(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) **N° de publication :** **2 964 480**
(à n'utiliser que pour les commandes de reproduction)

(21) **N° d'enregistrement national :** **10 57084**

(51) Int Cl[8] : ***G 06 F 9/44*** *(2006.01), G 06 K 19/073*

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 07.09.10.

(30) **Priorité :**

(43) **Date de mise à la disposition du public de la demande :** 09.03.12 Bulletin 12/10.

(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 :**

(71) **Demandeur(s) :** *OBERTHUR TECHNOLOGIES Société anonyme* — FR.

(72) **Inventeur(s) :** CHAMBEROT FRANCIS et MARTIN MARTINASSO LUDOVIC.

(73) **Titulaire(s) :** OBERTHUR TECHNOLOGIES Société anonyme.

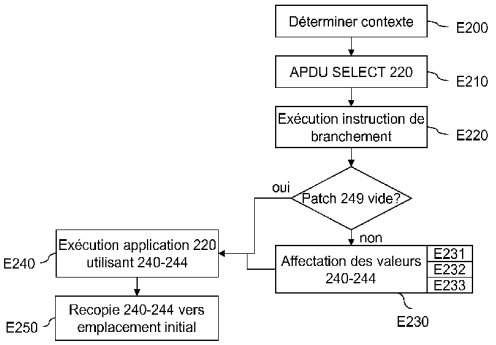
(74) **Mandataire(s) :** SANTARELLI.

(54) PROCEDE D'EXECUTION D'UNE APPLICATION DANS UN DISPOSITIF DE TRAITEMENT TEL QU'UNE CARTE A PUCE, ET DISPOSITIF DE TRAITEMENT ASSOCIE.

(57) La présente invention concerne l'exécution d'une application informatique (220) dans un dispositif de traitement tel qu'une carte à puce (1), comprenant un environnement logiciel (221) d'exécution de l'application.

Pour cette exécution, l'application information et/ou l'environnement logiciel comprend au moins une instruction de branchement pour appeler un patch logiciel autonome externe, et l'application informatique s'exécute en utilisant une donnée stockée (240-243) en mémoire (24) et fonction d'un contexte d'exécution courant (C0, C1 C2).

Selon l'invention, la donnée stockée fonction du contexte d'exécution courant est déterminée (E232), préalablement à son utilisation (E240) par l'application (220), par l'exécution d'un patch logiciel autonome (249) externe à ladite application et déclenchée au travers de ladite instruction de branchement.



FR 2 964 480 - A1

La présente invention concerne l'exécution d'une application informatique dans un dispositif de traitement tel qu'une carte à puce, et plus particulièrement un procédé d'exécution d'une application informatique dont l'exécution utilise une donnée stockée en mémoire qui est fonction d'un contexte d'exécution courant, et un dispositif de traitement associé.

Il est classique qu'une application informatique, notamment mémorisée en mémoire ROM (pour "*Read-Only Memory*") dans une carte à puce, tienne compte de certains paramètres définissant un contexte d'exécution courant pour choisir telle ou telle donnée à utiliser lors de son exécution. On parle alors de "*donnée contextuelle*".

C'est par exemple le cas d'une application de transaction financière, couramment utilisée dans les cartes à puce.

Dans ce cadre, une carte à puce peut être utilisée dans un terminal de paiement réalisant une communication avec contact pour réaliser une transaction financière à l'aide de cette application. Cette transaction est fortement sécurisée du fait de la saisie d'un numéro d'identification personnel (ou PIN pour "*Personal Identification Number*") et du fait d'échanges entre le terminal et un serveur distant de sécurité. Des transactions d'un montant élevé peuvent être ainsi réalisées.

La même carte à puce peut être dotée de moyens de communication sans contact (généralement connue sous l'appellation de carte "*contactless*") dans une utilisation de la même application informatique pour des transactions de plus faibles montants, ne requérant pas de connexion en ligne avec un serveur central, comme par exemple une fonctionnalité de porte-monnaie électronique. Le porte-monnaie électronique contient des unités de paiement préchargées. La sécurité des transactions électroniques de faibles montants est toutefois moindre, notamment en ce qu'il n'est généralement pas requis la saisie d'un PIN, ni l'utilisation d'un serveur distant de sécurité.

Le plafond de paiement pour la fonction de porte-monnaie électronique est moindre que celui pour la fonction de paiement à l'aide d'un terminal avec contact. Ces plafonds sont stockés en mémoire dans la carte à puce.

Aujourd'hui, la prise en compte d'un plafond variable selon l'utilisation porte-monnaie ou transaction avec contact (ou de toute autre donnée selon un contexte d'exécution variable) est intégrée directement dans le code de l'application (c'est-à-dire "en dur" dans le code), aux nombreux endroits idoines, par exemple à l'aide d'instructions conditionnelles renvoyant vers l'une ou l'autre des données en mémoire.

Cela pose un problème, notamment lorsque l'on souhaite faire évoluer l'application informatique, soit parce qu'initialement elle ne tient compte que d'un seul contexte d'exécution, soit parce que l'on souhaite ajouter un nouveau contexte d'exécution. En effet, dans ce cas, il est nécessaire d'intervenir directement dans le code à de nombreux endroits afin d'introduire les instructions conditionnelles adéquates et les renvois correspondants vers la mémoire.

Or ces modifications ne sont pas sans effet, notamment en ce qu'elles peuvent nécessiter la certification (selon les "*critères communs*" par exemple) à nouveau de l'ensemble de l'application informatique mémorisée en mémoire ROM. Une telle certification requiert toutefois plusieurs semaines de délais et engendre des dépenses non négligeables.

La présente invention vise à pallier ces inconvénients en réduisant voire supprimant la modification du code de l'application informatique et donc d'éventuelles nouvelles certifications onéreuses et longues pour tenir compte d'un contexte d'exécution nouveau ou changeant.

A cet effet, l'invention concerne notamment un procédé d'exécution d'une application informatique dans un dispositif de traitement comprenant un environnement logiciel d'exécution de l'application, ladite application informatique et/ou l'environnement logiciel comprenant au moins une instruction de branchement pour appeler un patch logiciel autonome externe, et ladite application informatique s'exécutant en utilisant au moins une donnée stockée

en mémoire, cette donnée utilisée étant fonction d'un contexte d'exécution courant,

procédé dans lequel la donnée stockée fonction du contexte d'exécution courant est déterminée, préalablement à son utilisation par l'application, par l'exécution d'un patch logiciel autonome externe à ladite application et déclenchée au travers de ladite instruction de branchement.

L'invention permet de s'affranchir d'une modification du code de l'application informatique (ou de se limiter à une unique modification) lorsque l'on souhaite obtenir une telle application informatique dont l'exécution tient compte d'un contexte d'exécution courant. Cet impact inexistant ou minime sur le code de l'application est en effet obtenu par l'utilisation d'un patch logiciel (section de code dont le but initial est d'apporter des correctifs/modifications à l'application) qui est autonome (en ce qu'il n'est pas intégré à l'application elle-même, mais vient se greffer sur elle – donc externe à l'application) et qui permet de déterminer la donnée contextuelle à utiliser par l'application.

Ainsi, les instructions conditionnelles de l'état de l'art peuvent être évitées, et du coup les nombreuses interventions tout au long du code. Seule une nouvelle instruction de branchement dans le code peut s'avérer nécessaire, ce qui, d'ailleurs, limite une nouvelle certification à une partie seulement de l'application et non à l'entièreté de celle-ci.

En outre, il est classique qu'à la conception initiale, l'application informatique comporte de nombreuses instructions de branchement vers d'hypothétiques patchs. Les patchs ne sont alors développés qu'au fur et à mesure des besoins, certaines instructions de branchement aboutissant du coup sur des patchs "vides". Dans ce cas, ladite instruction de branchement est une instruction de branchement prévue à la conception de l'application informatique et/ou de l'environnement logiciel pour appeler un hypothétique patch logiciel autonome externe non encore défini. Et la présente invention évite toute intervention directe dans le code de l'application ou du système d'exploitation car il est possible d'utiliser une telle instruction de branchement prévue initialement pour déclencher l'exécution du patch autonome selon l'invention.

En détournant l'utilisation classique des patchs logiciels correctifs pour en faire des outils de détermination et de sélection de données contextuelles utilisées par l'application logicielle, l'invention réduit drastiquement les interventions sur ces dernières, et du coup les coûts et délais de nouvelle certification, le cas échéant.

On notera que l'"*environnement logiciel d'exécution de l'application*" peut être constitué d'un ensemble de logiciels qui sont en cours d'exécution lorsque ladite application est exécutée. A titre principal, il s'agit notamment du système d'exploitation prévu dans le dispositif de traitement, type carte à puce.

L'invention s'applique à l'utilisation d'une ou plusieurs données spécifiques à un contexte d'exécution.

Dans un mode de réalisation, l'application informatique accède une zone mémoire principale pour obtenir ladite donnée à utiliser, et ledit patch logiciel renseigne, dans ladite zone mémoire principale, la donnée déterminée en fonction du contexte d'exécution courant.

Dans cette configuration, l'application informatique accède toujours à la même zone mémoire pour récupérer la donnée dont elle à besoin. Les opérations réalisées par le patch logiciel aux fins d'indiquer dans cette zone mémoire la donnée contextuelle qui convient sont donc totalement transparentes pour l'application. Cela garantit l'absence de modification de l'application pour introduire le choix de données contextuelles.

En particulier, le patch logiciel copie la donnée déterminée depuis une zone mémoire secondaire vers la zone mémoire principale. Cette configuration est simple à mettre en œuvre. On prévoira d'ailleurs éventuellement une recopie du contenu de la zone mémoire principale vers la zone mémoire secondaire à l'issue de l'exécution de l'application. Cela est valable notamment lorsque ces données contextuelles sont susceptibles d'évoluer au cours de l'exécution de l'application (par exemple un compteur du montant cumulé des transactions depuis la dernière transaction réalisée avec connexion à un serveur central). En particulier, il peut être prévu que la donnée déterminée en fonction du contexte d'exécution courant est renseignée par une écriture en mémoire non volatile réinscriptible.

En variante, le patch logiciel associe à la zone mémoire principale un pointeur vers une zone mémoire secondaire stockant la donnée déterminée. Cette disposition présente l'avantage que la zone mémoire principale (ou les zones mémoires principales dans le cas où l'on met en œuvre l'invention pour plusieurs données contextuelles simultanées) peuvent être de taille réduite. En effet, un pointeur (par exemple l'indication d'une autre adresse mémoire) est généralement compact: par exemple sur deux octets là où une donnée, type compteur de valeurs, est prévue sur six octets plus six octets de somme de contrôle (en anglais "*checksum*").

En outre, l'usage de pointeurs permet d'éviter l'étape de recopie du contenu de la zone mémoire principale vers la zone mémoire secondaire en cas d'évolution du contenu des données contextuelles au cours de l'exécution de l'application.

En particulier, la zone mémoire principale comprend des données par défaut d'un contexte par défaut, un champ binaire de partage et une adresse de partage,

et on renseigne, dans la zone mémoire principale, au moins une donnée déterminée différente des données par défaut, en utilisant le champ binaire de partage pour indiquer quelles données par défaut sont remplacées par l'au moins une donnée déterminée et en utilisant l'adresse de partage pour pointer sur l'emplacement mémoire stockant l'au moins une donnée déterminée.

Selon une caractéristique de l'invention, le patch logiciel détermine le contexte d'exécution courant pour sélectionner ladite donnée à utiliser parmi une pluralité de données associées à une pluralité respective de contextes d'exécution. C'est la donnée ainsi sélectionnée qui est renseignée dans la zone mémoire principale évoquée précédemment.

A titre d'exemple, à chaque contexte d'exécution peut être associé un fichier de données de partage dans lequel est défini l'ensemble des données spécifiques à ce contexte. Le patch selon l'invention peut alors permettre de "charger" ces données contextuelles d'un coup en indiquant ou pointant la zone mémoire principale vers ce fichier.

En particulier, la détermination du contexte d'exécution courant comprend la détection de l'une parmi :

– une interface de communication utilisée pour l'exécution de l'application, entre une interface de communication à contact et une interface de communication sans contact ;

– une information identifiant un utilisateur du dispositif de traitement au moment de l'exécution de ladite application ; et

– une information identifiant la détection d'une attaque sur le dispositif de traitement.

Le premier contexte d'exécution relatif à la nature de l'interface de communication peut simplement être déterminé en détectant la provenance du signal électrique d'alimentation du dispositif de traitement (soit via un contact électrique, soit via le module sans contact).

Le deuxième contexte peut par exemple découler d'une identification préalable de l'utilisateur (ou "*porteur*") du dispositif. Cette identification permet ainsi de "charger" des données spécifiques à cet utilisateur pour l'exécution de l'application, sans que cette dernière ne soit modifiée pour y indiquer explicitement l'existence de plusieurs utilisateurs possibles.

Le troisième contexte découle des mécanismes de détection d'attaques, par exemple d'attaque par faute, bien connus de l'homme de l'art. La présente invention offre ici une réalisation modulaire des préventions contre ces attaques car les mécanismes de détection des attaques, l'application informatique et le patch autonome apte à sélectionner des données en fonction de la détection ou non d'une attaque sont totalement indépendants.

Selon une caractéristique particulière, la pluralité de données associées à une pluralité respective de contextes d'exécution comprend l'une parmi :

– une pluralité de montants monétaires associée respectivement à une pluralité de moyens de paiement mettant en œuvre des moyens de communication différents ;

– une pluralité de numéros d'identification personnels (PIN) associée à une pluralité d'utilisateurs ; et

– au moins une donnée secrète et une donnée factice associées à l'absence et à la détection d'une attaque sur la carte à puce.

Ces données peuvent correspondre aux trois contextes d'exécution évoqués précédemment.

Dans un mode de réalisation, ladite instruction de branchement est intégrée dans le code de l'application informatique, avant les instructions mettant en œuvre ladite donnée stockée en mémoire.

Cette disposition permet soit d'utiliser directement les instructions de branchement vers d'hypothétiques patchs prévues dès la conception de l'application afin de ne pas intervenir à nouveau sur celle-ci, soit de modifier uniquement l'application en un seul emplacement du code, préalablement à la première instruction mettant en œuvre la donnée contextuelle (donc par exemple au niveau du début de l'application).

En variante, ladite instruction de branchement est intégrée à un système d'exploitation dans lequel est exécutée l'application informatique, ladite instruction de branchement étant prévue au sein d'instructions de code configurées pour traiter une commande de sélection de ladite application informatique.

Cette disposition permet d'éviter toute modification de l'application informatique. En outre, elle présente l'avantage d'une mise en œuvre simple de l'invention car associer l'instruction de branchement vers le patch au morceau de code traitant toute commande de sélection de l'application informatique garantit que la détermination et la sélection des données contextuelles sont bien réalisées avant leur utilisation.

Selon une caractéristique de l'invention, ladite application informatique est mémorisée dans une mémoire protégée en écriture du dispositif de traitement. Il peut s'agir d'une mémoire morte non réinscriptible, ou d'une mémoire Flash EEPROM protégée contre l'écriture par des moyens matériels. En effet, les applications pour lesquels les certifications sont généralement requises sont celles prévues pour être stockées dans les mémoires protégées en écriture. La mise en œuvre de l'invention s'avère donc particulièrement avantageuse pour ce type d'applications informatiques.

Selon une caractéristique de l'invention, le dispositif de traitement est une carte à puce.

L'invention a également pour objet un procédé d'exécution d'une application informatique mémorisée dans une mémoire protégée en écriture d'une carte à puce, ladite application informatique s'exécutant en utilisant une donnée stockée en mémoire qui est fonction d'un contexte d'exécution courant, comprenant les étapes suivantes:

à détection, par un système d'exploitation de la carte à puce, d'une commande de sélection de ladite application informatique, exécution d'une instruction de branchement prévue dans le système d'exploitation et configurée pour déclencher l'exécution d'un patch logiciel autonome,

le patch logiciel alors en exécution sélectionne, en fonction de l'interface de communication utilisée par la carte parmi une interface de communication avec contact et une interface de communication sans contact, une donnée à utiliser parmi au moins une première donnée contextuelle associée à une communication avec contact et au moins une deuxième donnée contextuelle associée à une communication sans contact.

Ce procédé présente des avantages similaires à ceux exposés précédemment, notamment celui d'éviter toute intervention dans le code de l'application informatique pour introduire une utilisation de données contextuelles.

L'invention a également pour objet un dispositif de traitement, par exemple une carte à puce, comprenant un environnement logiciel d'exécution et une application informatique dont l'exécution, dans l'environnement logiciel d'exécution, utilise au moins une donnée stockée en mémoire du dispositif de traitement, cette donnée utilisée étant fonction d'un contexte d'exécution courant,

ladite application informatique et/ou l'environnement logiciel comprenant au moins une instruction de branchement pour appeler un patch logiciel autonome externe,

et le dispositif de traitement comprenant un patch logiciel autonome externe à l'application et agencé pour déterminer ladite donnée à utiliser en

fonction du contexte d'exécution courant parmi une pluralité de données stockées en mémoire,

dispositif dans lequel ladite instruction de branchement est configurée pour déclencher l'exécution du patch logiciel autonome préalablement à l'utilisation de ladite donnée par l'application de sorte à sélectionner ladite donnée à utiliser parmi ladite pluralité de données.

Le dispositif de traitement ou a carte à puce présente des avantages similaires à ceux du procédé exposé ci-dessus. En particulier, il permet de ne pas avoir à modifier le code de l'application informatique (ou alors de façon très minime) pour introduire l'utilisation de données contextuelles.

De façon optionnelle, le dispositif de traitement peut comprendre des moyens se rapportant aux caractéristiques du procédé exposé précédemment, et notamment des zones mémoires principale et secondaire à partir desquelles sont réalisées des opérations de copie ou sont instanciés des pointeurs; un système d'exploitation définissant l'environnement logiciel d'exécution de l'application et dans lequel l'instruction de branchement est prévue au sein d'instructions de code configurées pour traiter une commande de sélection de l'application informatique.

Dans un mode de réalisation particulier, l'invention concerne également une carte à puce comprenant :

au moins une mémoire protégée en écriture mémorisant une application informatique dont l'exécution utilise une donnée stockée en mémoire et fonction d'un contexte d'exécution courant, et mémorisant un système d'exploitation définissant une environnement logiciel dans lequel s'exécute ladite application,

une interface de communication avec contact et une interface de communication sans contact pour communiquer avec l'extérieur de la carte,

une mémoire stockant au moins une première donnée contextuelle associée à une communication avec contact, et au moins une deuxième donnée contextuelle associée à une communication sans contact,

un patch logiciel autonome agencé pour sélectionner, en fonction d'une interface de communication utilisée, la donnée à utiliser par l'application parmi la première et la deuxième données,

carte dans laquelle le système d'exploitation comprend une instruction de branchement apte à être exécutée à la détection d'une commande de sélection de ladite application informatique, ladite instruction de branchement étant configurée pour déclencher l'exécution dudit patch de sorte à sélectionner ladite donnée à utiliser.

Cette carte à puce présente notamment les avantages évoqués précédemment en lien avec le procédé d'exécution selon l'invention.

L'invention concerne également un moyen de stockage d'informations comprenant des instructions pour un programme informatique adapté à mettre en œuvre le procédé d'exécution conforme à l'invention lorsque ce programme est chargé et exécuté par un système informatique.

L'invention concerne également un programme d'ordinateur lisible par un microprocesseur, comprenant des instructions pour la mise en œuvre du procédé d'exécution conforme à l'invention, lorsque ce programme est chargé et exécuté par le microprocesseur.

Les moyens de stockage d'information et programme d'ordinateur présentent des caractéristiques et avantages analogues aux procédés qu'ils mettent en œuvre.

D'autres particularités et avantages de l'invention apparaîtront dans la description ci-après, illustrée par les dessins ci-joints, dans lesquels :

- la **figure 1** illustre schématiquement une carte à puce mettant en œuvre la présente invention;

- la **figure 2** représente, sous forme de logigramme, des étapes générales d'exécution d'une application embarquée dans une carte à puce, selon la présente invention; et

- les **figures 3a-3c** illustrent un mode de réalisation des mécanismes d'indication de données contextuelles à l'application embarquée, en fonction du contexte d'exécution courant.

Dans l'exemple de réalisation décrit ci-après en référence aux **figures 1** à **3**, l'application informatique au sens de l'invention est une application de transaction financière embarquée dans un dispositif de traitement. Bien entendu, l'invention peut s'appliquer à d'autres types d'applications embarquées, telles qu'une application d'authentification ou de cryptographie.

En référence à la **figure 1**, un dispositif portatif de traitement de type carte à puce 1 comprend un corps de carte 10 et un module de carte 20. Le corps de carte 10 comprend, dans son épaisseur, une antenne 11 de communication sans contact reliée au module de carte 20, bien connue de l'homme de l'art.

Le module de carte 20 comprend, comme illustré schématiquement, un bus central 21 auquel sont reliés une mémoire morte non réinscriptible (mémoire ROM) 22, une mémoire vive RAM 23, une mémoire de stockage réinscriptible 24, type mémoire EEPROM, un microprocesseur 25, un module de communication sans contact 26 relié à l'antenne 11 par des bornes de connexion (non représentées), et une interface de communication par contact 27 présentant des contacts électriques affleurant la carte.

Le module de communication sans contact 26 (ou cellule radiofréquence) associé à l'antenne 11 contribue aux communications selon la norme ISO 14443 avec un lecteur sans contact 50 externe à la carte à puce. L'interface de communication par contact 27 contribue aux communications selon la norme ISO 7816 avec un lecteur avec contact 51 externe à la carte à puce.

Dans la mémoire ROM 22, sont mémorisées les instructions logicielles d'une application informatique de transaction financière 220 et d'un système d'exploitation 221 qui est exécuté lors de l'alimentation électrique de la carte à puce, soit par le lecteur 51 via les contacts électriques 27, soit par le lecteur 50 via le circuit de communication sans contact 11-26. L'invention s'applique toutefois également lorsque l'application informatique est mémorisée dans la mémoire réinscriptible 24.

L'application informatique 220 présente, lorsqu'elle est exécutée, une fonction de transaction "grand montant" utilisant le lecteur avec contact 51, et une fonction de transaction "faible montant" utilisant le lecteur sans contact 50. Ces deux fonctions sont réalisées sensiblement à l'aide des mêmes mécanismes, c'est-à-dire par les mêmes portions de code de l'application informatique 220.

Lors de cette exécution, l'application informatique utilise des données stockées en mémoire EEPROM 24, pour contrôler les mécanismes de transaction. Dans l'exemple de la figure, ces données sont au nombre de quatre comme suit:

- un compteur 240 du montant cumulé des transactions effectuées depuis la dernière connexion *on line* avec un serveur central;
- un compteur 241 du nombre de transactions effectuées depuis la dernière connexion *on line* avec le serveur central;
- des valeurs seuils haute et basse 242 pour le compteur 240: la valeur basse définit la valeur à partir de laquelle il est essayé de réaliser la transaction courante *on line* avec le serveur central, et la valeur haute définit la valeur à partir de laquelle toute nouvelle transaction est bloquée s'il n'est pas possible de réaliser une connexion avec le serveur central;
- des valeurs seuils haute et basse 243 pour le compteur 241, ayant la même finalité que les valeurs 242 eu égard au nombre de transactions 241;
- des valeurs 244 d'un premier seuil et d'un second seuil définissant respectivement un premier plafond de montant de transaction à partir duquel la transaction nécessite une connexion avec le serveur central, et un deuxième plafond de montant maximal autorisé.

L'application informatique 220 et/ou le système d'exploitation 221 comprennent des instructions de branchement vers d'hypothétiques patchs, prévues dès leur conception. De telles instructions de branchement prévues dès la conception sont connues de l'homme de l'art pour permettre notamment d'introduire à faible coût des patchs correctifs de l'application ou du système d'exploitation. En pratique ces instructions de branchement sont exécutées mais en l'absence du patch correspondant en mémoire (ou si celui-ci est vide),

aucune action ne se passe. En variante, de telles instructions de branchement peuvent être rajoutées au coup par coup à ces logiciels, nécessitant alors une recompilation avant réinstallation.

Selon l'invention, ces instructions de branchement vont permettre de lancer l'exécution d'un patch qui remplira les données 240-244 avec les données spécifiques au contexte de la transaction en cours (transaction avec contact ou transaction de porte-monnaie électronique sans contact).

Le microprocesseur 25 est apte à exécuter tout logiciel tel que le système d'exploitation 221 et l'application informatique 220 et d'éventuels patchs (section de code additionnelle d'un logiciel) tels qu'introduits par la suite.

Le microprocesseur 25 permet ainsi à la carte à puce 1 d'exécuter un procédé conforme à l'invention dont des exemples sont donnés par la suite. Chaque logiciel est composé d'une série d'instructions de commande du microprocesseur 25 qui sont chargées depuis une mémoire non volatile 22 ou 24 vers la mémoire vive RAM 23 ou qui sont exécutées directement depuis la mémoire non volatile.

En variante, l'ensemble microprocesseur 25 - mémoire non-volatile 22 - mémoire vive 23 peut être remplacé par un circuit à application spécifique qui comprend alors des moyens de mise en œuvre des différentes étapes du procédé selon l'invention.

La mémoire non volatile réinscriptible 24 stocke par ailleurs une pluralité de données 240'-244'; 240"-244" associées à une pluralité respective de contextes d'exécution C1-C2. L'invention s'applique également lorsque tout ou partie de ces données sont mémorisées dans une autre mémoire, par exemple la mémoire ROM 22 lorsque ces données concernent des données secrètes (clés de cryptage par exemple).

Dans notre exemple de la figure, quatre données 240'-244' sont associées à un contexte d'exécution C1 défini par l'utilisation de l'interface de communication avec contact 27 lors d'une transaction. Ces données comprennent :

- un compteur 240' du montant cumulé des transactions effectuées depuis la dernière connexion *on line* avec un serveur central ;

- un compteur 241' du nombre de transactions effectuées depuis la dernière connexion *on line* avec le serveur central ;

- des valeurs seuils haute et basse 242' pour le compteur 240': la valeur basse définit la valeur à partir de laquelle il est essayé de réaliser la transaction courante *on line* avec le serveur central, et la valeur haute définit la valeur à partir de laquelle toute nouvelle transaction est bloquée s'il n'est pas possible de réaliser une connexion avec le serveur central ;

- des valeurs seuils haute et basse 243' pour le compteur 241', ayant la même finalité que les valeurs 242' eu égard au nombre de transactions 241' ;

- des valeurs 244 d'un premier seuil et d'un second seuil définissant respectivement un premier plafond de montant de transaction à partir duquel la transaction nécessite une connexion avec le serveur central, et un deuxième plafond de montant maximal autorisé pour les transactions avec contact.

Par défaut, ces données peuvent être copiées au niveau des données 240-244 à la mise sous tension de la carte à puce. Cela définit le contexte C1 comme contexte par défaut.

De façon similaire, les quatre données 240"-244" sont associées à un contexte d'exécution C2 défini par l'utilisation de l'interface de communication sans contact 26+11 lors d'une transaction "faible montant". Ces données comprennent :

- un compteur 240" du montant cumulé des transactions effectuées depuis la dernière connexion *on line* avec un serveur central ;

- un compteur 241" du nombre de transactions effectuées depuis la dernière connexion *on line* avec le serveur central ;

- des valeurs seuils haute et basse 242" pour le compteur 240" ;

- des valeurs seuils haute et basse 243" pour le compteur 241" ;

- des valeurs 244" d'un seuil définissant un plafond de montant maximal autorisé pour les transactions sans contact.

En pratique l'utilisation du porte-monnaie électronique ne requiert pas d'identification par code PIN, ni de connexion avec un serveur central (transaction *offline*). De ce fait les valeurs seuils haute et basse 242" et 243"

sont de préférence plus faibles que les valeurs seuils haute et basse 242' et 243'. A noter que dans ce cas, les compteurs comptabilisent par exemple le montant cumulé et le nombre de transactions depuis la dernière connexion *on line* effectuée pour recharger le porte-monnaie électronique.

Enfin, la mémoire EEPROM 24 mémorise un patch logiciel autonome 249 constitué d'une suite d'instructions de code agencée pour déterminer un contexte d'exécution courant entre C1 et C2 et pour sélectionner les données 240'-244' ou 240"-244" correspondantes et les associer aux emplacements mémoires 240-244 utilisés par l'application. Bien entendu, un plus grand nombre de contextes peut être prévu.

En référence à la **figure 2**, on décrit maintenant les étapes principales d'une mise en œuvre de l'invention.

A la mise sous tension de la carte à puce 1 par le lecteur 50 ou 51, le système d'exploitation détermine, à l'aide de moyens logiciels appropriés, si les communications mises en œuvre le sont via l'interface sans contact 26 ou via l'interface avec contact 27. Cette information 230 est alors stockée en mémoire RAM 23 (étape E200) et indique directement dans quel contexte d'exécution C1 ou C2 se situent les communications en cours.

La détermination du mode de communication mis en œuvre peut notamment résulter de la détection du courant d'alimentation, soit par induction via le circuit 26, soit directement via les contacts électriques de l'interface 27.

En variante, cette détection peut être réalisée ultérieurement à la mise sous tension, par exemple au moment où l'application informatique de transaction 220 est appelée et sélectionnée.

Puis pour la mise en œuvre d'une transaction financière (avec ou sans contact), le terminal actif 50 ou 51 sélectionne l'application de transaction financière 220 par émission d'une commande APDU (pour "*Application Protocol Data Unit*") de type SELECT, à l'étape E210. En variante, la sélection de l'application 220 peut être implicite et automatique à la mise sous tension E200).

A réception de la commande APDU SELECT, le système d'exploitation 221 exécute son propre code destiné à traiter ces commandes

APDU SELECT. Ce code contient notamment une instruction de branchement vers le patch 249. Ainsi, le code du patch 249 est exécuté à cette occasion s'il n'est pas vide (étape E220), c'est-à-dire avant l'utilisation des données 240-244 par l'application 220. Si le patch 249 est vide, l'exécution du système d'exploitation 221 se poursuit vers l'étape E240.

Dans un mode de réalisation, un drapeau (non représenté) peut être prévu au niveau de la mémoire EEPROM 24 pour indiquer si le patch 249 est vide. Ainsi, le système d'exploitation 221 n'accède au patch 249 que s'il est non vide.

Au cours de l'étape E230 correspondant à l'exécution du code du patch 249 s'il est non vide, ce dernier :

- détermine (E231) le contexte d'exécution courant C1 ou C2 par simple lecture de l'information 230. En variante, la détermination de l'information 230 peut être réalisée à cette occasion en lieu et place de l'étape E200 ci-dessus ;
- détermine (E232), dans la mémoire EEPROM 24, les données spécifiques au contexte courant, à savoir 240'-244' pour le contexte C1 et 240"-244" pour le contexte C2 ;
- renseigne (E233), au niveau des données 240-244 qui vont être utilisées par l'application de transaction 220, les données déterminées à l'étape E232.

A titre illustratif, l'étape E233 peut consister à recopier les données déterminées 240'-244' ou 240"-244" au niveau des données 240-244, avec écrasement des données qui y sont présentes.

En variante, le patch 249 peut renseigner, dans les emplacements 240-244, des pointeurs appropriés vers les données déterminées 240'-244' ou 240"-244".

Puis à l'étape E240, l'application de transaction 220 est effectivement sélectionnée et lancée. Son exécution utilise alors les données 240-244 (contenant 240'-244' ou 240"-244") pour réaliser ladite transaction demandée par le lecteur actif 50 ou 51. A noter qu'en l'absence de patch 249, les données

240-244 comprennent des données par défaut qui correspondent à un unique contexte d'exécution envisagé.

Grâce à l'invention, en une seule action, c'est-à-dire à partir d'un seul point de branchement dans le système d'exploitation 221 (de façon similaire dans l'application 220 le cas échéant), on affecte les bonnes valeurs contextuelles à utiliser en fonction du contexte d'exécution courant.

Dans notre exemple, l'application 220 utilise ainsi les données spécifiques 240"-244" à une transaction de porte-monnaie électronique s'il s'agit d'une transaction avec un lecteur sans contact 50, et utilise les données spécifiques 240'-244' d'une transaction avec contact en cas d'utilisation du lecteur par contact 51.

Une étape E250 peut alors être prévue dans le cas où l'étape E233 consiste en une copie des valeurs 240'-244' ou 240"-244" dans les emplacements mémoires 240-244. Au cours de cette étape, les données mémorisées dans les données 240-244 (qui ont pu évoluer lors de l'exécution de l'application 220) sont recopiées dans les emplacements d'origine 240'-244' ou 240"-244" selon que le contexte d'exécution courant soit C1 ou C2 (indiqué dans l'information 230).

L'exécution de l'application 220 pour la transaction se termine alors.

Selon une variante de cette réalisation, l'instruction de branchement permettant l'exécution du patch 249 peut être prévue dans l'application 220 elle-même et non plus dans le système d'exploitation 221 définissant l'environnement d'exécution de l'application.

Dans ce cas, cette instruction de branchement est prévue au début des instructions de code de l'application 220, et en tout état de cause avant les instructions de code utilisant les données 240-244.

Ainsi, le système d'exécution 221 déclenche l'exécution de l'application, laquelle déclenche presque immédiatement l'exécution du patch 249 à l'aide de l'instruction de branchement. Les données 240'-244' ou 240"-244" selon le contexte courant C1 ou C2 indiquée dans 230 sont alors chargées ou renseignées dans les emplacements mémoires 240-244, de sorte à ce que l'application 220 utilise les bonnes données lors de son exécution.

Bien entendu, le cas échéant, une recopie des données 240-244 vers les données 240'-244' ou 240"-244" peut être mise en œuvre.

Dans une réalisation différente de l'invention, le patch 249 peut être intégré au sein du système d'exploitation 221 ou de l'application 220 (avant les instructions de code utilisant les données 240-244). L'invention prévoit ainsi d'intervenir qu'en un seul emplacement du logiciel modifié (220 ou 221), là où les techniques de l'état de l'art nécessitent l'intégration d'un grand nombre d'instructions conditionnelles.

Par ailleurs, bien que l'exemple précédant mette en œuvre cinq données spécifiques à chaque contexte d'exécution, l'invention s'applique à un nombre quelconque de telles données spécifiques. Le patch 249 est alors adapté pour tenir compte des spécificités de chaque contexte d'exécution: par exemple deux contextes d'exécution peuvent conduire à la copie d'un nombre différents de données spécifiques contextuelles car l'application peut nécessiter plus ou moins de données selon le contexte.

Selon différentes applications données ici à titre illustratif, ces données peuvent correspondre, en variante ou en combinaison :

– à des valeurs monétaires variables d'un contexte d'exécution à l'autre, comme suggéré dans l'exemple ci-dessus ;

– à des numéros d'identification personnels (PIN) qui sont différents selon l'utilisateur/porteur de la carte à puce au moment de la transaction ou qui sont différents selon que la transaction soit réalisée avec ou sans contact ;

– à une clé secrète (de cryptage ou d'authentification par exemple, mémorisée en mémoire ROM de préférence) utilisée dans un contexte où aucune attaque n'est détectée (par exemple par défaut) et à des données factices (*dummy data*) qui sont chargées dans les emplacements mémoires 240-244 dès qu'une attaque est détectée (afin dans ce cas de permettre une exécution de l'application 220 sans danger car les données 240-244 manipulées sont alors sans intérêt) ;

– à tout autre paramètre spécifique qui varie d'un contexte à l'autre.

Egalement, bien que l'exemple précédent mette en œuvre deux contextes d'exécution différents, un plus grand nombre peut être prévu, en

combinant par exemple plusieurs critères de contexte. A titre d'exemple, ces critères (pris individuellement ou en combinaison) peuvent comprendre le type de communication (avec ou sans contact), l'identification du porteur/utilisateur de la carte (par exemple via un menu de sélection initial – utilisation personnelle/utilisation professionnelle ou plusieurs personnes physiques différentes), la détection d'une attaque de la carte à l'aide de moyens matériels et/ou logiciels de détection. L'étape E200 est alors adaptée pour procéder à la détermination des critères définissant ces contextes.

On décrit maintenant en référence à la **figure 3**, un mode de réalisation des moyens d'affectation (E233) des données contextuelles en 240-244 en fonction du contexte déterminée en E200.

Les instructions de code de l'application de transaction 220 qui utilisent les données contextuelles stockées en mémoire pointent directement sur un fichier de données 2400 qui est associé à un contexte C0, défini ici comme contexte par défaut. Par exemple, ces instructions de code pointent sur l'adresse mémoire correspondant au début du fichier 2400.

Ce fichier 2400 comprend un entête 2410 et des données utiles 2420. L'entête 2410 comprend comprenant un identifiant unique du fichier (file_id 2411), une adresse des données utiles 2412 pointant sur le début de la section 2420 (@1000h), un champ binaire 2413 définissant les données utiles partagées comme décrit par la suite, et une adresse de partage 2414.

Par défaut, le champ binaire 2413 est nul (uniquement des '0') indiquant que les valeurs des données utiles 2412 doivent être utilisées.

Les données utiles 2420 correspondent à des données 240-244 spécifiques au contexte C0, mémorisées par exemple sous forme de structure TLV (*tag-length-value*).

Sans l'exécution du patch 249 ou lorsque le contexte d'exécution courant (dans 230) est le contexte C0, le champ binaire 2413 est nul indiquant que l'application 220 doit utiliser les données 240-244 de la section 2420 (**figure 3a**).

Les données spécifiques 240'-244' et 240''-244'' aux autres contextes (dans l'exemple de la figure, les contextes C1 et C2) sont également

mémorisées dans des fichiers similaires 2400' et 2400'', respectivement au niveau des adresses @2000h et @3000h.

Lorsque le contexte C1 est détecté comme étant le contexte d'exécution courant, le patch 249 vient modifier l'entête 2410 du fichier par défaut 2400 sur lequel pointe directement l'application 220, afin d'indiquer qu'il y a lieu désormais d'utiliser une ou plusieurs données spécifiques au contexte C1.

Dans l'exemple de la figure, on illustre l'utilisation d'une seule donnée spécifique au contexte C1, la donnée *data2* (notée *data2_C1*, en gras sur la **figure 3b**). La donnée *data2_C0* (barrée) est donc remplacée, pour les traitements de l'application, par cette donnée *data2_C1*.

Cette indication comprend la modification du champ binaire 2413 pour indiquer que la deuxième donnée *data2* est désormais à récupérer au niveau de l'adresse 2414 qui est elle-même précisée à la valeur @2000h. Par exemple le champ binaire consiste en une succession de bits ordonnés dont le moins significatif correspond à la première donnée *data1*. Dans ce cas, l'indication que la deuxième donnée *data2* est à récupérer à l'adresse 2414 prend la forme d'un champ 2413 valant '00010' (pour cinq données uniquement).

La récupération de *data2_C1* s'avère aisée pour l'application 220, en accédant à l'adresse 2414 et en identifiant la deuxième donnée dans la section 2420' à l'aide des tags de la structure TLV.

Par défaut, l'adresse 2414 peut être celle des données utiles 2420' du deuxième fichier 2400', soit @2000h, auquel cas l'indication du contexte C1 consiste uniquement à modifier le champ 2413 pour indiquer que la deuxième donnée qui doit être utilisée est celle à l'adresse @2000h déjà renseignée.

Comme montré sur la **figure 3b**, l'application 220 lorsqu'elle utilise les données contextuelles 240-244 accède à *data1_C0* (240), *data3_C0* (242), *data4_C0* (243) et *data5_C0* (244) dans le fichier 2400 et accède à *data2_C1* (241' – deuxième tag dans la section 2420') dans le fichier 2400' comme cela est indiqué au travers du champ binaire de partage 2413 et de l'adresse associée 2414.

La **figure 3c** illustre le cas où le contexte C2 est le contexte courant. Dans ce cas, le patch 249 a modifié le champ binaire 2413 pour indiquer que les deuxième et troisième données (*data2* et *data3*) sont spécifiques au contexte courant (donc ce champ vaut '00110') et a renseigné dans l'adresse 2414 l'adresse @3000h correspondant à la structure TLV associé au contexte C3.

Bien entendu, les cinq données *data1-data5* peuvent être différentes pour chaque contexte auquel cas le champ binaire 2413 prendra la valeur 11111 pour un contexte courant différent de C0, afin d'indiquer à l'application 220 d'aller chercher toutes les données contextuelles à l'adresse 2414. Lorsque l'intégralité du fichier 2400 est à utiliser pour chaque contexte, on peut s'affranchir de l'utilisation du champ 2413: seule la présence d'une adresse dans 2414 indique s'il y a des valeurs spécifiques à utiliser et où elles sont mémorisées. L'absence d'une adresse dans 2414 indique ainsi d'utiliser le contexte par défaut et les données 2420.

Ainsi, en jouant sur le champ binaire de partage 2413 et/ou sur l'adresse de partage 2414 on dirige efficacement l'exécution de l'application 220 vers l'utilisation des données spécifiques au contexte d'exécution courant. En outre, l'utilisation de pointeurs (adresse 2414) et d'un contexte par défaut permet d'éviter la mise en œuvre de l'étape E250 de recopie des données en cas d'évolution de celles-ci au cours de l'exécution de l'application 220.

Ces modifications de l'entête 2410 sont relativement simples à mettre en œuvre, de telle sorte que la présente invention relève d'une complexité réduite, tout en garantissant l'absence de modification de l'application 220.

Les exemples qui précèdent ne sont que des modes de réalisation de l'invention qui ne s'y limite pas.

REVENDICATIONS

1. Procédé d'exécution d'une application informatique (220) dans un dispositif de traitement (1) comprenant un environnement logiciel (221) d'exécution de l'application, ladite application informatique et/ou l'environnement logiciel comprenant au moins une instruction de branchement pour appeler un patch logiciel autonome externe, et ladite application informatique s'exécutant en utilisant au moins une donnée (240-244) stockée en mémoire (24), cette donnée utilisée étant fonction d'un contexte d'exécution courant (C0, C1, C2),

caractérisé en ce que la donnée stockée fonction du contexte d'exécution courant est déterminée (E232), préalablement à son utilisation (E240) par l'application (220), par l'exécution d'un patch logiciel autonome (249) externe à ladite application et déclenchée au travers de ladite instruction de branchement.

2. Procédé selon la revendication 1, dans lequel ladite instruction de branchement est une instruction de branchement prévue à la conception de l'application informatique et/ou de l'environnement logiciel pour appeler un hypothétique patch logiciel autonome externe non encore défini.

3. Procédé selon la revendication 1 ou 2, dans lequel l'application informatique (220) accède une zone mémoire principale (240-244, 2400) pour obtenir ladite donnée à utiliser, et ledit patch logiciel (249) renseigne (E233), dans ladite zone mémoire principale, la donnée déterminée en fonction du contexte d'exécution courant (C0, C1, C2).

4. Procédé selon la revendication 3, dans lequel le patch logiciel (249) copie (E233) la donnée déterminée depuis une zone mémoire secondaire (240'-244', 240"-244") vers la zone mémoire principale (240-244).

5. Procédé selon la revendication 3 ou 4, dans lequel la donnée déterminée en fonction du contexte d'exécution courant est renseignée par une écriture en mémoire non volatile réinscriptible (24).

6. Procédé selon la revendication 3, dans lequel le patch logiciel (249) associe à la zone mémoire principale un pointeur (2414) vers une zone mémoire secondaire (2420', 2420") stockant la donnée déterminée.

7. Procédé selon la revendication 3 ou 6, dans lequel la zone mémoire principale (2400) comprend des données par défaut (240-244) d'un contexte par défaut (C0), un champ binaire de partage (2413) et une adresse de partage (2414),

et on renseigne (233), dans la zone mémoire principale, au moins une donnée déterminée différente des données par défaut, en utilisant le champ binaire de partage (2413) pour indiquer quelles données par défaut sont remplacées par l'au moins une donnée déterminée et en utilisant l'adresse de partage (2414) pour pointer sur l'emplacement mémoire stockant l'au moins une donnée déterminée.

8. Procédé selon l'une des revendications précédentes, dans lequel le patch logiciel (249) détermine (E200, E231) le contexte d'exécution courant (C0, C1, C2) pour sélectionner (E232) ladite donnée à utiliser parmi une pluralité de données (240-244, 240'-244', 240"-244") associées à une pluralité respective de contextes d'exécution (C0, C1, C2).

9. Procédé selon la revendication 8, dans lequel la détermination (E200) du contexte d'exécution courant comprend la détection de l'une parmi :

– une interface de communication utilisée pour l'exécution de l'application, entre une interface de communication à contact (27) et une interface de communication sans contact (26) ;

– une information identifiant un utilisateur du dispositif de traitement au moment de l'exécution de ladite application ; et

– une information identifiant la détection d'une attaque sur le dispositif de traitement.

10. Procédé selon la revendication 8 ou 9, dans lequel la pluralité de données associées à une pluralité respective de contextes d'exécution comprend l'une parmi :

– une pluralité de montants monétaires associée respectivement à une pluralité de moyens de paiement mettant en œuvre des moyens de communication différents ;

– une pluralité de numéros d'identification personnels (PIN) associée à une pluralité d'utilisateurs ; et

– au moins une donnée secrète et une donnée factice associées à l'absence et à la détection d'une attaque sur la carte à puce.

11. Procédé selon l'une des revendications 1 à 10, dans lequel ladite instruction de branchement est intégrée à un système d'exploitation (221) dans lequel est exécutée l'application informatique (220), ladite instruction de branchement étant prévue au sein d'instructions de code configurées pour traiter une commande de sélection (SELECT) de ladite application informatique.

12. Procédé selon l'une des revendications précédentes, dans lequel ladite application informatique (220) est mémorisée dans une mémoire morte non réinscriptible (22) du dispositif de traitement (1).

13. Dispositif de traitement (1) comprenant un environnement logiciel (221) d'exécution et une application informatique (220) dont l'exécution, dans l'environnement logiciel d'exécution, utilise au moins une donnée (240-244) stockée en mémoire (24) du dispositif de traitement, cette donnée utilisée étant fonction d'un contexte d'exécution courant (C0, C1, C2),

ladite application informatique et/ou l'environnement logiciel comprenant au moins une instruction de branchement pour appeler un patch logiciel autonome externe,

caractérisé en ce qu'elle comprend:

un patch logiciel autonome (249) externe à l'application et agencé pour déterminer ladite donnée à utiliser en fonction du contexte d'exécution courant parmi une pluralité de données stockées (240-244, 240'-244', 240"-244") en mémoire,

et **en ce que** ladite instruction de branchement est configurée pour déclencher l'exécution du patch logiciel autonome (249) préalablement à l'utilisation de ladite donnée par l'application (220) de sorte à sélectionner ladite donnée à utiliser parmi ladite pluralité de données.

14. Moyen de stockage d'informations comprenant des instructions pour un programme informatique adapté à mettre en œuvre le procédé d'exécution selon l'une des revendications 1 à 12, lorsque ce programme est chargé et exécuté par un système informatique.

15. Programme d'ordinateur lisible par un microprocesseur, comprenant des instructions pour la mise en œuvre du procédé d'exécution selon l'une des revendications 1 à 12, lorsque ce programme est chargé et exécuté par le microprocesseur.

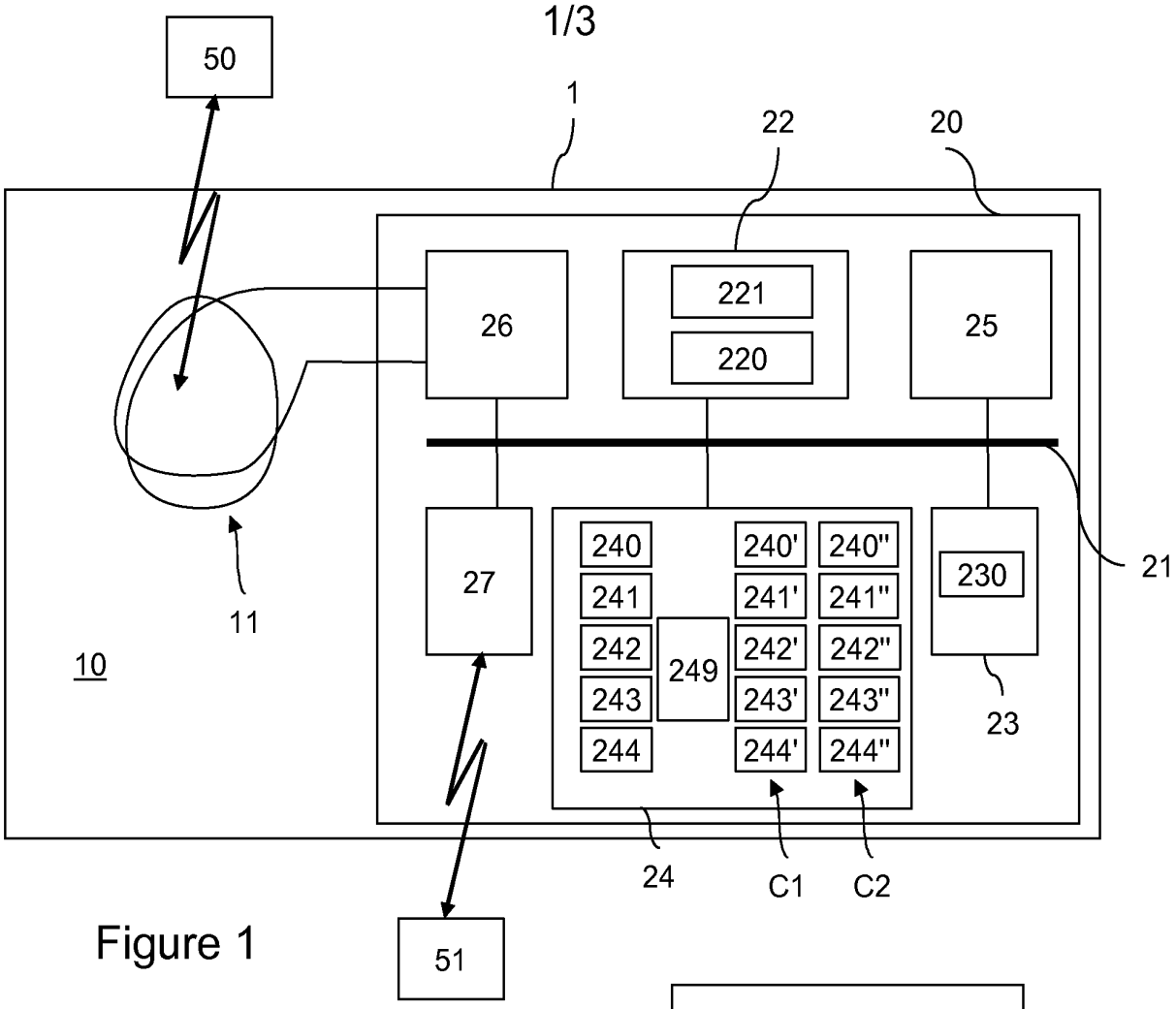


Figure 1

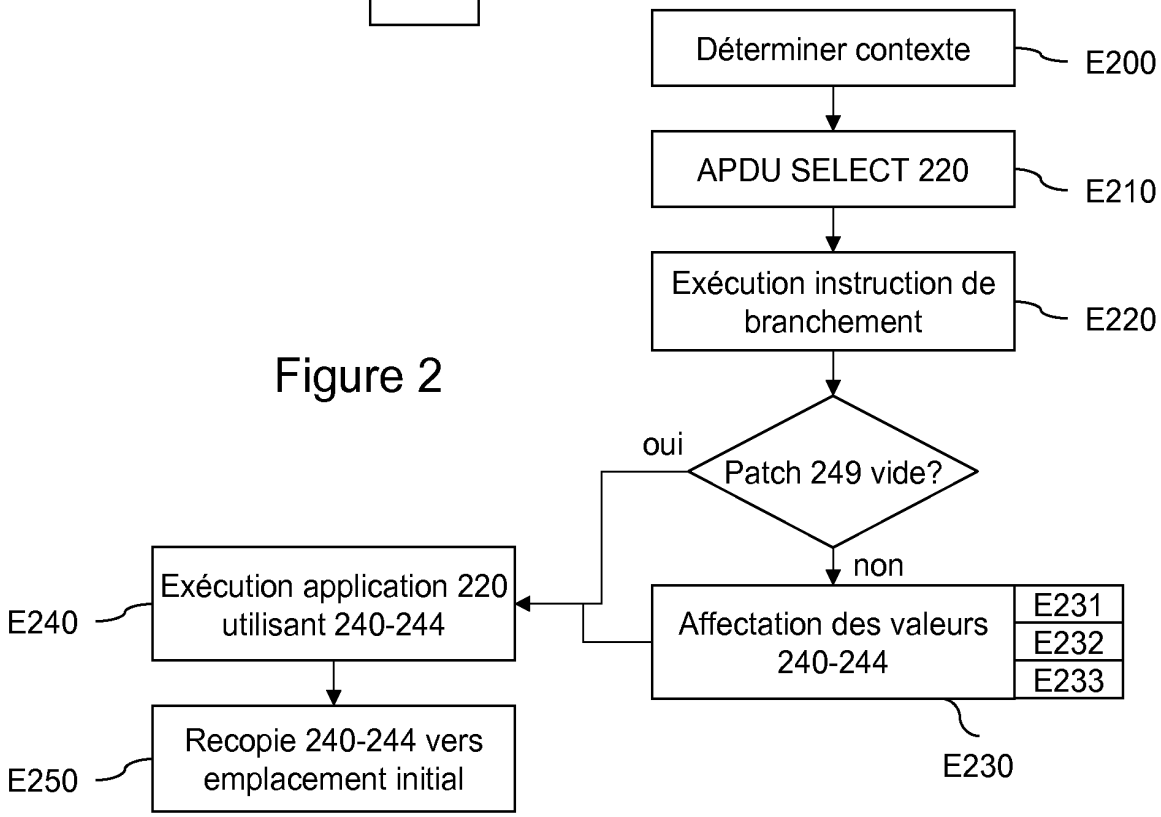


Figure 2

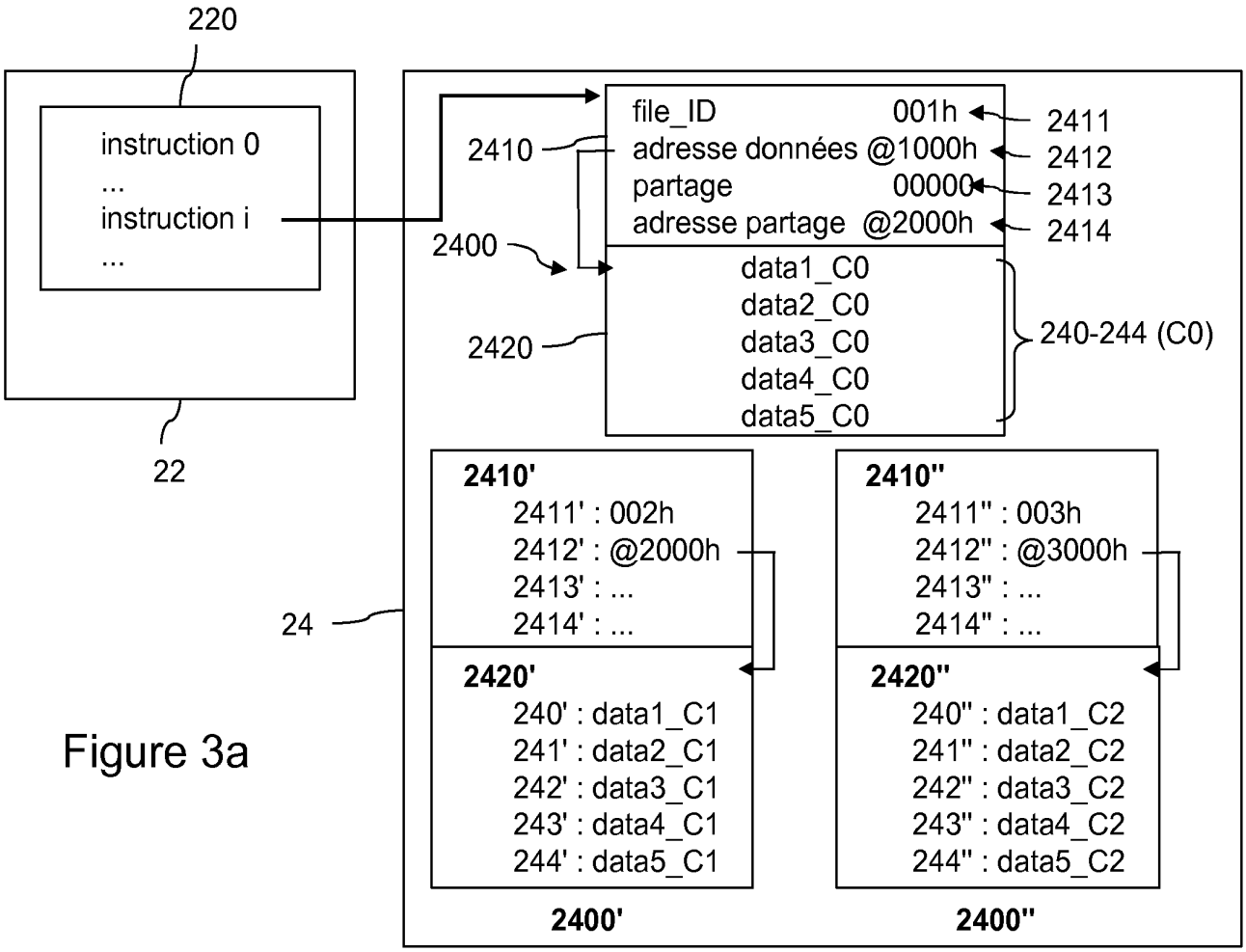

220
instruction 0
...
instruction i
...
22
2410
file_ID 001h
adresse données @1000h
partage 00000
adresse partage @2000h
2411
2412
2413
2414
2400
2420
data1_C0
data2_C0
data3_C0
data4_C0
data5_C0
240-244 (C0)
24
2410'
2411' : 002h
2412' : @2000h
2413' : ...
2414' : ...
2420'
240' : data1_C1
241' : data2_C1
242' : data3_C1
243' : data4_C1
244' : data5_C1
2400'
2410"
2411" : 003h
2412" : @3000h
2413" : ...
2414" : ...
2420"
240" : data1_C2
241" : data2_C2
242" : data3_C2
243" : data4_C2
244" : data5_C2
2400"

Figure 3a

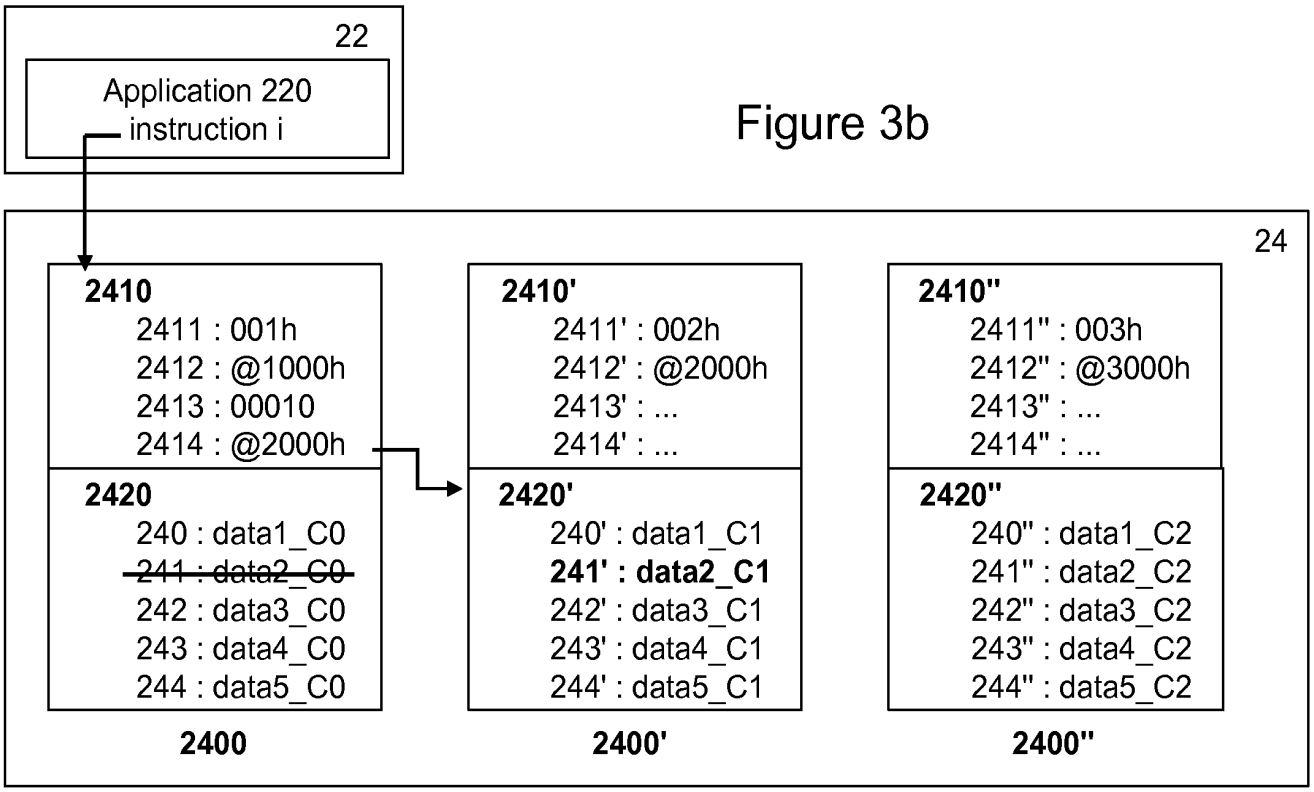


Figure 3b

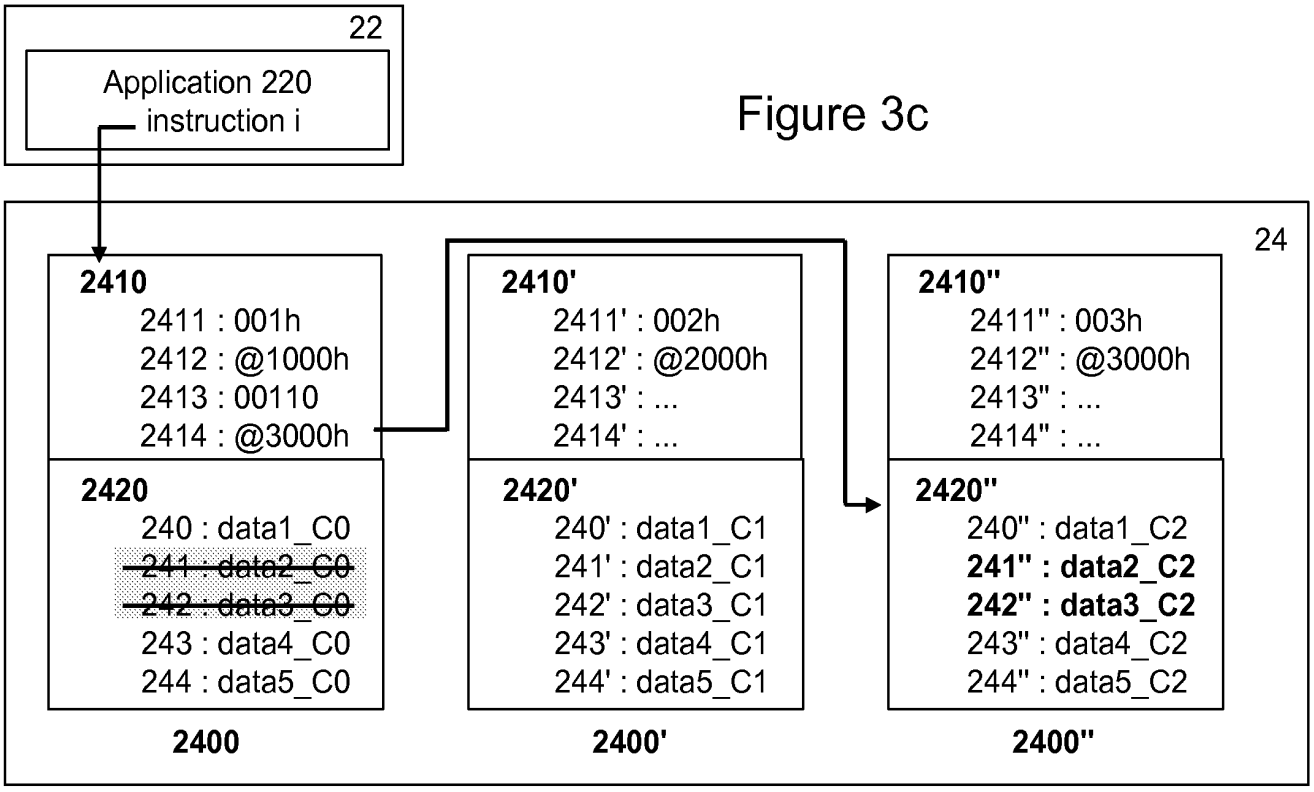


Figure 3c

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 740318
FR 1057084

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	US 2008/155573 A1 (DAWRA ANSHUL [US] ET AL) 26 juin 2008 (2008-06-26) * alinéa [0018] - alinéa [0029] * -----	1-15	G06F9/44 G06K19/073
A	US 2002/116478 A1 (PARADINAS PIERRE [FR] ET AL) 22 août 2002 (2002-08-22) * alinéa [0026] * * alinéa [0041] - alinéa [0042] * -----	1-15	
A	US 6 658 562 B1 (BONOMO RALPH [US] ET AL) 2 décembre 2003 (2003-12-02) * colonne 5, ligne 42 - colonne 6, ligne 22 * -----	1-15	
A	US 2007/157010 A1 (ZENZ INGO [DE] ET AL) 5 juillet 2007 (2007-07-05) * alinéa [0042] - alinéa [0048] * -----	1-15	
A	WO 2009/083979 A2 (SAFEND LTD [IL]; SEVER GIL [IL]; BERENGOLTZ PAVEL [IL]; HAZAMA HAY [IL) 9 juillet 2009 (2009-07-09) * page 3, ligne 18 - page 6, ligne 2 * -----	1-15	DOMAINES TECHNIQUES RECHERCHÉS (IPC) G06F
A	US 2002/178239 A1 (KINYON ROBERT [US] ET AL) 28 novembre 2002 (2002-11-28) * alinéa [0042] - alinéa [0049]; figure 2 * -----	1-15	

Date d'achèvement de la recherche	Examinateur
1 décembre 2010	Bijn, Koen

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 1057084 FA 740318

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 01-12-2010
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
US 2008155573 A1	26-06-2008	AUCUN	
US 2002116478 A1	22-08-2002	AUCUN	
US 6658562 B1	02-12-2003	AUCUN	
US 2007157010 A1	05-07-2007	EP 1971914 A1	24-09-2008
		WO 2007076944 A1	12-07-2007
WO 2009083979 A2	09-07-2009	AU 2008344956 A1	09-07-2009
		EP 2245540 A2	03-11-2010
US 2002178239 A1	28-11-2002	US 2007266124 A1	15-11-2007

EPO FORM P0465

Pour tout renseignement concernant cette annexe : voir Journal Officiel de l'Office européen des brevets, No.12/82