la taille du registre de processeur considéré, il est possible de diviser le transfert en plusieurs étapes de transfert, chaque étape transférant une quantité de données de taille au plus égale à la taille du registre de processeur considéré.

Le transfert peut également être mis en œuvre par un circuit électroniques dédiés (par exemple à l'aide d'un FPGA ou de toute forme de logique câblée réalisant une interface entre la mémoire volatile et la mémoire non volatile), ou encore par un contrôleur de mémoire. Ainsi, même en présence d'un processeur principal, un contrôleur de mémoire (par exemple de la mémoire source), tel que le contrôleur CTRL de mémoire Flash représenté sur la figure 3, peut prendre la main sur un bus de données reliant les mémoires source et cible ainsi que le processeur, et transférer des données de la mémoire source vers la mémoire cible, par exemple à l'aide d'une technique de type DMA (de l'anglais Direct Memory Access, signifiant Accès Direct à la Mémoire).

Le procédé comprend un calcul, postérieurement à l'initiation dudit transfert, d'une première empreinte desdites données. L'empreinte peut être tout simplement le résultat d'un code de redondance cyclique bien connu, tel qu'un code de type CRC (CRC16 ou CRC32 par exemple) appliqué aux données transférées. L'empreinte peut alternativement résulter de calculs de parité sur les données transférées, ou d'un calcul de type XOR (ou exclusif) des données sur elles mêmes, par exemple l'empreinte peut être un octet résultant du XOR de tous les octets composant les données transférées. L'empreinte peut également résulter d'une application de calculs bien plus complexes tels que des calculs cryptographiques. L'empreinte peut ainsi être

un hash (tel que MD5, SHA1 ou SHA256) des données transférées, ou le résultat d'un chiffrement des données transférées par un algorithme symétrique tel que DES, 3DES ou AES. L'empreinte peut même être une signature asymétrique (telle qu'une signature RSA, DSA, ou par courbes elliptiques) des données transférées. Ces différentes empreintes représentent chacune des complexités de mise en œuvre différentes et des niveaux de sécurité différents. Ainsi une parité à un bit est typiquement plus simple mais moins sûre qu'un XOR, qui est lui-même en général plus simple mais moins sûr qu'un CRC, qui est lui-même la plupart du temps plus simple mais moins sûr qu'un hash, un chiffrement symétrique, ou une signature asymétrique. Dans le cas d'une implémentation sous forme logicielle, la complexité est généralement proportionnelle à la durée d'exécution, ce qui signifie que les empreintes les plus complexes sont également les plus longues à calculer (durée accrue et donc performances réduites en termes de rapidité d'exécution). Cependant, les empreintes peuvent également être calculées en tout ou partie de manière matérielle, dans ce cas leur calcul peut être rapide (en cas d'accélération matérielle) voire « instantané ». Par « instantané », il faut comprendre que le calcul est effectué en parallèle avec le transfert et prend au plus autant de temps que le transfert (et ne le ralentit donc pas). Ainsi, une signature asymétrique, un hash ou un chiffrement symétrique peuvent être réalisés à l'aide d'un crypto-processeur. Le crypto-processeur peut être sollicité par le processeur (et communiquer avec lui par exemple par interruption) ou fonctionner de manière spontanée (en continu). Les crypto-processeurs sont bien connus, notamment dans le domaine des cartes à puces. Par exemple de nombreux composants de la famille ST23 de STMicroelectronics comprennent un crypto-processeur (notamment ST23YL18, ST23YL48, ST23YL80, ST23YT34, ST23YT66, ST23ZL18, ST23ZL34 ou ST23ZL48). Le calcul de la première empreinte étant postérieur à l'initiation du transfert, il n'est pas nécessaire de stocker le résultat du calcul dans la mémoire source ni dans la mémoire cible, dans la mesure ou le processeur (ou toute autre entité responsable du calcul de l'empreinte) est capable d'effectuer le calcul (ou de sous traiter le calcul à une entité extérieure telle qu'un crypto-processeur) sans impliquer ni la mémoire source ni la

mémoire cible. Ainsi, un processeur peut typiquement effectuer un XOR de tous les octets d'une donnée reçue dans un de ses registres (en supposant que la donnée remplit complètement ledit registre) à l'aide de ses seuls registres internes, et déterminer ainsi une empreinte correspondant à ce XOR sans avoir à faire appel à une mémoire externe. Le CRC constitue un type d'empreinte correspondant généralement à un bon compromis entre sécurité et simplicité.

Le calcul de la première empreinte étant effectué sur demande (après initiation d'un transfert, et non en prévision d'un éventuel transfert), le résultat est exploité dans la foulée et n'a donc pas besoin d'être stocké de manière permanente avec les données. Le calcul de la première empreinte peut être réalisé en tout ou partie en parallèle avec le transfert. Bien qu'un stockage permanent du résultat du calcul puisse être mis en œuvre, il est avantageusement remplacé par exemple par un stockage temporaire dans un registre d'un processeur mettant en œuvre le procédé.

Le procédé comprend une relecture desdites données à partir de la mémoire source du transfert. Ainsi, lorsque le procédé est mis en œuvre par un processeur, le processeur peut lire une deuxième fois le contenu de la mémoire source. Le procédé comprend alors un calcul, postérieurement à l'initiation de ladite relecture, d'une deuxième empreinte des données relues. Cette deuxième empreinte peut être calculée selon un mode opératoire identique à celui décrit précédemment pour la première empreinte.

Le procédé comprend enfin une validation du transfert si les deux empreintes correspondent. Les deux empreintes peuvent correspondre lorsqu'elles sont égales. Dans des schémas plus complexes, il est envisageable de calculer des empreintes différentes mais dont on peut dire qu'elles correspondent l'une à l'autre. Par exemple il est possible de générer un nombre aléatoire et d'effectuer un XOR entre la donnée lue de la mémoire source lors du transfert et ce nombre aléatoire, puis de chiffrer le résultat du XOR avec une clé symétrique afin de produire une première empreinte. On peut ensuite directement chiffrer la donnée relue pour produire la deuxième empreinte. Les deux empreintes ne sont alors pas égales même en l'absence

d'attaque, mais il est possible de vérifier que les empreintes correspondent en les déchiffrant, et en vérifiant que le déchiffrement de la première empreinte est égal au déchiffrement de la deuxième empreinte XOR le nombre aléatoire précédemment généré. Ces schémas complexes sont pertinents en présence d'accélération matérielle.

Dans le cas ou le transfert n'est pas validé, le procédé peut mettre en œuvre une contremesure qui peut comprendre notamment l'enregistrement dans un journal (log file en anglais) de l'événement considéré, ainsi qu'éventuellement d'un contexte associé (par exemple : détection d'une attaque sur le transfert de données de certificat X.509 par l'application de porte monnaie électronique n°3 à la date D et à l'heure H). La contremesure peut comprendre un blocage du dispositif protégé par le procédé (le dispositif mettant en œuvre le procédé pouvant être par exemple une carte à puce) afin d'empêcher toute tentative d'attaque ultérieure. Un tel blocage peut comprendre un effacement de toutes les données sensible, ou peut se contenter de bloquer le fonctionnement jusqu'à présentation d'un code de déblocage ou de toute authentification appropriée par une entité habilitée. La contremesure peut également mettre en œuvre une transmission de l'événement détecté (attaque d'un transfert de données) par tout moyen de communication disponible (réseau sans fil, réseau filaire, etc.). Dans le cas où aucun réseau n'est disponible au moment de l'attaque, le procédé peut prévoir l'envoi de l'information dès qu'un réseau devient disponible. Par exemple si un attaquant à subtiliser une carte bancaire, a tenté de l'attaquer sans succès puis l'a rendue à son propriétaire sans que ce dernier ne se rende compte qu'elle lui avait été temporairement dérobée, la carte peut transmettre l'information selon laquelle elle a été attaquée la prochaine fois que l'utilisateur la connecte à un terminal connecté à un réseau.

Un avantage du procédé est de permettre de protéger un dispositif mettant en œuvre des transferts de données contre des attaques (ou erreurs de transferts pouvant être liées à des problèmes matériels ou à l'environnement du dispositif). Par exemple, un dispositif tel qu'une carte à puce java peut être amené à authentifier un utilisateur de la carte afin de

l'autoriser ou non à effectuer une transaction à l'aide de cette carte (par exemple une transaction de paiement). L'authentification peut comprendre un transfert d'un code PIN stocké dans une mémoire de type EEPROM vers une mémoire RAM, et une réception par une interface de communication d'un code PIN saisi par un utilisateur, le code PIN saisi étant transféré dans la mémoire RAM. La carte à puce compare alors le code PIN saisi au code PIN stocké. Cependant un attaquant pourrait effectuer une attaque perturbant le fonctionnement de la carte, par exemple une attaque par injection de faute. Il peut s'agir notamment d'attaques bien connues telles que les attaques laser, les attaques par perturbation de la tension d'alimentation ou de la fréquence de fonctionnement. De telles attaques peuvent perturber le transfert. Si l'attaquant parvient par exemple à passer à zéro la valeur du code PIN lu depuis la mémoire EEPROM, il peut lui suffire de rentrer le code PIN zéro pour s'authentifier à la carte (en pratique d'autres contremesures devraient généralement être désactivées avant de parvenir à mener l'attaque à bien). La mise en œuvre du procédé sur une telle carte à puce permet de détecter la modification frauduleuse du code PIN et prévenir l'attaque.

Le procédé peut être mis en œuvre au niveau de la JVM (Java Virtual Machine) d'une carte à puce et protéger les opérations de transfert mises en œuvre par la JVM lorsqu'elle exécute des applets java.

Ainsi qu'illustré sur la figure 1, le procédé peut ainsi comprendre un transfert de données se décomposant en une lecture T1 d'une mémoire source (en l'occurrence une mémoire non volatile de type EEPROM) par un processeur µP, puis en une écriture T2 des données lues en EEPROM vers une mémoire cible (en l'occurrence une mémoire volatile de type RAM). Le procédé peut comprendre une relecture R à partir de l'EEPROM des données précédemment transférées. Le calcul des empreintes des données transférées et relues peut être réalisé par le microprocesseur µP. Par exemple, le microprocesseur peut effectuer une opération bien connue de type « read » (en langage assembleur) à l'adresse pertinente de l'EEPROM. Un registre du processeur reçoit alors la donnée à transférer en RAM (lue depuis l'EEPROM).

Le processeur peut alors calculer une empreinte de cette donnée. Par exemple il peut calculer un CRC de cette donnée, ou effectuer un XOR de tous les octets composants le registre contenant cette donnée, en supposant que la donnée remplit complètement le registre. Par la suite, le processeur peut écrire la donnée à l'adresse pertinente de la RAM, à l'aide par exemple d'une instruction assembleur bien connue de type « write ». Différents langages assembleur utilisent différents noms pour ces commandes « read » et « write », par exemple « LD », « MV », « MOV » etc.

Un mode de réalisation possible comprend de surcroît une génération de nombre aléatoire. Le procédé insère alors, entre le transfert et la relecture, une attente dont la durée est fonction de ce nombre aléatoire. Par exemple, le nombre aléatoire peut être un nombre de 16 bits représentant une durée en microsecondes. On peut ainsi insérer une attente comprise en 0ms et 65ms environ.

Cette attente gêne un attaquant qui pourrait être tenté, connaissant l'existence du procédé selon l'invention, de réitérer deux fois la même attaque au moment du transfert et au moment de la relecture (afin de rendre l'attaque indétectable). L'attente est avantageuse car elle introduit une difficulté de synchronisation, l'attaquant devant deviner à quel moment sont effectuées non seulement le transfert mais également la relecture.

Il est possible, pendant la période d'attente, d'effectuer des tâches annexes (par exemple en mode multitâches) afin de ne pas pénaliser les performances globales. Ainsi, des tâches qui auraient dues être effectuées postérieurement peuvent être commencées par anticipation durant la boucle d'attente. On peut privilégier dans ce cas des tâches postérieures administratives telles que le recyclage de zones mémoires préalablement allouées et désormais inutilisées (« garbage collector »). Evidemment, il convient d'exclure toute tâche postérieure sensible dont l'exécution serait subordonnée à un résultat lié au transfert.

Un mode de réalisation particulier de cette attente est représenté sur la

figure 1 (attente ATT entre la fin de l'écriture T2 et le début de la relecture R).

Un autre mode de réalisation possible comprend également une génération de nombre aléatoire. Le procédé effectue alors le transfert soit avant soit après la relecture en fonction de la valeur du nombre aléatoire généré.

Ainsi ce nombre aléatoire peut être un bit, dont la valeur 0 ou 1 détermine quelle opération (transfert ou relecture) doit être effectuée en premier.

Il peut également s'agir d'un nombre quelconque, une valeur seuil étant alors utilisée pour déterminer (selon que le nombre aléatoire et ou non plus grand que le seuil) quelle opération effectuer en premier. Le réglage de la valeur de seuil permet ainsi d'ajuster la probabilité que l'une des opérations se réalise avant l'autre (ce qui peut être pertinent dans certaines situations).

Le nombre aléatoire peut être le même nombre que celui éventuellement utilisé pour définir la durée d'une attente entre le transfert et la relecture. Ainsi, on peut décider de générer un seul nombre aléatoire de 16 bits compris entre -32768 et 32767, et définir ainsi, selon le signe du nombre aléatoire, quelle opération doit être effectuée en premier (transfert ou relecture), et selon la valeur absolue du nombre aléatoire, quelle durée doit séparer les deux opérations (par exemple une durée comprise entre 0ms et environ 32ms si le nombre aléatoire représente des microsecondes).

Un intérêt d'inverser aléatoirement les deux opérations tient au fait que l'attaquant peut dans certains cas n'être en mesure de reconnaître que l'une des opérations (le transfert ou la relecture). Une telle identification de l'opération peut être fonction par exemple de sa signature électromagnétique (les rayonnements électromagnétiques émis lors de l'exécution de ces opérations peuvent éventuellement être caractérisés) ou encore de la consommation électrique qu'elle génère (qui la encore peut éventuellement être caractérisée, par exemple par une attaque de type SPA).

L'homme du métier vise idéalement à concevoir des opérations indétectables. Cependant, certains progrès technologiques pourraient

permettre à l'avenir des modes de détections qui ne sont pas nécessairement envisagés à l'heure actuelle et contre lesquels on peut ainsi se prémunir par anticipation.

Un autre mode de réalisation possible comprend l'insertion d'une lecture de données entre le transfert et la relecture.

Ceci est avantageux car cette lecture « parasite » peut induire un attaquant en erreur. Il peut penser avoir repéré par exemple la relecture alors qu'il n'en est rien. Avantageusement, cette lecture est une lecture qui était nécessaire par la suite. Ainsi on ne pénalise pas les performances substantiellement, en ne faisant que modifier l'ordre des tâches. Il faut s'assurer dans ce cas que cette lecture n'était pas subordonnée au résultat du transfert que l'on cherche à protéger.

Selon une variante, il est au contraire avantageux d'utiliser une lecture totalement parasite au sens ou elle est fonctionnellement inutile (la valeur lue n'est pas pertinente et est dénuée d'intérêt pour la suite des opérations), son seul rôle étant de perturber l'attaquant potentiel qui tente d'altérer le transfert.

Un mode de réalisation particulier de cette lecture « parasite » est représenté sur la figure 1 (lecture L entre l'écriture T2 et la relecture R).

Selon un mode de réalisation possible, la mémoire non volatile et la mémoire volatile sont toutes deux connectées à un même bus de données (tel que le bus BUS_D des figures 3 et 4) et le procédé fixe la fréquence de fonctionnement du bus de données de manière aléatoire.

Ainsi, dans le cas d'une carte à puce, la puce reçoit typiquement un signal d'horloge de l'extérieur sur l'un de ses contacts afin de synchroniser les échanges avec un lecteur de carte à puce. C'est le cas notamment des cartes à puce ISO7816 traditionnelles.

Des cartes plus récentes fonctionnant en mode USB peuvent reconstituer le signal d'horloge sur la base des informations transitant sur les contacts D+

et D- de l'USB, sans nécessiter de signal d'horloge explicite.

Un attaquant peut déterminer un signal d'horloge externe (explicite ou implicite dans le cas de l'USB) et l'utiliser afin de faciliter la détermination des instants où se déroulent les étapes de transfert et de relecture.

Mais grâce à ce mode de réalisation, la carte à puce peut générer en interne un signal d'horloge de fréquence fluctuante et difficilement déterminable de l'extérieur qui rend l'attaque bien plus difficile. La carte à puce doit passer en mode horloge externe lorsqu'elle communique avec l'extérieur (lecteur de carte à puce) car sinon la communication est impossible (pas de synchronisation possible) mais peut repasser en mode horloge interne dès la communication terminée.

Selon un mode de réalisation possible, le procédé comprend l'ajout d'un bruit sur l'alimentation électrique des mémoires.

L'alimentation électrique des mémoires est généralement l'alimentation générale du dispositif mettant en œuvre le procédé. Par exemple, dans le cas d'une carte à puce, des contacts VCC et GND fournissent respectivement une tension d'alimentation et un signal de masse, qui permettent d'alimenter l'ensemble du microcontrôleur de la carte à puce, comprenant la ROM éventuelle, l'éventuelle EEPROM, la RAM, le processeur, les circuits d'entrée-sortie, etc.

Ce bruit peut être obtenu par exemple par échantillonnage (à l'aide d'un convertisseur analogique numérique) d'un bruit mesuré sur un composant électronique (par exemple un composant d'un microcontrôleur de carte à puce), puis par traitement du bruit numérique obtenu (par exemple par un crypto-processeur) afin d'accroître son caractère aléatoire (le bruit numérique peut ainsi être chiffré), puis le bruit numérique traité peut être reconverti en bruit analogique par un convertisseur numérique analogique, amplifié, puis superposé à l'alimentation électrique.

Ainsi, une attaque SPA ou des attaques de ce type, visant à reconnaître les opérations de transfert et de relecture sur la base de la consommation

électrique, sont rendues beaucoup plus difficiles.

Selon un mode de réalisation possible, la mémoire non volatile et la mémoire volatile étant toutes deux connectées à un même bus de données, l'empreinte est calculée à la volée par un calculateur d'empreintes de données relié à ce bus de données.

Le calculateur d'empreintes peut être une combinaison d'un processeur tel que le processeur µP de la figure 3 et d'un logiciel approprié. Mais le calculateur d'empreintes peut également être circuit électronique distinct dudit processeur, tel qu'un circuit CALC tel que représenté sur la figure 4. Le circuit CALC peut être un crypto-processeur associé au processeur. Il peut également et avantageusement s'agir d'un calculateur de CRC fonctionnant de façon autonome. Ainsi, un calculateur de CRC peut-il être connecté au bus de données et calculer en continu le CRC de toutes les données transitant sur le bus de données. De nombreux algorithmes de calcul de CRC sont connus, dans l'état de la technique. Un grand nombre d'entre eux est décrit dans « A PAINLESS GUIDE TO CRC ERROR DETECTION ALGORITHMS » version : 3, publié le 19 août 1993, de Ross N. Williams, dont le contenu est incorporé par référence. L'un d'eux est représenté ci-dessous sous forme de pseudo code, et il est parfaitement connu de l'implémenter sous forme matérielle à l'aide d'un circuit électronique.

```
function crc(bit array bitString[1..len], int polynomial) {
  shiftRegister := valeur initiale
// typiquement tous les bits de shiftRegister sont à 0 ou tous à 1
    for i from 1 to len
    {
      if most significant bit of shiftRegister xor bitString[i] = 1
        {
        shiftRegister := (shiftRegister left shift 1) xor polynomial
        } else
        {
```

```
            shiftRegister := (shiftRegister left shift 1)
            }
        }
    return shiftRegister
}
```

Le calculateur peut comprendre un registre d'initialisation, et un registre de lecture de résultat. Ces registres peuvent être adressables par un bus d'adresse associé au bus de données, tel que (par exemple) le bus BUS_A des figures 3 et 4. Le procédé peut ainsi initialiser le calcul de CRC via le registre d'initialisation en y écrivant une valeur particulière, par exemple l'octet zéro. Ceci peut être réalisé en envoyant un octet zéro sur le bus de données, après avoir préalablement indiqué sur le bus d'adresse la valeur de l'adresse du registre d'initialisation.

Il est également possible de décider que toute écriture dans le registre d'initialisation déclenche une initialisation du calcul de CRC (indépendamment de la valeur écrite – que ce soit la valeur zéro ou toute autre valeur).

Après cette étape d'initialisation, toutes les données transitant sur le bus de données sont alors prises en compte pour calculer le CRC.

A tout moment on peut lire le registre de lecture de résultat pour connaître la valeur courante du CRC. Cependant cela peut interrompre le calcul dans la mesure où la valeur du CRC lue est alors elle-même prise en compte dans le calcul du CRC à l'itération suivante, puisque cette valeur passe sur le bus de données.

Selon un mode de réalisation possible, le procédé (par exemple par l'intermédiaire d'un processeur tel que le processeur µP de la figure 4) écrit donc dans le registre d'initialisation juste avant le transfert de données, puis effectue le transfert, puis lit le registre de résultat du calculateur de CRC, en prenant soin de ne pas lire ou écrire de données sur le bus de données entre l'écriture dans le registre d'initialisation et le début du transfert, ni entre la fin du transfert et la lecture du registre de résultat du calculateur de CRC, ni en parallèle avec le transfert de données. La valeur lue dans le registre de

résultat correspond alors au CRC des données transférées.

Le registre d'initialisation et le registre de lecture de résultat peuvent avoir la (les) même(s) adresse(s). Ceci peut être avantageux notamment pour les dispositifs électroniques à mémoire contrainte (tels que de nombreuses cartes à puces), qui ont souvent des capacités d'adressage limitées.

Lorsque le registre d'initialisation et le registre de lecture de résultat sont de tailles différentes, il est possible de décider que la plage d'adresses utilisées pour adresser le plus petit des deux registres est entièrement contenue dans la plage d'adresses utilisées pour adresser le plus grand des deux registres. Ceci est possible car l'un des registres n'est utilisé qu'en écriture (le registre d'initialisation) alors que l'autre (le registre de lecture de résultat) n'est utilisé qu'en lecture. Le fait qu'une même adresse soit utilisée pour deux registres distincts n'est donc pas gênant. Le registre de lecture de résultat peut être plus grand que le registre d'initialisation, par exemple le registre de lecture de résultat peut avoir une taille de 32 bits alors que le registre d'initialisation peut avoir une taille de 8 bits.

Selon les modes de réalisation du calculateur de CRC précédemment décrits, le calculateur de CRC opère en continu, même lorsque le calcul du CRC des données transitant sur le bus de données ne présente pas d'intérêt. En particulier, après avoir calculé le CRC désiré lors d'un transfert, le calculateur de CRC continue à calculer le CRC global des données ayant transité sur le bus, à savoir les données transférées, concaténées à la valeur du CRC lue depuis le registre de résultat, concaténée à toutes les valeurs ultérieurement passées sur le bus de données.

Cependant, selon un autre mode de réalisation, il est possible d'utiliser un calculateur de CRC plus perfectionné acceptant des ordres plus élaborés. Par exemple le procédé peut écrire une valeur 0x00 (l'octet zéro) dans le registre d'initialisation pour réinitialiser la valeur du CRC calculé (recommencer un calcul neuf), comme précédemment indiqué, mais également écrire une autre valeur, par exemple la valeur 0x01, pour interrompre un calcul de CRC et permettre ainsi d'effectuer des échanges sur le bus de données sans que cela n'affecte le calcul de CRC en cours, puis écrire une autre valeur, par exemple

la valeur 0x02, dans le registre d'initialisation afin de reprendre le calcul du CRC là où il s'était arrêté. Ainsi, le calcul du CRC prend en compte les données passées sur le bus de données entre l'écriture 0x00 et l'écriture 0x01, puis entre l'écriture 0x02 et la lecture du résultat, mais pas entre l'écriture 0x01 et l'écriture 0x02. Dans ce cas, le calculateur de CRC doit être modifié pour ignorer (dans le calcul du CRC) les valeurs passant sur le bus de données lorsque que ces valeurs sont associées à une valeur du bus d'adresse correspondant à la valeur de l'adresse (ou à une valeur de la plage d'adresses) du registre d'initialisation (en l'occurrence les valeurs 0x01 et 0x02 seraient des valeurs parasites que l'on peut ignorer dans le calcul du CRC). On peut également ignorer les valeurs lues à une adresse correspondant à l'adresse (ou à une adresse de la plage d'adresses) du registre de lecture de résultat, lors que cette adresse (respectivement cette plage d'adresse) est différente de celle du registre d'initialisation. Bien entendu, des valeurs autres que les valeurs 0x00 , 0x01 et 0x02 indiquées ci-dessus sont possibles (n'importe quelle valeur d'octet est possible, le choix étant arbitraire). Il est également possible d'utiliser un registre d'initialisation de taille différente de 8 bits et un registre de lecture de résultat de taille différente de 32 bits.

Un avantage d'un calculateur d'empreinte connecté au bus de données (tel que le calculateur de CRC ci-dessus) est qu'il peut calculer des CRC sur des transferts de données de longueur arbitraire sans nécessiter d'utiliser un composant mémoire autre (c'est-à-dire sans nécessiter de transmissions supplémentaires sur le bus de données, de telles transmissions pouvant faire l'objet d'attaques, à l'exception de la transmission du CRC lui-même) et sans solliciter le processeur µP autrement qu'en laissant par exemple le soin au processeur µP de piloter le calculateur de CRC à l'aide des différentes commandes (0x00, 0x01, 0x02, ...).

Selon un mode de réalisation possible, le calculateur d'empreintes dispose d'un registre de stockage pour mémoriser une empreinte (par exemple un CRC) venant d'être calculé. Par exemple, au lieu de lire le registre de résultat afin d'obtenir la première empreinte, il est possible de prévoir de lancer

une commande au calculateur d'empreinte afin qu'il mémorise l'empreinte courante. Ainsi, dans le cas des calculateurs de CRC précédemment décrits, il est possible d'écrire une valeur (par exemple la valeur 0x03) dans le registre d'initialisation, afin d'ordonner au calculateur de CRC de stocker le CRC en interne (dans un registre de stockage du calculateur de CRC), sans que la valeur du CRC n'ait à transiter sur le bus de données (puisqu'elle ne quitte pas le calculateur de CRC), et ainsi sans que cette valeur de CRC ne puisse être sujette à une attaque sur le bus de données.

On peut également prévoir une autre commande pour comparer une empreinte (par exemple le CRC courant) à une empreinte précédemment stockée (par exemple le CRC mémorisé lors de l'envoi de la commande 0x03). Ainsi, le procédé peut, après relecture des données transmises (et calcul du CRC de ces données) écrire une valeur (par exemple la valeur 0x04) dans le registre d'initialisation, afin de déclencher une comparaison du CRC courant avec le CRC précédemment stocké. Il est entendu que les valeurs 0x03 et 0x04 transitant sur le bus de données ne sont pas prises en compte dans les calculs de CRC (comme expliqué précédemment pour les valeurs 0x01 et 0x02). Ainsi la commande « stockage de CRC » peut être codée par l'octet 0x03 et la commande « comparaison de CRC » peut être codée par l'octet 0x04, mais toutes autres valeurs d'octets seraient possibles (dans la mesure où une valeur unique est associée à chaque commande).

Selon une amélioration de ce mode de réalisation, le procédé paramètre les valeurs utilisées pour lancer des commandes au calculateur d'empreinte via le registre d'initialisation. Ainsi, au lieu d'utiliser les valeurs 0x00, 0x01, 0x02, 0x03 et 0x04 précédemment indiquées, il est possible de définir conventionnellement cinq valeurs aléatoires distinctes, qui peuvent être modifiées à intervalle régulier. Elles peuvent par exemple être modifiées à chaque utilisation, ou toutes les N utilisations où N est un nombre entier prédéfini, ou tous les J jours, où J est un nombre entier prédéfini (en admettant que le dispositif mettant en œuvre le procédé dispose d'une base de temps et donc de date, ce qui n'est pas nécessairement le cas). Ceci est avantageux car cela rend plus complexe le repérage de telles commandes par un

attaquant, et rend également plus complexe la simulation de telles commandes par un attaquant.

Selon un mode de réalisation possible mettant en œuvre un calculateur d'empreinte (tel que le calculateur de CRC des modes de réalisations précédents) connecté au bus de données, le calcul de la première empreinte est effectué sur les données écrites dans la mémoire cible du transfert lors de l'étape de transfert.

Ainsi, lorsqu'un processeur, contrôleur, ou autre dispositif effectue un transfert en effectuant deux transactions sur le bus de données (telles que les transactions T1 et T2 représentées sur les figures 1 et 2), on protège la chaîne complète de la transaction.

Par exemple, un processeur peut tout d'abord lire une donnée de la mémoire source vers un de ses registres, puis écrire la valeur de ce registre vers la mémoire cible. La donnée transférée circule alors deux fois sur le bus de données, et peut être attaquée à deux reprises. Un calcul d'empreinte sur la lecture de la mémoire source vers le registre de processeur ne garantit pas l'absence d'attaque durant l'étape ultérieure d'écriture du registre vers la mémoire cible. En revanche un calcul d'empreinte durant l'étape d'écriture du registre vers la mémoire cible permet de détecter l'attaque, même s'il ne permet pas de dire si l'attaque a été effectuée lors du premier ou du second passage sur le bus de données.

Ainsi, le calcul d'empreinte par une calculateur d'empreinte connecté au bus de données est-il avantageux par rapport à un calcul d'empreinte par une entité telle qu'un processeur pilotant un transfert de données dont on veut calculer l'empreinte, et ce même si le processeur (ou autre entité) délègue le calcul de l'empreinte à un matériel dédié (crypto-processeur ou autre) sur la base de la donnée reçue de la mémoire source.

En effet, alors que le processeur (ou autre entité jouant un rôle de gestionnaire de transfert, tel que contrôleur mémoire) ne manipule qu'une seule donnée (par exemple la donnée d'un registre qui reçoit la donnée

transférée depuis la mémoire source puis la transfère à la mémoire cible), le calculateur d'empreinte connecté au bus de données voit passer deux données potentiellement distinctes (en raison par exemple d'une attaque sur le bus).

Selon un mode de réalisation dont un exemple particulier est illustré sur la figure 2, dans lequel un processeur µP calcule la première empreinte, il est possible d'effectuer le transfert en entier (lecture T1 de la mémoire source puis écriture T2 vers la mémoire cible) puis de lire la donnée écrite dans la mémoire cible (troisième passage R1 sur le bus de données) et de calculer la première empreinte sur la base de cette donnée lue (R1) dans la mémoire cible afin de détecter une attaque aussi bien durant l'étape de lecture T1 que durant l'étape d'écriture T2, en partant du principe qu'il serait très difficile pour un attaquant d'effectuer lors de la lecture R1 de la mémoire cible une attaque inverse de celle qu'il aurait effectuée lors de l'écriture T2 de la mémoire cible (c'est-à-dire lors de la deuxième étape du transfert) afin de dissimuler cette attaque au processeur µP.

Un mode de réalisation se rapporte à un programme d'ordinateur comprenant une suite d'instructions mettant en œuvre les étapes du procédé selon l'une des revendications précédentes lorsqu'elles sont exécutées par un processeur. Il peut s'agir notamment d'un programme écrit en langage assembleur exécutable par une processeur tel qu'un processeur de carte à puce. Il peut également s'agir d'un programme écrit en langage de haut niveau (de type C, C++, C#, ou java, Visual Basic, etc.) à l'exception éventuellement de certaines parties (par exemple les parties touchant au matériel) qui peuvent être écrites en assembleur.

Un mode de réalisation concerne un support de stockage non transitoire lisible par ordinateur comprenant un programme d'ordinateur selon un mode de réalisation de l'invention. Le support de stockage peut être notamment une mémoire non volatile (EEPROM, ROM, Flash, etc.) qui peut se trouver dans

une carte à puce, une clé USB, un token, etc.

Un mode de réalisation concerne un dispositif électronique sécurisé comprenant une mémoire non volatile (par exemple une ROM, une EEPROM ou une Flash) ainsi qu'une mémoire volatile (par exemple une RAM).

Le dispositif électronique comprend un circuit de transfert de données entre la mémoire non volatile et la mémoire volatile. Il peut s'agir d'un processeur µP combiné à un logiciel adapté (réalisant par exemple une opération de lecture de la mémoire source suivie d'une écriture dans la mémoire cible), ou d'un circuit électronique dédié, par exemple un FPGA ou toute forme de logique câblée réalisant une interface entre la mémoire volatile et la mémoire non volatile. Il peut également s'agir d'un contrôleur de mémoire (par exemple un contrôleur de mémoire Flash) adapté pour prendre la main sur un bus de données le reliant à l'autre mémoire (par exemple une mémoire RAM), en utilisant par exemple un technique de type DMA. Un tel contrôleur CTRL est représenté sur la figure 3.

Le dispositif électronique comprend un calculateur d'empreintes de données. Il peut s'agir d'un processeur µP combiné à un logiciel adapté (réalisant par exemple un calcul de CRC ou d'autre type d'empreinte). Il peut également s'agir d'un circuit électronique dédié, par exemple un FPGA ou toute forme de logique câblée réalisant un calcul d'empreinte. Un tel calculateur CALC est représenté sur la figure 4. Le calculateur d'empreintes peut également comprendre une combinaison d'un circuit dédié et d'un processeur associé à un logiciel.

Le dispositif électronique comprend un circuit de protection agencé pour valider un transfert de données du circuit de transfert de données lorsqu'une première empreinte qui est calculée postérieurement à l'initiation du transfert des données sur les données transférées correspond à une deuxième empreinte qui est calculée postérieurement à l'initiation d'une relecture desdites données de la mémoire source du transfert sur les données relues. Le circuit de protection peut être un processeur µP combiné à un logiciel

adapté (réalisant par exemple une comparaison des empreintes calculées et prenant ou non, selon la comparaison, une décision de validation de transfert). Le circuit de protection peut également être un circuit électronique dédié, par exemple un FPGA ou toute forme de logique câblée réalisant une comparaison d'empreinte. Il peut s'agit notamment du calculateur CALC représenté sur la figure 4, qui peut comprendre une fonction de comparaison. Le circuit de protection peut également comprendre une combinaison d'un circuit dédié et d'un processeur associé à un logiciel.

Les figures 3 et 4 illustrent deux possibilités d'architecture pour de tels dispositifs électroniques sécurisés. L'EEPROM de la figure 4 pourrait être toute autre mémoire non volatile (Flash, ROM, etc.), de même que la Flash de la figure 3 pourrait être toute autre mémoire non volatile (EEPROM, ROM, etc.).

Le dispositif peut être par exemple une carte à puce, un HSM (Hardware Security Module, à savoir un équipement capable notamment de stocker des clés cryptographiques de manière sécurisée), un accélérateur SSL ou TLS, ou encore un serveur manipulant des données sensible.

Selon un mode de réalisation possible, le dispositif comprend un générateur de nombres aléatoires, le dispositif étant agencé pour insérer, entre un transfert de données et la relecture desdites données, une attente dont la durée est fonction d'un nombre aléatoire généré par le générateur de nombres aléatoires.

Selon un mode de réalisation possible, le dispositif comprend un générateur de nombres aléatoires, le dispositif étant agencé pour effectuer un transfert de données soit avant soit après la relecture desdites données en fonction de la valeur d'un nombre aléatoire généré par le générateur de nombres aléatoires.

Selon un mode de réalisation possible, le dispositif est agencé pour insérer une lecture de données entre un transfert de données et la relecture desdits données transférées.

Selon un mode de réalisation possible, le dispositif comprend un bus de données (tel que le bus BUS_D représenté sur les figures 3 et 4) reliant la mémoire non volatile et la mémoire volatile, le dispositif étant agencé pour fixer la fréquence de fonctionnement du bus de données de manière aléatoire.

Selon un mode de réalisation possible, le dispositif est agencé pour ajouter un bruit sur l'alimentation électrique des mémoires (qui peut être l'alimentation électrique commune à tout le dispositif).

Selon un mode de réalisation possible, le dispositif comprend un bus de données (tel que le bus BUS_D représenté sur les figures 3 et 4) reliant la mémoire non volatile et la mémoire volatile, le calculateur d'empreintes de données étant relié à ce bus de données et étant agencé pour calculer les empreintes à la volée à partir des données circulant sur le bus de données.

L'invention ne se limite pas aux formes de réalisation décrites ci-avant à titre d'exemple ; elle s'étend à d'autres variantes.

Ainsi, il a été décrit ci-avant un dispositif pouvant être une carte à puce, notamment une carte bancaire. Cependant, l'invention s'applique à toute autre carte à puce (cartes à puces de santé, cartes d'identité électronique, cartes d'accès, cartes SIM, etc.) ainsi qu'à tout dispositif de sécurité tel qu'un passeport électronique, un chronotachygraphe électronique, un tag RFID, ou une clé USB, un token d'identification, etc.

Les mémoires utilisables ne se limitent pas aux exemples donnés à titre purement illustratif mais couvrent tout type de mémoire.

Le procédé de sécurisation peut évidemment être combiné avec d'autres

procédés de sécurisation et/ou à des protections matérielles afin d'accroître la protection.

Les modes de réalisations décrits en relation avec le procédé selon l'invention peuvent être transposés aux dispositifs, ainsi qu'aux programmes d'ordinateur et aux supports de stockage du programme selon l'invention, et réciproquement.

REVENDICATIONS

1. Procédé de sécurisation de transfert de données, comprenant:
- un transfert de données (T1, T2) entre une mémoire non volatile (EEPROM) et une mémoire volatile (RAM),
- un calcul, postérieurement à l'initiation dudit transfert, d'une première empreinte desdites données,
- une relecture (R, R2) desdites données à partir de la mémoire source (EEPROM) du transfert,
- un calcul, postérieurement à l'initiation de ladite relecture (R, R2), d'une deuxième empreinte des données relues, et
- une validation du transfert (T1, T2) si les deux empreintes correspondent.

2. Procédé selon la revendication 1, comprenant une génération de nombre aléatoire, le procédé insérant entre le transfert (T1, T2) et la relecture (R) une attente (ATT) dont la durée est fonction de ce nombre aléatoire.

3. Procédé selon la revendication 1 ou 2, comprenant une génération de nombre aléatoire, le procédé effectuant le transfert (T1, T2) soit avant soit après la relecture (R, R2) en fonction de la valeur du nombre aléatoire généré.

4. Procédé selon l'une des revendications précédentes, comprenant l'insertion d'une lecture de données (L) entre le transfert (T1, T2) et la relecture (R).

5. Procédé selon l'une des revendications précédentes, dans lequel, la mémoire non volatile (EEPROM) et la mémoire volatile (RAM) étant toutes deux connectées à un même bus de données (BUS_D), le procédé fixe la fréquence de fonctionnement du bus de données (BUS_D) de manière aléatoire.

6. Procédé selon l'une des revendications précédentes, comprenant l'ajout d'un bruit sur l'alimentation électrique des mémoires (RAM, EEPROM).

7. Procédé selon l'une des revendications précédentes, dans lequel, la mémoire non volatile (EEPROM) et la mémoire volatile (RAM) étant toutes deux connectées à un même bus de données (BUS_D), l'empreinte est calculée à la volée par un calculateur d'empreintes de données relié à ce bus de données (BUS_D).

8. Procédé selon la revendication 7, dans lequel le calcul de la première empreinte est effectué sur les données écrites dans la mémoire cible (RAM) du transfert lors de l'étape de transfert (T1, T2).

9. Programme d'ordinateur comprenant une suite d'instructions mettant en œuvre les étapes du procédé selon l'une des revendications précédentes lorsqu'elles sont exécutées par un processeur (µP).

10. Support de stockage non transitoire lisible par ordinateur comprenant un programme d'ordinateur selon la revendication 9.

11. Dispositif électronique sécurisé comprenant une mémoire non volatile (EEPROM, FLASH) ainsi qu'une mémoire volatile (RAM), un circuit de transfert de données (µP) entre la mémoire non volatile (EEPROM, FLASH) et la mémoire volatile (RAM), un calculateur d'empreintes de données (CALC), et un circuit de protection (µP) agencé pour valider un transfert de données du circuit de transfert de données (µP) lorsqu'une première empreinte qui est calculée postérieurement à l'initiation du transfert des données sur les données transférées correspond à une deuxième empreinte qui est calculée postérieurement à l'initiation d'une relecture desdites données de la mémoire source (EEPROM, FLASH, RAM) du transfert sur les données relues.

12. Dispositif selon la revendication 11, comprenant un générateur de nombres aléatoires, le dispositif étant agencé pour insérer, entre un transfert de données et la relecture desdites données, une attente dont la durée est fonction d'un nombre aléatoire généré par le générateur de nombres aléatoires.

13. Dispositif selon la revendication 11 ou 12, comprenant un générateur de nombres aléatoires, le dispositif étant agencé pour effectuer un transfert de données soit avant soit après la relecture desdites données en fonction de la valeur d'un nombre aléatoire généré par le générateur de nombres aléatoires.

14. Dispositif selon l'une des revendications 11 à 13, le dispositif étant agencé pour insérer une lecture de données entre un transfert de données et la relecture desdits données transférées.

15. Dispositif selon l'une des revendications 11 à 14, comprenant un bus de données (BUS_D) reliant la mémoire non volatile (EEPROM, FLASH) et la mémoire volatile (RAM), le dispositif étant agencé pour fixer la fréquence de fonctionnement du bus de données (BUS_D) de manière aléatoire.

16. Dispositif selon l'une des revendications 11 à 15, le dispositif étant agencé pour ajouter un bruit sur l'alimentation électrique des mémoires (RAM, EEPROM, FLASH).

17. Dispositif selon l'une des revendications 11 à 16, comprenant un bus de données (BUS_D) reliant la mémoire non volatile (EEPROM, FLASH) et la mémoire volatile (RAM), le calculateur d'empreintes de données (CALC) étant relié à ce bus de données (BUS_D) et étant agencé pour calculer les empreintes à la volée à partir des données circulant sur le bus de données (BUS_D).

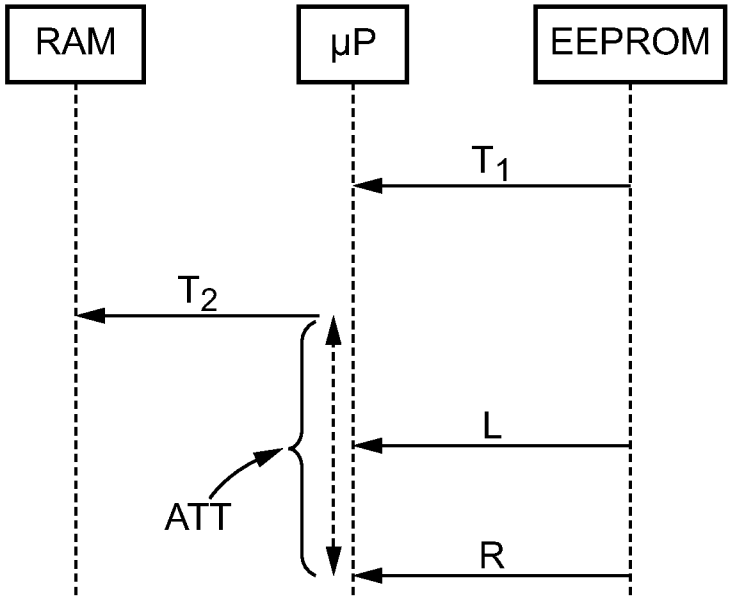

FIG. 1

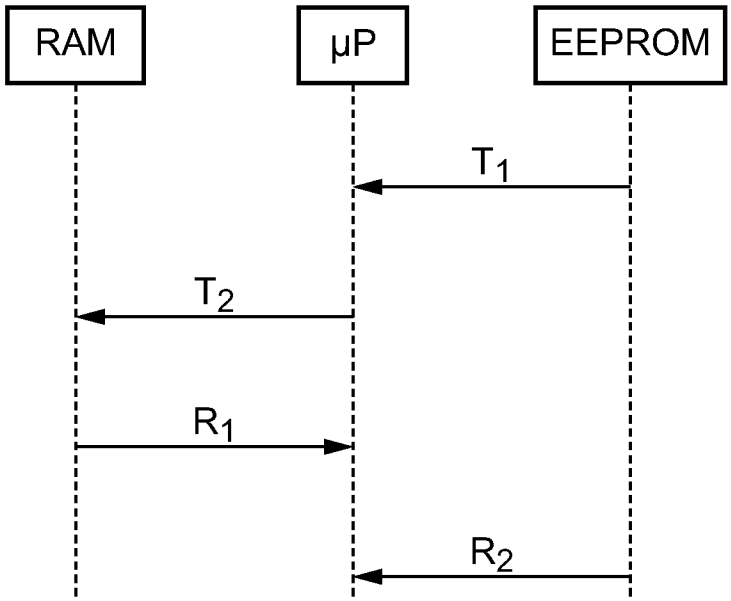

FIG. 2

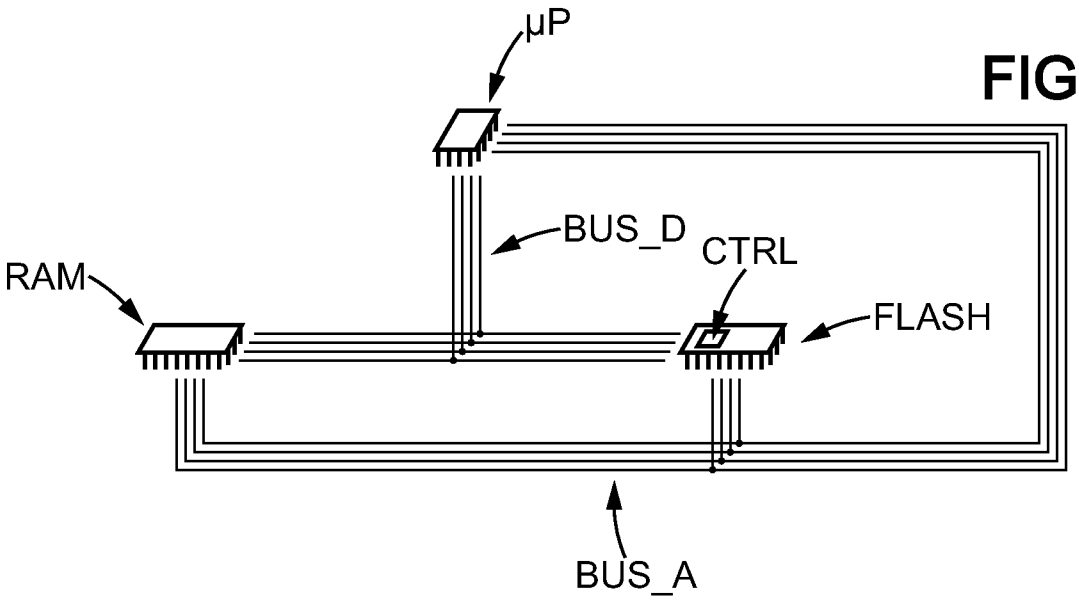

FIG. 3

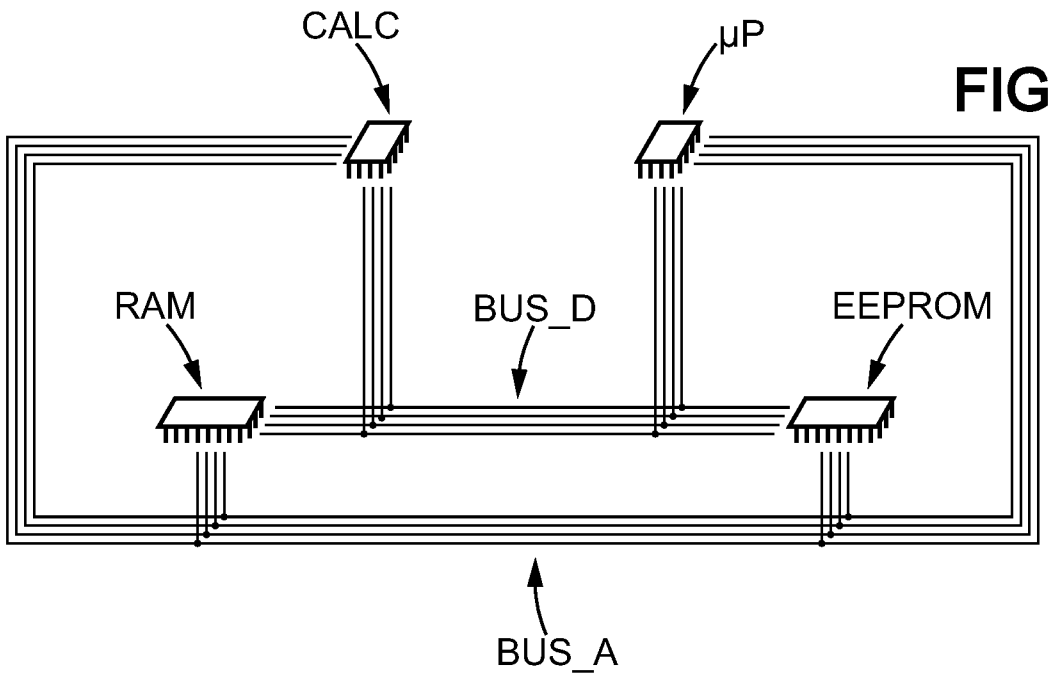

FIG. 4

INTERNATIONAL SEARCH REPORT

International application No
PCT/FR2012/051304

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/24
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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2007/168793 A1 (SEO KYUNG-DUCK [KR]) 19 July 2007 (2007-07-19)	1,2, 6-12,16, 17
Y	paragraph [0028]; figure 4 paragraph [0033] - paragraph [0037]; figure 5 paragraph [0020]; figure 2	3-5, 13-15

-/--

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier 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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
15 August 2012	28/08/2012

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	Vinck, Bart

INTERNATIONAL SEARCH REPORT

International application No
PCT/FR2012/051304

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	MICHAEL TUNSTALL ET AL: "Efficient Use of Random Delays in Embedded Software", 9 May 2007 (2007-05-09), INFORMATION SECURITY THEORY AND PRACTICES. SMART CARDS, MOBILE AND UBIQUITOUS COMPUTING SYSTEMS; [LECTURE NOTES IN COMPUTER SCIENCE;;LNCS], SPRINGER BERLIN HEIDELBERG, BERLIN, HEIDELBERG, PAGE(S) 27 - 38, XP019079371, ISBN: 978-3-540-72353-0 abstract Introduction -----	2,12
Y	HAGAI BAR-EL ET AL: "The Sorcerer's Apprentice Guide to Fault Attacks", INTERNET CITATION, 7 May 2004 (2004-05-07), XP002329915, Retrieved from the Internet: URL:http://web.archive.org/web/20041016071838/eprint.iacr.org/2004/100 [retrieved on 2005-05-27] abstract Section 1 Section 5.1.2 Section 5.2 -----	3-5, 13-15

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/FR2012/051304

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2007168793 A1	19-07-2007	JP 2007183937 A	19-07-2007
		KR 20070074308 A	12-07-2007
		US 2007168793 A1	19-07-2007

RAPPORT DE RECHERCHE INTERNATIONALE

Demande internationale n°
PCT/FR2012/051304

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
INV. G06F21/24
ADD.

Selon la classification internationale des brevets (CIB) ou à la fois selon la classification nationale et la CIB

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultée (système de classification suivi des symboles de classement)
G06F

Documentation consultée autre que la documentation minimale dans la mesure où ces documents relèvent des domaines sur lesquels a porté la recherche

Base de données électronique consultée au cours de la recherche internationale (nom de la base de données, et si cela est réalisable, termes de recherche utilisés)
EPO-Internal

C. DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
X	US 2007/168793 A1 (SEO KYUNG-DUCK [KR]) 19 juillet 2007 (2007-07-19)	1,2, 6-12,16, 17
Y	alinéa [0028]; figure 4 alinéa [0033] - alinéa [0037]; figure 5 alinéa [0020]; figure 2	3-5, 13-15
	----- -/--	

[X] Voir la suite du cadre C pour la fin de la liste des documents

[X] Les documents de familles de brevets sont indiqués en annexe

* Catégories spéciales de documents cités:

"A" document définissant l'état général de la technique, non considéré comme particulièrement pertinent

"E" document antérieur, mais publié à la date de dépôt international ou après cette date

"L" document pouvant jeter un doute sur une revendication de priorité ou cité pour déterminer la date de publication d'une autre citation ou pour une raison spéciale (telle qu'indiquée)

"O" document se référant à une divulgation orale, à un usage, à une exposition ou tous autres moyens

"P" document publié avant la date de dépôt international, mais postérieurement à la date de priorité revendiquée

"T" document ultérieur publié après la date de dépôt international ou la date de priorité et n'appartenant pas à l'état de la technique pertinent, mais cité pour comprendre le principe ou la théorie constituant la base de l'invention

"X" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme nouvelle ou comme impliquant une activité inventive par rapport au document considéré isolément

"Y" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme impliquant une activité inventive lorsque le document est associé à un ou plusieurs autres documents de même nature, cette combinaison étant évidente pour une personne du métier

"&" document qui fait partie de la même famille de brevets

Date à laquelle la recherche internationale a été effectivement achevée	Date d'expédition du présent rapport de recherche internationale
15 août 2012	28/08/2012

Nom et adresse postale de l'administration chargée de la recherche internationale	Fonctionnaire autorisé
Office Européen des Brevets, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Vinck, Bart

RAPPORT DE RECHERCHE INTERNATIONALE

Demande internationale n°
PCT/FR2012/051304

C(suite). DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie*	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
A	MICHAEL TUNSTALL ET AL: "Efficient Use of Random Delays in Embedded Software", 9 mai 2007 (2007-05-09), INFORMATION SECURITY THEORY AND PRACTICES. SMART CARDS, MOBILE AND UBIQUITOUS COMPUTING SYSTEMS; [LECTURE NOTES IN COMPUTER SCIENCE;;LNCS], SPRINGER BERLIN HEIDELBERG, BERLIN, HEIDELBERG, PAGE(S) 27 - 38, XP019079371, ISBN: 978-3-540-72353-0 abrégé Introduction	2,12

Y	HAGAI BAR-EL ET AL: "The Sorcerer's Apprentice Guide to Fault Attacks", INTERNET CITATION, 7 mai 2004 (2004-05-07), XP002329915, Extrait de l'Internet: URL:http://web.archive.org/web/20041016071838/eprint.iacr.org/2004/100 [extrait le 2005-05-27] abrégé Section 1 Section 5.1.2 Section 5.2	3-5, 13-15

RAPPORT DE RECHERCHE INTERNATIONALE

Renseignements relatifs aux membres de familles de brevets

Demande internationale n°
PCT/FR2012/051304

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
US 2007168793 A1	19-07-2007	JP 2007183937 A	19-07-2007
		KR 20070074308 A	12-07-2007
		US 2007168793 A1	19-07-2007