(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
13 février 2003 (13.02.2003)

PCT

(10) Numéro de publication internationale
WO 03/012604 A2

(51) Classification internationale des brevets[7] : G06F 1/00

(21) Numéro de la demande internationale :
PCT/FR02/02340

(22) Date de dépôt international : 4 juillet 2002 (04.07.2002)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
01/10246 31 juillet 2001 (31.07.2001) FR

(71) Déposant *(pour tous les États désignés sauf US)* : **VALIDY** [FR/FR]; Zone Industrielle, 5, rue Jean Charcot, F-26100 Romans sur Isere (FR).

(72) Inventeurs; et

(75) Inventeurs/Déposants *(pour US seulement)* : **CUENOD, Jean-Christophe** [FR/FR]; 11, rue Edgar Degas, F-78360 Montesson (FR). **SGRO, Gilles** [FR/FR]; 3, impasse Jules Ferry, F-26300 Bourg de Peage (FR).

(74) Mandataire : **THIBAULT, Jean-Marc**; Cabinet Beau de Loménie, 51, avenue Jean-Jaurès, B.P. 7073, F-69301 Lyon Cedex 07 (FR).

(81) États désignés *(national)* : AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) États désignés *(régional)* : brevet ARIPO (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), brevet eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), brevet européen (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), brevet OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Publiée :

— *sans rapport de recherche internationale, sera republiée dès réception de ce rapport*

En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

(54) **Title:** METHOD FOR PROTECTING A SOFTWARE USING A SO-CALLED RENAMING PRINCIPLE AGAINST ITS UNAUTHORISED USE

(54) **Titre :** PROCEDE POUR PROTEGER UN LOGICIEL A L'AIDE D'UN PRINCIPE DIT DE "RENOMMAGE" CONTRE SON UTILISATION NON AUTORISEE

(57) **Abstract:** The invention concerns a method for protecting, from a unit, a vulnerable software against unauthorised use, said vulnerable software operating on a data processing system. The inventive method consists in defining: a set of dependent functions whereof the dependent functions are executable in a unit; a set of triggering commands, said triggering commands capable of being executed in the data processing system and of triggering execution of the dependent functions in a unit; for each triggering command, an instruction; a method for renaming the instructions; and reinstating means for implementation in a unit during a use phase, and enabling to recover the dependent function to be executed, from the renamed instruction.

(57) **Abrégé :** L'invention concerne un procédé pour protéger, à partir d'une unité, un logiciel vulnérable contre son utilisation non autorisée, ledit logiciel vulnérable fonctionnant sur un système de traitement de données. Le procédé selon l'invention consiste à définir : - un ensemble de fonctions dépendantes dont les fonctions dépendantes sont susceptibles d'être exécutées dans une unité, - un ensemble de commandes déclenchantes, ces commandes déclenchantes étant susceptibles d'être exécutées dans le système de traitement de données et de déclencher l'exécution dans une unité, des fonctions dépendantes, - pour chaque commande déclenchante, une consigne, - une méthode de renommage des consignes, - et des moyens de rétablissement destinés à être mis en uvre dans une unité au cours d'une phase d'utilisation, et permettant de retrouver la fonction dépendante à exécuter, à partir de la consigne renommée.

WO 03/012604 A2

PROCEDE POUR PROTEGER UN LOGICIEL A L'AIDE D'UN PRINCIPE DIT DE « RENOMMAGE » CONTRE SON UTILISATION NON AUTORISEE

La présente invention concerne le domaine technique des systèmes de traitement de données au sens général et elle vise, plus précisément, les moyens pour protéger, contre son utilisation non autorisée, un logiciel fonctionnant sur lesdits systèmes de traitement de données.

L'objet de l'invention vise, plus particulièrement, les moyens pour protéger un logiciel contre son utilisation non autorisée, à partir d'une unité de traitement et de mémorisation, une telle unité étant communément matérialisée par une carte à puce ou par une clé matérielle sur port USB.

Dans le domaine technique ci-dessus, le principal inconvénient concerne l'emploi non autorisé de logiciels par des utilisateurs n'ayant pas acquitté des droits de licence. Cette utilisation illicite de logiciels cause un préjudice manifeste pour les éditeurs de logiciels, les distributeurs de logiciels et/ou toute personne intégrant de tels logiciels dans des produits. Pour éviter de telles copies illicites, il a été proposé dans l'état de la technique, diverses solutions pour protéger des logiciels.

Ainsi, il est connu une solution de protection consistant à mettre en oeuvre un système matériel de protection, tel qu'un élément physique appelé clé de protection ou "dongle" en terminologie anglo-saxonne. Une telle clé de protection devrait garantir l'exécution du logiciel uniquement en présence de la clé. Or, il doit être constaté qu'une telle solution est inefficace car elle présente l'inconvénient d'être facilement contournable. Une personne mal intentionnée ou pirate peut, à l'aide d'outils spécialisés, tels que des désassembleurs, supprimer les instructions de contrôle de la clé de protection. Il devient alors possible de réaliser des copies illicites correspondant à des versions modifiées des logiciels n'ayant plus aucune protection. De plus, cette solution ne peut pas être généralisée à tous les logiciels, dans la mesure où il est difficile de connecter plus de deux clés de protection sur un même système.

L'objet de l'invention vise justement à remédier aux inconvénients énoncés ci-dessus en proposant un procédé pour protéger un logiciel contre son utilisation non autorisée, à partir d'une unité de traitement et de mémorisation ad hoc, dans la

mesure où la présence d'une telle unité est nécessaire pour que le logiciel soit complètement fonctionnel.

Pour atteindre un tel objectif, l'objet de l'invention concerne un procédé pour protéger, à partir d'au moins une unité vierge comportant au moins des moyens de traitement et des moyens de mémorisation, un logiciel vulnérable contre son utilisation non autorisée, ledit logiciel vulnérable fonctionnant sur un système de traitement de données. Le procédé selon l'invention consiste :

→ dans une phase de protection :

- à définir :
  - un ensemble de fonctions dépendantes dont les fonctions dépendantes sont susceptibles d'être exécutées dans une unité,
  - un ensemble de commandes déclenchantes pour cet ensemble de fonctions dépendantes, ces commandes déclenchantes étant susceptibles d'être exécutées dans le système de traitement de données et de déclencher l'exécution dans une unité, des fonctions dépendantes,
  - pour chaque commande déclenchante, une consigne correspondant au moins en partie à l'information transmise depuis le système de traitement de données vers une unité, afin de déclencher l'exécution de la fonction dépendante correspondante dans une unité, cette consigne se présentant sous la forme d'au moins un argument de la commande déclenchante,
  - une méthode de renommage des consignes permettant de renommer les consignes afin d'obtenir des commandes déclenchantes à consignes renommées,
  - et des moyens de rétablissement destinés à être mis en œuvre dans une unité au cours d'une phase d'utilisation, et permettant de retrouver la fonction dépendante à exécuter, à partir de la consigne renommée,
- à construire des moyens d'exploitation permettant de transformer l'unité vierge en une unité capable de mettre en œuvre les moyens de rétablissement,
- à créer un logiciel protégé :

- en choisissant, au moins un traitement algorithmique qui, lors de l'exécution du logiciel vulnérable, utilise au moins un opérande et permet d'obtenir au moins un résultat,
- en choisissant au moins une portion du source du logiciel vulnérable contenant au moins un traitement algorithmique choisi,
- en produisant le source du logiciel protégé à partir du source du logiciel vulnérable, en modifiant au moins une portion choisie du source du logiciel vulnérable pour obtenir au moins une portion modifiée du source du logiciel protégé, cette modification étant telle que :
  - lors de l'exécution du logiciel protégé une première partie d'exécution est exécutée dans le système de traitement de données et une seconde partie d'exécution est exécutée dans une unité, obtenue à partir de l'unité vierge après chargement d'informations,
  - la seconde partie d'exécution exécute au moins la fonctionnalité d'au moins un traitement algorithmique choisi,
  - au moins un traitement algorithmique choisi est décomposé de manière que lors de l'exécution du logiciel protégé, ce traitement algorithmique est exécuté, au moyen de la seconde partie d'exécution, en utilisant des fonctions dépendantes,
  - pour au moins un traitement algorithmique choisi, des commandes déclenchantes à consignes renommées sont intégrées dans le source du logiciel protégé, de manière que lors de l'exécution du logiciel protégé, chaque commande déclenchante à consigne renommée est exécutée par la première partie d'exécution et déclenche dans l'unité, le rétablissement, au moyen des moyens de rétablissement, de la consigne et l'exécution, au moyen de la seconde partie d'exécution, de la fonction dépendante correspondante,
  - et un ordonnancement des commandes déclenchantes à consignes renommées est choisi parmi l'ensemble des ordonnancements permettant l'exécution du logiciel protégé,

- et en produisant :
  - une première partie objet du logiciel protégé, à partir du source du logiciel protégé, cette première partie objet étant telle que lors de l'exécution du logiciel protégé, apparaît une première partie d'exécution qui est exécutée dans le système de traitement de données et dont au moins une portion prend en compte que les commandes déclenchantes à consignes renommées sont exécutées selon l'ordonnancement choisi,
  - et une seconde partie objet du logiciel protégé, contenant les moyens d'exploitation, cette seconde partie objet étant telle que, après chargement dans l'unité vierge et lors de l'exécution du logiciel protégé, apparaît la seconde partie d'exécution au moyen de laquelle les consignes sont rétablies et les fonctions dépendantes sont exécutées,
- et à charger la seconde partie objet dans l'unité vierge, en vue d'obtenir l'unité,

→ et dans une phase d'utilisation au cours de laquelle est exécuté le logiciel protégé:

- en présence de l'unité et à chaque fois qu'une commande déclenchante à consigne renommée, contenue dans une portion de la première partie d'exécution l'impose, à rétablir dans l'unité, l'identité de la fonction dépendante correspondante et à exécuter celle-ci, de sorte que cette portion est exécutée correctement et que, par conséquent, le logiciel protégé est complètement fonctionnel,
- et en l'absence de l'unité, malgré la demande d'une portion de la première partie d'exécution de déclencher l'exécution d'une fonction dépendante dans l'unité, à ne pas pouvoir répondre correctement à cette demande, de sorte qu'au moins cette portion n'est pas exécutée correctement et que, par conséquent, le logiciel protégé n'est pas complètement fonctionnel.

Selon une variante de réalisation, le procédé selon l'invention consiste :

→ dans la phase de protection :

- à définir pour au moins une fonction dépendante, une famille de fonctions

dépendantes algorithmiquement équivalentes, mais déclenchées par des commandes déclenchantes dont les consignes renommées sont différentes,

- et à modifier le logiciel protégé :
  - en choisissant dans le source du logiciel protégé au moins une commande déclenchante à consigne renommée,
  - et en modifiant au moins une portion choisie du source du logiciel protégé en remplaçant au moins la consigne renommée d'une commande déclenchante à consigne renommée choisie, par une autre consigne renommée, déclenchant une fonction dépendante de la même famille.

Selon une variante de réalisation, le procédé selon l'invention consiste :

→ dans la phase de protection :

- à définir :
  - en tant que méthode de renommage des consignes, une méthode de chiffrement pour chiffrer les consignes,
  - et en tant que moyens de rétablissement, des moyens mettant en oeuvre une méthode de déchiffrement pour déchiffrer les consignes renommées et rétablir ainsi l'identité des fonctions dépendantes à exécuter dans l'unité.

Selon une forme préférée de réalisation, le procédé selon l'invention consiste :

→ dans la phase de protection :

- à modifier le logiciel protégé:
  - en choisissant au moins une variable utilisée dans au moins un traitement algorithmique choisi, qui lors de l'exécution du logiciel protégé, définit partiellement l'état du logiciel protégé,
  - en modifiant au moins une portion choisie du source du logiciel protégé, cette modification étant telle que lors de l'exécution du logiciel protégé, au moins une variable choisie ou au moins une copie de variable choisie réside dans l'unité,
  - et en produisant :
    - la première partie objet du logiciel protégé, cette première partie

objet étant telle que lors de l'exécution du logiciel protégé, au moins une portion de la première partie d'exécution prend aussi en compte qu'au moins une variable ou au moins une copie de variable réside dans l'unité,

- et la seconde partie objet du logiciel protégé, cette seconde partie objet étant telle que, après chargement dans l'unité et lors de l'exécution du logiciel protégé, apparaît la seconde partie d'exécution au moyen de laquelle au moins une variable choisie, ou au moins une copie de variable choisie réside aussi dans l'unité,

→ et dans la phase d'utilisation :

- en présence de l'unité à chaque fois qu'une portion de la première partie d'exécution l'impose, à utiliser une variable ou une copie de variable résidant dans l'unité, de sorte que cette portion est exécutée correctement et que, par conséquent, le logiciel protégé est complètement fonctionnel,
- et en l'absence de l'unité, malgré la demande d'une portion de la première partie d'exécution d'utiliser une variable ou une copie de variable résidant dans l'unité, à ne pouvoir répondre correctement à cette demande, de sorte qu'au moins cette portion n'est pas exécutée correctement et que, par conséquent, le logiciel protégé n'est pas complètement fonctionnel.

Selon une autre forme préférée de réalisation, le procédé selon l'invention consiste :

→ dans la phase de protection :

- à définir :
  - un jeu de fonctions élémentaires, sous-ensenble de l'ensemble des fonctions dépendantes,
  - et un jeu de commandes élémentaires pour ce jeu de fonctions élémentaires, ce jeu de commandes élémentaires étant un sous-ensemble de l'ensemble des commandes déclenchantes,
- à construire les moyens d'exploitation permettant à l'unité, d'exécuter aussi les fonctions élémentaires dudit jeu, l'exécution de ces fonctions élémentaires étant déclenchée par l'exécution dans le système de traitement

de données, des commandes élémentaires dont la consigne a été renommée,

- et à modifier le logiciel protégé:
  - en modifiant au moins une portion choisie du source du logiciel protégé, cette modification étant telle que la décomposition d'au moins un traitement algorithmique choisi en fonctions dépendantes n'utilise que des fonctions élémentaires,
  - en produisant :
    - la première partie objet du logiciel protégé, cette première partie objet étant telle que lors de l'exécution du logiciel protégé, au moins une portion de la première partie d'exécution exécute aussi les commandes élémentaires selon l'ordonnancement choisi,
    - et la seconde partie objet du logiciel protégé contenant aussi les moyens d'exploitation, cette seconde partie objet étant telle que, après chargement dans l'unité et lors de l'exécution du logiciel protégé, apparaît la seconde partie d'exécution au moyen de laquelle sont aussi exécutées les fonctions élémentaires déclenchées par la première partie d'exécution,

→ et dans la phase d'utilisation :

- en présence de l'unité et à chaque fois qu'une commande élémentaire contenue dans une portion de la première partie d'exécution l'impose, à exécuter la fonction élémentaire correspondante dans l'unité, de sorte que cette portion est exécutée correctement et que, par conséquent, le logiciel protégé est complètement fonctionnel,
- et en l'absence de l'unité, malgré la demande d'une portion de la première partie d'exécution, de déclencher l'exécution d'une fonction élémentaire dans l'unité, à ne pas pouvoir répondre correctement à cette demande, de sorte qu'au moins cette portion n'est pas exécutée correctement et que, par conséquent, le logiciel protégé n'est pas complètement fonctionnel.

Selon une autre forme préférée de réalisation, le procédé selon l'invention consiste :

→ dans la phase de protection :

- à définir :

- au moins une caractéristique d'exécution de logiciel, susceptible d'être surveillée au moins en partie dans l'unité,
- au moins un critère à respecter pour au moins une caractéristique d'exécution de logiciel,
- des moyens de détection à mettre en oeuvre dans l'unité et permettant de détecter qu'au moins une caractéristique d'exécution de logiciel ne respecte pas au moins un critère associé,
- et des moyens de coercition à mettre en oeuvre dans l'unité et permettant d'informer le système de traitement de données et/ou de modifier l'exécution d'un logiciel, lorsqu'au moins un critère n'est pas respecté,

• à construire les moyens d'exploitation permettant à l'unité, de mettre aussi en oeuvre les moyens de détection et les moyens de coercition,

• et à modifier le logiciel protégé :

- en choisissant au moins une caractéristique d'exécution de logiciel à surveiller, parmi les caractéristiques d'exécution susceptibles d'être surveillées,
- en choisissant au moins un critère à respecter pour au moins une caractéristique d'exécution de logiciel choisie,
- en choisissant dans le source du logiciel protégé, des fonctions élémentaires pour lesquelles au moins une caractéristique d'exécution de logiciel choisie, est à surveiller,
- en modifiant au moins une portion choisie du source du logiciel protégé, cette modification étant telle que lors de l'exécution du logiciel protégé, au moins une caractéristique d'exécution choisie est surveillée au moyen de la seconde partie d'exécution, et le non respect d'un critère aboutit à une information du système de traitement de données et/ou à une modification de l'exécution du logiciel protégé,
- et en produisant la seconde partie objet du logiciel protégé contenant les moyens d'exploitation mettant aussi en oeuvre les moyens de détection et les moyens de coercition, cette seconde partie objet étant

telle que, après chargement dans l'unité et lors de l'exécution du logiciel protégé, au moins une caractéristique d'exécution de logiciel est surveillée et le non respect d'un critère aboutit à une information du système de traitement de données et/ou à une modification de l'exécution du logiciel protégé,

→ et dans la phase d'utilisation :

- en présence de l'unité :
  - tant que tous les critères correspondant à toutes les caractéristiques d'exécution surveillées de toutes les portions modifiées du logiciel protégé sont respectés, à permettre le fonctionnement nominal de ces portions du logiciel protégé et par conséquent à permettre le fonctionnement nominal du logiciel protégé,
  - et si au moins un des critères correspondant à une caractéristique d'exécution surveillée d'une portion du logiciel protégé n'est pas respecté, à en informer le système de traitement de données et/ou à modifier le fonctionnement de la portion du logiciel protégé, de sorte que le fonctionnement du logiciel protégé est modifié.

Selon une variante de réalisation, le procédé selon l'invention consiste :

→ dans la phase de protection :

- à définir :
  - en tant que caractéristique d'exécution de logiciel susceptible d'être surveillée, une variable de mesure de l'utilisation d'une fonctionnalité d'un logiciel,
  - en tant que critère à respecter, au moins un seuil associé à chaque variable de mesure,
  - et des moyens d'actualisation permettant de mettre à jour au moins une variable de mesure,
- à construire les moyens d'exploitation permettant à l'unité de mettre aussi en œuvre les moyens d'actualisation,
- et à modifier le logiciel protégé :
  - en choisissant en tant que caractéristique d'exécution de logiciel à

surveiller, au moins une variable de mesure de l'utilisation d'une fonctionnalité d'un logiciel,

- en choisissant :
  - ➢ au moins une fonctionnalité du logiciel protégé dont l'utilisation est susceptible d'être surveillée grâce à une variable de mesure,
  - ➢ au moins une variable de mesure servant à quantifier l'utilisation de ladite fonctionnalité,
  - ➢ au moins un seuil associé à une variable de mesure choisie correspondant à une limite d'utilisation de ladite fonctionnalité,
  - ➢ et au moins une méthode de mise à jour d'une variable de mesure choisie en fonction de l'utilisation de ladite fonctionnalité,
- et en modifiant au moins une portion choisie du source du logiciel protégé, cette modification étant telle que, lors de l'exécution du logiciel protégé, la variable de mesure est actualisée au moyen de la seconde partie d'exécution, en fonction de l'utilisation de ladite fonctionnalité et au moins un dépassement de seuil est pris en compte,

→ et dans la phase d'utilisation, en présence de l'unité, et dans le cas où il est détecté au moins un dépassement de seuil correspondant à au moins une limite d'utilisation, à en informer le système de traitement de données et/ou à modifier le fonctionnement de la portion du logiciel protégé, de sorte que le fonctionnement du logiciel protégé est modifié.

Selon une variante de réalisation, le procédé selon l'invention consiste :

→ dans la phase de protection :

- à définir :
  - pour au moins une variable de mesure, plusieurs seuils associés,
  - et des moyens de coercition différents correspondant à chacun de ces seuils,
- et à modifier le logiciel protégé :
  - en choisissant dans le source du logiciel protégé, au moins une variable de mesure choisie à laquelle doivent être associés plusieurs seuils correspondants à des limites différentes d'utilisation de la fonctionnalité,

- en choisissant au moins deux seuils associés à la variable de mesure choisie,
- et en modifiant au moins une portion choisie du source du logiciel protégé, cette modification étant telle que, lors de l'exécution du logiciel protégé, les dépassements des divers seuils sont pris en compte, au moyen de la seconde partie d'exécution, de manière différente,

→ et dans la phase d'utilisation :

- en présence de l'unité :
  - dans le cas où il est détecté le dépassement d'un premier seuil, à enjoindre le logiciel protégé de ne plus utiliser la fonctionnalité correspondante,
  - et dans le cas où il est détecté le dépassement d'un deuxième seuil, à rendre inopérante la fonctionnalité correspondante et/ou au moins une portion du logiciel protégé.

Selon une variante de réalisation, le procédé selon l'invention consiste :

→ dans la phase de protection :

- à définir des moyens de rechargement permettant de créditer au moins une utilisation supplémentaire pour au moins une fonctionnalité de logiciel surveillée par une variable de mesure,
- à construire les moyens d'exploitation permettant aussi à l'unité de mettre en œuvre les moyens de rechargement,
- et à modifier le logiciel protégé :
  - en choisissant dans le source du logiciel protégé, au moins une variable de mesure choisie permettant de limiter l'utilisation d'une fonctionnalité à laquelle au moins une utilisation supplémentaire doit pouvoir être créditée,
  - et en modifiant au moins une portion choisie, cette modification étant telle que dans une phase dite de rechargement, au moins une utilisation supplémentaire d'au moins une fonctionnalité correspondant à une variable de mesure choisie peut être créditée,

→ et dans la phase de rechargement :

- à réactualiser au moins une variable de mesure choisie et/ou au moins un seuil associé, de manière à permettre au moins une utilisation supplémentaire de la fonctionnalité.

Selon une variante de réalisation, le procédé selon l'invention consiste :

→ dans la phase de protection :

- à définir :
  - en tant que caractéristique d'exécution de logiciel susceptible d'être surveillée, un profil d'utilisation de logiciel,
  - et en tant que critère à respecter, au moins un trait d'exécution de logiciel,
- et à modifier le logiciel protégé :
  - en choisissant en tant que caractéristique d'exécution de logiciel à surveiller au moins un profil d'utilisation de logiciel,
  - en choisissant au moins un trait d'exécution qu'au moins un profil d'utilisation choisi doit respecter,
  - et en modifiant au moins une portion choisie du source du logiciel protégé, cette modification étant telle que, lors de l'exécution du logiciel protégé, la seconde partie d'exécution respecte tous les traits d'exécution choisis,

→ et dans la phase d'utilisation en présence de l'unité, et dans le cas où il est détecté qu'au moins un trait d'exécution n'est pas respecté, à en informer le système de traitement de données et/ou à modifier le fonctionnement de la portion du logiciel protégé, de sorte que le fonctionnement du logiciel protégé est modifié.

Selon une variante de réalisation, le procédé selon l'invention consiste :

→ dans la phase de protection :

- à définir :
  - un jeu d'instructions dont les instructions sont susceptibles d'être exécutées dans l'unité,
  - un jeu de commandes d'instructions pour ce jeu d'instructions, ces

commandes d'instructions étant susceptibles d'être exécutées dans le système de traitement de données et de déclencher dans l'unité l'exécution des instructions,

- en tant que profil d'utilisation, l'enchaînement des instructions,
- en tant que trait d'exécution, un enchaînement souhaité pour l'exécution des instructions,
- en tant que moyens de détection, des moyens permettant de détecter que l'enchaînement des instructions ne correspond pas à celui souhaité,
- et en tant que moyens de coercition, des moyens permettant d'informer le système de traitement de données et/ou de modifier le fonctionnement de la portion de logiciel protégé lorsque l'enchaînement des instructions ne correspond pas à celui souhaité,

• à construire les moyens d'exploitation permettant aussi à l'unité d'exécuter les instructions du jeu d'instructions, l'exécution de ces instructions étant déclenchée par l'exécution dans le système de traitement de données, des commandes d'instructions,

• et à modifier le logiciel protégé :

- en modifiant au moins une portion choisie du source du logiciel protégé :
  - ➢ en transformant les fonctions élémentaires en instructions,
  - ➢ en spécifiant l'enchaînement que doivent respecter au moins certaines des instructions lors de leur exécution dans l'unité,
  - ➢ et en transformant les commandes élémentaires en commandes d'instructions correspondant aux instructions utilisées,

→ et dans la phase d'utilisation, en présence de l'unité, dans le cas où il est détecté que l'enchaînement des instructions exécutées dans l'unité ne correspond pas à celui souhaité, à en informer le système de traitement de données et/ou à modifier le fonctionnement de la portion du logiciel protégé, de sorte que le fonctionnement du logiciel protégé est modifié.

Selon une variante de réalisation, le procédé selon l'invention consiste :

→ dans la phase de protection :

• à définir :

- en tant que jeu d'instructions, un jeu d'instructions dont au moins certaines instructions travaillent sur des registres et utilisent au moins un opérande en vue de rendre un résultat,
- pour au moins une partie des instructions travaillant sur des registres :
  - ➢ une partie définissant la fonctionnalité de l'instruction,
  - ➢ et une partie définissant l'enchaînement souhaité pour l'exécution des instructions et comportant des champs de bits correspondant à :
    - ◊ un champ d'identification de l'instruction,
    - ◊ et pour chaque opérande de l'instruction :
      - * un champ drapeau,
      - * et un champ d'identification prévue de l'opérande,
- pour chaque registre appartenant aux moyens d'exploitation et utilisé par le jeu d'instructions, un champ d'identification générée dans lequel est automatiquement mémorisée l'identification de la dernière instruction ayant rendu son résultat dans ce registre,
- en tant que moyens de détection, des moyens permettant, lors de l'exécution d'une instruction, pour chaque opérande, lorsque le champ drapeau l'impose, de contrôler l'égalité entre le champ d'identification générée correspondant au registre utilisé par cet opérande, et le champ d'identification prévue de l'origine de cet opérande,
- et en tant que moyens de coercition, des moyens permettant de modifier le résultat des instructions, si au moins une des égalités contrôlées est fausse.

Selon une autre forme préférée de réalisation, le procédé selon l'invention consiste :

→ dans la phase de protection :

- à modifier le logiciel protégé :
  - en choisissant dans le source du logiciel protégé, au moins un branchement conditionnel effectué dans au moins un traitement algorithmique choisi,

- en modifiant au moins une portion choisie du source du logiciel protégé, cette modification étant telle que lors de l'exécution du logiciel protégé, la fonctionnalité d'au moins un branchement conditionnel choisi, est exécutée, au moyen de la seconde partie d'exécution, dans l'unité,
- et en produisant :
  - la première partie objet du logiciel protégé, cette première partie objet étant telle que lors de l'exécution du logiciel protégé, la fonctionnalité d'au moins un branchement conditionnel choisi est exécutée dans l'unité,
  - et la seconde partie objet du logiciel protégé, cette seconde partie objet étant telle que, après chargement dans l'unité et lors de l'exécution du logiciel protégé, apparaît la seconde partie d'exécution au moyen de laquelle la fonctionnalité d'au moins un branchement conditionnel choisi est exécutée,

→ et dans la phase d'utilisation :

- en présence de l'unité et à chaque fois qu'une portion de la première partie d'exécution l'impose, à exécuter la fonctionnalité d'au moins un branchement conditionnel dans l'unité, de sorte que cette portion est exécutée correctement et que, par conséquent, le logiciel protégé est complètement fonctionnel,
- et en l'absence de l'unité et malgré la demande d'une portion de la première partie d'exécution d'exécuter la fonctionnalité d'un branchement conditionnel dans l'unité, à ne pas pouvoir répondre correctement à cette demande, de sorte qu'au moins cette portion n'est pas exécutée correctement et que par conséquent, le logiciel protégé n'est pas complètement fonctionnel.

Selon une variante de réalisation, le procédé selon l'invention consiste, dans la phase de protection, à modifier le logiciel protégé :

- en choisissant, dans le source du logiciel protégé au moins une série de branchements conditionnels choisis,
- en modifiant au moins une portion choisie du source du logiciel

protégé, cette modification étant telle que lors de l'exécution du logiciel protégé, la fonctionnalité globale d'au moins une série choisie de branchements conditionnels est exécutée au moyen de la seconde partie d'exécution, dans l'unité,

- et en produisant :
  - la première partie objet du logiciel protégé, cette première partie objet étant telle que lors de l'exécution du logiciel protégé, la fonctionnalité d'au moins une série choisie de branchements conditionnels est exécutée dans l'unité,
  - et la seconde partie objet du logiciel protégé, cette seconde partie objet étant telle que, après chargement dans l'unité et lors de l'exécution du logiciel protégé, apparaît la seconde partie d'exécution au moyen de laquelle la fonctionnalité globale d'au moins une série choisie de branchements conditionnels est exécutée.

Le procédé selon l'invention permet ainsi de protéger l'utilisation d'un logiciel par la mise en oeuvre d'une unité de traitement et de mémorisation qui présente la particularité de contenir une partie du logiciel en cours d'exécution. Il s'ensuit que toute version dérivée du logiciel tentant de fonctionner sans l'unité de traitement et de mémorisation impose de recréer la partie du logiciel contenue dans l'unité de traitement et de mémorisation lors de l'exécution, sous peine que cette version dérivée du logiciel ne soit pas complètement fonctionnelle.

Diverses autres caractéristiques ressortent de la description faite ci-dessous en référence aux dessins annexés qui montrent, à titre d'exemples non limitatifs, des formes de réalisation et de mise en oeuvre de l'objet de l'invention.

Les **fig. 10** et **11** sont des schémas blocs fonctionnels illustrant les diverses représentations d'un logiciel respectivement non protégé et protégé par le procédé conforme à l'invention.

Les **fig. 20** à **22** illustrent à titre d'exemples, diverses formes de réalisation d'un dispositif de mise en oeuvre du procédé conforme à l'invention.

Les **fig. 30** et **31** sont des schémas blocs fonctionnels explicitant le principe général du procédé conforme à l'invention.

Les **fig. 40** à **43** sont des schémas illustrant le procédé de protection selon l'invention mettant en oeuvre le principe de protection par variable.

Les **fig. 60** à **64** sont des schémas illustrant le procédé de protection selon l'invention mettant en oeuvre le principe de protection par fonctions élémentaires.

Les **fig. 70** à **74** sont des schémas illustrant le procédé de protection selon l'invention mettant en oeuvre le principe de protection par détection et coercition.

Les **fig. 80** à **85** sont des schémas illustrant le procédé de protection selon l'invention mettant en oeuvre le principe de protection par renommage.

Les **fig. 90** à **92** sont des schémas illustrant le procédé de protection selon l'invention mettant en oeuvre le principe de protection par branchement conditionnel.

La **fig. 100** est un schéma illustrant les différentes phases de mise en oeuvre de l'objet de l'invention.

La **fig. 110** illustre un exemple de réalisation d'un système permettant la mise en oeuvre du stade de construction de la phase de protection conforme à l'invention.

La **fig. 120** illustre un exemple de réalisation d'une unité de pré-personnalisation utilisée dans le procédé de protection conforme à l'invention.

La **fig. 130** illustre un exemple de réalisation d'un système permettant la mise en oeuvre du stade de confection d'outils de la phase de protection conforme à l'invention.

La **fig. 140** illustre un exemple de réalisation d'un système permettant la mise en oeuvre du procédé de protection selon l'invention.

La **fig. 150** illustre un exemple de réalisation d'une unité de personnalisation utilisée dans le procédé de protection conforme à l'invention.

Dans la suite de la description, les définitions suivantes seront utilisées :

- Un système de traitement de données **3** est un système capable d'exécuter un programme.
- Une unité de traitement et de mémorisation est une unité capable :
  - d'accepter des données fournies par un système de traitement de données **3**,
  - de restituer des données au système de traitement de données **3**,
  - de stocker des données au moins en partie de manière secrète et de conserver au moins une partie de celles-ci même lorsque l'unité est

hors tension,

- et d'effectuer du traitement algorithmique sur des données, une partie ou la totalité de ce traitement étant secret.

- Une unité **6** est une unité de traitement et de mémorisation mettant en oeuvre le procédé selon l'invention.
- Une unité vierge **60** est une unité qui ne met pas en oeuvre le procédé selon l'invention, mais qui peut recevoir des informations la transformant en une unité **6**.
- Une unité pré-personnalisée **66** est une unité vierge **60** ayant reçue une partie des informations lui permettant, après réception d'informations complémentaires, d'être transformée en une unité **6**.
- Le chargement d'informations dans une unité vierge **60** ou une unité pré-personnalisée **66** correspond à un transfert d'informations dans l'unité vierge **60** ou l'unité pré-personnalisée **66**, et à un stockage desdites informations transférées. Eventuellement, le transfert peut comporter un changement de format des informations.
- Une variable, une donnée ou une fonction contenue dans le système de traitement de données **3** sera indiquée par une majuscule, tandis qu'une variable, une donnée ou une fonction contenue dans l'unité **6** sera indiquée par une minuscule.
- Un "logiciel protégé", est un logiciel ayant été protégé par au moins un principe de protection mis en oeuvre par le procédé conforme à l'invention.
- Un "logiciel vulnérable", est un logiciel n'ayant été protégé par aucun principe de protection mis en oeuvre par le procédé conforme à l'invention.
- Dans le cas où la différenciation entre un logiciel vulnérable et un logiciel protégé n'a pas d'importance, le terme "logiciel" est utilisé.
- Un logiciel se présente sous diverses représentations selon l'instant considéré dans son cycle de vie :
  - une représentation source,
  - une représentation objet,
  - une distribution,

– ou une représentation dynamique.

- Une représentation source d'un logiciel est comprise comme une représentation qui après transformation, donne une représentation objet. Une représentation source peut se présenter selon différents niveaux, d'un niveau conceptuel abstrait à un niveau exécutable directement par un système de traitement de données ou une unité de traitement et de mémorisation.
- Une représentation objet d'un logiciel correspond à un niveau de représentation qui après transfert dans une distribution puis chargement dans un système de traitement de données ou une unité de traitement et de mémorisation, peut être exécuté. Il peut s'agir, par exemple, d'un code binaire, d'un code interprété, etc.
- Une distribution est un support physique ou virtuel contenant la représentation objet, cette distribution devant être mise à disposition de l'utilisateur pour lui permettre d'utiliser le logiciel.
- Une représentation dynamique correspond à l'exécution du logiciel à partir de sa distribution.
- Une portion de logiciel correspond à une partie quelconque de logiciel et peut, par exemple correspondre, à une ou plusieurs instructions consécutives ou non, et/ou à un ou plusieurs blocs fonctionnels consécutifs ou non, et/ou à une ou plusieurs fonctions, et/ou un ou plusieurs sous programmes, et/ou un ou plusieurs modules. Une portion d'un logiciel peut correspondre aussi à la totalité de ce logiciel.

Les **fig. 10** et **11** illustrent les diverses représentations respectivement d'un logiciel vulnérable **2v** au sens général, et d'un logiciel protégé **2p** selon le procédé conforme à l'invention.

La **fig. 10** illustre diverses représentations d'un logiciel vulnérable **2v** apparaissant au cours de son cycle de vie. Le logiciel vulnérable **2v** peut ainsi apparaître sous l'une des représentations suivantes :

- une représentation source **2vs**,
- une représentation objet **2vo**,

- une distribution **2vd**. Cette distribution peut se présenter communément sous la forme d'un moyen de distribution physique tel qu'un CDROM ou sous la forme de fichiers distribués à travers un réseau (GSM, Internet, ...),
- ou d'une représentation dynamique **2ve** correspondant à l'exécution du logiciel vulnérable **2v** sur un système de traitement de données **3** de tous types connus, qui comporte de manière classique, au moins un processeur **4**.

La **fig. 11** illustre diverses représentations d'un logiciel protégé **2p** apparaissant au cours de son cycle de vie. Le logiciel protégé **2p** peut ainsi apparaître sous l'une des représentations suivantes :

- une représentation source **2ps** comportant une première partie source destinée au système de traitement de données **3** et une seconde partie source destinée à l'unité **6**, une partie de ces parties source pouvant communément être contenue dans des fichiers communs,
- une représentation objet **2po** comportant une première partie objet **2pos** destinée au système de traitement de données **3** et une seconde partie objet **2pou** destinée à l'unité **6**,
- une distribution **2pd** comportant :
  - une première partie de distribution **2pds** contenant la première partie objet **2pos**, cette première partie de distribution **2pds** étant destinée au système de traitement de données **3** et pouvant se présenter communément sous la forme d'un moyen de distribution physique tel qu'un CDROM, ou sous la forme de fichiers distribués à travers un réseau (GSM, Internet, ...),
  - et une seconde partie de distribution **2pdu** se présentant sous la forme :
    - d'au moins une unité pré-personnalisée **66** sur laquelle une partie de la seconde partie objet **2pou** a été chargée et pour laquelle l'utilisateur doit terminer la personnalisation en chargeant des informations complémentaires, afin d'obtenir une unité **6**, ces informations complémentaires étant obtenues, par exemple, par chargement ou téléchargement à travers un réseau,

- ou d'au moins une unité **6** sur laquelle la seconde partie objet **2pou** a été chargée,

- ou une représentation dynamique **2pe** correspondant à l'exécution du logiciel protégé **2p**. Cette représentation dynamique **2pe** comporte une première partie d'exécution **2pes** qui est exécutée dans le système de traitement de données **3** et une seconde partie d'exécution **2peu** qui est exécutée dans l'unité **6**.

Dans le cas où la différenciation entre les différentes représentations du logiciel protégé **2p** n'a pas d'importance, les expressions première partie du logiciel protégé et deuxième partie du logiciel protégé sont utilisées.

La mise en oeuvre du procédé selon l'invention conformément à la représentation dynamique de la **fig. 11**, utilise un dispositif **1p** comportant un système de traitement de données **3** relié par une liaison **5** à une unité **6**. Le système de traitement de données **3** est de tous types et comporte, de manière classique, au moins un processeur **4**. Le système de traitement de données **3** peut être un ordinateur ou faire partie, par exemple, de diverses machines, dispositifs, produits fixes ou mobiles, ou véhicules au sens général. La liaison **5** peut être réalisée de toute manière possible, telle que par exemple par une ligne série, un bus USB, une liaison radio, une liaison optique, une liaison réseau ou une connexion électrique directe sur un circuit du système de traitement de données **3**, etc. Il est à noter que l'unité **6** peut éventuellement se trouver physiquement à l'intérieur du même circuit intégré que le processeur **4** du système de traitement de données **3**. Dans ce cas, l'unité **6** peut être considérée comme un co-processeur par rapport au processeur **4** du système de traitement de données **3** et la liaison **5** est interne au circuit intégré.

Les **fig. 20** à **22** montrent de manière illustrative et à titre non limitatif, diverses formes de réalisation du dispositif **1p** permettant la mise en oeuvre du procédé de protection conforme à l'invention.

Dans l'exemple de réalisation illustré à la **fig. 20**, le dispositif de protection **1p** comporte, en tant que système de traitement de données **3**, un ordinateur et, en tant qu'unité **6**, une carte à puce **7** et son interface **8** communément appelé lecteur de carte. L'ordinateur **3** est relié à l'unité **6** par une liaison **5**. Lors de l'exécution d'un logiciel protégé **2p**, la première partie d'exécution **2pes** qui est exécutée dans

l'ordinateur **3** et la seconde partie d'exécution **2peu** qui est exécutée dans la carte à puce **7** et son interface **8**, doivent toutes les deux être fonctionnelles afin que le logiciel protégé **2p** soit complètement fonctionnel.

Dans l'exemple de réalisation illustré à la **fig. 21**, le dispositif de protection **1p** équipe un produit **9** au sens général, comportant divers organes **10** adaptés à la ou les fonctions assumées par un tel produit **9**. Le dispositif de protection **1p** comporte, d'une part, un système de traitement de données **3** embarqué dans le produit **9** et, d'autre part, une unité **6** associée au produit **9.** Pour que le produit **9** soit entièrement fonctionnel, le logiciel protégé **2p** doit être complètement fonctionnel. Ainsi, lors de l'exécution du logiciel protégé **2p**, la première partie d'exécution **2pes** qui est exécutée dans le système de traitement de données **3** et la seconde partie d'exécution **2peu** qui est exécutée dans l'unité **6,** doivent toutes les deux être fonctionnelles. Ce logiciel protégé **2p** permet donc de manière indirecte, de protéger contre une utilisation non autorisée, le produit **9** ou l'une de ses fonctionnalités. Par exemple, le produit **9** peut être une installation, un système, une machine, un jouet, un appareil électroménager, un téléphone, etc..

Dans l'exemple de réalisation illustré à la **fig. 22**, le dispositif de protection **1p** inclut plusieurs ordinateurs, ainsi qu'une partie d'un réseau de communication. Le système de traitement de données **3** est un premier ordinateur relié par une liaison **5** de type réseau, à une unité **6** constituée par un deuxième ordinateur. Pour la mise en oeuvre de l'invention, le deuxième ordinateur **6** est utilisé comme un serveur de licences pour un logiciel protégé **2p**. Lors de l'exécution du logiciel protégé **2p**, la première partie d'exécution **2pes** qui est exécutée dans le premier ordinateur **3** et la seconde partie d'exécution **2peu** qui est exécutée dans le deuxième ordinateur **6**, doivent toutes les deux être fonctionnelles afin que le logiciel protégé **2p** soit complètement fonctionnel.

La **fig. 30** permet d'expliciter de manière plus précise, le procédé de protection conforme à l'invention. Il est à noter qu'un logiciel vulnérable **2v** est considéré comme étant exécuté totalement dans un système de traitement de données **3**. Par contre, dans le cas de la mise en oeuvre d'un logiciel protégé **2p**, le système de traitement de données **3** comporte des moyens de transfert **12** reliés par la liaison **5**, à des moyens de transfert **13** faisant partie de l'unité **6** permettant de faire

communiquer entre elles, la première partie d'exécution **2pes** et la seconde partie d'exécution **2peu** du logiciel protégé **2p**.

Il doit être considéré que les moyens de transfert **12, 13** sont de nature logicielle et/ou matérielle et sont aptes à assurer et, éventuellement, à optimiser la communication des données entre le système de traitement de données **3** et l'unité **6**. Ces moyens de transfert **12, 13** sont adaptés pour permettre de disposer d'un logiciel protégé **2p** qui est, de préférence, indépendant du type de la liaison **5** utilisée. Ces moyens de transfert **12, 13** ne font pas partie de l'objet de l'invention et ne sont pas décrits plus précisément car ils sont bien connus de l'Homme de l'art. La première partie du logiciel protégé **2p** comporte des commandes. Lors de l'exécution du logiciel protégé **2p**, l'exécution de ces commandes par la première partie d'exécution **2pes** permet la communication entre la première partie d'exécution **2pes** et la seconde partie d'exécution **2peu**. Dans la suite de la description, ces commandes sont représentées par **IN**, **OUT** ou **TRIG**.

Tel qu'illustré à la **fig. 31**, pour permettre la mise en oeuvre de la seconde partie d'exécution **2peu** du logiciel protégé **2p**, l'unité **6** comporte des moyens de protection **14**. Les moyens de protection **14** comportent des moyens de mémorisation **15** et des moyens de traitement **16**.

Par souci de simplification dans la suite de la description, il est choisi de considérer, lors de l'exécution du logiciel protégé **2p**, la présence de l'unité **6** ou l'absence de l'unité **6**. En réalité, une unité **6** présentant des moyens de protection **14** inadaptés à l'exécution de la seconde partie d'exécution **2peu** du logiciel protégé **2p** est aussi considérée comme absente, à chaque fois que l'exécution du logiciel protégé **2p** n'est pas correct. En d'autres termes :

- une unité **6** physiquement présente et comportant des moyens de protection **14** adaptés à l'exécution de la seconde partie d'exécution **2peu** du logiciel protégé **2p**, est toujours considérée comme présente,
- une unité **6** physiquement présente mais comportant des moyens de protection **14** inadaptés, c'est-à-dire ne permettant pas la mise en oeuvre correcte de la seconde partie d'exécution **2peu** du logiciel protégé **2p** est considérée comme présente, lorsqu'elle fonctionne correctement, et comme absente lorsqu'elle ne fonctionne pas correctement,

- et une unité **6** physiquement absente est toujours considérée comme absente.

Dans le cas où l'unité **6** est constituée par une carte à puce **7** et son interface **8**, les moyens de transfert **13** sont décomposés en deux parties dont l'une se trouve sur l'interface **8** et dont l'autre se trouve sur la carte à puce **7**. Dans cet exemple de réalisation, l'absence de la carte à puce **7** est considérée comme équivalente à l'absence de l'unité **6**. En d'autres termes, en l'absence de la carte à puce **7** et/ou de son interface **8**, les moyens de protection **14** ne sont pas accessibles et ne permettent donc pas l'exécution de la seconde partie d'exécution **2peu** du logiciel protégé, de sorte que le logiciel protégé **2p** n'est pas complètement fonctionnel.

Conformément à l'invention, le procédé de protection vise à mettre en œuvre un principe de protection, dit par "renommage" dont une description est effectuée en relation des **fig. 80** à **85**.

Pour la mise en œuvre du principe de protection par renommage, il est défini :

- un ensemble de fonctions dépendantes, dont les fonctions dépendantes sont susceptibles d'être exécutées, au moyen de la seconde partie d'exécution **2peu**, dans l'unité **6**, et éventuellement de transférer des données entre le système de traitement de données **3** et l'unité **6**, cet ensemble de fonctions dépendantes pouvant être fini ou infini,
- un ensemble de commandes déclenchantes pour ces fonctions dépendantes, ces commandes déclenchantes étant susceptibles d'être exécutées dans le système de traitement de données **3** et de déclencher dans l'unité **6**, l'exécution de fonctions dépendantes correspondantes,
- pour chaque commande déclenchante, une consigne correspondant au moins en partie à l'information transmise par la première partie d'exécution **2pes**, à la seconde partie d'exécution **2peu**, afin de déclencher l'exécution de la fonction dépendante correspondante, cette consigne se présentant sous la forme d'au moins un argument de la commande déclenchante,
- une méthode de renommage des consignes destinée à être mise en oeuvre lors de la modification du logiciel vulnérable, une telle méthode permettant de renommer les consignes afin d'obtenir des commandes déclenchantes à consignes renommées permettant de dissimuler l'identité des fonctions

dépendantes correspondantes,

- et des moyens de rétablissement **20** destinés à être mis en oeuvre dans l'unité **6** lors de la phase d'utilisation et permettant de retrouver la consigne initiale, à partir de la consigne renommée, afin de retrouver la fonction dépendante à exécuter.

Pour la mise en oeuvre du principe de protection par renommage, il est aussi construit des moyens d'exploitation permettant de transformer une unité vierge **60** en une unité **6** mettant au moins en oeuvre les moyens de rétablissement **20**.

Pour la mise en oeuvre du principe de protection par renommage, il est aussi choisi, dans le source du logiciel vulnérable **2vs :**

- au moins un traitement algorithmique utilisant au moins un opérande et rendant au moins un résultat,
- et au moins une portion du source du logiciel vulnérable **2vs**, contenant au moins un traitement algorithmique choisi.

Le source du logiciel vulnérable **2vs** est ensuite modifié, de manière à obtenir le source du logiciel protégé **2ps**. Cette modification est telle que notamment :

- lors de l'exécution du logiciel protégé **2p**, au moins une portion de la première partie d'exécution **2pes**, qui est exécutée dans le système de traitement de données **3**, prend en compte que la fonctionnalité d'au moins un traitement algorithmique choisi est exécutée dans l'unité **6**,
- lors de l'exécution du logiciel protégé **2p**, la seconde partie d'exécution **2peu**, qui est exécutée dans l'unité **6**, exécute au moins la fonctionnalité d'au moins un traitement algorithmique choisi,
- chaque traitement algorithmique choisi est décomposé de manière que lors de l'exécution du logiciel protégé **2p**, chaque traitement algorithmique choisi est exécuté, au moyen de la seconde partie d'exécution **2peu**, en utilisant des fonctions dépendantes. De préférence, chaque traitement algorithmique choisi est décomposé en fonctions dépendantes $\mathbf{fd_n}$ (avec **n** variant de **1** à **N**), à savoir :
  - éventuellement une ou plusieurs fonctions dépendantes permettant la mise à disposition d'un ou de plusieurs opérandes pour l'unité **6**,

- des fonctions dépendantes dont certaines utilisent la ou les opérandes et qui en combinaison, exécutent la fonctionnalité du traitement algorithmique choisi, utilisant ce ou ces opérandes,
- et éventuellement une ou plusieurs fonctions dépendantes permettant la mise à disposition par l'unité **6,** pour le système de traitement de données **3** du résultat du traitement algorithmique choisi,

- lors de l'exécution du logiciel protégé **2p**, la seconde partie d'exécution **2peu** exécute les fonctions dépendantes $\mathbf{fd_n}$,
- lors de l'exécution du logiciel protégé **2p**, les fonctions dépendantes sont déclenchées par des commandes déclenchantes à consignes renommées,
- et un ordonnancement des commandes déclenchantes est choisi parmi l'ensemble des ordonnancements permettant l'exécution du logiciel protégé **2p**.

La première partie d'exécution **2pes** du logiciel protégé **2p,** exécutée dans le système de traitement de données **3**, exécute des commandes déclenchantes à consignes renommées transférant vers l'unité **6** des consignes renommées, et déclenchant dans l'unité **6** le rétablissement au moyen des moyens de rétablissement **20**, des consignes, puis l'exécution au moyen de la seconde partie d'exécution **2peu**, de chacune des fonctions dépendantes $\mathbf{fd_n}$ précédemment définies.

En d'autres termes, le principe de protection par renommage consiste à renommer les consignes des commandes déclenchantes, de manière à obtenir des commandes déclenchantes à consignes renommées dont l'exécution dans le système de traitement de données **3**, déclenche dans l'unité **6**, l'exécution des fonctions dépendantes qui auraient été déclenchées par les commandes déclenchantes à consignes non renommées, sans toutefois que l'examen du logiciel protégé **2p** ne permette de déterminer l'identité des fonctions dépendantes exécutées.

La **fig. 80** illustre un exemple d'exécution d'un logiciel vulnérable **2v**. Dans cet exemple, il apparaît au cours de l'exécution du logiciel vulnérable **2v** dans le système de traitement de données **3**, à un instant donné, le calcul de $\mathbf{Z \leftarrow F(X, Y)}$ correspondant à l'affectation à une variable **Z** du résultat d'un traitement algorithmique représenté par une fonction **F** et utilisant les opérandes **X** et **Y**.

Les **fig. 81** et **82** illustrent un exemple de mise en oeuvre de l'invention.

La **fig. 81** illustre la mise en oeuvre partielle de l'invention. Dans cet exemple, lors de l'exécution dans le système de traitement de données **3** de la première partie d'exécution **2pes** du logiciel protégé **2p** et en présence de l'unité **6,** il apparaît :

- aux instants t_1, t_2, l'exécution des commandes déclenchantes CD_1, CD_2 déclenchant dans l'unité **6**, l'exécution au moyen de la seconde partie d'exécution **2peu**, des fonctions dépendantes fd_1, fd_2 correspondantes qui assurent le transfert des données **X, Y** depuis le système de traitement de données **3** vers des zones de mémorisation respectivement **x, y** situées dans les moyens de mémorisation **15** de l'unité **6**, ces commandes déclenchantes CD_1, CD_2 étant représentées respectivement par **OUT(x, X)**, **OUT(y, Y)**,
- aux instants t_3 à t_{N-1}, l'exécution des commandes déclenchantes CD_3 à CD_{N-1}, déclenchant dans l'unité **6**, l'exécution au moyen de la seconde partie d'exécution **2peu**, des fonctions dépendantes fd_3 à fd_{N-1} correspondantes, ces commandes déclenchantes CD_3 à CD_{N-1} étant représentées, respectivement, par **TRIG**(fd_3) à **TRIG**(fd_{N-1}). La suite des fonctions dépendantes fd_3 à fd_{N-1} exécutées en combinaison est algorithmiquement équivalente à la fonction **F**. Plus précisément, l'exécution de ces commandes déclenchantes conduit à l'exécution dans l'unité **6**, des fonctions dépendantes fd_3 à fd_{N-1} qui se servent du contenu des zones de mémorisation **x, y** et rendent le résultat dans une zone de mémorisation **z** de l'unité **6**,
- et à l'instant t_N, l'exécution d'une commande déclenchante CD_N déclenchant dans l'unité **6**, l'exécution au moyen de la seconde partie d'exécution **2peu**, de la fonction dépendante fd_N assurant le transfert du résultat du traitement algorithmique contenu dans la zone de mémorisation **z** de l'unité **6** vers le système de traitement de données **3**, afin de l'affecter à la variable **Z**, cette commande étant représentée par **IN(z)**.

Dans cet exemple, pour mettre en oeuvre complètement l'invention, il est choisi en tant que consigne, le premier argument des commandes déclenchantes **OUT** et l'argument des commandes déclenchantes **TRIG** et **IN**. Les consignes ainsi choisies

sont renommées par la méthode de renommage des consignes. De cette manière, les consignes des commandes déclenchantes CD_1 à CD_N à savoir **x, y,** fd_3, fd_{N-1}, **z** sont renommées de manière à obtenir respectivement **R(x), R(y),** $R(fd_3)$..., $R(fd_{N-1})$, **R(z).**

La **fig. 82** illustre la mise en oeuvre complète de l'invention. Dans cet exemple, lors de l'exécution dans le système de traitement de données **3,** de la première partie d'exécution **2pes** du logiciel protégé **2p**, et en présence de l'unité **6**, il apparaît :

- aux instants t_1, t_2, l'exécution des commandes déclenchantes à consignes renommées $CDCR_1$, $CDCR_2$, transférant vers l'unité **6**, les consignes renommées **R(x), R(y)** ainsi que les données **X, Y** déclenchant dans l'unité **6** le rétablissement au moyen des moyens de rétablissement **20**, des consignes renommées pour rétablir les consignes à savoir l'identité des zones de mémorisation **x, y**, puis l'exécution au moyen de la seconde partie d'exécution **2peu**, des fonctions dépendantes fd_1, fd_2 correspondantes qui assurent le transfert des données **X, Y** depuis le système de traitement de données **3** vers les zones de mémorisation respectivement **x, y** situées dans les moyens de mémorisations **15** de l'unité **6,** ces commandes déclenchantes à consignes renommées $CDCR_1$, $CDCR_2$ étant représentées respectivement par **OUT (R(x), X), OUT (R(y), Y),**
- aux instants t_3 à t_{N-1}, l'exécution des commandes déclenchantes à consignes renommées $CDCR_3$ à $CDCR_{N-1}$, transférant vers l'unité **6**, les consignes renommées $R(fd_3)$ à $R(fd_{N-1})$, déclenchant dans l'unité **6** le rétablissement au moyen des moyens de rétablissement **20**, des consignes, à savoir fd_3 à fd_{N-1}, puis l'exécution au moyen de la seconde partie d'exécution **2peu**, des fonctions dépendantes fd_3 à fd_{N-1}, ces commandes déclenchantes à consignes renommées $CDCR_3$ à $CDCR_{N-1}$ étant représentées respectivement par **TRIG** $(R(fd_3))$ à **TRIG** $(R(fd_{N-1}))$,
- et à l'instant t_N, l'exécution de la commande déclenchante à consigne renommée $CDCR_N$ transférant vers l'unité **6**, la consigne renommée **R(z)** déclenchant dans l'unité **6** le rétablissement au moyen des moyens de rétablissement **20**, de la consigne à savoir l'identité de la zone de mémorisation **z,** puis l'exécution au moyen de la seconde partie d'exécution

2peu, de la fonction dépendante $\mathbf{fd_N}$ assurant le transfert du résultat du traitement algorithmique contenu dans la zone de mémorisation **z** de l'unité **6** vers le système de traitement de données **3**, afin de l'affecter à la variable **Z**, cette commande déclenchante à consigne renommée $\mathbf{CDCR_N}$ étant représentée par **IN (R(z))**.

Dans l'exemple illustré, les commandes déclenchantes à consignes renommées 1 à N sont exécutées successivement. Il est à noter que deux améliorations peuvent être apportées :

- La première amélioration concerne le cas où plusieurs traitements algorithmiques sont déportés dans l'unité **6** et au moins le résultat d'un traitement algorithmique est utilisé par un autre traitement algorithmique. Dans ce cas, certaines commandes déclenchantes à consignes renommées servant au transfert, peuvent être éventuellement supprimées.
- La deuxième amélioration vise à opter pour un ordonnancement pertinent des commandes déclenchantes à consignes renommées parmi l'ensemble des ordonnancements permettant l'exécution du logiciel protégé **2p**. A cet égard, il est préférable de choisir un ordonnancement des commandes déclenchantes à consignes renommées qui dissocie temporellement l'exécution des fonctions dépendantes, en intercalant, entre celles-ci des portions de code exécuté dans le système de traitement de données **3** et comportant ou non des commandes déclenchantes à consignes renommées servant à la détermination d'autres données. Les **fig. 83** et **84** illustrent le principe d'une telle réalisation.

La **fig. 83** montre un exemple d'exécution d'un logiciel vulnérable **2v.** Dans cet exemple, il apparaît, au cours de l'exécution du logiciel vulnérable **2v**, dans le système de traitement de données **3**, l'exécution de deux traitement algorithmiques aboutissant à la détermination de **Z** et **Z'**, telles que $\mathbf{Z \leftarrow F\ (X, Y)}$ et $\mathbf{Z' \leftarrow F'\ (X', Y')}$.

La **fig. 84** illustre un exemple de mise en œuvre du procédé selon l'invention pour lequel les deux traitements algorithmiques choisis à la **fig. 83** sont déportés dans l'unité **6**. Selon un tel exemple, lors de l'exécution dans le système de traitement de données **3** de la première partie d'exécution **2pes** du logiciel protégé **2p** et en

présence de l'unité **6**, il apparaît, comme expliqué ci-dessus, l'exécution des commandes déclenchantes à consignes renommées $\mathbf{CDCR_1}$ à $\mathbf{CDCR_N}$ correspondant à la détermination de **Z** et l'exécution des commandes déclenchantes à consignes renommées $\mathbf{CDCR'_1}$ à $\mathbf{CDCR'_M}$ correspondant à la détermination de **Z'**. Comme illustré, les commandes déclenchantes à consignes renommées $\mathbf{CDCR_1}$ à $\mathbf{CDCR_N}$ ne sont pas exécutées consécutivement, dans la mesure où les commandes déclenchantes à consignes renommées $\mathbf{CDCR'_1}$ à $\mathbf{CDCR'_M}$ ainsi que d'autres portions de code sont intercalées. Dans l'exemple, il est ainsi réalisé l'ordonnancement suivant : $\mathbf{CDCR_1}$, portion de code intercalé, $\mathbf{CDCR'_1}$, $\mathbf{CDCR_2}$, portion de code intercalé, $\mathbf{CDCR'_2}$, $\mathbf{CDCR'_3}$, portion de code intercalé, $\mathbf{CDCR'_4}$, $\mathbf{CDCR_3}$, $\mathbf{CDCR_4}$, ..., $\mathbf{CDCR_N}$, $\mathbf{CDCR'_M}$.

Il est à noter que, lors de l'exécution d'une portion de la première partie d'exécution **2pes** du logiciel protégé **2p**, les commandes déclenchantes à consignes renommées exécutées dans le système de traitement de données **3**, déclenchent dans l'unité **6** le rétablissement de l'identité des fonctions dépendantes correspondantes puis l'exécution de celles-ci. Ainsi, il apparaît qu'en présence de l'unité **6**, cette portion est exécutée correctement et que, par conséquent, le logiciel protégé **2p** est complètement fonctionnel.

La **fig. 85** illustre un exemple de tentative d'exécution du logiciel protégé **2p**, alors que l'unité **6** est absente. Dans cet exemple, lors de l'exécution dans le système de traitement de données **3** de la première partie d'exécution **2pes** du logiciel protégé **2p**, à tous les instants, l'exécution d'une commande déclenchante à consigne renommée ne peut pas déclencher le rétablissement de la consigne ni l'exécution de la fonction dépendante correspondante, en raison de l'absence de l'unité **6**. La valeur à affecter à la variable **Z** ne peut donc pas être déterminée correctement.

Il apparaît donc, qu'en l'absence de l'unité **6**, au moins une demande d'une portion de la première partie d'exécution **2pes** du logiciel protégé **2p**, de déclencher le rétablissement d'une consigne et l'exécution d'une fonction dépendante dans l'unité **6** ne peut pas être satisfaite correctement, de sorte qu'au moins cette portion n'est pas exécutée correctement et que, par conséquent, le logiciel protégé **2p** n'est pas complètement fonctionnel.

Grâce à ce principe de protection par renommage, l'examen dans le logiciel

protégé **2p** des commandes déclenchantes à consignes renommées ne permet pas de déterminer l'identité des fonctions dépendantes devant être exécutées dans l'unité **6**. Il est à noter que le renommage des consignes est effectué lors de la modification du logiciel vulnérable **2v** en un logiciel protégé **2p**.

Selon une variante du principe de protection par renommage, il est défini pour au moins une fonction dépendante, une famille de fonctions dépendantes algorithmiquement équivalentes mais déclenchées par des commandes déclenchantes à consignes renommées différentes. Selon cette variante, pour au moins un traitement algorithmique utilisant des fonctions dépendantes, ce traitement algorithmique est décomposé en fonctions dépendantes qui pour au moins l'une d'entre elles est remplacée par une fonction dépendante de la même famille au lieu de conserver plusieurs occurrences de la même fonction dépendante. A cette fin, des commandes déclenchantes à consignes renommées sont modifiées pour tenir compte du remplacement de fonctions dépendantes par des fonctions dépendantes de la même famille. En d'autres termes, deux fonctions dépendantes de la même famille ont des consignes différentes et par conséquent des commandes déclenchantes à consignes renommées différentes et, il n'est pas possible, à l'examen du logiciel protégé **2p**, de déceler que les fonctions dépendantes appelées sont algorithmiquement équivalentes.

Selon une première réalisation préférée de la variante du principe de protection par renommage, il est défini pour au moins une fonction dépendante, une famille de fonctions dépendantes algorithmiquement équivalente, en concaténant un champ de bruit à l'information définissant la partie fonctionnelle de la fonction dépendante à exécuter dans l'unité **6**.

Selon une deuxième réalisation préférée de la variante du principe de protection par renommage, il est défini pour au moins une fonction dépendante, une famille de fonctions dépendantes algorithmiquement équivalente en utilisant des champs d'identification.

Selon une variante préférée de réalisation du principe de protection par renommage, il est défini comme méthode de renommage des consignes une méthode de chiffrement permettant de chiffrer les consignes pour les transformer en consignes renommées. Il est rappelé que le renommage des consignes est effectué dans la phase de protection **P**. Pour cette variante préférée, les moyens de rétablissement **20** sont

des moyens mettant en oeuvre une méthode de déchiffrement permettant de déchiffrer les consignes renommées et de rétablir ainsi l'identité des fonctions dépendantes à exécuter dans l'unité **6**. Ces moyens de rétablissement sont mis en oeuvre dans l'unité **6** et peuvent être de nature logicielle ou matérielle. Ces moyens de rétablissement **20** sont sollicités dans la phase d'utilisation **U** à chaque fois qu'une commande déclenchante à consigne renommée est exécutée dans le système de traitement de données **3** dans le but de déclencher dans l'unité **6**, l'exécution d'une fonction dépendante.

Selon une autre caractéristique avantageuse de l'invention, le procédé de protection vise à mettre en oeuvre un principe de protection dit par "variable" dont une description est effectuée en relation des **fig. 40** à **43**.

Pour la mise en oeuvre du principe de protection par variable, il est choisi dans le source du logiciel vulnérable **2vs** au moins une variable qui lors de l'exécution du logiciel vulnérable **2v**, définit partiellement l'état de celui-ci. Par état d'un logiciel, il doit être compris l'ensemble des informations, à un instant donné, nécessaires à l'exécution complète de ce logiciel, de sorte que l'absence d'une telle variable choisie nuit à l'exécution complète de ce logiciel. Il est aussi choisi au moins une portion du source du logiciel vulnérable **2vs** contenant au moins une variable choisie.

Au moins une portion choisie du source du logiciel vulnérable **2vs** est alors modifié, de manière à obtenir le source du logiciel protégé **2ps.** Cette modification est telle que lors de l'exécution du logiciel protégé **2p**, au moins une portion de la première partie d'exécution **2pes** qui est exécutée dans le système de traitement de données **3**, prend en compte qu'au moins une variable choisie ou au moins une copie de variable choisie réside dans l'unité **6**.

La **fig. 40** illustre un exemple d'exécution d'un logiciel vulnérable **2v.** Dans cet exemple, il apparaît au cours de l'exécution du logiciel vulnérable **2v** dans le système de traitement de données **3** :

- à l'instant t_1, l'affectation de la donnée **X** à la variable V_1, représentée par $V_1 \leftarrow X$,
- à l'instant t_2, l'affectation de la valeur de la variable V_1 à la variable **Y**, représentée par $Y \leftarrow V_1$,

- et à l'instant t_3, l'affectation de la valeur de la variable V_1 à la variable **Z**, représentée par $Z \leftarrow V_1$.

La **fig. 41** illustre un exemple d'une première forme de mise en oeuvre de l'invention pour laquelle la variable réside dans l'unité **6**. Dans cet exemple, lors de l'exécution dans le système de traitement de données **3** de la première partie d'exécution **2pes** du logiciel protégé **2p** , et en présence de l'unité **6**, il apparaît :

- à l'instant t_1, l'exécution d'une commande de transfert déclenchant le transfert de la donnée **X** depuis le système de traitement de données **3** vers la variable v_1 située dans les moyens de mémorisation **15** de l'unité **6**, cette commande de transfert étant représentée par **OUT**(v_1, **X**) et correspondant au final à l'affectation de la donnée **X** à la variable v_1,
- à l'instant t_2, l'exécution d'une commande de transfert déclenchant le transfert de la valeur de la variable v_1 résidant dans l'unité **6** vers le système de traitement de données **3** afin de l'affecter à la variable **Y**, cette commande de transfert étant représentée par **IN**(v_1) et correspondant au final à l'affectation de la valeur de la variable v_1 à la variable **Y**,
- et à l'instant t_3, l'exécution d'une commande de transfert déclenchant le transfert de la valeur de la variable v_1 résidant dans l'unité **6** vers le système de traitement de données **3** afin de l'affecter à la variable **Z**, cette commande de transfert étant représentée par **IN**(v_1) et correspondant au final à l'affectation de la valeur de la variable v_1 à la variable **Z**.

Il est à noter que lors de l'exécution du logiciel protégé **2p**, au moins une variable réside dans l'unité **6**. Ainsi, lorsqu'une portion de la première partie d'exécution **2pes** du logiciel protégé **2p** l'impose, et en présence de l'unité **6**, la valeur de cette variable résidant dans l'unité **6** est transférée vers le système de traitement de données **3** pour être utilisée par la première partie d'exécution **2pes** du logiciel protégé **2p**, de sorte que cette portion est exécutée correctement et que, par conséquent, le logiciel protégé **2p** est complètement fonctionnel.

La **fig. 42** illustre un exemple d'une deuxième forme de mise en oeuvre de l'invention pour laquelle une copie de la variable réside dans l'unité **6**. Dans cet exemple, lors de l'exécution dans le système de traitement de données **3** de la

première partie d'exécution **2pes** du logiciel protégé **2p**, et en présence de l'unité **6**, il apparaît :

- à l'instant t_1, l'affectation de la donnée **X** à la variable V_1 située dans le système de traitement de données **3**, ainsi que l'exécution d'une commande de transfert déclenchant le transfert de la donnée **X** depuis le système de traitement de données **3** vers la variable v_1 située dans les moyens de mémorisation **15** de l'unité **6**, cette commande de transfert étant représentée par **OUT**(v_1, **X**),
- à l'instant t_2, l'affectation de la valeur de la variable V_1 à la variable **Y**,
- et à l'instant t_3, l'exécution d'une commande de transfert déclenchant le transfert de la valeur de la variable v_1 résidant dans l'unité **6** vers le système de données **3** afin de l'affecter à la variable **Z**, cette commande de transfert étant représentée par **IN**(v_1).

Il est à noter que lors de l'exécution du logiciel protégé **2p**, au moins une copie d'une variable réside dans l'unité **6**. Ainsi, lorsqu'une portion de la première partie d'exécution **2pes** du logiciel protégé **2p** l'impose, et en présence de l'unité **6**, la valeur de cette copie de variable résidant dans l'unité **6** est transférée vers le système de traitement de données **3** pour être utilisée par la première partie d'exécution **2pes** du logiciel protégé **2p**, de sorte que cette portion est exécutée correctement et que, par conséquent, le logiciel protégé **2p** est complètement fonctionnel.

La **fig. 43** illustre un exemple de tentative d'exécution du logiciel protégé **2p**, alors que l'unité **6** est absente. Dans cet exemple, lors de l'exécution dans le système de traitement de données **3** de la première partie d'exécution **2pes** du logiciel protégé **2p** :

- à l'instant t_1, l'exécution de la commande de transfert **OUT**(v_1, **X**) ne peut pas déclencher le transfert de la donnée **X** vers la variable v_1, compte tenu de l'absence de l'unité **6**,
- à l'instant t_2, l'exécution de la commande de transfert **IN**(v_1) ne peut pas déclencher le transfert de la valeur de la variable v_1 vers le système de traitement de données **3**, compte tenu de l'absence de l'unité **6**,
- et à l'instant t_3, l'exécution de la commande de transfert **IN**(v_1) ne peut pas

déclencher le transfert de la valeur de la variable v_1 vers le système de traitement de données **3**, compte tenu de l'absence de l'unité **6**.

Il apparaît donc qu'en l'absence de l'unité **6**, au moins une demande d'une portion de la première partie d'exécution **2pes** d'utiliser une variable ou une copie de variable résidant dans l'unité **6**, ne peut pas être satisfaite correctement, de sorte qu'au moins cette portion n'est pas exécutée correctement et que, par conséquent, le logiciel protégé **2p** n'est pas complètement fonctionnel.

Il est à noter que les transferts de données entre le système de traitement de données **3** et l'unité **6** illustrés dans les exemples qui précèdent n'utilisent que des affectations simples mais que l'Homme de l'Art saura les combiner avec d'autres opérations pour aboutir à des opérations complexes telles que par exemple **OUT(v1, 2 * X + 3)** ou bien **Z ← (5 * v1 + v2).**

Selon une autre caractéristique avantageuse de l'invention, le procédé de protection vise à mettre en œuvre un principe de protection, dit par "fonctions élémentaires", dont une description est effectuée en relation des **fig. 60** à **64**.

Pour la mise en oeuvre du principe de protection par fonctions élémentaires, il est défini :

- un jeu de fonctions élémentaires dont les fonctions élémentaires sont susceptibles d'être exécutées, au moyen de la seconde partie d'exécution **2peu**, dans l'unité **6**, et éventuellement de transférer des données entre le système de traitement de données **3** et l'unité **6**,
- et un jeu de commandes élémentaires pour ce jeu de fonctions élémentaires, ces commandes élémentaires étant susceptibles d'être exécutées dans le système de traitement de données **3** et de déclencher l'exécution dans l'unité **6**, des fonctions élémentaires correspondantes.

Pour la mise en oeuvre du principe de protection par fonctions élémentaires, il est aussi construit des moyens d'exploitation permettant de transformer une unité vierge **60** en une unité **6** capable d'exécuter les fonctions élémentaires, l'exécution de ces fonctions élémentaires étant déclenchée par l'exécution dans le système de traitement de données **3**, de commandes élémentaires.

Pour la mise en œuvre du principe de protection par fonctions élémentaires, il est aussi choisi, dans le source du logiciel vulnérable **2vs**, au moins un traitement

algorithmique utilisant au moins un opérande et rendant au moins un résultat. Il est aussi choisi au moins une portion du source du logiciel vulnérable **2vs** contenant au moins un traitement algorithmique choisi.

Au moins une portion choisie du source du logiciel vulnérable **2vs** est alors modifiée, de manière à obtenir le source du logiciel protégé **2ps**. Cette modification est telle que notamment :

- lors de l'exécution du logiciel protégé **2p**, au moins une portion de la première partie d'exécution **2pes**, qui est exécutée dans le système de traitement de données **3**, prend en compte que la fonctionnalité d'au moins un traitement algorithmique choisi est exécutée dans l'unité **6**,
- lors de l'exécution du logiciel protégé **2p**, la seconde partie d'exécution **2peu**, qui est exécutée dans l'unité **6**, exécute au moins la fonctionnalité d'au moins un traitement algorithmique choisi,
- chaque traitement algorithmique choisi est décomposé de manière que lors de l'exécution du logiciel protégé **2p**, chaque traitement algorithmique choisi est exécuté, au moyen de la seconde partie d'exécution **2peu**, en utilisant des fonctions élémentaires. De préférence, chaque traitement algorithmique choisi est décomposé en fonctions élémentaires $\mathbf{fe_n}$ (avec **n** variant de 1 à N), à savoir :
  - éventuellement une ou plusieurs fonctions élémentaires permettant la mise à disposition d'un ou de plusieurs opérandes pour l'unité **6**,
  - des fonctions élémentaires dont certaines utilisent la ou les opérandes et qui en combinaison, exécutent la fonctionnalité du traitement algorithmique choisi, utilisant ce ou ces opérandes,
  - et éventuellement une ou plusieurs fonctions élémentaires permettant la mise à disposition par l'unité **6**, pour le système de traitement de données **3** du résultat du traitement algorithmique choisi,
- et un ordonnancement des commandes élémentaires est choisi parmi l'ensemble des ordonnancements permettant l'exécution du logiciel protégé **2p**.

La première partie d'exécution **2pes** du logiciel protégé **2p**, qui est exécutée

dans le système de traitement de données **3**, exécute des commandes élémentaires $\mathbf{CFE_n}$ (avec **n** variant de 1 à N), déclenchant dans l'unité **6**, l'exécution au moyen de la seconde partie d'exécution **2peu**, de chacune des fonctions élémentaires $\mathbf{fe_n}$ précédemment définies.

La **fig. 60** illustre un exemple d'exécution d'un logiciel vulnérable **2v**. Dans cet exemple, il apparaît, au cours de l'exécution du logiciel vulnérable **2v** dans le système de traitement de données **3**, à un instant donné, le calcul de $\mathbf{Z \leftarrow F(X, Y)}$ correspondant à l'affectation à une variable **Z** du résultat d'un traitement algorithmique représenté par une fonction **F** et utilisant des opérandes **X** et **Y**.

La **fig. 61** illustre un exemple de mise en oeuvre de l'invention pour lequel le traitement algorithmique choisi à la **fig. 60** est déporté dans l'unité **6**. Dans cet exemple, lors de l'exécution dans le système de traitement de données **3** de la première partie d'exécution **2pes** du logiciel protégé **2p** et en présence de l'unité **6**, il apparaît :

- aux instants $\mathbf{t_1}$, $\mathbf{t_2}$, l'exécution des commandes élémentaires $\mathbf{CFE_1}$, $\mathbf{CFE_2}$ déclenchant dans l'unité **6**, l'exécution au moyen de la seconde partie d'exécution **2peu**, des fonctions élémentaires $\mathbf{fe_1}$, $\mathbf{fe_2}$ correspondantes qui assurent le transfert des données **X**, **Y** depuis le système de traitement de données **3** vers des zones de mémorisation respectivement **x**, **y** situées dans les moyens de mémorisation **15** de l'unité **6**, ces commandes élémentaires $\mathbf{CFE_1}$, $\mathbf{CFE_2}$ étant représentées respectivement par **OUT(x, X)**, **OUT(y, Y)**,
- aux instants $\mathbf{t_3}$ à $\mathbf{t_{N-1}}$, l'exécution des commandes élémentaires $\mathbf{CFE_3}$ à $\mathbf{CFE_{N-1}}$, déclenchant dans l'unité **6**, l'exécution au moyen de la seconde partie d'exécution **2peu**, des fonctions élémentaires $\mathbf{fe_3}$ à $\mathbf{fe_{N-1}}$ correspondantes, ces commandes élémentaires $\mathbf{CFE_3}$ à $\mathbf{CFE_{N-1}}$ étant représentées, respectivement, par $\mathbf{TRIG(fe_3)}$ à $\mathbf{TRIG(fe_{N-1})}$. La suite des fonctions élémentaires $\mathbf{fe_3}$ à $\mathbf{fe_{N-1}}$ exécutées en combinaison est algorithmiquement équivalente à la fonction **F**. Plus précisément, l'exécution de ces commandes élémentaires conduit à l'exécution dans l'unité **6**, des fonctions élémentaires $\mathbf{fe_3}$ à $\mathbf{fe_{N-1}}$ qui se servent du contenu des zones de mémorisation **x**, **y** et rendent le résultat dans une zone de

mémorisation **z** de l'unité **6**,

- et à l'instant t_N, l'exécution d'une commande élémentaire CFE_N déclenchant dans l'unité **6**, l'exécution au moyen de la seconde partie d'exécution **2peu**, de la fonction élémentaire fe_N assurant le transfert du résultat du traitement algorithmique, contenu dans la zone de mémorisation **z** de l'unité **6** vers le système de traitement de données **3**, afin de l'affecter à la variable **Z**, cette commande élémentaire CFE_N étant représentée par **IN(z)**.

Dans l'exemple illustré, les commandes élémentaires 1 à N sont exécutées successivement. Il est à noter que deux améliorations peuvent être apportées :

- La première amélioration concerne le cas où plusieurs traitements algorithmiques sont déportés dans l'unité **6** et au moins le résultat d'un traitement algorithmique est utilisé par un autre traitement algorithmique. Dans ce cas, certaines commandes élémentaires servant au transfert, peuvent être éventuellement supprimées.
- La deuxième amélioration vise à opter pour un ordonnancement pertinent des commandes élémentaires parmi l'ensemble des ordonnancements permettant l'exécution du logiciel protégé **2p**. A cet égard, il est préférable de choisir un ordonnancement des commandes élémentaires qui dissocie temporellement l'exécution des fonctions élémentaires, en intercalant, entre celles-ci des portions de code exécuté dans le système de traitement de données **3** et comportant ou non des commandes élémentaires servant à la détermination d'autres données. Les **fig. 62** et **63** illustrent le principe d'une telle réalisation.

La **fig. 62** montre un exemple d'exécution d'un logiciel vulnérable **2v**. Dans cet exemple, il apparaît au cours de l'exécution du logiciel vulnérable **2v**, dans le système de traitement de données **3**, l'exécution de deux traitements algorithmiques aboutissant à la détermination de **Z** et **Z'**, telles que **Z ← F (X, Y)** et **Z' ←F'(X', Y')**.

La **fig. 63** illustre un exemple de mise en œuvre du procédé selon l'invention pour lequel les deux traitements algorithmiques choisis à la **fig. 62** sont déportés dans l'unité **6**. Selon un tel exemple, lors de l'exécution dans le système de traitement de données **3** de la première partie d'exécution **2pes** du logiciel protégé **2p** et en présence de l'unité **6**, il apparaît, comme expliqué ci-dessus, l'exécution des

commandes élémentaires CFE_1 à CFE_N correspondant à la détermination de **Z** et l'exécution des commandes élémentaires CFE'_1 à CFE'_M correspondant à la détermination de **Z'**. Comme illustré, les commandes élémentaires CFE_1 à CFE_N ne sont pas exécutées consécutivement, dans la mesure où les commandes élémentaires CFE'_1 à CFE'_M, ainsi que d'autres portions de code sont intercalées. Dans l'exemple il est ainsi réalisé l'ordonnancement suivant : CFE_1, portion de code intercalé, CFE'_1, CFE_2, portion de code intercalé, CFE'_2, CFE'_3, portion de code intercalé, CFE'_4, CFE_3, CFE_4, ..., CFE_N, CFE'_M.

Il est à noter que, lors de l'exécution du logiciel protégé **2p**, en présence de l'unité **6**, à chaque fois qu'une commande élémentaire contenue dans une portion de la première partie d'exécution **2pes** du logiciel protégé **2p** l'impose, la fonction élémentaire correspondante est exécutée dans l'unité **6**. Ainsi, il apparaît, qu'en présence de l'unité **6**, cette portion est exécutée correctement et que, par conséquent, le logiciel protégé **2p** est complètement fonctionnel.

La **fig. 64** illustre un exemple de tentative d'exécution du logiciel protégé **2p**, alors que l'unité **6** est absente. Dans cet exemple, lors de l'exécution dans le système de traitement de données **3**, de la première partie d'exécution **2pes** du logiciel protégé **2p**, à tous les instants, l'exécution d'une commande élémentaire ne peut pas déclencher l'exécution de la fonction élémentaire correspondante, en raison de l'absence de l'unité **6**. La valeur à affecter à la variable **Z** ne peut donc pas être déterminée correctement.

Il apparaît donc, qu'en l'absence de l'unité **6**, au moins une demande d'une portion de la première partie d'exécution **2pes** du logiciel protégé **2p**, de déclencher l'exécution d'une fonction élémentaire dans l'unité **6** ne peut pas être satisfaite correctement, de sorte qu'au moins cette portion n'est pas exécutée correctement et que, par conséquent, le logiciel protégé **2p** n'est pas complètement fonctionnel.

Selon une autre caractéristique avantageuse de l'invention, le procédé de protection vise à mettre en œuvre un principe de protection, dit par "détection et coercition", dont une description est effectuée en relation des **fig. 70** à **74**.

Pour la mise en œuvre du principe de protection par détection et coercition, il est défini :

- au moins une caractéristique d'exécution de logiciel susceptible d'être

surveillée au moins en partie dans l'unité **6**,

- au moins un critère à respecter pour au moins une caractéristique d'exécution de logiciel,
- des moyens de détection **17** à mettre en oeuvre dans l'unité **6** et permettant de détecter qu'au moins une caractéristique d'exécution de logiciel ne respecte pas au moins un critère associé,
- et des moyens de coercition **18** à mettre en oeuvre dans l'unité **6** et permettant d'informer le système de traitement de données **3** et/ou de modifier l'exécution d'un logiciel, lorsqu'au moins un critère n'est pas respecté.

Pour la mise en oeuvre du principe de protection par détection et coercition, il est aussi construit des moyens d'exploitation permettant de transformer une unité vierge **60** en une unité **6** mettant au moins en oeuvre les moyens de détection **17** et les moyens de coercition **18**.

La **fig. 70** illustre les moyens nécessaires à la mise en oeuvre de ce principe de protection par détection et coercition. L'unité **6** comporte les moyens de détection **17** et les moyens de coercition **18** appartenant aux moyens de traitement **16**. Les moyens de coercition **18** sont informés du non respect d'un critère par les moyens de détection **17**.

D'une manière plus précise, les moyens de détection **17** utilisent des informations en provenance des moyens de transfert **13** et/ou des moyens de mémorisation **15** et/ou des moyens de traitement **16**, afin de surveiller une ou plusieurs caractéristiques d'exécution de logiciel. A chaque caractéristique d'exécution de logiciel est fixé au moins un critère à respecter.

Dans le cas où il est détecté qu'au moins une caractéristique d'exécution de logiciel ne respecte pas au moins un critère, les moyens de détection **17** en informent les moyens de coercition **18**. Ces moyens de coercition **18** sont adaptés pour modifier, de la manière appropriée, l'état de l'unité **6**.

Pour la mise en oeuvre du principe de protection par détection et coercition, il est aussi choisi :

- au moins une caractéristique d'exécution de logiciel à surveiller, parmi les caractéristiques d'exécution susceptibles d'être surveillées,

- au moins un critère à respecter pour au moins une caractéristique d'exécution de logiciel choisie,
- dans le source du logiciel vulnérable **2vs**, au moins un traitement algorithmique pour lequel au moins une caractéristique d'exécution de logiciel est à surveiller,
- et dans le source du logiciel vulnérable **2vs**, au moins une portion contenant au moins un traitement algorithmique choisi.

Au moins une portion choisie du source du logiciel vulnérable **2vs** est ensuite modifiée, de manière à obtenir le source du logiciel protégé **2ps**. Cette modification est telle que notamment lors de l'exécution du logiciel protégé **2p :**

- au moins une portion de la première partie d'exécution **2pes**, qui est exécutée dans le système de traitement de données **3,** prend en compte qu'au moins une caractéristique d'exécution de logiciel choisie est à surveiller, au moins en partie dans l'unité **6,**
- et la seconde partie d'exécution **2peu,** qui est exécutée dans l'unité **6,** surveille au moins en partie, une caractéristique d'exécution de logiciel choisie.

Lors de l'exécution du logiciel protégé **2p**, protégé par ce principe de protection par détection et coercition, en présence de l'unité **6** :

- tant que tous les critères correspondants à toutes les caractéristiques d'exécution surveillées de toutes les portions modifiées du logiciel protégé **2p** sont respectés, ces portions modifiées du logiciel protégé **2p** fonctionnent de manière nominale et, par conséquent, le logiciel protégé **2p** fonctionne de manière nominale,
- et si au moins un des critères correspondant à une caractéristique d'exécution surveillée d'une portion du logiciel protégé **2p** n'est pas respecté, le système de traitement de données **3** en est informé et/ou le fonctionnement de la portion du logiciel protégé **2p** est modifié, de sorte que le fonctionnement du logiciel protégé **2p** est modifié.

Bien entendu, en l'absence de l'unité **6**, au moins une demande d'une portion de la première partie d'exécution **2pes** du logiciel protégé **2p** d'utiliser l'unité **6** ne peut

pas être satisfaite correctement de sorte qu'au moins cette portion ne s'exécute pas correctement et que par conséquent le logiciel protégé **2p** n'est pas complètement fonctionnel.

Pour la mise en oeuvre du principe de protection par détection et coercition, deux types de caractéristiques d'exécution de logiciel sont utilisées préférentiellement.

Le premier type de caractéristique d'exécution de logiciel correspond à une variable de mesure de l'exécution d'un logiciel et le second type correspond à un profil d'utilisation d'un logiciel. Ces deux types de caractéristiques peuvent être utilisées indépendamment ou en combinaison.

Pour la mise en oeuvre du principe de protection par détection et coercition utilisant, comme caractéristique d'exécution, une variable de mesure de l'exécution de logiciel, il est défini :

- dans les moyens de mémorisation **15**, la possibilité de mémoriser au moins une variable de mesure servant à quantifier l'utilisation d'au moins une fonctionnalité de logiciel,
- dans les moyens de détection **17**, la possibilité de surveiller au moins un seuil associé à chaque variable de mesure,
- et des moyens d'actualisation permettant de mettre à jour chaque variable de mesure en fonction de l'utilisation de la fonctionnalité à laquelle elle est associée.

Il est aussi construit des moyens d'exploitation mettant en oeuvre, en plus des moyens de détection **17** et des moyens de coercition **18**, les moyens d'actualisation.

Il est aussi choisi, dans le source du logiciel vulnérable **2vs** :

- au moins une fonctionnalité du logiciel vulnérable **2v** dont l'utilisation est susceptible d'être surveillée grâce à une variable de mesure,
- au moins une variable de mesure servant à quantifier l'utilisation de ladite fonctionnalité,
- au moins un seuil associé à la variable de mesure correspondant à une limite d'utilisation de ladite fonctionnalité,
- et au moins une méthode de mise à jour de la variable de mesure en fonction de l'utilisation de ladite fonctionnalité.

Le source du logiciel vulnérable **2vs** est ensuite modifiée, de manière à obtenir le source du logiciel protégé **2ps**, cette modification étant telle que, lors de l'exécution du logiciel protégé **2p,** la seconde partie d'exécution **2peu :**

- actualise la variable de mesure en fonction de l'utilisation de ladite fonctionnalité,
- et prend en compte au moins un dépassement de seuil.

En d'autres termes, lors de l'exécution du logiciel protégé **2p**, la variable de mesure est mise à jour en fonction de l'utilisation de ladite fonctionnalité, et lorsque le seuil est dépassé, les moyens de détection **17** en informent les moyens de coercition **18** qui prennent une décision adaptée pour informer le système de traitement de données **3** et/ou modifier les traitements effectués par les moyens de traitement **16** permettant de modifier le fonctionnement de la portion du logiciel protégé **2p**, de sorte que le fonctionnement du logiciel protégé **2p** est modifié.

Pour la mise en oeuvre d'une première variante préférée de réalisation du principe de protection par détection et coercition utilisant, comme caractéristique, une variable de mesure, il est défini :

- pour au moins une variable de mesure, plusieurs seuils associés,
- et des moyens de coercition différents correspondant à chacun de ces seuils.

Il est aussi choisi, dans le source du logiciel vulnérable **2vs** :

- au moins une variable de mesure servant à quantifier l'utilisation d'au moins une fonctionnalité du logiciel et à laquelle doivent être associés plusieurs seuils correspondant à des limites différentes d'utilisation de ladite fonctionnalité,
- et au moins deux seuils associés à la variable de mesure.

Le source du logiciel vulnérable **2vs** est ensuite modifié, de manière à obtenir le source du logiciel protégé **2ps**, cette modification étant telle que, lors de l'exécution du logiciel protégé **2p**, la seconde partie d'exécution **2peu** :

- actualise la variable de mesure en fonction de l'utilisation de ladite fonctionnalité,
- et prend en compte, de manière différente, les dépassements des divers seuils.

En d'autres termes, de manière classique, lors de l'exécution du logiciel protégé **2p**, lorsque le premier seuil est dépassé, l'unité **6** informe le système de traitement de données **3** enjoignant le logiciel protégé **2p** de ne plus utiliser cette fonctionnalité. Si le logiciel protégé **2p** continue d'utiliser cette fonctionnalité, le second seuil pourra être dépassé. Dans le cas où le second seuil est dépassé, les moyens de coercition **18** peuvent rendre inopérante la fonctionnalité choisie et/ou rendre inopérant le logiciel protégé **2p**.

Pour la mise en oeuvre d'une deuxième variante préférée de réalisation du principe de protection par détection et coercition utilisant, comme caractéristique, une variable de mesure, il est défini des moyens de rechargement permettant de créditer au moins une utilisation supplémentaire pour au moins une fonctionnalité de logiciel surveillée par une variable de mesure.

Il est aussi construit des moyens d'exploitation mettant en oeuvre, en plus des moyens de détection **17**, des moyens de coercition **18** et des moyens d'actualisation, les moyens de rechargement.

Il est aussi choisi, dans le source du logiciel vulnérable **2vs**, au moins une variable de mesure servant à limiter l'utilisation d'au moins une fonctionnalité du logiciel et à laquelle au moins une utilisation supplémentaire doit pouvoir être créditée.

Le source du logiciel vulnérable **2vs** est ensuite modifié, de manière à obtenir le source du logiciel protégé **2ps**, cette modification étant telle que, dans une phase dite de rechargement, au moins une utilisation supplémentaire d'au moins une fonctionnalité correspondant à une variable de mesure choisie peut être créditée.

Il est procédé, dans la phase de rechargement, à la réactualisation d'au moins une variable de mesure choisie et/ou d'au moins un seuil associé, de manière à permettre au moins une utilisation supplémentaire de la fonctionnalité correspondante. En d'autres termes, il est possible, dans la phase de rechargement, de créditer des utilisations supplémentaires d'au moins une fonctionnalité du logiciel protégé **2p**.

Pour la mise en oeuvre du principe de protection par détection et coercition utilisant, comme caractéristique, un profil d'utilisation de logiciel, il est défini en tant que critère à respecter pour ce profil d'utilisation, au moins un trait d'exécution de

logiciel.

Il est aussi choisi, dans le source du logiciel vulnérable **2vs** :

- au moins un profil d'utilisation à surveiller,
- et au moins un trait d'exécution qu'au moins un profil d'utilisation choisi doit respecter.

Le source du logiciel vulnérable **2vs** est ensuite modifié, de manière à obtenir le source du logiciel protégé **2ps**, cette modification étant telle que, lors de l'exécution du logiciel protégé **2p**, la seconde partie d'exécution **2peu** respecte tous les traits d'exécution choisis. En d'autres termes, l'unité **6** surveille elle-même la manière dont la seconde partie d'exécution **2peu** est exécutée et peut informer le système de traitement de données **3** et/ou modifier le fonctionnement du logiciel protégé **2p**, dans le cas où au moins un trait d'exécution n'est pas respecté.

Lors de l'exécution du logiciel protégé **2p**, protégé par ce principe, en présence de l'unité **6** :

- tant que tous les traits d'exécution de toutes les portions modifiées du logiciel protégé **2p** sont respectés, ces portions modifiées du logiciel protégé **2p** fonctionnent de manière nominale et, par conséquent, le logiciel protégé **2p** fonctionne de manière nominale,
- et si au moins un trait d'exécution d'une portion de logiciel protégé **2p** n'est pas respecté, le système de traitement de données **3** en est informé et/ou le fonctionnement de la portion du logiciel protégé **2p** est modifié, de sorte que le fonctionnement du logiciel protégé **2p** est modifié.

Il peut être envisagé la surveillance de différents traits d'exécution, comme par exemple la surveillance de la présence d'instructions comportant un marqueur ou la surveillance de l'enchaînement d'exécution pour au moins une partie des instructions.

Pour la mise en oeuvre du principe de protection par détection et coercition utilisant comme trait d'exécution à respecter, la surveillance de l'enchaînement d'exécution pour au moins une partie des instructions, il est défini :

- un jeu d'instructions dont les instructions sont susceptibles d'être exécutées dans l'unité **6**,
- un jeu de commandes d'instruction pour ce jeu d'instructions, ces

commandes d'instructions sont susceptibles d'être exécutées dans le système de traitement de données **3.** L'exécution de chacune de ces commandes d'instructions dans le système de traitement de données **3** déclenche dans l'unité **6**, l'exécution de l'instruction correspondante,

- des moyens de détection **17** permettant de détecter que l'enchaînement des instructions ne correspond pas à celui souhaité,
- et des moyens de coercition **18** permettant d'informer le système de traitement de données **3** et/ou de modifier l'exécution d'un logiciel lorsque l'enchaînement des instructions ne correspond pas à celui souhaité.

Il est aussi construit des moyens d'exploitation permettant, à l'unité **6**, d'exécuter aussi les instructions du jeu d'instructions, l'exécution de ces instructions étant déclenchée par l'exécution dans le système de traitement de données **3** des commandes d'instructions.

Il est aussi choisi, dans le source du logiciel vulnérable **2vs**, au moins un traitement algorithmique devant être déporté dans l'unité **6** et pour lequel l'enchaînement d'au moins une partie des instructions est à surveiller.

Le source du logiciel vulnérable **2vs** est ensuite modifié de manière à obtenir le source du logiciel protégé **2ps**, cette modification est telle que, lors de l'exécution du logiciel protégé **2p** :

- la seconde partie d'exécution **2peu** exécute au moins la fonctionnalité du traitement algorithmique choisi,
- le traitement algorithmique choisi est décomposé en instructions,
- l'enchaînement que doivent respecter au moins certaines des instructions lors de leur exécution dans l'unité **6** est spécifié,
- et la première partie d'exécution **2pes** du logiciel protégé **2p** exécute des commandes d'instructions qui déclenchent l'exécution des instructions dans l'unité **6.**

Lors de l'exécution du logiciel protégé **2p**, protégé par ce principe, en présence de l'unité **6** :

- tant que l'enchaînement des instructions de toutes les portions modifiées du logiciel protégé **2p** correspond à celui souhaité, ces portions modifiées du

logiciel protégé **2p** fonctionnent de manière nominale et, par conséquent, le logiciel protégé **2p** fonctionne de manière nominale,

- et si l'enchaînement des instructions d'une portion de logiciel protégé **2p** exécutées dans l'unité **6** ne correspond pas à celui souhaité, le système de traitement de données **3** en est informé et/ou le fonctionnement de la portion du logiciel protégé **2p** est modifié, de sorte que le fonctionnement du logiciel protégé **2p** est modifié.

La **fig. 71** illustre un exemple de mise en oeuvre du principe de protection par détection et coercition utilisant, comme trait d'exécution à respecter la surveillance de l'enchaînement d'exécution d'au moins une partie des instructions, dans le cas où l'enchaînement souhaité est respecté.

La première partie d'exécution **2pes** du logiciel protégé **2p**, exécutée dans le système de traitement de données **3**, exécute des commandes d'instructions $\mathbf{CI_i}$ déclenchant, dans l'unité **6** l'exécution des instructions $\mathbf{i_i}$ appartenant au jeu d'instructions. Dans le jeu d'instructions, au moins certaines des instructions comportent chacune une partie définissant la fonctionnalité de l'instruction et une partie permettant de vérifier l'enchaînement souhaité pour l'exécution des instructions. Dans cet exemple, les commandes d'instructions $\mathbf{CI_i}$ sont représentées par $\mathbf{TRIG(i_i)}$ et l'enchaînement souhaité pour l'exécution des instructions est $\mathbf{i_n}$, $\mathbf{i_{n+1}}$ et $\mathbf{i_{n+2}}$. L'exécution dans l'unité **6**, de l'instruction $\mathbf{i_n}$ donne le résultat **a** et l'exécution de l'instruction $\mathbf{i_{n+1}}$ donne le résultat **b**. L'instruction $\mathbf{i_{n+2}}$ utilise comme opérande, les résultats **a** et **b** des instructions $\mathbf{i_n}$ et $\mathbf{i_{n+1}}$ et son exécution donne le résultat **c**.

Compte tenu que cet enchaînement des instructions exécutées dans l'unité **6** correspond à celui souhaité, il en résulte un fonctionnement normal ou nominal du logiciel protégé **2p**.

La **fig. 72** illustre un exemple de mise en oeuvre du principe de protection par détection et coercition utilisant, comme trait d'exécution à respecter, la surveillance de l'enchaînement d'exécution d'au moins une partie des instructions, dans le cas où l'enchaînement souhaité n'est pas respecté.

Selon cet exemple, l'enchaînement souhaité pour l'exécution des instructions est toujours $\mathbf{i_n}$, $\mathbf{i_{n+1}}$ et $\mathbf{i_{n+2}}$. Toutefois, l'enchaînement d'exécution des instructions est modifié par le remplacement de l'instruction $\mathbf{i_n}$ par l'instruction $\mathbf{i'_n}$, de sorte que

l'enchaînement effectivement exécuté est $\mathbf{i'_n}$, $\mathbf{i_{n+1}}$ et $\mathbf{i_{n+2}}$. L'exécution de l'instruction $\mathbf{i'_n}$ donne le résultat **a**, c'est-à-dire le même résultat que l'exécution de l'instruction $\mathbf{i_n}$. Toutefois, au plus tard lors de l'exécution de l'instruction $\mathbf{i_{n+2}}$, les moyens de détection **17** détectent que l'instruction $\mathbf{i'_n}$ ne correspond pas à l'instruction souhaitée pour générer le résultat **a** utilisé comme opérande de l'instruction $\mathbf{i_{n+2}}$. Les moyens de détection **17** en informent les moyens de coercition **18** qui modifient en conséquence, le fonctionnement de l'instruction $\mathbf{i_{n+2}}$, de sorte que l'exécution de l'instruction $\mathbf{i_{n+2}}$ donne le résultat **c'** pouvant être différent de **c**. Bien entendu, si l'exécution de l'instruction $\mathbf{i'_n}$ donne un résultat **a'** différent du résultat **a** de l'instruction $\mathbf{i_n}$, il est clair que le résultat de l'instruction $\mathbf{i_{n+2}}$ peut aussi être différent de **c**.

Dans la mesure où l'enchaînement d'exécution des instructions exécutées dans l'unité **6** ne correspond pas à celui souhaité, il peut donc être obtenu une modification du fonctionnement du logiciel protégé **2p**.

Les **fig. 73** et **74** illustrent une variante préférée de réalisation du principe de protection par détection et coercition utilisant, comme trait d'exécution à respecter, la surveillance de l'enchaînement d'exécution d'au moins une partie des instructions. Selon cette variante préférée, il est défini un jeu d'instructions dont au moins certaines instructions travaillent sur des registres et utilisent au moins un opérande en vue de rendre un résultat.

Comme illustré à la **fig. 73,** il est défini pour au moins certaines des instructions travaillant sur des registres, une partie **PF** définissant la fonctionnalité de l'instruction et une partie **PE** définissant l'enchaînement souhaité pour l'exécution des instructions. La partie **PF** correspond au code opération connu de l'Homme de l'art. La partie **PE** définissant l'enchaînement souhaité, comporte des champs de bits correspondant à :

- un champ d'identification de l'instruction **CII,**
- et pour chaque opérande **k** de l'instruction, avec **k** variant de 1 à **K**, et **K** nombre d'opérandes de l'instruction :
  - un champ drapeau $\mathbf{CD_k}$, indiquant s'il convient de vérifier la provenance de l'opérande **k,**
  - et un champ d'identification prévue $\mathbf{CIP_k}$ de l'opérande, indiquant l'identité attendue de l'instruction ayant généré le contenu de

l'opérande **k**.

Comme illustré à la **fig. 74**, le jeu d'instructions comporte **V** registres appartenant aux moyens de traitement **16**, chaque registre étant nommé $\mathbf{R_v}$, avec **v** variant de 1 à **V**. Pour chaque registre $\mathbf{R_v}$, il est défini deux champs, à savoir :

- un champ fonctionnel $\mathbf{CF_v}$, connu de l'Homme de l'art et permettant de stocker le résultat de l'exécution des instructions,
- et un champ d'identification générée $\mathbf{CIG_v}$ permettant de mémoriser l'identité de l'instruction ayant généré le contenu du champ fonctionnel $\mathbf{CF_v}$. Ce champ d'identification générée $\mathbf{CIG_v}$ est mis à jour automatiquement avec le contenu du champ d'identification de l'instruction **CII** ayant généré le champ fonctionnel $\mathbf{CF_v}$. Ce champ d'identification générée $\mathbf{CIG_v}$ n'est pas accessible, ni modifiable par aucune instruction et sert uniquement aux moyens de détection **17**.

Lors de l'exécution d'une instruction, les moyens de détection **17** effectuent pour chaque opérande **k** les opérations suivantes :

- le champ drapeau $\mathbf{CD_k}$ est lu,
- si le champ drapeau $\mathbf{CD_k}$ l'impose, le champ d'identification prévue $\mathbf{CIP_k}$ et le champ d'identification générée $\mathbf{CIG_v}$ correspondant au registre utilisé par l'opérande **k** sont lus tous les deux,
- l'égalité des deux champs $\mathbf{CIP_k}$ et $\mathbf{CIG_v}$ est contrôlée,
- et si l'égalité est fausse, les moyens de détection **17** considèrent que l'enchaînement d'exécution des instructions n'est pas respecté.

Les moyens de coercition **18** permettent de modifier le résultat des instructions lorsque les moyens de détection **17** les ont informés d'un enchaînement d'instructions non respecté. Une réalisation préférentielle consiste à modifier la partie fonctionnelle **PF** de l'instruction en cours d'exécution ou la partie fonctionnelle **PF** d'instructions ultérieures.

Selon une autre caractéristique avantageuse de l'invention, le procédé de protection vise à mettre en œuvre un principe de protection dit par "branchement conditionnel" dont la description est effectuée en relation des **fig. 90** à **92**.

Pour la mise en œuvre du principe de protection par branchement conditionnel,

il est choisi dans le source du logiciel vulnérable **2vs**, au moins un branchement conditionnel **BC**. Il est aussi choisi au moins une portion du source du logiciel vulnérable **2vs** contenant au moins un branchement conditionnel **BC** choisi.

Au moins une portion choisie du source du logiciel vulnérable **2vs** est alors modifié, de manière à obtenir le source du logiciel protégé **2ps.** Cette modification est telle que notamment lors de l'exécution du logiciel protégé **2p :**

- au moins une portion de la première partie d'exécution **2pes**, qui est exécutée dans le système de traitement de données **3**, prend en compte que la fonctionnalité d'au moins un branchement conditionnel **BC** choisi est exécutée dans l'unité **6**,
- et la seconde partie d'exécution **2peu**, qui est exécutée dans l'unité **6**, exécute au moins la fonctionnalité d'au moins un branchement conditionnel **BC** choisi et met à la disposition du système de traitement de données **3**, une information permettant à la première partie d'exécution **2pes**, de poursuivre son exécution à l'endroit choisi.

La première partie d'exécution **2pes** du logiciel protégé **2p**, exécutée dans le système de traitement de données **3**, exécute des commandes de branchements conditionnels, déclenchant dans l'unité **6**, l'exécution au moyen de la seconde partie d'exécution **2peu**, de branchements conditionnels déportés **bc** dont la fonctionnalité est équivalente à la fonctionnalité des branchements conditionnels **BC** choisis.

La **fig. 90** illustre un exemple d'exécution d'un logiciel vulnérable **2v.** Dans cet exemple, il apparaît, au cours de l'exécution du logiciel vulnérable **2v** dans le système de traitement de données **3** à un instant donné, un branchement conditionnel **BC** indiquant au logiciel vulnérable **2v** l'endroit où poursuivre son déroulement, à savoir l'un des trois endroits possibles $\mathbf{B_1}$, $\mathbf{B_2}$ ou $\mathbf{B_3}$. Il doit être compris que le branchement conditionnel **BC** prend la décision de poursuivre l'exécution du logiciel à l'endroit $\mathbf{B_1}$, $\mathbf{B_2}$ ou $\mathbf{B_3}$.

La **fig. 91** illustre un exemple de mise en oeuvre de l'invention pour lequel le branchement conditionnel choisi pour être déporté dans l'unité **6**, correspond au branchement conditionnel **BC**. Dans cet exemple, lors de l'exécution dans le système de traitement de données **3** de la première partie d'exécution **2pes** du logiciel protégé **2p** et en présence de l'unité **6**, il apparaît :

- à l'instant t_1, l'exécution de la commande de branchement conditionnel CBC_1 déclenchant dans l'unité **6**, l'exécution au moyen de la seconde partie d'exécution **2peu,** du branchement conditionnel déporté **bc** algorithmiquement équivalent au branchement conditionnel **BC**, cette commande de branchement conditionnel CBC_1 étant représentée par **TRIG(bc)**,
- et à l'instant t_2, le transfert de l'unité **6** vers le système de traitement de données **3**, de l'information permettant à la première partie d'exécution **2pes**, de poursuivre son exécution à l'endroit choisi, à savoir l'endroit B_1, B_2 ou B_3.

Il est à noter que lors de l'exécution d'une portion de la première partie d'exécution **2pes** du logiciel protégé **2p**, les commandes de branchements conditionnels exécutées dans le système de traitement de données **3** déclenchent l'exécution des branchements conditionnels déportés correspondants dans l'unité **6**. Ainsi, il apparaît qu'en présence de l'unité **6**, cette portion est exécutée correctement et que par conséquent, le logiciel protégé **2p** est complètement fonctionnel.

La **fig. 92** illustre une tentative d'exécution du logiciel protégé **2p**, alors que l'unité **6** est absente. Dans cet exemple, lors de l'exécution dans le système de traitement de données **3** de la première partie d'exécution **2pes** du logiciel protégé **2p** :

- à l'instant t_1, l'exécution de la commande de branchement conditionnel CBC_1, ne peut déclencher l'exécution du branchement conditionnel déporté **bc**, compte tenu de l'absence de l'unité **6**,
- et à l'instant t_2, le transfert de l'information permettant à la première partie d'exécution **2pes** de poursuivre à l'endroit choisi échoue compte tenu de l'absence de l'unité **6**.

Il apparaît donc qu'en l'absence de l'unité **6**, au moins une demande d'une portion de la première partie d'exécution **2pes** de déclencher l'exécution d'un branchement conditionnel déporté dans l'unité **6**, ne peut pas être satisfaite correctement, de sorte qu'au moins cette portion n'est pas exécutée correctement et que, par conséquent, le logiciel protégé **2p** n'est pas complètement fonctionnel.

Dans la description qui précède en relation des **fig. 90** à **92**, l'objet de l'invention vise à déporter dans l'unité **6**, un branchement conditionnel. Bien entendu, une réalisation préférée de l'invention peut consister à déporter dans l'unité **6**, une série de branchements conditionnels dont la fonctionnalité globale est équivalente à l'ensemble des fonctionnalités des branchements conditionnels qui ont été déportés. L'exécution de la fonctionnalité globale de cette série de branchements conditionnels déportés aboutit à la mise à disposition, pour le système de traitement de données **3**, d'une information permettant à la première partie d'exécution **2pes** du logiciel protégé **2p** de poursuivre son exécution à l'endroit choisi.

Dans la description qui précède en relation des **fig. 40** à **92**, cinq principes différents de protection d'un logiciel ont été explicités de manière générale indépendamment les uns des autres. Le procédé de protection conforme à l'invention, est mis en oeuvre en utilisant le principe de protection par renommage, éventuellement associé à un ou plusieurs autres principes de protection. Dans le cas où le principe de protection par renommage est complété par la mise en œuvre d'au moins un autre principe de protection, le principe de protection par renommage est complété avantageusement par le principe de protection par variable et/ou le principe de protection par fonctions élémentaires et/ou le principe de protection par branchement conditionnel.

Et lorsque le principe de protection par fonctions élémentaires est aussi mis en œuvre, celui-ci peut être complété à son tour par le principe de protection par détection et coercition et/ou le principe de protection par branchement conditionnel.

Et lorsque le principe de protection par détection et coercition est aussi mis en œuvre, celui-ci peut être complété à son tour par le principe de protection par branchement conditionnel.

Selon la variante préférée de réalisation, le principe de protection par renommage est complété par le principe de protection par variable et par le principe de protection par fonctions élémentaires, complété par le principe de protection par détection et coercition, complété par le principe de protection par branchement conditionnel.

Dans le cas où un principe de protection est appliqué, en complément du principe de protection par renommage, sa description effectuée précédemment doit

comporter, pour tenir compte de sa mise en oeuvre combinée, les modifications suivantes :

- la notion de logiciel vulnérable doit être comprise comme logiciel vulnérable vis-à-vis du principe de protection en cours de description. Ainsi, dans le cas où un principe de protection a déjà été appliqué au logiciel vulnérable, l'expression « logiciel vulnérable » doit être interprétée par le lecteur comme l'expression « logiciel protégé par le ou les principes de protection déjà appliqué(s)» ;
- la notion de logiciel protégé doit être comprise comme logiciel protégé vis-à-vis du principe de protection en cours de description. Ainsi, dans le cas où un principe de protection a déjà été appliqué, l'expression « logiciel protégé » doit être interprétée par le lecteur comme l'expression « nouvelle version du logiciel protégé »;
- et le ou les choix effectué(s) pour la mise en oeuvre du principe de protection en cours de description doit(vent) tenir compte du ou des choix effectué(s) pour la mise en oeuvre du ou des principe(s) de protection déjà appliqué(s).

La suite de la description permet de mieux comprendre la mise en oeuvre du procédé de protection conforme à l'invention. Ce procédé de protection selon l'invention fait intervenir, comme cela apparaît plus précisément à la **fig. 100** :

- d'abord, une phase de protection **P** au cours de laquelle un logiciel vulnérable **2v** est modifié en un logiciel protégé **2p,**
- ensuite, une phase d'utilisation **U** au cours de laquelle le logiciel protégé **2p** est mis en oeuvre. Dans cette phase d'utilisation **U** :
  - en présence de l'unité **6** et à chaque fois qu'une portion de la première partie d'exécution **2pes** exécutée dans le système de traitement de données **3** l'impose, une fonctionnalité imposée est exécutée dans l'unité **6**, de sorte que cette portion est exécutée correctement et que, par conséquent, le logiciel protégé **2p** est complètement fonctionnel,
  - en l'absence de l'unité **6** et malgré la demande d'une portion de la première partie d'exécution **2pes** d'exécuter une fonctionnalité dans

l'unité **6**, cette demande ne peut pas être honorée correctement, de sorte qu'au moins cette portion n'est pas exécutée correctement et que par conséquent, le logiciel protégé **2p** n'est pas complètement fonctionnel,

- et éventuellement une phase de rechargement **R** au cours de laquelle est créditée au moins une utilisation supplémentaire d'une fonctionnalité protégée par la mise en oeuvre de la deuxième variante préférée de réalisation du principe de protection par détection et coercition utilisant comme caractéristique, une variable de mesure.

La phase de protection **P** peut être décomposée en deux sous-phases de protection $\mathbf{P_1}$ et $\mathbf{P_2}$. La première, dite sous-phase de protection amont $\mathbf{P_1}$, est mise en oeuvre indépendamment du logiciel vulnérable **2v** à protéger. La deuxième, dite sous-phase de protection aval $\mathbf{P_2}$ est dépendante du logiciel vulnérable **2v** à protéger. Il est à noter que les sous-phases de protection amont $\mathbf{P_1}$ et aval $\mathbf{P_2}$ peuvent être réalisées avantageusement par deux personnes différentes ou deux équipes différentes. Par exemple, la sous-phase de protection amont $\mathbf{P_1}$ peut être réalisée par une personne ou une société assurant le développement de systèmes de protection de logiciels, tandis que la sous-phase de protection aval $\mathbf{P_2}$ peut être réalisée par une personne ou une société assurant le développement de logiciels devant être protégés. Bien entendu, il est clair que les sous-phases de protection amont $\mathbf{P_1}$ et aval $\mathbf{P_2}$ peuvent aussi être réalisées par la même personne ou la même équipe.

La sous-phase de protection amont $\mathbf{P_1}$ fait intervenir plusieurs stades $\mathbf{S_{11}}$, ..., $\mathbf{S_{1i}}$ pour chacun desquels différentes tâches ou travaux sont à effectuer.

Le premier stade de cette sous-phase de protection amont $\mathbf{P_1}$ est appelé "stade de définitions $\mathbf{S_{11}}$". Lors de ce stade de définitions $\mathbf{S_{11}}$:

- il est choisi :
  - le type de l'unité **6**. A titre illustratif, il peut-être choisi en tant qu'unité **6**, un lecteur **8** de cartes à puce et la carte à puce **7** associée au lecteur,
  - et les moyens de transfert **12**, **13** destinés à être mis en oeuvre respectivement dans le système de traitement de données **3** et dans l'unité **6**, au cours de la phase d'utilisation **U** et aptes à assurer le transfert de données entre le système de traitement de données **3** et l'unité **6**,

- il est défini :
  - un ensemble de fonctions dépendantes dont les fonctions dépendantes sont susceptibles d'être exécutées dans une unité **6**,
  - un ensemble de commandes déclenchantes pour cet ensemble de fonctions dépendantes, ces commandes déclenchantes étant susceptibles d'être exécutées dans le système de traitement de données **3** et de déclencher l'exécution dans une unité **6**, des fonctions dépendantes,
  - pour chaque commande déclenchante, une consigne correspondant au moins en partie à l'information transmise depuis le système de traitement de données **3** vers une unité **6**, afin de déclencher l'exécution de la fonction dépendante correspondante dans une unité **6**, cette consigne se présentant sous la forme d'au moins un argument de la commande déclenchante,
  - une méthode de renommage des consignes permettant de renommer les consignes afin d'obtenir des commandes déclenchantes à consignes renommées,
  - et des moyens de rétablissement **20** destinés à être mis en œuvre dans une unité **6** au cours d'une phase d'utilisation **U**, et permettant de retrouver la fonction dépendante à exécuter, à partir de la consigne renommée,
- et dans le cas où le procédé de protection selon l'invention met en oeuvre une variante du principe de protection par renommage, il est aussi défini pour au moins une fonction dépendante, une famille de fonctions dépendantes algorithmiquement équivalentes, mais déclenchées par des commandes déclenchantes dont les consignes renommées sont différentes,
- et dans le cas où le procédé de protection selon l'invention met en oeuvre une variante préférée du principe de protection par renommage, il est aussi défini :
  - en tant que méthode de renommage des consignes, une méthode de chiffrement pour chiffrer les consignes,

- et en tant que moyens de rétablissement **20**, des moyens mettant en oeuvre une méthode de déchiffrement pour déchiffrer les consignes renommées et rétablir ainsi l'identité des fonctions dépendantes à exécuter dans l'unité **6**.

- et dans le cas où le procédé de protection selon l'invention met en oeuvre le principe de protection par fonctions élémentaires, il est aussi défini :
  - un jeu de fonctions élémentaires, sous-ensenble de l'ensemble des fonctions dépendantes,
  - et un jeu de commandes élémentaires pour ce jeu de fonctions élémentaires, ce jeu de commandes élémentaires étant un sous-ensemble de l'ensemble des commandes déclenchantes,
- et dans le cas où le procédé de protection selon l'invention met en oeuvre le principe de protection par détection et coercition, il est aussi défini :
  - au moins une caractéristique d'exécution de logiciel, susceptible d'être surveillée au moins en partie dans l'unité **6**,
  - au moins un critère à respecter pour au moins une caractéristique d'exécution de logiciel,
  - des moyens de détection **17** à mettre en oeuvre dans l'unité **6** et permettant de détecter qu'au moins une caractéristique d'exécution de logiciel ne respecte pas au moins un critère associé,
  - et des moyens de coercition **18** à mettre en oeuvre dans l'unité **6** et permettant d'informer le système de traitement de données **3** et/ou de modifier l'exécution d'un logiciel, lorsqu'au moins un critère n'est pas respecté,
- et dans le cas où le procédé de protection selon l'invention met en oeuvre le principe de protection par détection et coercition utilisant comme caractéristique une variable de mesure de l'exécution du logiciel, il est aussi défini :
  - en tant que caractéristique d'exécution de logiciel susceptible d'être surveillée, une variable de mesure de l'utilisation d'une fonctionnalité d'un logiciel,

- en tant que critère à respecter, au moins un seuil associé à chaque variable de mesure,
- et des moyens d'actualisation permettant de mettre à jour au moins une variable de mesure,

- et dans le cas où le procédé de protection selon l'invention met en oeuvre une première variante préférée de réalisation du principe de protection par détection et coercition utilisant comme caractéristique une variable de mesure de l'exécution du logiciel, il est aussi défini :
  - pour au moins une variable de mesure, plusieurs seuils associés,
  - et des moyens de coercition différents correspondant à chacun de ces seuils,
- et dans le cas où le procédé de protection selon l'invention met en oeuvre une seconde variante préférée de réalisation du principe de protection par détection et coercition utilisant comme caractéristique une variable de mesure de l'exécution du logiciel, il est aussi défini des moyens de rechargement permettant de créditer au moins une utilisation supplémentaire pour au moins une fonctionnalité de logiciel surveillée par une variable de mesure,
- et dans le cas où le procédé de protection selon l'invention met en oeuvre le principe de protection par détection et coercition utilisant comme caractéristique un profil d'utilisation de logiciel, il est aussi défini :
  - en tant que caractéristique d'exécution de logiciel susceptible d'être surveillée, un profil d'utilisation de logiciel,
  - et en tant que critère à respecter, au moins un trait d'exécution de logiciel,
- et dans le cas où le procédé de protection selon l'invention met en oeuvre le principe de protection par détection et coercition utilisant comme trait d'exécution à respecter, la surveillance de l'enchaînement d'exécution, il est aussi défini :
  - un jeu d'instructions dont les instructions sont susceptibles d'être exécutées dans l'unité **6**,

- un jeu de commandes d'instructions pour ce jeu d'instructions, ces commandes d'instructions étant susceptibles d'être exécutées dans le système de traitement de données **3** et de déclencher dans l'unité **6** l'exécution des instructions,
- en tant que profil d'utilisation, l'enchaînement des instructions,
- en tant que trait d'exécution, un enchaînement souhaité pour l'exécution des instructions,
- en tant que moyens de détection **17**, des moyens permettant de détecter que l'enchaînement des instructions ne correspond pas à celui souhaité,
- et en tant que moyens de coercition **18**, des moyens permettant d'informer le système de traitement de données **3** et/ou de modifier le fonctionnement de la portion de logiciel protégé **2p** lorsque l'enchaînement des instructions ne correspond pas à celui souhaité,

• et dans le cas où le procédé de protection selon l'invention met en oeuvre une variante préférée de réalisation du principe de protection par détection et coercition utilisant comme trait d'exécution à respecter, la surveillance de l'enchaînement d'exécution, il est aussi défini :

- en tant que jeu d'instructions, un jeu d'instructions dont au moins certaines instructions travaillent sur des registres et utilisent au moins un opérande en vue de rendre un résultat,
- pour au moins une partie des instructions travaillant sur des registres :
  - ➢ une partie **PF** définissant la fonctionnalité de l'instruction,
  - ➢ et une partie définissant l'enchaînement souhaité pour l'exécution des instructions et comportant des champs de bits correspondant à :
    - ◊ un champ d'identification de l'instruction **CII**,
    - ◊ et pour chaque opérande de l'instruction :
      - * un champ drapeau $\mathbf{CD_k}$,
      - * et un champ d'identification prévue $\mathbf{CIP_k}$ de l'opérande,
- pour chaque registre appartenant aux moyens d'exploitation et utilisé par le jeu d'instructions, un champ d'identification générée $\mathbf{CIG_v}$ dans

lequel est automatiquement mémorisée l'identification de la dernière instruction ayant rendu son résultat dans ce registre,

- en tant que moyens de détection **17**, des moyens permettant, lors de l'exécution d'une instruction, pour chaque opérande, lorsque le champ drapeau $\mathbf{CD_k}$ l'impose, de contrôler l'égalité entre le champ d'identification générée $\mathbf{CIG_v}$ correspondant au registre utilisé par cet opérande, et le champ d'identification prévue $\mathbf{CIP_k}$ de l'origine de cet opérande,
- et en tant que moyens de coercition **18**, des moyens permettant de modifier le résultat des instructions, si au moins une des égalités contrôlées est fausse.

Lors de la sous-phase de protection amont, le stade de définition $\mathbf{S_{11}}$ est suivi par un stade appelé "stade de construction $\mathbf{S_{12}}$". Lors d'un tel stade $\mathbf{S_{12}}$, sont construits les moyens de transfert **12**, **13** et les moyens d'exploitation correspondant aux définitions du stade de définition $\mathbf{S_{11}}$.

Lors de ce stade de construction $\mathbf{S_{12}}$, il est donc procédé :

- à la construction des moyens de transfert **12**, **13** permettant, au cours de la phase d'utilisation **U**, le transfert de données entre le système de traitement de données **3** et l'unité **6**,
- à la construction des moyens d'exploitation permettant à l'unité **6**, au cours de la phase d'utilisation **U** de mettre en oeuvre les moyens de rétablissement,
- et lorsque le principe de protection par fonctions élémentaires est aussi mis en oeuvre, à la construction des moyens d'exploitation permettant aussi à l'unité **6**, au cours de la phase d'utilisation **U** d'exécuter les fonctions élémentaires du jeu de fonctions élémentaires,
- et lorsque le principe de protection par détection et coercition est aussi mis en oeuvre, à la construction :
  - des moyens d'exploitation permettant à l'unité **6**, au cours de la phase d'utilisation **U** de mettre aussi en oeuvre les moyens de détection **17** et les moyens de coercition **18**,

- et éventuellement des moyens d'exploitation permettant à l'unité **6**, au cours de la phase d'utilisation **U** de mettre aussi en oeuvre les moyens d'actualisation,
- et éventuellement des moyens d'exploitation permettant à l'unité **6**, au cours de la phase de rechargement de mettre aussi en oeuvre les moyens de rechargement,
- et éventuellement des moyens d'exploitation permettant aussi à l'unité **6**, au cours de la phase d'utilisation **U** d'exécuter les instructions du jeu d'instruction.

La construction des moyens d'exploitation est réalisée de manière classique, par l'intermédiaire d'une unité de développement de programme en prenant en compte les définitions intervenues au stade de définitions S_{11}. Une telle unité est décrite dans la suite de la description à la **fig. 110**.

Lors de la sous-phase de protection amont P_1, le stade de construction S_{12} peut être suivi par un stade appelé "stade de pré-personnalisation S_{13}". Lors de ce stade de pré-personnalisation S_{13} au moins une partie des moyens de transfert **13** et/ou les moyens d'exploitation sont chargés dans au moins une unité vierge **60** en vue d'obtenir au moins une unité pré-personnalisée **66**. Il est à noter qu'une partie des moyens d'exploitation, une fois transférée dans une unité pré-personnalisée **66**, n'est plus directement accessible de l'extérieur de cette unité pré-personnalisée **66**. Le transfert des moyens d'exploitation dans une unité vierge **60** peut être réalisé par l'intermédiaire d'une unité de pré-personnalisation adaptée, qui est décrite dans la suite de la description à la **fig. 120**. Dans le cas d'une unité pré-personnalisée **66**, constituée d'une carte à puce **7** et de son lecteur **8**, la pré-personnalisation ne concerne que la carte à puce **7**.

Lors de la sous-phase de protection amont P_1, il peut être mis en oeuvre, après le stade de définition S_{11} et, éventuellement après le stade de construction S_{12}, un stade appelé "stade de confection d'outils S_{14}". Lors de ce stade de confection d'outils S_{14} sont confectionnés des outils permettant d'aider à la génération de logiciels protégés ou d'automatiser la protection de logiciels. De tels outils permettent :

- d'aider à choisir ou de choisir automatiquement dans le logiciel vulnérable **2v** à protéger :

- le ou les traitements algorithmiques susceptibles d'être décomposées en fonctions dépendantes déportables dans l'unité **6** et pour lesquelles les consignes des commandes déclenchantes peuvent être renommées,
- la ou les portions susceptibles d'être modifiées,
- et lorsque le principe de protection par variable est aussi mis en oeuvre, la ou les variables susceptibles d'être déportées dans l'unité **6**,
- et lorsque le principe de protection par fonctions élémentaires est aussi mis en oeuvre, le ou les traitements algorithmiques susceptibles d'être décomposées en fonctions élémentaires déportables dans l'unité **6**,
- et lorsque le principe de protection par détection et coercition est aussi mis en oeuvre, la ou les caractéristiques d'exécution à surveiller et, éventuellement, le ou les traitements algorithmiques susceptibles d'être décomposées en instructions déportables dans l'unité **6**,
- et lorsque le principe de protection par branchement conditionnel est aussi mis en oeuvre, le ou les branchements conditionnels dont la fonctionnalité est susceptible d'être déportée dans l'unité **6**,

• et, éventuellement, d'aider à générer des logiciels protégés ou d'automatiser la protection de logiciels.

Ces différents outils peuvent être réalisés indépendamment ou en combinaison et chaque outil peut prendre diverses formes, comme par exemple préprocesseur, assembleur, compilateur, etc.

La sous-phase de protection amont $\mathbf{P_1}$ est suivie par une sous-phase de protection aval $\mathbf{P_2}$ dépendante du logiciel vulnérable **2v** à protéger. Cette sous-phase de protection aval $\mathbf{P_2}$ fait intervenir également plusieurs stades. Le premier stade correspondant à la mise en oeuvre du principe de protection par renommage est appelé "stade de création $\mathbf{S_{21}}$". Lors de ce stade de création $\mathbf{S_{21}}$, il est utilisé les choix intervenus au stade de définition $\mathbf{S_{11}}$. A l'aide de ces choix et éventuellement d'outils construits au stade de confection d'outils $\mathbf{S_{14}}$, il est créé le logiciel protégé **2p** :

• en choisissant, au moins un traitement algorithmique qui, lors de l'exécution du logiciel vulnérable **2v**, utilise au moins un opérande et permet d'obtenir au moins un résultat,

- en choisissant au moins une portion du source du logiciel vulnérable **2vs** contenant au moins un traitement algorithmique choisi,
- en produisant le source du logiciel protégé **2ps** à partir du source du logiciel vulnérable **2vs**, en modifiant au moins une portion choisie du source du logiciel vulnérable **2vs** pour obtenir au moins une portion modifiée du source du logiciel protégé **2ps**, cette modification étant telle que :
  - lors de l'exécution du logiciel protégé **2p** une première partie d'exécution **2pes** est exécutée dans le système de traitement de données **3** et une seconde partie d'exécution **2peu** est exécutée dans une unité **6**, obtenue à partir de l'unité vierge **60** après chargement d'informations,
  - la seconde partie d'exécution **2peu** exécute au moins la fonctionnalité d'au moins un traitement algorithmique choisi,
  - au moins un traitement algorithmique choisi est décomposé de manière que lors de l'exécution du logiciel protégé **2p,** ce traitement algorithmique est exécuté, au moyen de la seconde partie d'exécution **2peu**, en utilisant des fonctions dépendantes,
  - pour au moins un traitement algorithmique choisi, des commandes déclenchantes à consignes renommées sont intégrées dans le source du logiciel protégé **2ps**, de manière que lors de l'exécution du logiciel protégé **2p**, chaque commande déclenchante à consigne renommée est exécutée par la première partie d'exécution **2pes** et déclenche dans l'unité **6**, le rétablissement, au moyen des moyens de rétablissement **20**, de la consigne et l'exécution, au moyen de la seconde partie d'exécution **2peu**, de la fonction dépendante correspondante,
  - et un ordonnancement des commandes déclenchantes à consignes renommées est choisi parmi l'ensemble des ordonnancements permettant l'exécution du logiciel protégé **2p**,
- et en produisant :
  - une première partie objet **2pos** du logiciel protégé **2p**, à partir du source du logiciel protégé **2ps**, cette première partie objet **2pos** étant telle que lors de l'exécution du logiciel protégé **2p**, apparaît une

première partie d'exécution **2pes** qui est exécutée dans le système de traitement de données **3** et dont au moins une portion prend en compte que les commandes déclenchantes à consignes renommées sont exécutées selon l'ordonnancement choisi,

– et une seconde partie objet **2pou** du logiciel protégé **2p**, contenant les moyens d'exploitation, cette seconde partie objet **2pou** étant telle que, après chargement dans l'unité vierge **60** et lors de l'exécution du logiciel protégé **2p**, apparaît la seconde partie d'exécution **2peu** au moyen de laquelle les consignes sont rétablies et les fonctions dépendantes sont exécutées.

Pour la mise en oeuvre d'une variante du principe de protection par renommage, le logiciel protégé **2p** est modifié :

- en choisissant dans le source du logiciel protégé **2ps** au moins une commande déclenchante à consigne renommée,
- et en modifiant au moins une portion choisie du source du logiciel protégé **2ps** en remplaçant au moins la consigne renommée d'une commande déclenchante à consigne renommée choisie, par une autre consigne renommée, déclenchant une fonction dépendante de la même famille.

Lors de la sous-phase de protection aval $\mathbf{P_2}$, et lorsqu'au moins un autre principe de protection est appliqué en plus du principe de protection par renommage, il est mis en oeuvre un "stade de modification $\mathbf{S_{22}}$". Lors de ce stade de modification $\mathbf{S_{22}}$, il est utilisé les définitions intervenues au stade de définitions $\mathbf{S_{11}}$. A l'aide de ces définitions et éventuellement d'outils construits au stade de confection d'outils $\mathbf{S_{14}}$, le logiciel protégé **2p** est modifié pour permettre la mise en oeuvre des principes de protection selon l'un des arrangements définis ci-avant.

Lorsque le principe de protection par variable est mis en oeuvre, le logiciel protégé **2p** est modifié :

- en choisissant au moins une variable utilisée dans au moins un traitement algorithmique choisi, qui lors de l'exécution du logiciel protégé **2p**, définit partiellement l'état du logiciel protégé **2p**,
- en modifiant au moins une portion choisie du source du logiciel protégé **2ps**, cette modification étant telle que lors de l'exécution du logiciel protégé

2p, au moins une variable choisie ou au moins une copie de variable choisie réside dans l'unité **6**,

- et en produisant :
  - la première partie objet **2pos** du logiciel protégé **2p**, cette première partie objet **2pos** étant telle que lors de l'exécution du logiciel protégé **2p**, au moins une portion de la première partie d'exécution **2pes** prend aussi en compte qu'au moins une variable ou au moins une copie de variable réside dans l'unité **6**,
  - et la seconde partie objet **2pou** du logiciel protégé **2p**, cette seconde partie objet **2pou** étant telle que, après chargement dans l'unité **6** et lors de l'exécution du logiciel protégé **2p**, apparaît la seconde partie d'exécution **2peu** au moyen de laquelle au moins une variable choisie, ou au moins une copie de variable choisie réside aussi dans l'unité **6**.

Lorsque le principe de protection par fonctions élémentaires est mis en oeuvre, le logiciel protégé **2p** est modifié :

- en modifiant au moins une portion choisie du source du logiciel protégé **2ps**, cette modification étant telle que la décomposition d'au moins un traitement algorithmique choisi en fonctions dépendantes n'utilise que des fonctions élémentaires,
- en produisant :
  - la première partie objet **2pos** du logiciel protégé **2p**, cette première partie objet **2pos** étant telle que lors de l'exécution du logiciel protégé **2p**, au moins une portion de la première partie d'exécution **2pes** exécute aussi les commandes élémentaires selon l'ordonnancement choisi,
  - et la seconde partie objet **2pou** du logiciel protégé **2p** contenant aussi les moyens d'exploitation, cette seconde partie objet **2pou** étant telle que, après chargement dans l'unité **6** et lors de l'exécution du logiciel protégé **2p**, apparaît la seconde partie d'exécution **2peu** au moyen de laquelle sont aussi exécutées les fonctions élémentaires déclenchées par la première partie d'exécution **2pes**.

Lorsque le principe de protection par détection et coercition est mis en oeuvre, le logiciel protégé **2p** est modifié :

- en choisissant au moins une caractéristique d'exécution de logiciel à surveiller, parmi les caractéristiques d'exécution susceptibles d'être surveillées,
- en choisissant au moins un critère à respecter pour au moins une caractéristique d'exécution de logiciel choisie,
- en choisissant dans le source du logiciel protégé **2ps**, des fonctions élémentaires pour lesquelles au moins une caractéristique d'exécution de logiciel choisie, est à surveiller,
- en modifiant au moins une portion choisie du source du logiciel protégé **2ps**, cette modification étant telle que lors de l'exécution du logiciel protégé **2p**, au moins une caractéristique d'exécution choisie est surveillée au moyen de la seconde partie d'exécution **2peu**, et le non respect d'un critère aboutit à une information du système de traitement de données **3** et/ou à une modification de l'exécution du logiciel protégé **2p**,
- et en produisant la seconde partie objet **2pou** du logiciel protégé **2p** contenant les moyens d'exploitation mettant aussi en oeuvre les moyens de détection **17** et les moyens de coercition **18**, cette seconde partie objet **2pou** étant telle que, après chargement dans l'unité **6** et lors de l'exécution du logiciel protégé **2p**, au moins une caractéristique d'exécution de logiciel est surveillée et le non respect d'un critère aboutit à une information du système de traitement de données **3** et/ou à une modification de l'exécution du logiciel protégé **2p**.

Pour la mise en oeuvre du principe de protection par détection et coercition utilisant comme caractéristique une variable de mesure de l'exécution du logiciel, le logiciel protégé **2p** est modifié :

- en choisissant en tant que caractéristique d'exécution de logiciel à surveiller, au moins une variable de mesure de l'utilisation d'une fonctionnalité d'un logiciel,
- en choisissant :

- au moins une fonctionnalité du logiciel protégé **2p** dont l'utilisation est susceptible d'être surveillée grâce à une variable de mesure,
- au moins une variable de mesure servant à quantifier l'utilisation de ladite fonctionnalité,
- au moins un seuil associé à une variable de mesure choisie correspondant à une limite d'utilisation de ladite fonctionnalité,
- et au moins une méthode de mise à jour d'une variable de mesure choisie en fonction de l'utilisation de ladite fonctionnalité,

• et en modifiant au moins une portion choisie du source du logiciel protégé **2ps**, cette modification étant telle que, lors de l'exécution du logiciel protégé **2p**, la variable de mesure est actualisée au moyen de la seconde partie d'exécution **2peu**, en fonction de l'utilisation de ladite fonctionnalité et au moins un dépassement de seuil est pris en compte.

Pour la mise en oeuvre d'une première variante préférée de réalisation du principe de protection par détection et coercition utilisant, comme caractéristique, une variable de mesure, le logiciel protégé **2p** est modifié :

• en choisissant dans le source du logiciel protégé **2ps**, au moins une variable de mesure choisie à laquelle doivent être associés plusieurs seuils correspondants à des limites différentes d'utilisation de la fonctionnalité,

• en choisissant au moins deux seuils associés à la variable de mesure choisie,

• et en modifiant au moins une portion choisie du source du logiciel protégé **2ps**, cette modification étant telle que, lors de l'exécution du logiciel protégé **2p**, les dépassements des divers seuils sont pris en compte, au moyen de la seconde partie d'exécution **2peu**, de manière différente.

Pour la mise en oeuvre d'une seconde variante préférée de réalisation du principe de protection par détection et coercition utilisant comme caractéristique, une variable de mesure, le logiciel protégé **2p** est modifié :

• en choisissant dans le source du logiciel protégé **2ps**, au moins une variable de mesure choisie permettant de limiter l'utilisation d'une fonctionnalité à laquelle au moins une utilisation supplémentaire doit pouvoir être créditée,

- et en modifiant au moins une portion choisie, cette modification étant telle que dans une phase dite de rechargement, au moins une utilisation supplémentaire d'au moins une fonctionnalité correspondant à une variable de mesure choisie peut être créditée.

Pour la mise en oeuvre du principe de protection par détection et coercition utilisant comme caractéristique, un profil d'utilisation de logiciel, le logiciel protégé **2p** est modifié :

- en choisissant en tant que caractéristique d'exécution de logiciel à surveiller au moins un profil d'utilisation de logiciel,
- en choisissant au moins un trait d'exécution qu'au moins un profil d'utilisation choisi doit respecter,
- et en modifiant au moins une portion choisie du source du logiciel protégé **2ps**, cette modification étant telle que, lors de l'exécution du logiciel protégé **2p**, la seconde partie d'exécution **2peu** respecte tous les traits d'exécution choisis.

Pour la mise en oeuvre du principe de protection par détection et coercition utilisant comme trait d'exécution à respecter, la surveillance de l'enchaînement d'exécution, le logiciel protégé **2p** est modifié :

- en modifiant au moins une portion choisie du source du logiciel protégé **2ps** :
  - en transformant les fonctions élémentaires en instructions,
  - en spécifiant l'enchaînement que doivent respecter au moins certaines des instructions lors de leur exécution dans l'unité **6**,
  - et en transformant les commandes élémentaires en commandes d'instructions correspondant aux instructions utilisées.

Lorsque le principe de protection par branchement conditionnel est mis en oeuvre, le logiciel protégé **2p** est modifié :

- en choisissant dans le source du logiciel protégé **2ps**, au moins un branchement conditionnel effectué dans au moins un traitement algorithmique choisi,
- en modifiant au moins une portion choisie du source du logiciel protégé

2ps, cette modification étant telle que lors de l'exécution du logiciel protégé **2p**, la fonctionnalité d'au moins un branchement conditionnel choisi, est exécutée, au moyen de la seconde partie d'exécution **2peu**, dans l'unité **6,**

- et en produisant :
  - la première partie objet **2pos** du logiciel protégé **2p**, cette première partie objet **2pos** étant telle que lors de l'exécution du logiciel protégé **2p**, la fonctionnalité d'au moins un branchement conditionnel choisi est exécutée dans l'unité **6**,
  - et la seconde partie objet **2pou** du logiciel protégé **2p**, cette seconde partie objet **2pou** étant telle que, après chargement dans l'unité **6** et lors de l'exécution du logiciel protégé **2p**, apparaît la seconde partie d'exécution **2peu** au moyen de laquelle la fonctionnalité d'au moins un branchement conditionnel choisi est exécutée.

Pour la mise en oeuvre d'une réalisation préférée du principe de protection par branchement conditionnel, le logiciel protégé **2p** est modifié :

- en choisissant, dans le source du logiciel protégé **2ps** au moins une série de branchements conditionnels choisis,
- en modifiant au moins une portion choisie du source du logiciel protégé **2ps**, cette modification étant telle que lors de l'exécution du logiciel protégé **2p**, la fonctionnalité globale d'au moins une série choisie de branchements conditionnels est exécutée au moyen de la seconde partie d'exécution **2peu**, dans l'unité **6**,
- et en produisant :
  - la première partie objet **2pos** du logiciel protégé **2p**, cette première partie objet **2pos** étant telle que lors de l'exécution du logiciel protégé **2p**, la fonctionnalité d'au moins une série choisie de branchements conditionnels est exécutée dans l'unité **6**,
  - et la seconde partie objet **2pou** du logiciel protégé **2p**, cette seconde partie objet **2pou** étant telle que, après chargement dans l'unité **6** et lors de l'exécution du logiciel protégé **2p**, apparaît la seconde partie d'exécution **2peu** au moyen de laquelle la fonctionnalité globale d'au

moins une série choisie de branchements conditionnels est exécutée.

Bien entendu, les principes de protection selon l'invention peuvent être appliqués directement lors du développement d'un nouveau logiciel sans nécessiter la réalisation préalable de logiciels protégés intermédiaires. De cette façon, les stades de création S_{21} et de modification S_{22} peuvent être effectués concomitamment de manière à obtenir directement le logiciel protégé **2p**.

Lors de la sous-phase de protection aval P_2, il est mis en oeuvre après le stade de création S_{21} du logiciel protégé **2p**, et éventuellement après le stade de modification S_{22}, un stade appelé "stade de personnalisation S_{23}". Lors de ce stade de personnalisation S_{23}, la seconde partie objet **2pou** contenant les moyens d'exploitation est chargée dans au moins une unité vierge **60**, en vue d'obtenir au moins une unité **6**, ou une partie de la seconde partie objet **2pou** contenant éventuellement les moyens d'exploitation est chargée dans au moins une unité pré-personnalisée **66**, en vue d'obtenir au moins une unité **6**. Le chargement de ces informations de personnalisation permet de rendre opérationnelle au moins une unité **6**. Il est à noter qu'une partie de ces informations, une fois transférée dans une unité **6**, n'est pas directement accessible de l'extérieur de cette unité **6**. Le transfert des informations de personnalisation dans une unité vierge **60** ou une unité pré-personnalisée **66** peut être réalisé par l'intermédiaire d'une unité de personnalisation adaptée qui est décrite dans la suite de la description à la **fig. 150**. Dans le cas d'une unité **6**, constituée d'une carte à puce **7** et de son lecteur **8**, la personnalisation ne concerne que la carte à puce **7**.

Pour la mise en oeuvre de la phase de protection **P**, différents moyens techniques sont décrits plus précisément en relation des **fig. 110, 120, 130, 140 et 150**.

La **fig. 110** illustre un exemple de réalisation d'un système **25** permettant de mettre en oeuvre le stade de construction S_{12} prenant en compte les définitions intervenues au stade de définitions S_{11} et au cours duquel sont construits les moyens de transfert **12, 13** et éventuellement, les moyens d'exploitation destinés à l'unité **6**. Un tel système **25** comporte une unité de développement de programme ou station de travail se présentant classiquement sous la forme d'un ordinateur comprenant une unité centrale, un écran, des périphériques du type clavier-souris, et comportant,

notamment, les programmes suivants : éditeurs de fichiers, assembleurs, pré-processeurs, compilateurs, interpréteurs, debuggers et éditeurs de liens.

La **fig. 120** illustre un exemple de réalisation d'une unité de pré-personnalisation **30** permettant de charger au moins en partie les moyens de transfert **13** et/ou les moyens d'exploitation dans au moins une unité vierge **60** en vue d'obtenir au moins une unité pré-personnalisée **66**. Cette unité de pré-personnalisation **30** comporte un moyen de lecture et d'écriture **31** permettant de pré-personnaliser de manière électrique, une unité vierge **60**, de manière à obtenir une unité pré-personnalisée **66** dans laquelle les moyens de transfert **13** et/ou d'exploitation ont été chargés. L'unité de pré-personnalisation **30** peut aussi comporter des moyens de personnalisation physique **32** de l'unité vierge **60** pouvant se présenter par exemple, sous la forme d'une imprimante. Dans le cas où l'unité **6** est constituée par une carte à puce **7** et son lecteur **8**, la pré-personnalisation concerne généralement uniquement la carte à puce **7**.

La **fig. 130** illustre un exemple de réalisation d'un système **35** permettant d'effectuer la confection des outils permettant d'aider à générer des logiciels protégés ou d'automatiser la protection de logiciels. Un tel système **35** comporte une unité de développement de programme ou station de travail se présentant classiquement sous la forme d'un ordinateur comprenant une unité centrale, un écran, des périphériques du type clavier-souris, et comportant, notamment, les programmes suivants : éditeurs de fichiers, assembleurs, pré-processeurs, compilateurs, interpréteurs, debuggers et éditeurs de liens.

La **fig. 140** illustre un exemple de réalisation d'un système **40** permettant de créer directement un logiciel protégé **2p** ou de modifier un logiciel vulnérable **2v** en vue d'obtenir un logiciel protégé **2p**. Un tel système **40** comporte une unité de développement de programme ou station de travail se présentant classiquement sous la forme d'un ordinateur comprenant une unité centrale, un écran, des périphériques du type clavier-souris, et comportant, notamment, les programmes suivants : éditeurs de fichiers, assembleurs, pré-processeurs, compilateurs, interpréteurs, debuggers et éditeurs de liens, ainsi que des outils permettant d'aider à générer des logiciels protégés ou d'automatiser la protection de logiciels.

La **fig. 150** illustre un exemple de réalisation d'une unité de personnalisation **45**

permettant de charger la seconde partie objet **2pou** dans au moins une unité vierge **60** en vue d'obtenir au moins une unité **6** ou une partie de la seconde partie objet **2pou** dans au moins une unité pré-personnalisée **66** en vue d'obtenir au moins une unité **6**. Cette unité de personnalisation **45** comporte un moyen de lecture et d'écriture **46** permettant de personnaliser de manière électrique, au moins une unité vierge **60** ou au moins une unité pré-personnalisée **66,** de manière à obtenir au moins une unité **6**. A l'issue de cette personnalisation, une unité **6** comporte les informations nécessaires à l'exécution du logiciel protégé **2p**. L'unité de personnalisation **45** peut aussi comporter des moyens de personnalisation physique **47** pour au moins une unité **6** pouvant se présenter par exemple, sous la forme d'une imprimante. Dans le cas où une unité **6** est constituée par une carte à puce **7** et son lecteur **8**, la personnalisation concerne généralement uniquement la carte à puce **7**.

Le procédé de protection de l'invention peut être mis en oeuvre avec les améliorations suivantes :

- Il peut être prévu d'utiliser conjointement plusieurs unités de traitement et de mémorisation dans lesquelles est répartie la seconde partie objet **2pou** du logiciel protégé **2p** de manière que leur utilisation conjointe permette d'exécuter le logiciel protégé **2p**, l'absence d'au moins une de ces unités de traitement et de mémorisation empêchant l'utilisation du logiciel protégé **2p**.
- De même, après le stade de pré-personnalisation S_{13} et lors du stade de personnalisation S_{23}, la partie de la seconde partie objet **2pou** nécessaire pour transformer l'unité pré-personnalisée **66** en une unité **6** peut être contenue dans une unité de traitement et de mémorisation utilisée par l'unité de personnalisation **45** afin de limiter l'accès à cette partie de la seconde partie objet **2pou**. Bien entendu, cette partie de la seconde partie objet **2pou** peut être répartie dans plusieurs unités de traitement et de mémorisation de manière que cette partie de la seconde partie objet **2pou** soit accessible uniquement lors de l'utilisation conjointe de ces unités de traitement et de mémorisation.

REVENDICATIONS :

1 - Procédé pour protéger, à partir d'au moins une unité vierge **(60)** comportant au moins des moyens de mémorisation **(15)** et des moyens de traitement **(16)**, un logiciel vulnérable **(2v)** contre son utilisation non autorisée, ledit logiciel vulnérable **(2v)** fonctionnant sur un système de traitement de données **(3)**, caractérisé en ce qu'il consiste :

→ dans une phase de protection **(P)** :

- à définir :
  - un ensemble de fonctions dépendantes dont les fonctions dépendantes sont susceptibles d'être exécutées dans une unité **(6)**,
  - un ensemble de commandes déclenchantes pour cet ensemble de fonctions dépendantes, ces commandes déclenchantes étant susceptibles d'être exécutées dans le système de traitement de données **(3)** et de déclencher l'exécution dans une unité **(6)**, des fonctions dépendantes,
  - pour chaque commande déclenchante, une consigne correspondant au moins en partie à l'information transmise depuis le système de traitement de données **(3)** vers une unité **(6)**, afin de déclencher l'exécution de la fonction dépendante correspondante dans une unité **(6)**, cette consigne se présentant sous la forme d'au moins un argument de la commande déclenchante,
  - une méthode de renommage des consignes permettant de renommer les consignes afin d'obtenir des commandes déclenchantes à consignes renommées,
  - et des moyens de rétablissement **(20)** destinés à être mis en œuvre dans une unité **(6)** au cours d'une phase d'utilisation **(U)**, et permettant de retrouver la fonction dépendante à exécuter, à partir de la consigne renommée,
- à construire des moyens d'exploitation permettant de transformer l'unité vierge **(60)** en une unité **(6)** capable de mettre en œuvre les moyens de

rétablissement **(20)**,

- à créer un logiciel protégé **(2p)** :
  - en choisissant, au moins un traitement algorithmique qui, lors de l'exécution du logiciel vulnérable **(2v)**, utilise au moins un opérande et permet d'obtenir au moins un résultat,
  - en choisissant au moins une portion du source du logiciel vulnérable **(2vs)** contenant au moins un traitement algorithmique choisi,
  - en produisant le source du logiciel protégé **(2ps)** à partir du source du logiciel vulnérable **(2vs)**, en modifiant au moins une portion choisie du source du logiciel vulnérable **(2vs)** pour obtenir au moins une portion modifiée du source du logiciel protégé **(2ps)**, cette modification étant telle que :
    - lors de l'exécution du logiciel protégé **(2p)** une première partie d'exécution **(2pes)** est exécutée dans le système de traitement de données **(3)** et une seconde partie d'exécution **(2peu)** est exécutée dans une unité **(6)**, obtenue à partir de l'unité vierge **(60)** après chargement d'informations,
    - la seconde partie d'exécution **(2peu)** exécute au moins la fonctionnalité d'au moins un traitement algorithmique choisi,
    - au moins un traitement algorithmique choisi est décomposé de manière que lors de l'exécution du logiciel protégé **(2p)**, ce traitement algorithmique est exécuté, au moyen de la seconde partie d'exécution **(2peu)**, en utilisant des fonctions dépendantes,
    - pour au moins un traitement algorithmique choisi, des commandes déclenchantes à consignes renommées sont intégrées dans le source du logiciel protégé **(2ps)**, de manière que lors de l'exécution du logiciel protégé **(2p)**, chaque commande déclenchante à consigne renommée est exécutée par la première partie d'exécution **(2pes)** et déclenche dans l'unité **(6)**, le rétablissement, au moyen des moyens de rétablissement **(20)**, de la consigne et l'exécution, au moyen de la seconde partie d'exécution **(2peu)**, de la fonction dépendante correspondante,

➢ et un ordonnancement des commandes déclenchantes à consignes renommées est choisi parmi l'ensemble des ordonnancements permettant l'exécution du logiciel protégé **(2p)**,

– et en produisant :

➢ une première partie objet **(2pos)** du logiciel protégé **(2p)**, à partir du source du logiciel protégé **(2ps)**, cette première partie objet **(2pos)** étant telle que lors de l'exécution du logiciel protégé **(2p)**, apparaît une première partie d'exécution **(2pes)** qui est exécutée dans le système de traitement de données **(3)** et dont au moins une portion prend en compte que les commandes déclenchantes à consignes renommées sont exécutées selon l'ordonnancement choisi,

➢ et une seconde partie objet **(2pou)** du logiciel protégé **(2p)**, contenant les moyens d'exploitation, cette seconde partie objet **(2pou)** étant telle que, après chargement dans l'unité vierge **(60)** et lors de l'exécution du logiciel protégé **(2p)**, apparaît la seconde partie d'exécution **(2peu)** au moyen de laquelle les consignes sont rétablies et les fonctions dépendantes sont exécutées,

- et à charger la seconde partie objet **(2pou)** dans l'unité vierge **(60)**, en vue d'obtenir l'unité **(6)**,

→ et dans une phase d'utilisation **(U)** au cours de laquelle est exécuté le logiciel protégé **(2p)**:

- en présence de l'unité **(6)** et à chaque fois qu'une commande déclenchante à consigne renommée, contenue dans une portion de la première partie d'exécution **(2pes)** l'impose, à rétablir dans l'unité **(6)**, l'identité de la fonction dépendante correspondante et à exécuter celle-ci, de sorte que cette portion est exécutée correctement et que, par conséquent, le logiciel protégé **(2p)** est complètement fonctionnel,
- et en l'absence de l'unité **(6)**, malgré la demande d'une portion de la première partie d'exécution **(2pes)** de déclencher l'exécution d'une fonction dépendante dans l'unité **(6)**, à ne pas pouvoir répondre correctement à cette demande, de sorte qu'au moins cette portion n'est pas exécutée

correctement et que, par conséquent, le logiciel protégé **(2p)** n'est pas complètement fonctionnel.

2- Procédé selon la revendication 1, caractérisé en ce qu'il consiste :

→ dans la phase de protection **(P)** :

- à définir pour au moins une fonction dépendante, une famille de fonctions dépendantes algorithmiquement équivalentes, mais déclenchées par des commandes déclenchantes dont les consignes renommées sont différentes,
- et à modifier le logiciel protégé **(2p) :**
  - en choisissant dans le source du logiciel protégé **(2ps)** au moins une commande déclenchante à consigne renommée,
  - et en modifiant au moins une portion choisie du source du logiciel protégé **(2ps)** en remplaçant au moins la consigne renommée d'une commande déclenchante à consigne renommée choisie, par une autre consigne renommée, déclenchant une fonction dépendante de la même famille.

3 - Procédé selon la revendication 1 ou 2, caractérisé en ce qu'il consiste :

→ dans la phase de protection **(P)** :

- à définir :
  - en tant que méthode de renommage des consignes, une méthode de chiffrement pour chiffrer les consignes,
  - et en tant que moyens de rétablissement **(20)**, des moyens mettant en oeuvre une méthode de déchiffrement pour déchiffrer les consignes renommées et rétablir ainsi l'identité des fonctions dépendantes à exécuter dans l'unité **(6)**.

4 - Procédé selon l'une des revendications 1 à 3, caractérisé en ce qu'il consiste :

→ dans la phase de protection **(P)** :

- à modifier le logiciel protégé **(2p)**:
  - en choisissant au moins une variable utilisée dans au moins un traitement algorithmique choisi, qui lors de l'exécution du logiciel protégé **(2p)**, définit partiellement l'état du logiciel protégé **(2p)**,
  - en modifiant au moins une portion choisie du source du logiciel

protégé **(2ps)**, cette modification étant telle que lors de l'exécution du logiciel protégé **(2p)**, au moins une variable choisie ou au moins une copie de variable choisie réside dans l'unité **(6)**,

- et en produisant :
  - la première partie objet **(2pos)** du logiciel protégé **(2p)**, cette première partie objet **(2pos)** étant telle que lors de l'exécution du logiciel protégé **(2p)**, au moins une portion de la première partie d'exécution **(2pes)** prend aussi en compte qu'au moins une variable ou au moins une copie de variable réside dans l'unité **(6)**,
  - et la seconde partie objet **(2pou)** du logiciel protégé **(2p)**, cette seconde partie objet **(2pou)** étant telle que, après chargement dans l'unité **(6)** et lors de l'exécution du logiciel protégé **(2p)**, apparaît la seconde partie d'exécution **(2peu)** au moyen de laquelle au moins une variable choisie, ou au moins une copie de variable choisie réside aussi dans l'unité **(6)**,

→ et dans la phase d'utilisation **(U)** :

- en présence de l'unité **(6)** à chaque fois qu'une portion de la première partie d'exécution **(2pes)** l'impose, à utiliser une variable ou une copie de variable résidant dans l'unité **(6)**, de sorte que cette portion est exécutée correctement et que, par conséquent, le logiciel protégé **(2p)** est complètement fonctionnel,
- et en l'absence de l'unité **(6)**, malgré la demande d'une portion de la première partie d'exécution **(2pes)** d'utiliser une variable ou une copie de variable résidant dans l'unité **(6)**, à ne pouvoir répondre correctement à cette demande, de sorte qu'au moins cette portion n'est pas exécutée correctement et que, par conséquent, le logiciel protégé **(2p)** n'est pas complètement fonctionnel.

5 - Procédé selon l'une des revendications 1 à 3, caractérisé en ce qu'il consiste :

→ dans la phase de protection **(P)** :

- à définir :
  - un jeu de fonctions élémentaires, sous-ensenble de l'ensemble des

fonctions dépendantes,

- et un jeu de commandes élémentaires pour ce jeu de fonctions élémentaires, ce jeu de commandes élémentaires étant un sous-ensemble de l'ensemble des commandes déclenchantes,

- à construire les moyens d'exploitation permettant à l'unité **(6)**, d'exécuter aussi les fonctions élémentaires dudit jeu, l'exécution de ces fonctions élémentaires étant déclenchée par l'exécution dans le système de traitement de données **(3)**, des commandes élémentaires dont la consigne a été renommée,
- et à modifier le logiciel protégé **(2p)**:
  - en modifiant au moins une portion choisie du source du logiciel protégé **(2ps)**, cette modification étant telle que la décomposition d'au moins un traitement algorithmique choisi en fonctions dépendantes n'utilise que des fonctions élémentaires,
  - en produisant :
    - la première partie objet **(2pos)** du logiciel protégé **(2p)**, cette première partie objet **(2pos)** étant telle que lors de l'exécution du logiciel protégé **(2p)**, au moins une portion de la première partie d'exécution **(2pes)** exécute aussi les commandes élémentaires selon l'ordonnancement choisi,
    - et la seconde partie objet **(2pou)** du logiciel protégé **(2p)** contenant aussi les moyens d'exploitation, cette seconde partie objet **(2pou)** étant telle que, après chargement dans l'unité **(6)** et lors de l'exécution du logiciel protégé **(2p)**, apparaît la seconde partie d'exécution **(2peu)** au moyen de laquelle sont aussi exécutées les fonctions élémentaires déclenchées par la première partie d'exécution **(2pes)**,

→ et dans la phase d'utilisation **(U)** :

- en présence de l'unité **(6)** et à chaque fois qu'une commande élémentaire contenue dans une portion de la première partie d'exécution **(2pes)** l'impose, à exécuter la fonction élémentaire correspondante dans l'unité **(6)**,

de sorte que cette portion est exécutée correctement et que, par conséquent, le logiciel protégé **(2p)** est complètement fonctionnel,

- et en l'absence de l'unité **(6)**, malgré la demande d'une portion de la première partie d'exécution **(2pes)**, de déclencher l'exécution d'une fonction élémentaire dans l'unité **(6)**, à ne pas pouvoir répondre correctement à cette demande, de sorte qu'au moins cette portion n'est pas exécutée correctement et que, par conséquent, le logiciel protégé **(2p)** n'est pas complètement fonctionnel.

6 - Procédé selon la revendication 5, caractérisé en ce qu'il consiste :

→ dans la phase de protection **(P)** :

- à définir :
  - au moins une caractéristique d'exécution de logiciel, susceptible d'être surveillée au moins en partie dans l'unité **(6)**,
  - au moins un critère à respecter pour au moins une caractéristique d'exécution de logiciel,
  - des moyens de détection **(17)** à mettre en oeuvre dans l'unité **(6)** et permettant de détecter qu'au moins une caractéristique d'exécution de logiciel ne respecte pas au moins un critère associé,
  - et des moyens de coercition **(18)** à mettre en oeuvre dans l'unité **(6)** et permettant d'informer le système de traitement de données **(3)** et/ou de modifier l'exécution d'un logiciel, lorsqu'au moins un critère n'est pas respecté,
- à construire les moyens d'exploitation permettant à l'unité **(6)**, de mettre aussi en oeuvre les moyens de détection **(17)** et les moyens de coercition **(18)**,
- et à modifier le logiciel protégé **(2p)** :
  - en choisissant au moins une caractéristique d'exécution de logiciel à surveiller, parmi les caractéristiques d'exécution susceptibles d'être surveillées,
  - en choisissant au moins un critère à respecter pour au moins une caractéristique d'exécution de logiciel choisie,

- en choisissant dans le source du logiciel protégé **(2ps)**, des fonctions élémentaires pour lesquelles au moins une caractéristique d'exécution de logiciel choisie, est à surveiller,
- en modifiant au moins une portion choisie du source du logiciel protégé **(2ps)**, cette modification étant telle que lors de l'exécution du logiciel protégé **(2p)**, au moins une caractéristique d'exécution choisie est surveillée au moyen de la seconde partie d'exécution **(2peu)**, et le non respect d'un critère aboutit à une information du système de traitement de données **(3)** et/ou à une modification de l'exécution du logiciel protégé **(2p)**,
- et en produisant la seconde partie objet **(2pou)** du logiciel protégé **(2p)** contenant les moyens d'exploitation mettant aussi en oeuvre les moyens de détection **(17)** et les moyens de coercition **(18)**, cette seconde partie objet **(2pou)** étant telle que, après chargement dans l'unité **(6)** et lors de l'exécution du logiciel protégé **(2p)**, au moins une caractéristique d'exécution de logiciel est surveillée et le non respect d'un critère aboutit à une information du système de traitement de données **(3)** et/ou à une modification de l'exécution du logiciel protégé **(2p)**,

→ et dans la phase d'utilisation **(U)** :

- en présence de l'unité **(6)** :
  - tant que tous les critères correspondant à toutes les caractéristiques d'exécution surveillées de toutes les portions modifiées du logiciel protégé **(2p)** sont respectés, à permettre le fonctionnement nominal de ces portions du logiciel protégé **(2p)** et par conséquent à permettre le fonctionnement nominal du logiciel protégé **(2p)**,
  - et si au moins un des critères correspondant à une caractéristique d'exécution surveillée d'une portion du logiciel protégé **(2p)** n'est pas respecté, à en informer le système de traitement de données **(3)** et/ou à modifier le fonctionnement de la portion du logiciel protégé **(2p)**, de sorte que le fonctionnement du logiciel protégé **(2p)** est modifié.

7 - Procédé selon la revendication 6, pour limiter l'utilisation d'un logiciel protégé **(2p)**, caractérisé en ce qu'il consiste :

→ dans la phase de protection **(P)** :

- à définir :
  - en tant que caractéristique d'exécution de logiciel susceptible d'être surveillée, une variable de mesure de l'utilisation d'une fonctionnalité d'un logiciel,
  - en tant que critère à respecter, au moins un seuil associé à chaque variable de mesure,
  - et des moyens d'actualisation permettant de mettre à jour au moins une variable de mesure,
- à construire les moyens d'exploitation permettant à l'unité **(6)** de mettre aussi en œuvre les moyens d'actualisation,
- et à modifier le logiciel protégé **(2p)** :
  - en choisissant en tant que caractéristique d'exécution de logiciel à surveiller, au moins une variable de mesure de l'utilisation d'une fonctionnalité d'un logiciel,
  - en choisissant :
    - ➢ au moins une fonctionnalité du logiciel protégé **(2p)** dont l'utilisation est susceptible d'être surveillée grâce à une variable de mesure,
    - ➢ au moins une variable de mesure servant à quantifier l'utilisation de ladite fonctionnalité,
    - ➢ au moins un seuil associé à une variable de mesure choisie correspondant à une limite d'utilisation de ladite fonctionnalité,
    - ➢ et au moins une méthode de mise à jour d'une variable de mesure choisie en fonction de l'utilisation de ladite fonctionnalité,
  - et en modifiant au moins une portion choisie du source du logiciel protégé **(2ps)**, cette modification étant telle que, lors de l'exécution du logiciel protégé **(2p)**, la variable de mesure est actualisée au moyen de la seconde partie d'exécution **(2peu)**, en fonction de l'utilisation de

ladite fonctionnalité et au moins un dépassement de seuil est pris en compte,

→ et dans la phase d'utilisation **(U)**, en présence de l'unité **(6)**, et dans le cas où il est détecté au moins un dépassement de seuil correspondant à au moins une limite d'utilisation, à en informer le système de traitement de données **(3)** et/ou à modifier le fonctionnement de la portion du logiciel protégé **(2p)**, de sorte que le fonctionnement du logiciel protégé **(2p)** est modifié.

8 - Procédé selon la revendication 7, caractérisé en ce qu'il consiste :

→ dans la phase de protection **(P)** :

- à définir :
  - pour au moins une variable de mesure, plusieurs seuils associés,
  - et des moyens de coercition différents correspondant à chacun de ces seuils,
- et à modifier le logiciel protégé **(2p)** :
  - en choisissant dans le source du logiciel protégé **(2ps)**, au moins une variable de mesure choisie à laquelle doivent être associés plusieurs seuils correspondants à des limites différentes d'utilisation de la fonctionnalité,
  - en choisissant au moins deux seuils associés à la variable de mesure choisie,
  - et en modifiant au moins une portion choisie du source du logiciel protégé **(2ps)**, cette modification étant telle que, lors de l'exécution du logiciel protégé **(2p)**, les dépassements des divers seuils sont pris en compte, au moyen de la seconde partie d'exécution **(2peu)**, de manière différente,

→ et dans la phase d'utilisation **(U)** :

- en présence de l'unité **(6)** :
  - dans le cas où il est détecté le dépassement d'un premier seuil, à enjoindre le logiciel protégé **(2p)** de ne plus utiliser la fonctionnalité correspondante,
  - et dans le cas où il est détecté le dépassement d'un deuxième seuil, à

rendre inopérante la fonctionnalité correspondante et/ou au moins une portion du logiciel protégé **(2p)**.

9 - Procédé selon la revendication 7 ou 8, caractérisé en ce qu'il consiste :

→ dans la phase de protection **(P)** :

- à définir des moyens de rechargement permettant de créditer au moins une utilisation supplémentaire pour au moins une fonctionnalité de logiciel surveillée par une variable de mesure,
- à construire les moyens d'exploitation permettant aussi à l'unité **(6)** de mettre en œuvre les moyens de rechargement,
- et à modifier le logiciel protégé **(2p)** :
  - en choisissant dans le source du logiciel protégé **(2ps)**, au moins une variable de mesure choisie permettant de limiter l'utilisation d'une fonctionnalité à laquelle au moins une utilisation supplémentaire doit pouvoir être créditée,
  - et en modifiant au moins une portion choisie, cette modification étant telle que dans une phase dite de rechargement, au moins une utilisation supplémentaire d'au moins une fonctionnalité correspondant à une variable de mesure choisie peut être créditée,

→ et dans la phase de rechargement :

- à réactualiser au moins une variable de mesure choisie et/ou au moins un seuil associé, de manière à permettre au moins une utilisation supplémentaire de la fonctionnalité.

10 - Procédé selon la revendication 6, caractérisé en ce qu'il consiste :

→ dans la phase de protection **(P)** :

- à définir :
  - en tant que caractéristique d'exécution de logiciel susceptible d'être surveillée, un profil d'utilisation de logiciel,
  - et en tant que critère à respecter, au moins un trait d'exécution de logiciel,
- et à modifier le logiciel protégé **(2p)** :
  - en choisissant en tant que caractéristique d'exécution de logiciel à

surveiller au moins un profil d'utilisation de logiciel,

- en choisissant au moins un trait d'exécution qu'au moins un profil d'utilisation choisi doit respecter,
- et en modifiant au moins une portion choisie du source du logiciel protégé **(2ps)**, cette modification étant telle que, lors de l'exécution du logiciel protégé **(2p)**, la seconde partie d'exécution **(2peu)** respecte tous les traits d'exécution choisis,

→ et dans la phase d'utilisation **(U)** en présence de l'unité **(6)**, et dans le cas où il est détecté qu'au moins un trait d'exécution n'est pas respecté, à en informer le système de traitement de données **(3)** et/ou à modifier le fonctionnement de la portion du logiciel protégé **(2p)**, de sorte que le fonctionnement du logiciel protégé **(2p)** est modifié.

11 - Procédé selon la revendication 10, caractérisé en ce qu'il consiste :

→ dans la phase de protection **(P)** :

- à définir :
  - un jeu d'instructions dont les instructions sont susceptibles d'être exécutées dans l'unité **(6)**,
  - un jeu de commandes d'instructions pour ce jeu d'instructions, ces commandes d'instructions étant susceptibles d'être exécutées dans le système de traitement de données **(3)** et de déclencher dans l'unité **(6)** l'exécution des instructions,
  - en tant que profil d'utilisation, l'enchaînement des instructions,
  - en tant que trait d'exécution, un enchaînement souhaité pour l'exécution des instructions,
  - en tant que moyens de détection **(17)**, des moyens permettant de détecter que l'enchaînement des instructions ne correspond pas à celui souhaité,
  - et en tant que moyens de coercition **(18)**, des moyens permettant d'informer le système de traitement de données **(3)** et/ou de modifier le fonctionnement de la portion de logiciel protégé **(2p)** lorsque l'enchaînement des instructions ne correspond pas à celui souhaité,

- à construire les moyens d'exploitation permettant aussi à l'unité **(6)** d'exécuter les instructions du jeu d'instructions, l'exécution de ces instructions étant déclenchée par l'exécution dans le système de traitement de données **(3)**, des commandes d'instructions,
- et à modifier le logiciel protégé **(2p)** :
  - en modifiant au moins une portion choisie du source du logiciel protégé **(2ps)** :
    - ➢ en transformant les fonctions élémentaires en instructions,
    - ➢ en spécifiant l'enchaînement que doivent respecter au moins certaines des instructions lors de leur exécution dans l'unité **(6)**,
    - ➢ et en transformant les commandes élémentaires en commandes d'instructions correspondant aux instructions utilisées,

→ et dans la phase d'utilisation **(U)**, en présence de l'unité **(6)**, dans le cas où il est détecté que l'enchaînement des instructions exécutées dans l'unité **(6)** ne correspond pas à celui souhaité, à en informer le système de traitement de données **(3)** et/ou à modifier le fonctionnement de la portion du logiciel protégé **(2p)**, de sorte que le fonctionnement du logiciel protégé **(2p)** est modifié.

12 - Procédé selon la revendication 11, caractérisé en ce qu'il consiste :

→ dans la phase de protection **(P)** :

- à définir :
  - en tant que jeu d'instructions, un jeu d'instructions dont au moins certaines instructions travaillent sur des registres et utilisent au moins un opérande en vue de rendre un résultat,
  - pour au moins une partie des instructions travaillant sur des registres :
    - ➢ une partie **(PF)** définissant la fonctionnalité de l'instruction,
    - ➢ et une partie définissant l'enchaînement souhaité pour l'exécution des instructions et comportant des champs de bits correspondant à :
      - ◊ un champ d'identification de l'instruction **(CII)**,
      - ◊ et pour chaque opérande de l'instruction :
        - * un champ drapeau **(CD_k)**,

* et un champ d'identification prévue **(CIP_k)** de l'opérande,

- pour chaque registre appartenant aux moyens d'exploitation et utilisé par le jeu d'instructions, un champ d'identification générée **(CIG_v)** dans lequel est automatiquement mémorisée l'identification de la dernière instruction ayant rendu son résultat dans ce registre,
- en tant que moyens de détection **(17)**, des moyens permettant, lors de l'exécution d'une instruction, pour chaque opérande, lorsque le champ drapeau **(CD_k)** l'impose, de contrôler l'égalité entre le champ d'identification générée **(CIG_v)** correspondant au registre utilisé par cet opérande, et le champ d'identification prévue **(CIP_k)** de l'origine de cet opérande,
- et en tant que moyens de coercition **(18)**, des moyens permettant de modifier le résultat des instructions, si au moins une des égalités contrôlées est fausse.

13 - Procédé selon l'une des revendications 1 à 12, caractérisé en ce qu'il consiste :

→ dans la phase de protection **(P)** :

- à modifier le logiciel protégé **(2p)** :
  - en choisissant dans le source du logiciel protégé **(2ps)**, au moins un branchement conditionnel effectué dans au moins un traitement algorithmique choisi,
  - en modifiant au moins une portion choisie du source du logiciel protégé **(2ps)**, cette modification étant telle que lors de l'exécution du logiciel protégé **(2p)**, la fonctionnalité d'au moins un branchement conditionnel choisi, est exécutée, au moyen de la seconde partie d'exécution **(2peu)**, dans l'unité **(6)**,
  - et en produisant :
    - ➢ la première partie objet **(2pos)** du logiciel protégé **(2p)**, cette première partie objet **(2pos)** étant telle que lors de l'exécution du logiciel protégé **(2p)**, la fonctionnalité d'au moins un branchement

conditionnel choisi est exécutée dans l'unité **(6)**,

- ➢ et la seconde partie objet **(2pou)** du logiciel protégé **(2p)**, cette seconde partie objet **(2pou)** étant telle que, après chargement dans l'unité **(6)** et lors de l'exécution du logiciel protégé **(2p)**, apparaît la seconde partie d'exécution **(2peu)** au moyen de laquelle la fonctionnalité d'au moins un branchement conditionnel choisi est exécutée,

→ et dans la phase d'utilisation **(U)** :

- en présence de l'unité **(6)** et à chaque fois qu'une portion de la première partie d'exécution **(2pes)** l'impose, à exécuter la fonctionnalité d'au moins un branchement conditionnel dans l'unité **(6)**, de sorte que cette portion est exécutée correctement et que, par conséquent, le logiciel protégé **(2p)** est complètement fonctionnel,
- et en l'absence de l'unité **(6)** et malgré la demande d'une portion de la première partie d'exécution **(2pes)** d'exécuter la fonctionnalité d'un branchement conditionnel dans l'unité **(6)**, à ne pas pouvoir répondre correctement à cette demande, de sorte qu'au moins cette portion n'est pas exécutée correctement et que par conséquent, le logiciel protégé **(2p)** n'est pas complètement fonctionnel.

14 - Procédé selon la revendication 13, caractérisé en ce qu'il consiste, dans la phase de protection **(P)**, à modifier le logiciel protégé **(2p)** :

- en choisissant, dans le source du logiciel protégé **(2ps)** au moins une série de branchements conditionnels choisis,
- en modifiant au moins une portion choisie du source du logiciel protégé **(2ps)**, cette modification étant telle que lors de l'exécution du logiciel protégé **(2p)**, la fonctionnalité globale d'au moins une série choisie de branchements conditionnels est exécutée au moyen de la seconde partie d'exécution **(2peu)**, dans l'unité **(6)**,
- et en produisant :
  - ➢ la première partie objet **(2pos)** du logiciel protégé **(2p)**, cette première partie objet **(2pos)** étant telle que lors de l'exécution du

logiciel protégé **(2p)**, la fonctionnalité d'au moins une série choisie de branchements conditionnels est exécutée dans l'unité **(6)**,

- et la seconde partie objet **(2pou)** du logiciel protégé **(2p)**, cette seconde partie objet **(2pou)** étant telle que, après chargement dans l'unité **(6)** et lors de l'exécution du logiciel protégé **(2p)**, apparaît la seconde partie d'exécution **(2peu)** au moyen de laquelle la fonctionnalité globale d'au moins une série choisie de branchements conditionnels est exécutée.

15 - Procédé selon l'une des revendications 1 à 14, caractérisé en ce qu'il consiste à décomposer la phase de protection **(P)** en une sous-phase de protection amont **(P1)**, indépendante du logiciel à protéger et une sous-phase de protection aval **(P2)**, dépendante du logiciel à protéger.

16 - Procédé selon la revendication 15, caractérisé en ce qu'il consiste, lors de la sous-phase de protection amont **(P1)**, à faire intervenir un stade de définitions **(S11)** dans lequel sont effectuées toutes les définitions.

17- Procédé selon la revendication 16, caractérisé en ce qu'il consiste, après le stade de définitions **(S11)**, à faire intervenir un stade de construction **(S12)** dans lequel sont construits les moyens d'exploitation.

18 - Procédé selon la revendication 17, caractérisé en ce qu'il consiste, après le stade de construction **(S12)**, à faire intervenir un stade de pré-personnalisation **(S13)**, consistant à charger dans une unité vierge **(60)**, au moins une partie des moyens d'exploitation en vue d'obtenir une unité pré-personnalisée **(66)**.

19 - Procédé selon la revendication 16 ou 17, caractérisé en ce qu'il consiste, lors de la sous-phase de protection amont **(P1)**, à faire intervenir un stade de confection d'outils **(S14)** dans lequel sont confectionnés des outils permettant d'aider à générer des logiciels protégés ou d'automatiser la protection de logiciels.

20 - Procédé selon les revendications 15 et 18, caractérisé en ce qu'il consiste à décomposer la sous-phase de protection aval **(P2)**, en :

- un stade de création **(S21)** dans lequel est créé le logiciel protégé **(2p)**, à partir du logiciel vulnérable **(2v)**,
- éventuellement, un stade de modification **(S22)** dans lequel est modifié le

logiciel protégé **(2p)**,

- et un stade de personnalisation **(S23)** dans lequel :
  - la seconde partie objet **(2pou)** du logiciel protégé **(2p)** contenant les moyens d'exploitation est chargée dans au moins une unité vierge **(60)** en vue d'obtenir au moins une unité **(6)**,
  - ou une partie de la seconde partie objet **(2pou)** du logiciel protégé **(2p)** contenant éventuellement les moyens d'exploitation est chargée dans au moins une unité pré-personnalisée **(66)** en vue d'obtenir au moins une unité **(6)**.

21 - Procédé selon les revendications 19 et 20, caractérisé en ce qu'il consiste, lors du stade de création **(S21)** et éventuellement du stade de modification **(S22)**, à utiliser au moins l'un des outils d'aide à la génération de logiciels protégés ou d'automatisation de la protection de logiciels.

22 - Système pour la mise en oeuvre du procédé conforme à la revendication 17, caractérisé en ce qu'il comporte une unité de développement de programmes, servant, lors du stade de construction **(S12)**, à effectuer la construction des moyens d'exploitation destinés à l'unité **(6)**, prenant en compte les définitions intervenues au stade de définitions **(S11)**.

23 - Système pour la mise en oeuvre du procédé conforme à la revendication 18, caractérisé en ce qu'il comporte une unité de pré-personnalisation **(30)** permettant de charger au moins une partie des moyens d'exploitation dans au moins une unité vierge **(60)**, en vue d'obtenir au moins une unité pré-personnalisée **(66)**.

24 - Système pour la mise en oeuvre du procédé conforme à la revendication 19, caractérisé en ce qu'il comporte une unité de développement de programmes, servant à effectuer lors du stade de confection d'outils **(S14)**, la confection d'outils d'aide à la génération de logiciels protégés ou d'automatisation de la protection de logiciels.

25 - Système pour la mise en oeuvre du procédé conforme à la revendication 20 ou 21, caractérisé en ce qu'il comporte une unité de développement de programmes servant à créer ou à modifier un logiciel protégé **(2p)**.

26 - Système pour la mise en oeuvre du procédé conforme à la revendication 20, caractérisé en ce qu'il comporte une unité de personnalisation **(45)** permettant de charger :

- la seconde partie objet **(2pou)** dans au moins une unité vierge **(60)**, en vue d'obtenir au moins une unité **(6)**,
- ou une partie de la seconde partie objet **(2pou)** dans au moins une unité pré-personnalisée **(66)**, en vue d'obtenir au moins une unité **(6)**.

27 - Unité pré-personnalisée **(66)**, caractérisée en ce qu'elle est obtenue par le système conforme à la revendication 23.

28 - Unité **(6)** permettant d'exécuter un logiciel protégé **(2p)** et d'empêcher son utilisation non autorisée, caractérisée en ce qu'elle contient la seconde partie objet **(2pou)** du logiciel protégé **(2p)** chargée à l'aide d'une unité de personnalisation **(45)** conforme à la revendication 26.

29 - Ensemble d'unités **(6)**, caractérisé en ce que la seconde partie objet **(2pou)** du logiciel protégé **(2p)**, chargée à l'aide d'une unité de personnalisation **(45)** conforme à la revendication 26, est répartie sur plusieurs unités de traitement et de mémorisation de manière que leur utilisation conjointe permette d'exécuter le logiciel protégé **(2p)**.

30 - Ensemble de distribution **(2pd)** d'un logiciel protégé **(2p)**, caractérisé en ce qu'il comporte :

- une première partie de distribution **(2pds)** contenant la première partie objet **(2pos)** et destinée à fonctionner dans un système de traitement de données **(3)**,
- et une seconde partie de distribution **(2pdu)** se présentant sous la forme :
  - d'une unité vierge **(60)**,
  - ou d'une unité pré-personnalisée **(66)** conforme à la revendication 27, capable après chargement d'informations de personnalisation, de se transformer en une unité **(6)**,
  - ou d'une unité **(6)** conforme à la revendication 28.

31 - Ensemble de distribution **(2pd)** d'un logiciel protégé **(2p)** selon la revendication 30, caractérisé en ce que la première partie de distribution **(2pds)** se présente sous la forme d'un moyen de distribution physique, CDROM par exemple, ou sous la forme de fichiers distribués au travers d'un réseau.

32 - Ensemble de distribution **(2pd)** d'un logiciel protégé **(2p)** selon la

revendication 30, caractérisé en ce que la seconde partie de distribution **(2pdu)**, se présentant sous la forme d'unités vierges **(60)**, d'unités pré-personnalisées **(66)** ou d'unités **(6)**, comporte au moins une carte à puce **(7)**.

33 - Unité de traitement et de mémorisation caractérisée en ce qu'elle contient la partie de la seconde partie objet **(2pou)** nécessaire pour transformer une unité pré-personnalisée **(66)** conforme à la revendication 27 en une unité **(6)** conforme à la revendication 28.

34 - Ensemble d'unités de traitement et de mémorisation caractérisé en ce que les unités de traitement et de mémorisation utilisées conjointement, contiennent la partie de la seconde partie objet **(2pou)** nécessaire pour transformer une unité pré-personnalisée **(66)** conforme à la revendication 27 en une unité **(6)** conforme à la revendication 28.

FIG. 10

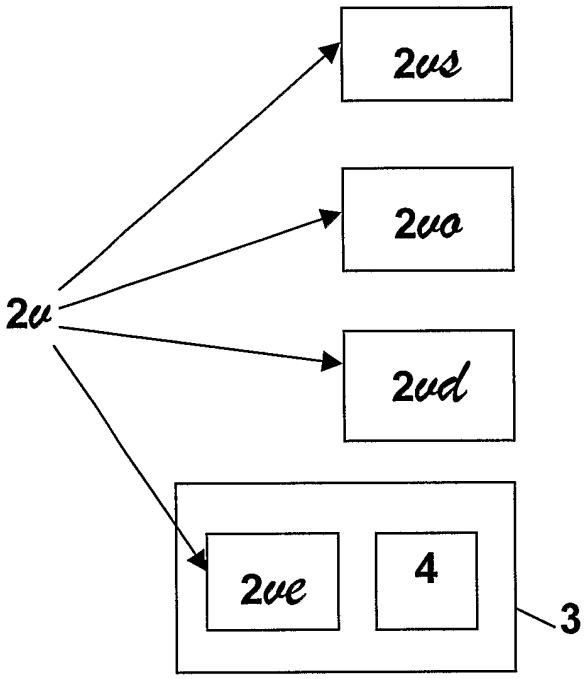

2vs
2vo
2v
2vd
2ve
4
3

FIG. 11

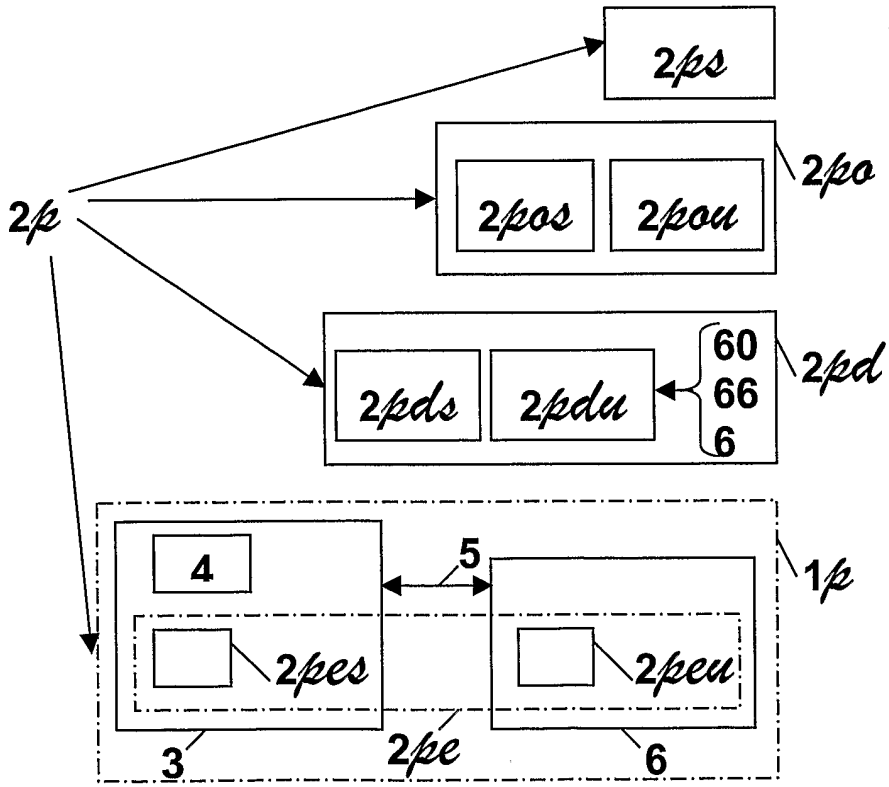

2ps
2p
2pos
2pou
2po
2pds
2pdu
60
66
6
2pd
4
5
1p
2pes
2peu
3
2pe
6

FIG. 20

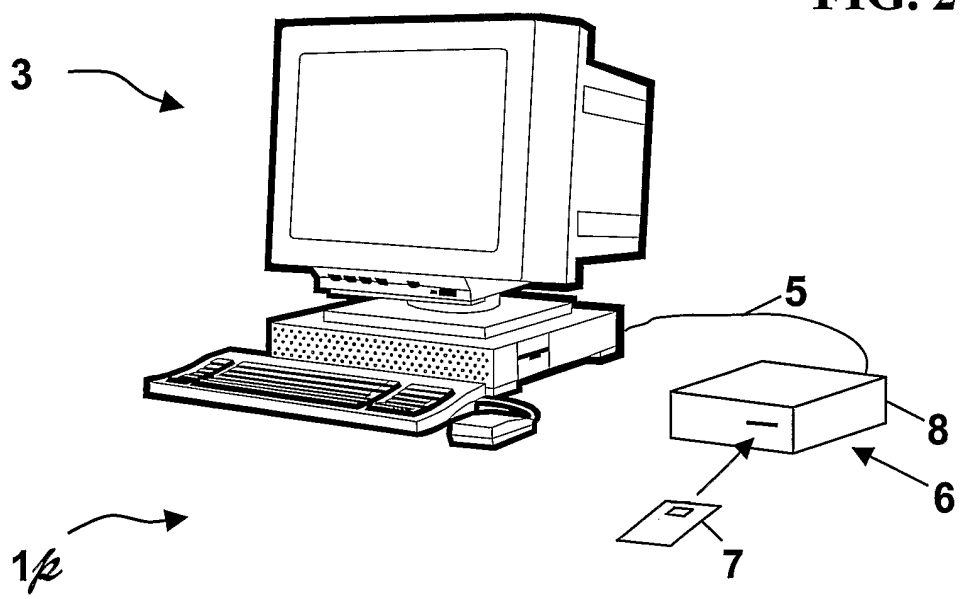


FIG. 21

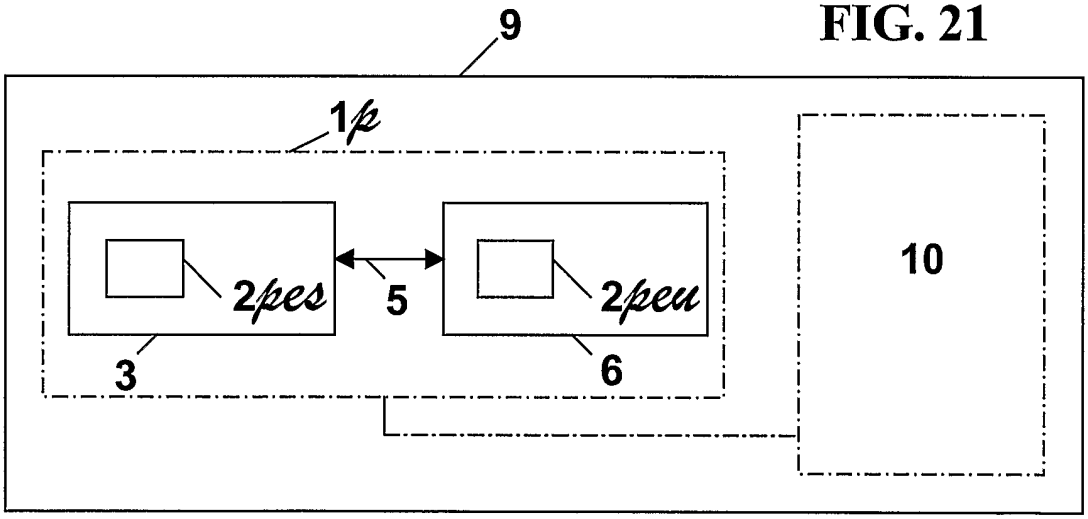


FIG. 22

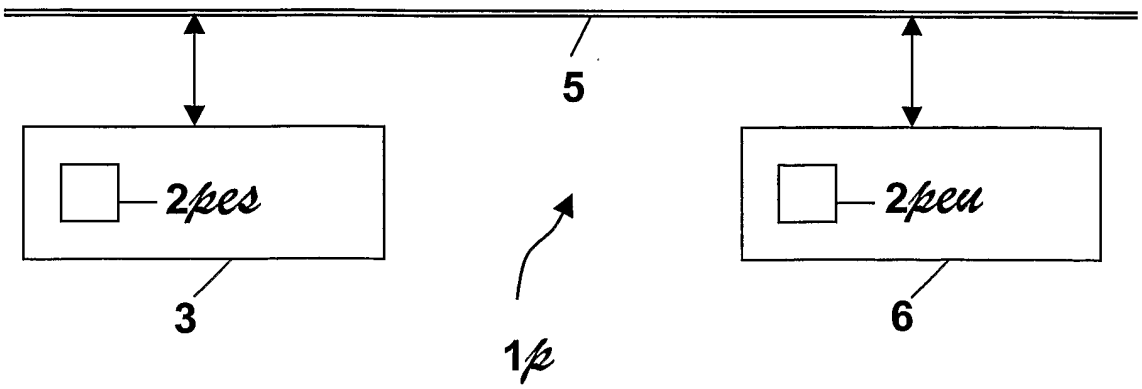


FIG. 30

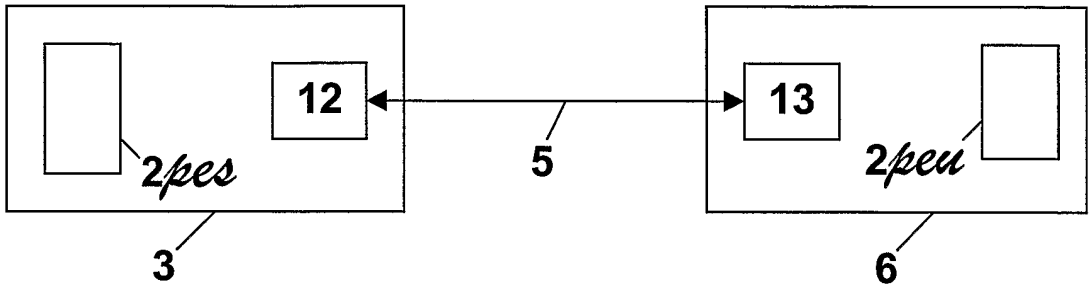


FIG. 31

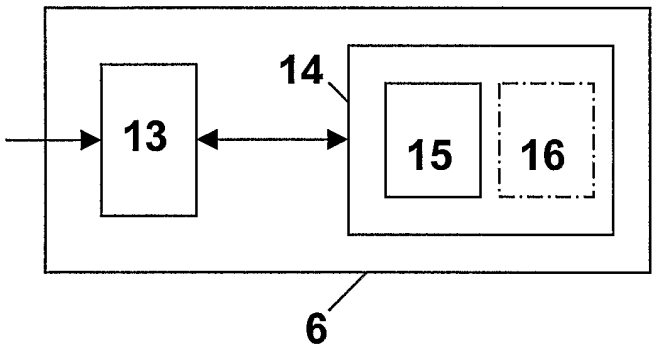


FIG. 40

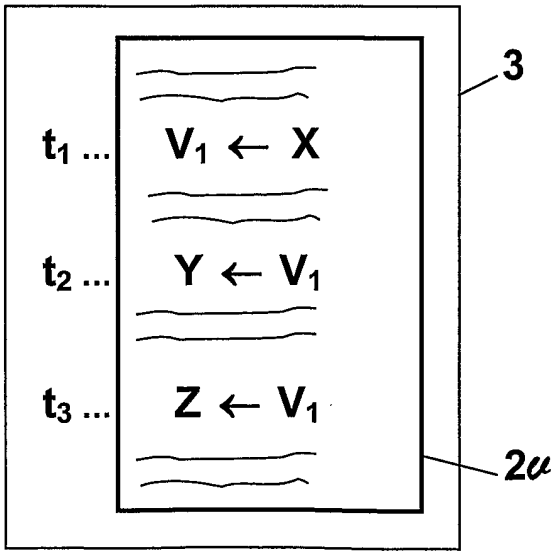


FIG. 41

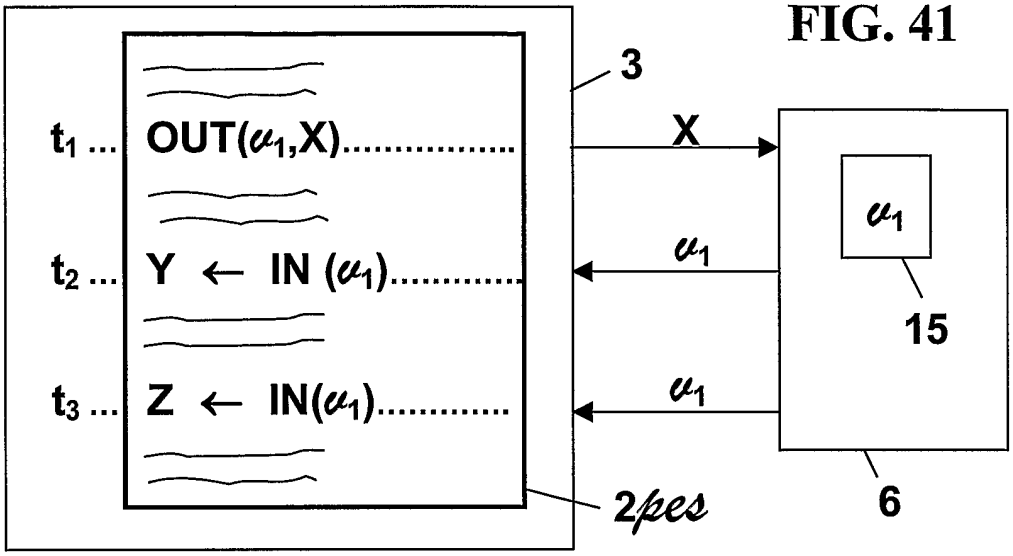


FIG. 42

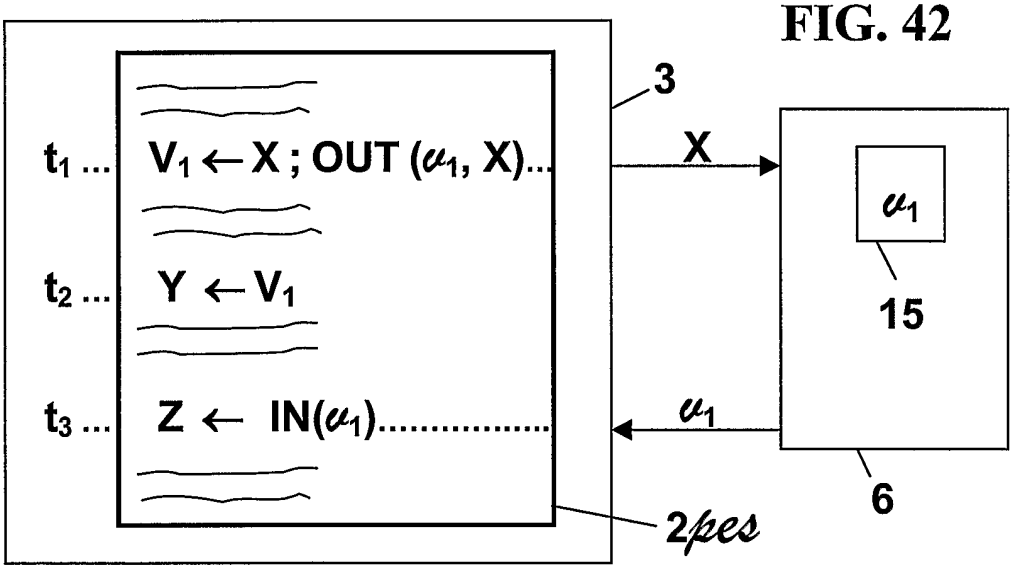


FIG. 43

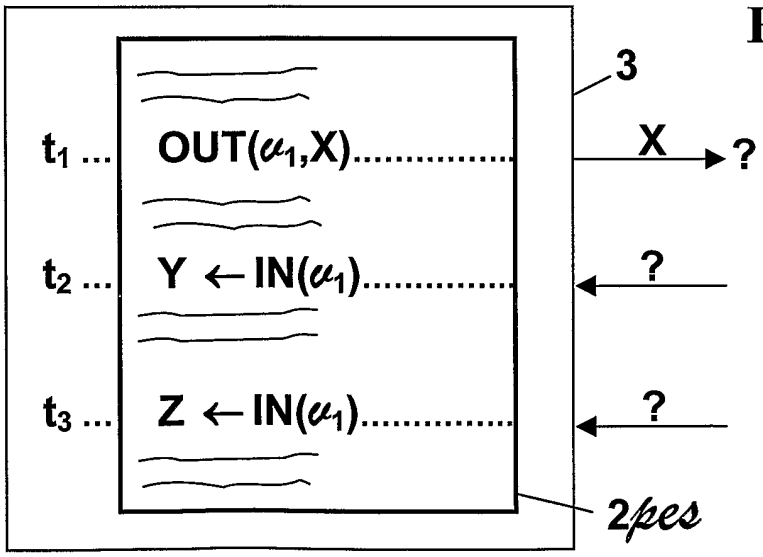


FIG. 60

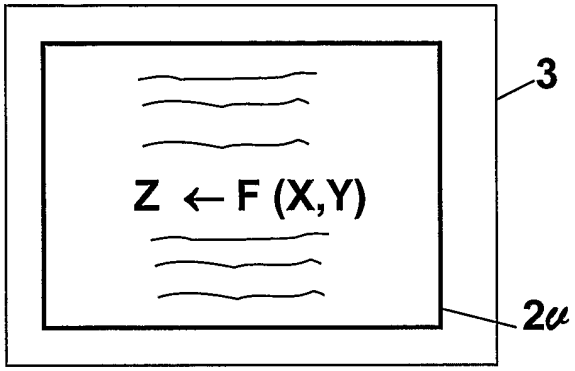


FIG. 61

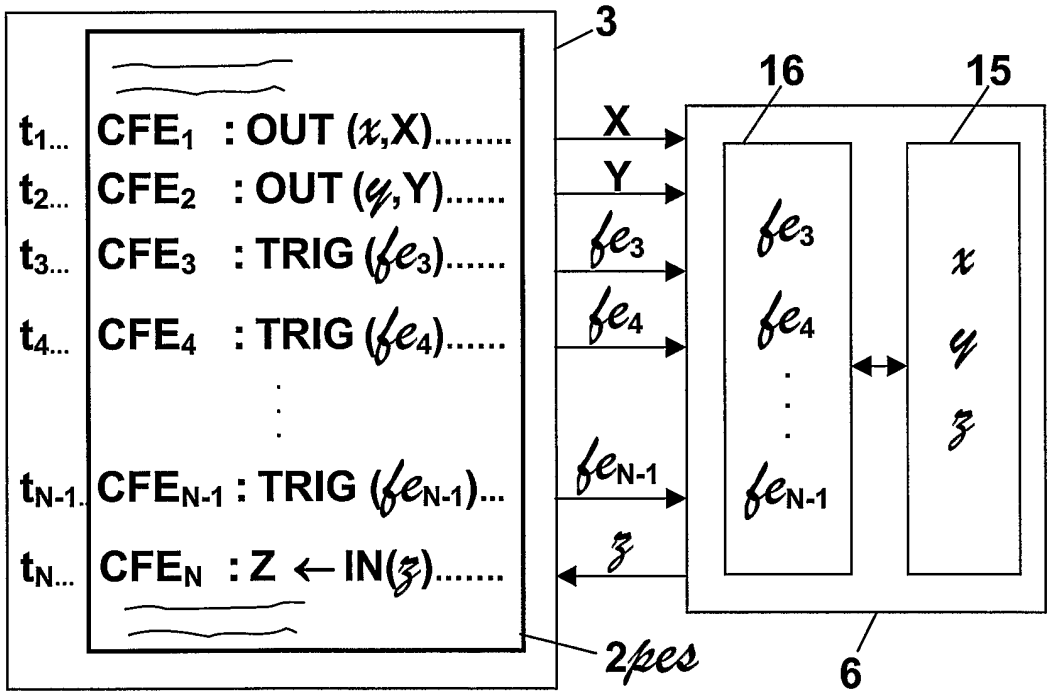


FIG. 62

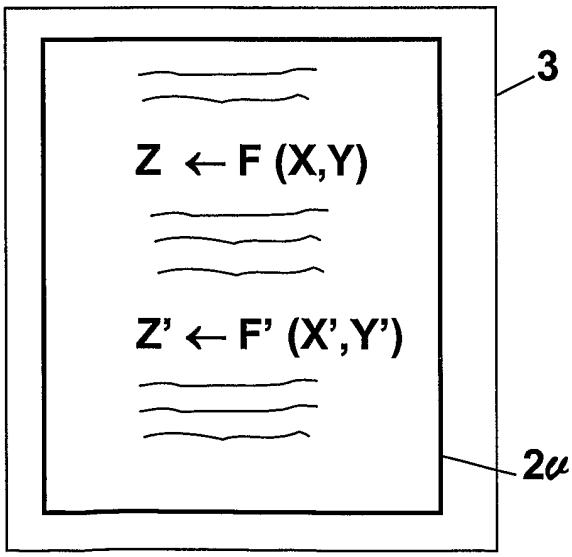


FIG. 63

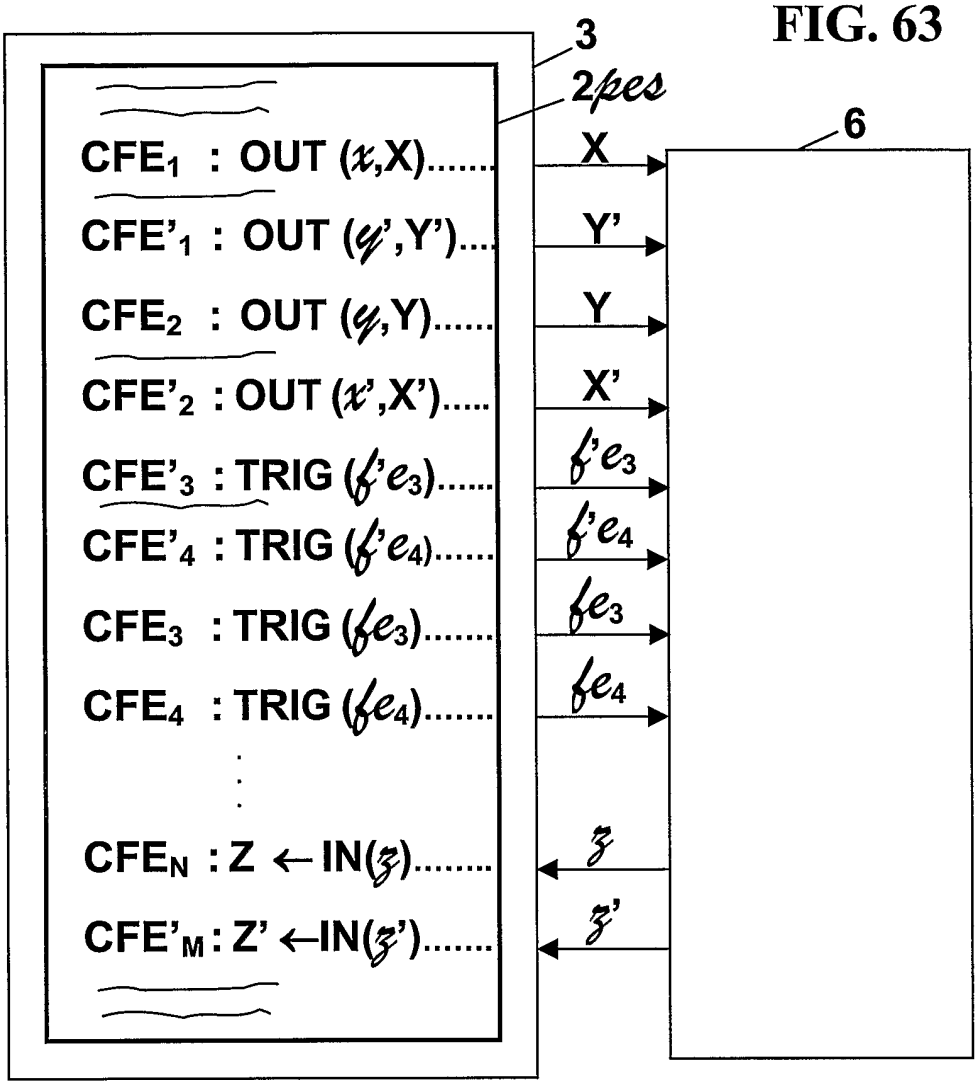

3
2pes
6
CFE1 : OUT (x,X).....
CFE'1 : OUT (y',Y')....
CFE2 : OUT (y,Y).....
CFE'2 : OUT (x',X').....
CFE'3 : TRIG (fe'3).....
CFE'4 : TRIG (fe'4).....
CFE3 : TRIG (fe3).....
CFE4 : TRIG (fe4).....
CFEN : Z ← IN(z).....
CFE'M : Z' ←IN(z').....
X
Y'
Y
X'
fe'3
fe'4
fe3
fe4
z
z'

FIG. 64

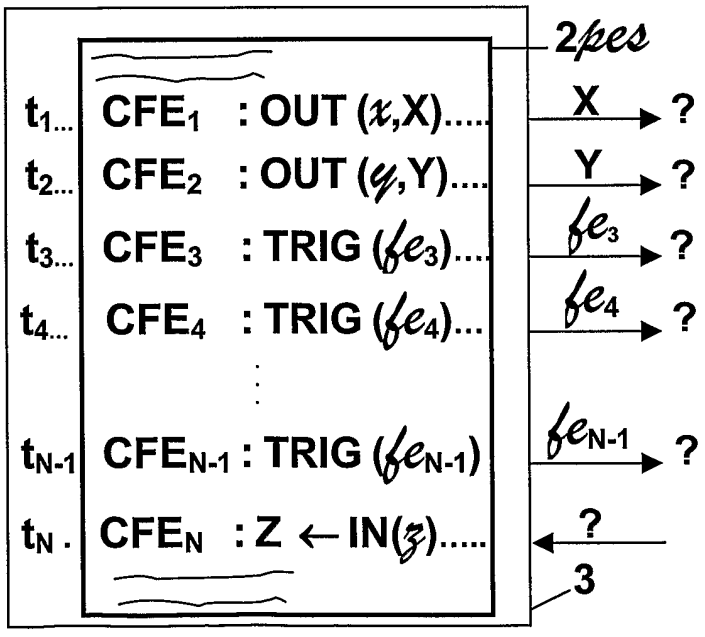

2pes
t1...
t2...
t3...
t4...
tN-1
tN .
CFE1 : OUT (x,X).....
CFE2 : OUT (y,Y)....
CFE3 : TRIG (fe3)....
CFE4 : TRIG (fe4)...
CFEN-1 : TRIG (feN-1)
CFEN : Z ← IN(z).....
X
Y
fe3
fe4
feN-1
?
3

FIG. 70

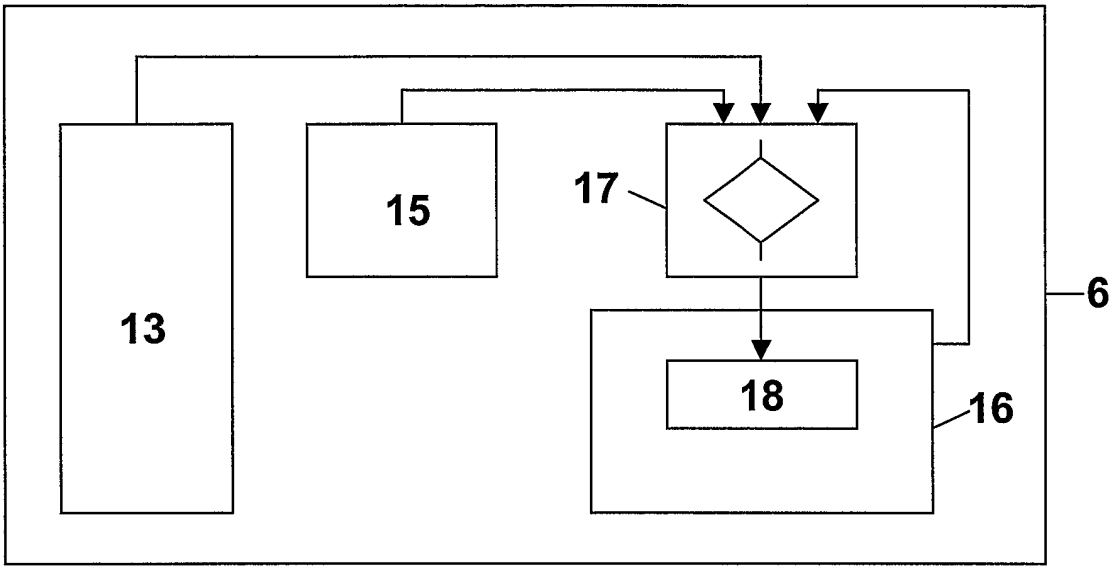

13
15
17
18
16
6

FIG. 71

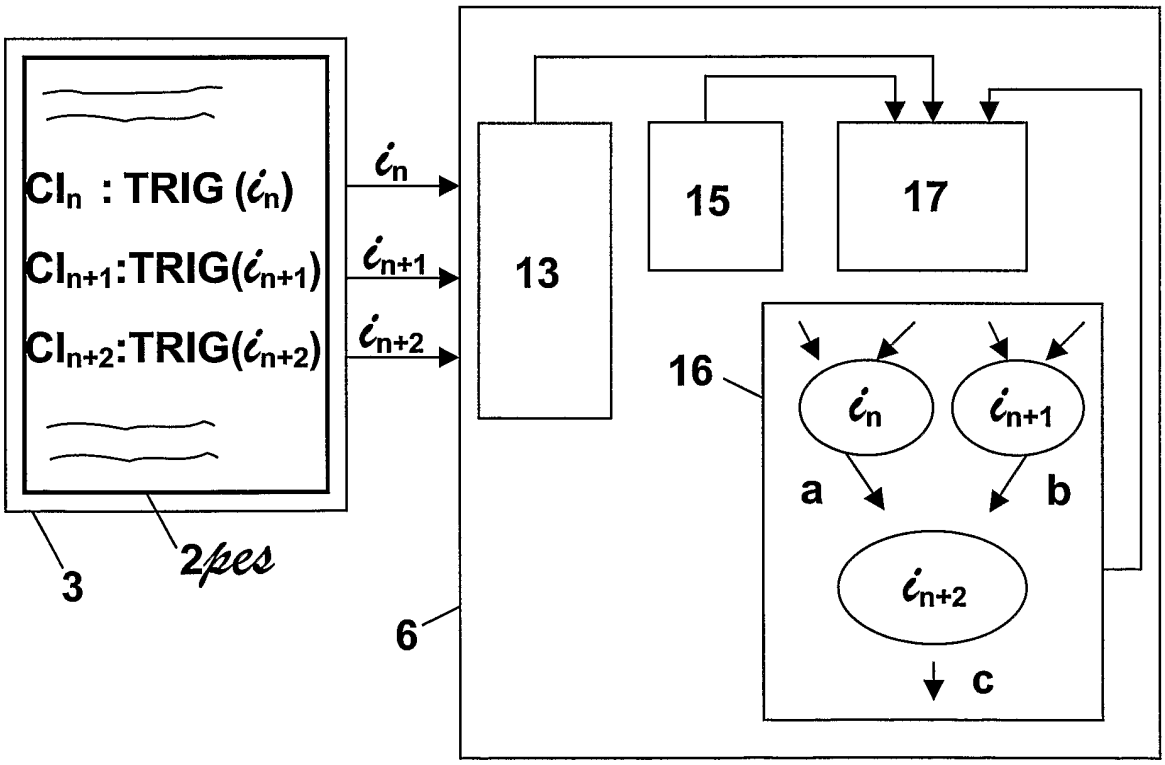

CIn : TRIG (in)
CIn+1:TRIG(in+1)
CIn+2:TRIG(in+2)
in
in+1
in+2
3
2pes
13
15
17
16
in
in+1
a
b
in+2
c
6

FIG. 72

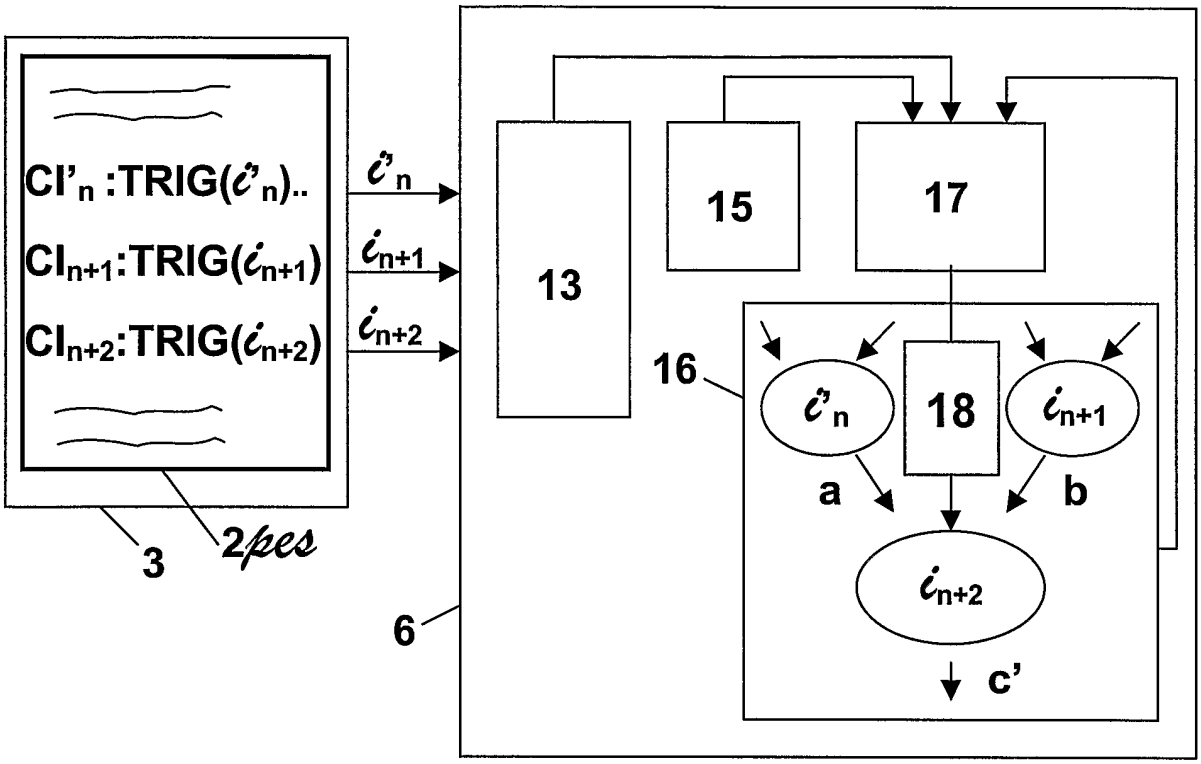


FIG. 73

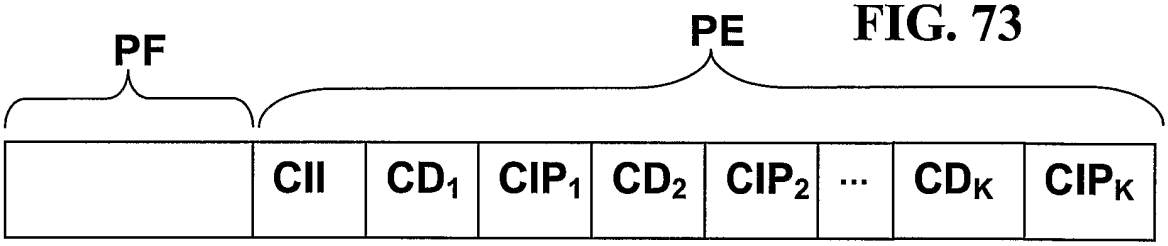


FIG. 74

FIG. 80

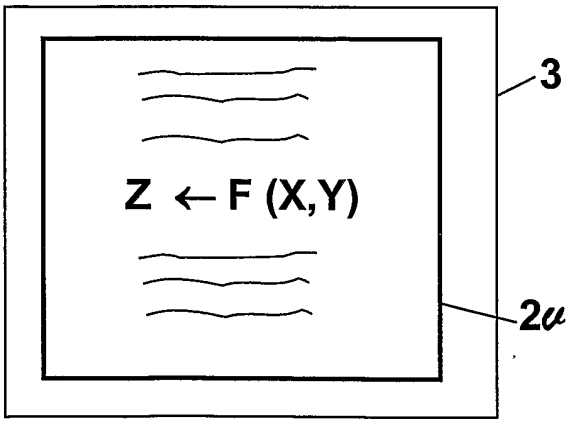


FIG. 81

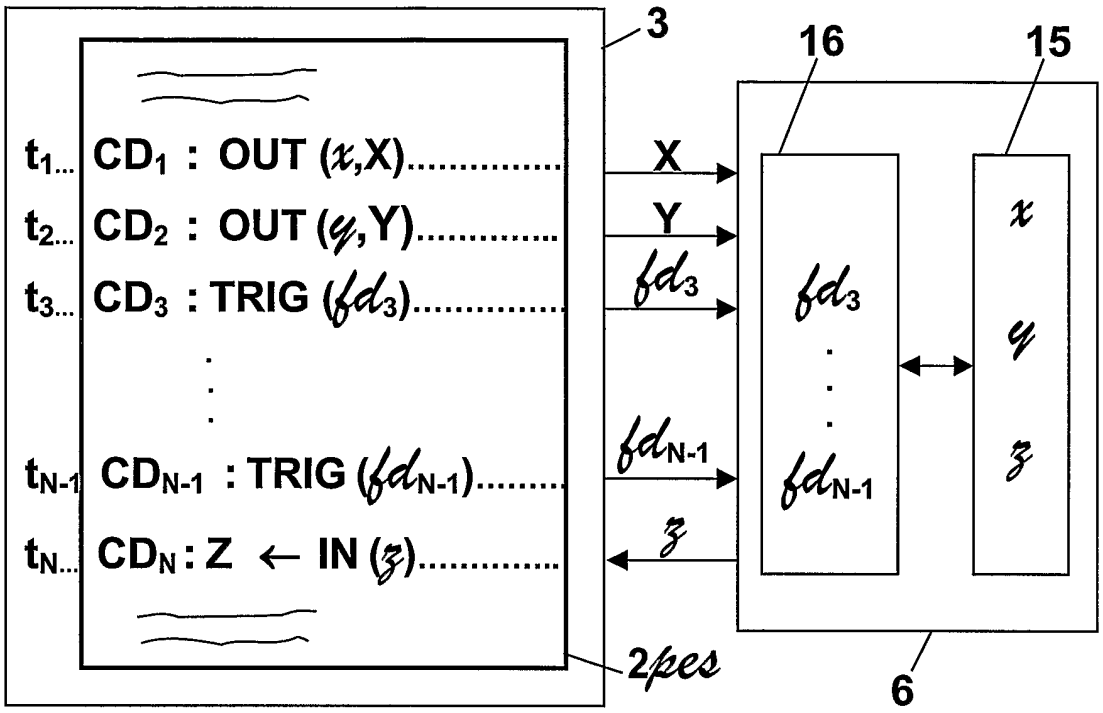
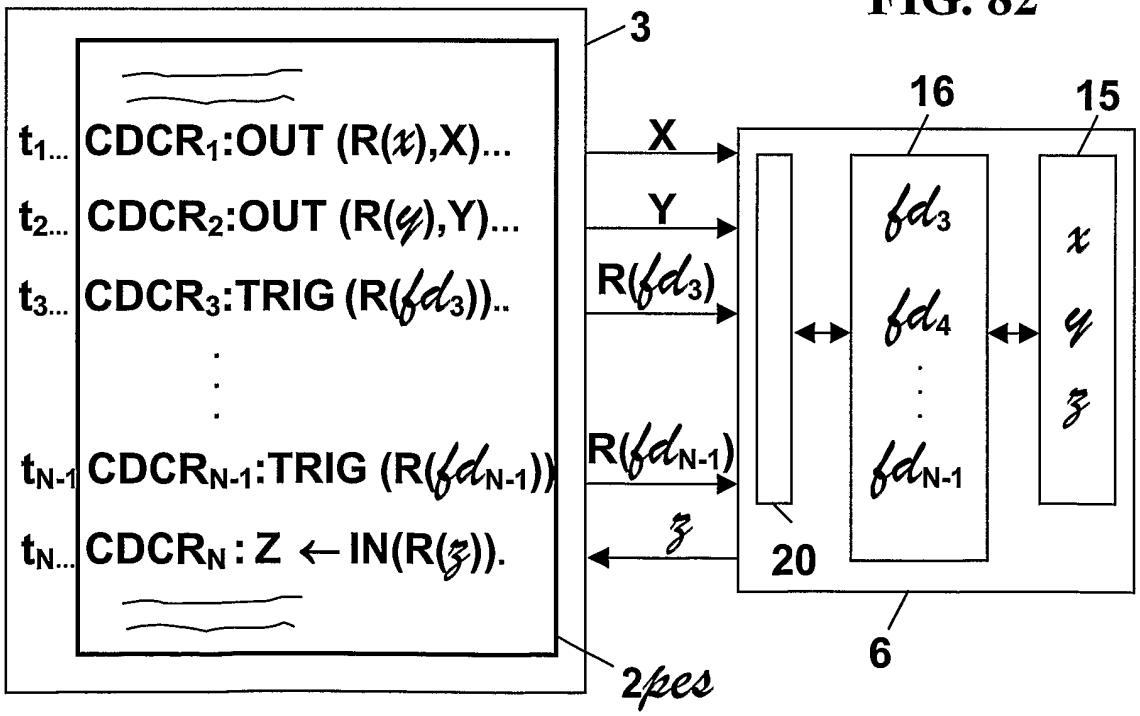

3
16
15
t1... CD1 : OUT (x,X).....
t2... CD2 : OUT (y,Y).....
t3... CD3 : TRIG (fd3).....
tN-1 CDN-1 : TRIG (fdN-1).....
tN... CDN : Z ← IN (z).....
X
Y
fd3
fdN-1
z
fd3
fdN-1
x
y
z
2pes
6

FIG. 82


3
16
15
t1... CDCR1:OUT (R(x),X)...
t2... CDCR2:OUT (R(y),Y)...
t3... CDCR3:TRIG (R(fd3))..
tN-1 CDCRN-1:TRIG (R(fdN-1))
tN... CDCRN : Z ← IN(R(z)).
X
Y
R(fd3)
R(fdN-1)
z
fd3
fd4
fdN-1
x
y
z
20
2pes
6

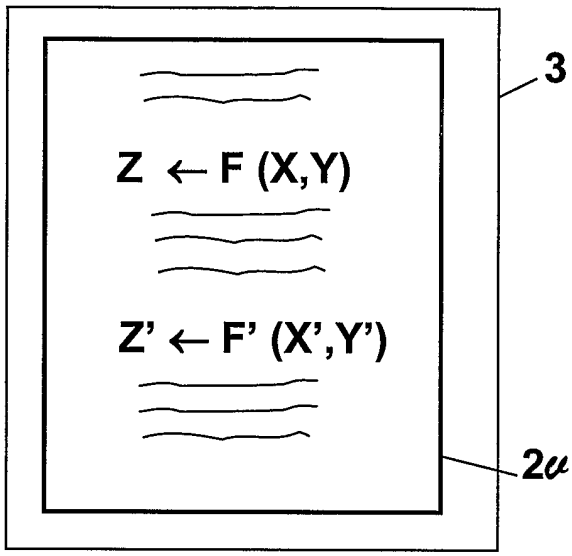

Z ← F (X,Y)
Z' ← F' (X',Y')
3
2u

FIG. 83

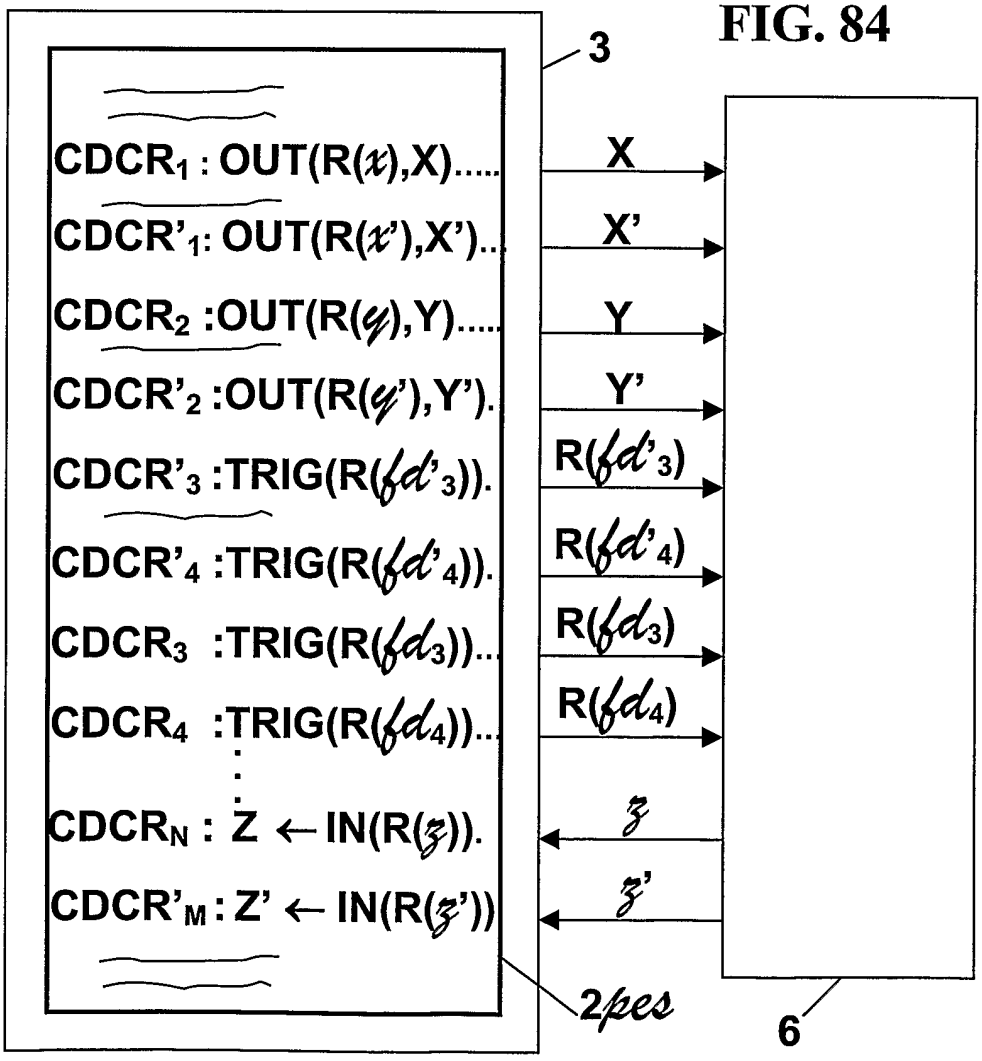

CDCR1 : OUT(R(x),X).....
CDCR'1: OUT(R(x'),X')..
CDCR2 :OUT(R(y),Y).....
CDCR'2 :OUT(R(y'),Y').
CDCR'3 :TRIG(R(fd'3)).
CDCR'4 :TRIG(R(fd'4)).
CDCR3 :TRIG(R(fd3))..
CDCR4 :TRIG(R(fd4))..
CDCRN : Z ← IN(R(z)).
CDCR'M : Z' ← IN(R(z'))
3
X
X'
Y
Y'
R(fd'3)
R(fd'4)
R(fd3)
R(fd4)
z
z'
2pes
6

FIG. 84

FIG. 85

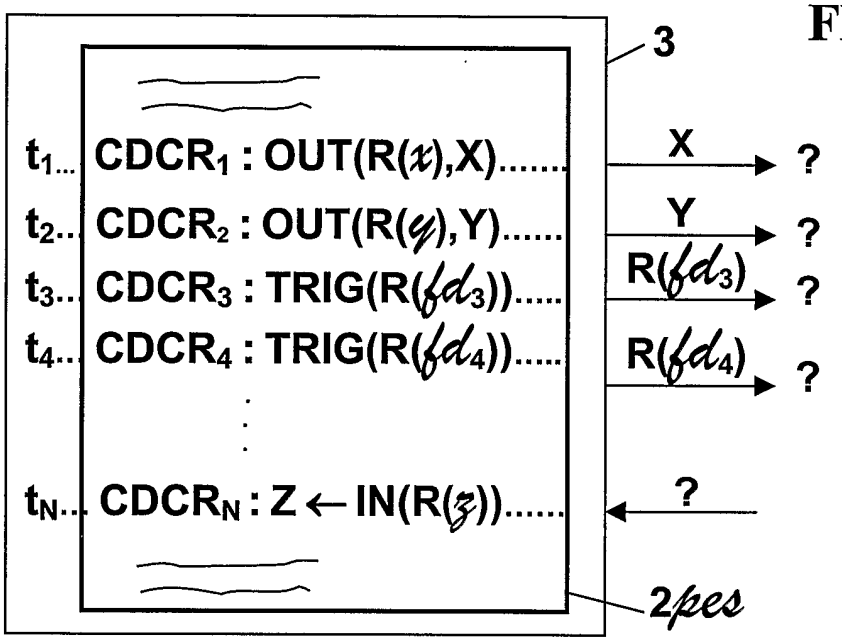


FIG. 90

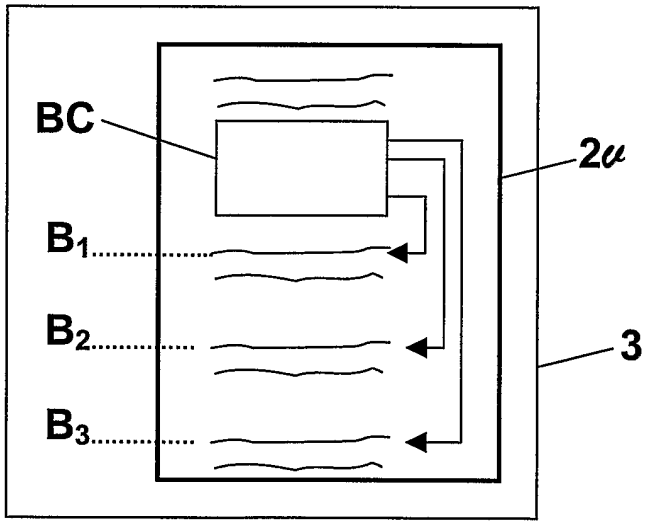


FIG. 91

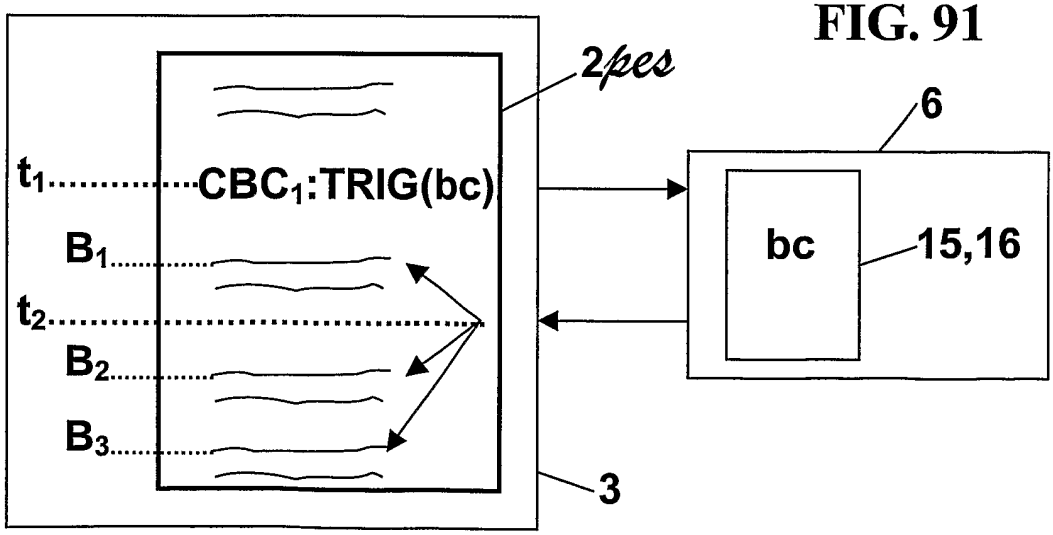


FIG. 92

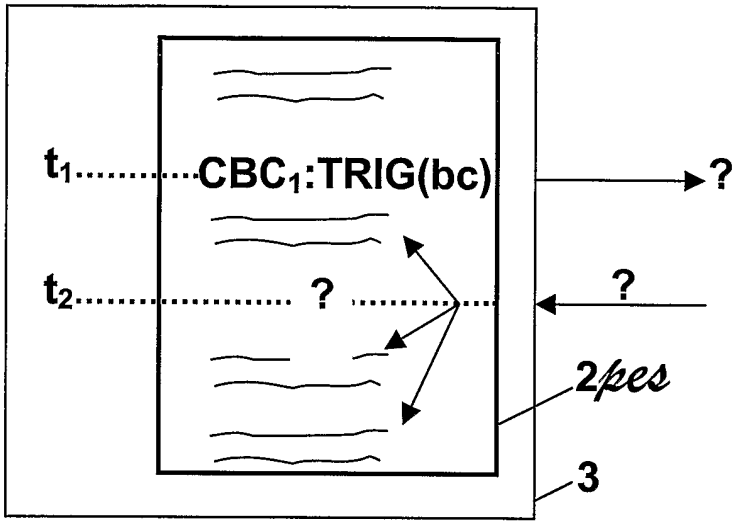


FIG. 100

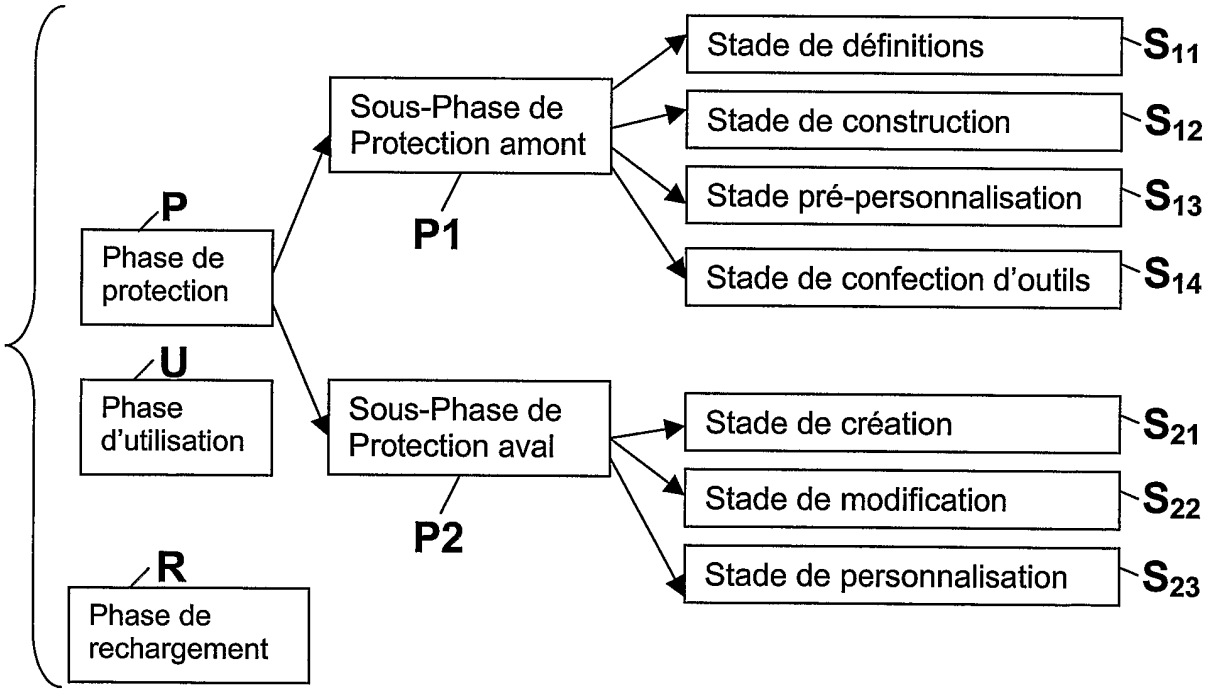
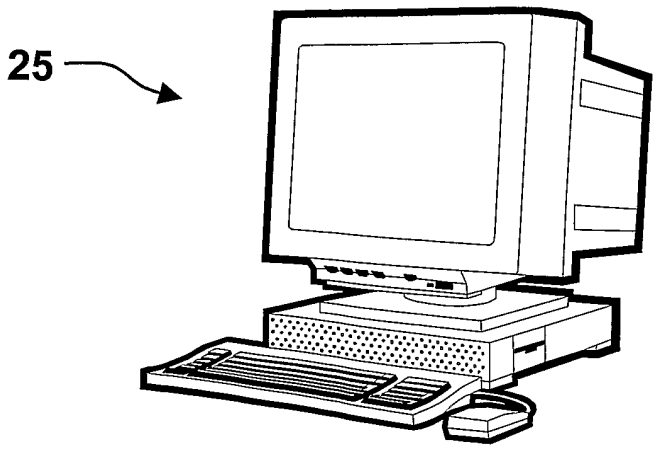


FIG. 110



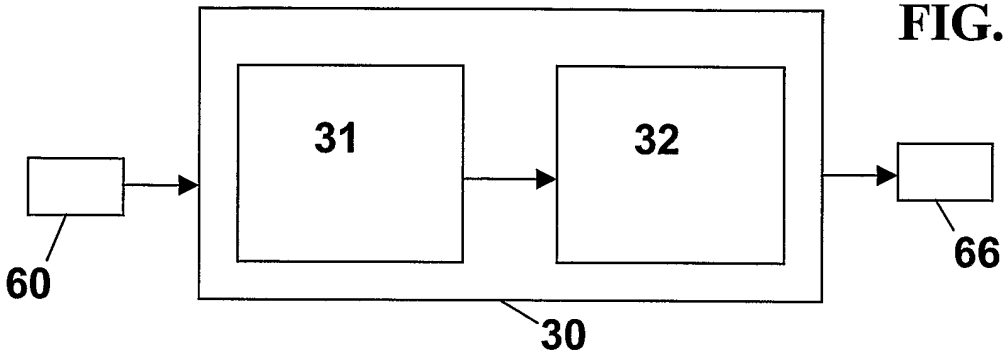


FIG. 120

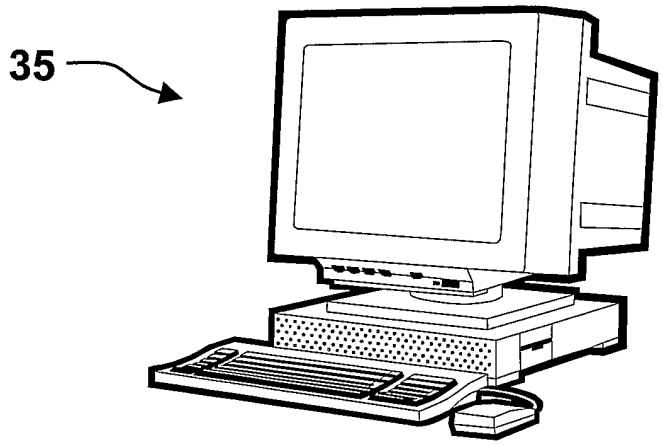


FIG. 130

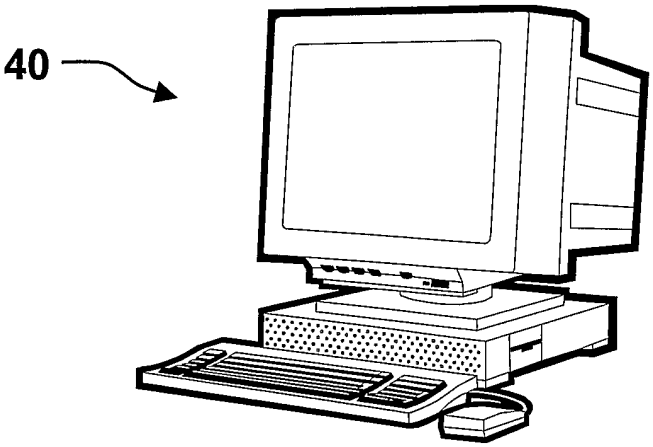


FIG. 140

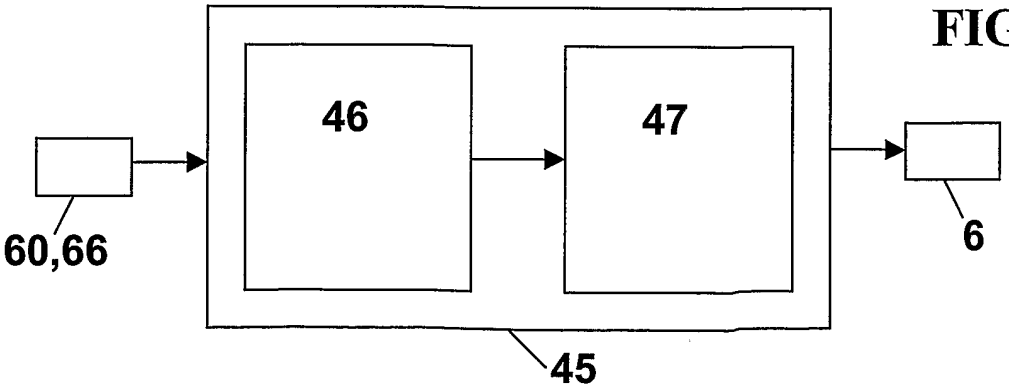


FIG. 150