(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Numéro de publication : **0 402 210 B1**

(12) # FASCICULE DE BREVET EUROPEEN

(45) Date de publication du fascicule du brevet : **30.08.95 Bulletin 95/35**

(51) Int. Cl.[6] : **G06F 11/08**

(21) Numéro de dépôt : **90401476.8**

(22) Date de dépôt : **01.06.90**

(54) **Procédé pour vérifier l'intégrité d'un logiciel ou de données, et système pour la mise en oeuvre de ce procédé.**

(30) Priorité : **06.06.89 FR 8907429**

(43) Date de publication de la demande : **12.12.90 Bulletin 90/50**

(45) Mention de la délivrance du brevet : **30.08.95 Bulletin 95/35**

(84) Etats contractants désignés : **AT BE CH DE DK ES FR GB GR IT LI LU NL SE**

(56) Documents cités :
EP-A- 0 280 035
DE-A- 3 709 524
IEEE SYMPOSIUM ON SECURITY AND PRIVACY, Oakland, CA, 18-21 avril 1988, pages 52-58, IEEE, New York, US; M.K. JOSEPH et al.: "A fault tolerance approach to computer viruses"
IEEE SYMPOSIUM ON SECURITY AND PRIVACY, Oakland, CA, 1-3 mai 1989, pages 312-318, IEEE, New York, US; G.I. DAVIDA et al.: "Defending systems against viruses through cryptographic authentication"

(73) Titulaire : **BULL CP8**
68 route de Versailles,
B.P. 45
F-78430 Louveciennes (FR)

(72) Inventeur : **Oisel, André**
5, Les Nouveaux Horizons
F-78990 Elancourt (FR)
Inventeur : **Ugon, Michel**
6, rue des Cépages
F-78310 Maurepas (FR)

(74) Mandataire : **Corlu, Bernard Edouard et al**
Direction de la Propriété Intellectuelle BULL SA,
Poste courrier: LV59C18,
68 route de Versailles
F-78430 Louveciennes (FR)

EP 0 402 210 B1

Description

L'invention est relative à un système pour vérifier l'intégrité d'un logiciel ou de données, et à un système pour la mise en oeuvre de ce procédé. Elle permet de vérifier que des informations telles que des données qui doivent demeurer constantes d'une utilisation à l'autre d'un système informatique, et qui sont mémorisées sur un support, en des endroits précis de ce support, ou bien encore que des informations constituant les instructions de fonctionnement d'un programme informatique, encore appelé logiciel, n'ont pas été altérées de façon volontaire ou accidentelle entre des utilisations successives. En effet, pour qu'un logiciel fonctionne correctement, il ne faut pas que ses instructions (ou bien encore que des données nécessaires entre deux utilisations successives) soient modifiées de façon anarchique.

En effet, depuis que les ordinateurs sont d'un usage quasi généralisé, n'importe quel utilisateur peut accéder aux programmes d'application, ou bien encore aux systèmes d'exploitation des ordinateurs.

Il arrive, en raison de cette possibilité d'accès généralisé, que des utilisateurs commettent des erreurs de manipulation, qui se traduisent par une altération du programme ou des données nécessaires au fonctionnement du système informatique, ou bien encore que de façon volontaire, certaines personnes mal intentionnées modifient la structure des programmes, ou le contenu des données d'un système informatique, afin de perturber le fonctionnement du système. Dans ce dernier cas, le problème est encore plus crucial que lors d'une modification suite à une erreur, puisque la modification volontaire peut consister à introduire dans le programme, ou dans les données, des éléments parasites tels que des instructions qui, lorsqu'ils sont pris en compte par le programme, entraînent une auto-modification de ce dernier, qui peut aller jusqu'à sa destruction complète, par effet de diffusion de proche en proche.

Par ailleurs, la plupart des logiciels sont maintenant protégés contre la recopie, et contiennent à cet effet des moyens qui permettent que lors d'une copie, la copie, ou bien même l'exemplaire original soit pollué et/ou se dégénère au fur et à mesure des utilisations successives. Ceci peut poser un problème lorsqu'un utilisateur de bonne foi a acheté une copie faite en fraude et voit rapidement le logiciel devenir inutilisable.

Ce genre de pollution des logiciels ou des données entraîne une dégénérescence progressive du logiciel et est beaucoup plus difficile à détecter qu'une pollution entraînant un dysfonctionnement immédiat du logiciel. En effet, ce dernier type de pollution, entraînant un dysfonctionnement immédiat, est remarquable très rapidement, puisque généralement ceci se traduit par des résultats de traitement aberrants.

Par contre, une pollution entraînant une dégénérescence de proche en proche, surtout lorsqu'elle est faite dans un but nuisible, n'est pas forcément détectable à la seule lecture des résultats, car elle peut être introduite d'une façon telle que les résultats effectués lors des premières utilisations soient corrects ou tout au moins le semblent, les erreurs détectables se manifestant à l'issue d'un certain nombre d'utilisations.

Enfin, un dernier type de pollution peut consister, après qu'un logiciel correct ait été chargé en vue de son exécution, notamment lorsque le système fonctionne en réseau, à faire en sorte que des instructions parasites soient intercalées ou substituées à certaines des instructions normales de fonctionnement du logiciel. Ceci peut se faire à distance, par exemple par l'intermédiaire de lignes de transmission.

Dans ce cas, l'utilisateur qui a chargé le logiciel, même s'il sait que le logiciel original qu'il a chargé est correct, ne peut pas forcément détecter l'intrusion et la modification faite de l'extérieur, et il en résulte qu'il lance le programme qui va donner des résultats de traitement immédiatement aberrants.

Les instructions ou données mises en oeuvre dans un système informatique sont codées sous forme de mots binaires, comportant chacun un nombre déterminé de bits. Un format usuel est l'octet, c'est-à-dire qu'un mot contient huit bits, pouvant avoir un état logique "0" ou un état logique "1". Selon le type d'instructions ou de données, il peut être nécessaire d'employer pour une même instruction ou une même donnée plusieurs mots, et un logiciel est constitué par une succession de ces mots. Les altérations, dont il a été fait mention auparavant, peuvent consister, soit à rajouter des mots binaires, donc des instructions ou des données parasites, soit à modifier l'état d'un ou plusieurs bits de certains mots du programme original.

Une première solution connue, pour détecter des altérations, et qui peut s'avérer satisfaisante lorsqu'une instruction ou une donnée a été modifiée de façon involontaire, consiste à signer le logiciel, c'est-à-dire que suite aux instructions ou aux données, un ou plusieurs mots binaires constituent la signature du logiciel et/ou des données. A cet effet, on considère que le programme original est un message M auquel on fait subir une transformation S = f(M), dont le résultat S constitue une signature, que l'on appose à un endroit déterminé du logiciel. Par exemple, la signature peut être constituée par le dernier mot du logiciel.

Afin de vérifier la signature au moment du chargement ultérieur du logiciel, en vue d'une utilisation, la signature du logiciel chargé est recalculée et comparée avec la signature mémorisée sur le support. En cas de concordance, ceci signifie que l'original n'a pas été altéré. Un tel procédé est connu, par exemple, de EP-A-280 035.

Cependant, en cas de fraude volontaire, un frau-

deur compétent qui connaîtrait la fonction ou l'algorithme utilisé pour calculer la signature, peut faire en sorte qu'à chaque intrusion illicite, la signature mémorisée soit modifiée, et qu'en conséquence, lors de l'utilisation suivante, il y ait concordance entre la signature recalculée et la signature dernièrement mémorisée sur le support du logiciel ou de données. Ainsi, une personne chargée de vérifier la conformité de la signature, en mettant en oeuvre l'algorithme de transformation du message, trouve une concordance et ne découvre pas l'intrusion.

Par ailleurs, cette méthode est difficilement applicable dans le cas où le logiciel est de taille importante, car chaque calcul de signature, en vue d'une vérification, nécessite un certain temps, pendant lequel le logiciel n'est pas utilisé pour la fonction pour laquelle il a été prévu.

L'invention a pour but de remédier à ces inconvénients, en proposant un procédé et un système qui permettent de vérifier en toutes circonstances, de façon rapide et fiable, qu'un logiciel ou des données sont conformes et n'ont pas été altérés de façon volontaire ou non entre deux utilisations.

L'invention est définie par les revendications indépendantes jointes.

Elle concerne un procédé pour vérifier l'intégrité, par rapport à un message original, d'un message subséquent contenant des informations, consistant à faire calculer par des moyens de traitement une signature à partir du message original auquel on applique un algorithme, caractérisé en ce qu'il comprend les étapes consistant à :

- diviser le message original en un nombre total de modules
- utiliser les dits moyens de traitement pour appliquer le dit algorithme à chaque module du message original pour calculer une signature respective de chaque module ;
- utiliser un objet portatif électronique possédant des circuits de traitement et des moyens de mémorisation incluant une zone de mémoire protégée non volatile accessible seulement par les dits circuits de traitement, et mémoriser les dites signatures dans la zone de mémoire protégée et le dit algorithme dans les moyens de mémorisation ;

et vérifier les messages subséquents dans l'objet portatif par les étapes consistant à :

- sélectionner, dans les messages subséquents, un nombre partiel de modules inférieur au nombre total de modules, le dit nombre partiel étant tel que la probabilité de détecter une altération dans le message subséquent, en considérant seulement le dit nombre partiel de modules, a une valeur déterminée ;
- utiliser les circuits de traitement pour appliquer le dit algorithme à chaque module sélectionné de façon à calculer une signature respective ;
- sélectionner dans la zone de mémoire protégée les signatures des modules du message original qui correspondent aux modules sélectionnés du message subséquent et comparer ces signatures avec celles du message subséquent ; et
- faire envoyer par l'objet portatif une réponse indiquant si l'intégrité entre les messages original et subséquent est confirmée, c'est-à-dire si les dites signatures du message original correspondent respectivement à celles du message subséquent.

L'invention est donc particulièrement avantageuse car elle peut être mise en oeuvre à tout moment de la durée de vie du logiciel ou des données. En effet, la mémorisation des signatures, à l'intérieur d'un objet portatif, qui peut être avantageusement un dispositif du genre carte à mémoire et à microcircuits électroniques, peut être faite à tout moment, par n'importe quel utilisateur qui désire se protéger. Ceci peut être fait par le concepteur, et dans ce cas l'objet portatif est fourni avec le logiciel ; ceci peut être effectué par un utilisateur qui désire vérifier que, au fur et à mesure des utilisations, son logiciel reste intègre. Dans un tel cas, l'utilisateur se procure un objet portatif spécifique vierge, et fait se dérouler le programme de calcul de signatures au moment qu'il désire, en vue de leur mémorisation, puis de façon arbitraire, à l'aide de cet objet portatif, il peut parfois vérifier l'intégrité du logiciel.

Par ailleurs, étant donné que le calcul des signatures en vue de leur comparaison avec les signatures initialement stockées s'effectue à l'intérieur de l'objet portatif, et puisque le contrôle de la conformité des signatures s'effectue en extrayant des signatures mémorisées dans une zone accessible seulement par les circuits de traitement de l'objet portatif, il est impossible à un fraudeur de tromper le système. En effet, ce sont les circuits de l'objet portatif qui recalculent chaque signature et les envoient aux circuits de comparaison qui ne sont accessibles que sous contrôle des circuits de traitement. En conséquence, si le message n'est pas intègre, il est impossible d'envoyer une fausse signature à l'adresse des circuits de comparaison, de sorte que toute modification ou intrusion est immédiatement détectée.

De plus, le fait de faire porter la vérification du message, non pas sur le nombre total m de modules, mais sur un nombre partiel p inférieur à m permet de réduire avantageusement le temps de calcul nécessaire à la vérification. Dans la mesure où le calcul est effectué dans un objet portatif, c'est-à-dire un dispositif ayant des moyens de calcul forcément limités par le volume disponible, cette réduction du temps de calcul est précieuse.

D'autres caractéristiques et avantages de l'invention apparaîtront avec la description ci-après, faite en regard des figures 1 à 6 annexées illustrant le

principe, et certains moyens de mise en oeuvre de la présente invention.

Sur les figures 1A à 1D, on a illustré la structure connue en soi d'une suite d'informations, à laquelle la présente invention est susceptible de s'appliquer.

La figure 1A illustre le schéma type de la structure d'un logiciel. Un logiciel est composé d'une suite d'instructions, dans l'exemple numéro 1 à numéro k, codées sous forme binaire, à l'aide de mots repérables chacun par leur adresse. Dans l'exemple illustré, le logiciel comporte m mots binaires, m pouvant être égal ou supérieur à k. En effet, selon le type d'instructions, le codage d'une même instruction peut affecter plusieurs mots binaires. C'est ce que l'on a illustré sur cette figure 1A où l'instruction N° 2 est codée sur deux mots, les mots 2 et 3. Chaque mot comporte généralement, dans un système informatique structuré, un nombre n de bits déterminés, qui est généralement de la taille d'un octet, c'est-à-dire de huit bits, ou un multiple de l'octet. Bien entendu, ceci n'est pas limitatif et l'invention pourrait également s'appliquer à n'importe quelle autre structure de logiciel dans lequel le format des mots pourrait varier de l'un à l'autre.

Un logiciel ainsi structuré, avec des mots de même longueur, c'est-à-dire n bits, si on le prend dans un ordre déterminé constitue un message de longueur totale, exprimée en bits L = m x n. De façon générale, la longueur totale du message est constituée par la somme de l'ensemble des bits constituant chacun des mots utilisés pour la séquence d'instructions.

L'invention est également applicable à la vérification de l'intégrité de séquences de données, ce que l'on a illustré par la figure 1B. En effet, il peut être nécessaire de vérifier l'intégrité de données qui sont susceptibles d'être utilisées de façon répétitive, car par exemple nécessaires au déroulement d'un programme.

Les données peuvent être indépendantes du programme, comme illustré par la figure 1B sur laquelle on a représenté un ensemble de m mots de n bits contenant un nombre j de données.

Comme les instructions, les données peuvent être codées sur plusieurs mots binaires, ce qui explique que le nombre j de données puisse être différent du nombre n de mots contenant ces données, c'est-à-dire que j peut être inférieur ou égal à m.

La figure 1C illustre un exemple particulier de logiciel dans lequel sont mélangées des instructions et des données (x données et y instructions).

Il faut remarquer, d'une façon générale, qu'une instruction peut être considérée comme une donnée particulière.

Généralement, un logiciel est implanté sur un support particulier, par exemple dans la mémoire centrale d'un ordinateur, ou bien encore sur le disque dur contenant les programmes d'un micro-ordinateur. Egalement, il peut être utilisé directement à partir du support d'origine, tel que la disquette sur laquelle il a été implanté au départ. Une non intégrité du logiciel, volontaire ou non, se traduit par une modification de l'un ou l'autre des mots dont il a été fait mention précédemment.

La figure 1D montre qu'un mot est en fait constitué par un ensemble de bits pouvant prendre la valeur "0" ou "1", et la modification d'un mot se traduit par le changement d'état de l'un des bits au moins. Une autre modification peut consister à superposer aux instructions ou données d'origine, des instructions ou données parasites. Ceci peut survenir lorsque le logiciel est chargé sur un autre support que son support d'origine, par exemple lorsque le logiciel est implanté sur un disque dur, afin d'être utilisé à partir de ce disque dur.

Lorsqu'une instruction ou une donnée, soit parasite, soit modifiée, est lue, il peut se produire alors des phénomènes incontrôlables.

La présente invention consiste à considérer que la suite des instructions ou données, que l'on désignera dans la suite de la description indifféremment par l'un ou l'autre de ces termes, ou par le terme information, puisque l'invention s'applique indifféremment à la vérification de l'intégrité de logiciels et/ou de données, constitue un message de longueur L exprimée en bits, dont la valeur de L est égale au nombre de bits constituant le message.

Une première solution, applicable directement aux messages de taille réduite, consiste à précalculer une signature électronique du message, en utilisant un algorithme de transformation A de ce message, ledit algorithme pouvant prendre en compte en outre au moins une clé secrète, puis en stockant la signature obtenue S dans la mémoire électronique d'une carte à microcalculateur possédant l'algorithme utilisé pour le précalcul de signature, et l'objet portatif comporte en outre une zone de mémoire accessible par les seuls circuits de traitement qu'il incorpore, contenant la clé secrète au cas où elle a été utilisée lors du précalcul de la signature.

Au moment de la vérification de l'intégrité d'un message, l'utilisateur connecte l'objet portatif au système contenant le logiciel, et lance un programme de vérification qui a pour effet de faire recalculer, par les circuits de traitement de l'objet portatif, à l'aide de l'algorithme qu'il incorpore, et le cas échéant de la clé secrète, une autre signature prenant en compte la suite de bits constituant le message à vérifier.

La signature recalculée est alors comparée, à l'aide des circuits de traitement de l'objet portatif, à la signature initialement calculée et stockée dans la mémoire de l'objet portatif, et accessible seulement par les circuits de traitement de ce dernier. Si le message ayant servi de base au calcul de la première signature est identique à celui ayant servi à la vérification, c'est-à-dire dans le cas où le logiciel et/ou les

données n'ont pas été altérés, alors la signature enregistrée dans la mémoire de l'objet portatif est identique à la signature recalculée, ce qui peut être indiqué à la personne effectuant la vérification.

On voit immédiatement un autre intérêt de l'invention : puisque la signature calculée pour le contrôle l'est par les circuits de traitement de l'objet portatif, et est comparée à l'intérieur de l'objet portatif, par les mêmes circuits de traitement, il devient impossible de simuler une fausse signature, car elle ne serait pas prise en compte par les circuits de traitement, contrairement à ce qui se passait dans l'art antérieur où un fraudeur pouvait associer à un logiciel et/ou à des données une fausse signature correspondant au logiciel ou aux données remaniées.

La figure 2 illustre le principe d'un mode de mise en oeuvre de la présente invention, en vue d'effectuer le calcul de la signature qui doit être stockée dans la mémoire de l'objet portatif servant par la suite au contrôle, et en utilisant une clé secrète K.

Le message M de longueur L, est par exemple découpé en blocs élémentaires B1, B2 ... Bf, dont chacun contient un nombre de bits compatible avec le format de travail des circuits de traitement effectuant le calcul de la signature. On pourrait par exemple envisager que chaque bloc contienne un bit, mais avec les circuits de traitement actuels, chaque bloc sera plutôt constitué par un nombre de bits multiple de l'octet. Le premier bloc B1 peut par exemple être constitué par le premier mot du message servant de base à la signature, le second bloc B2 peut être constitué par le second mot du message, et ainsi de suite jusqu'au dernier bloc Bf qui est constitué par le dernier mot du message. Bien entendu, il est tout à fait envisageable de compliquer la tâche à un fraudeur, en utilisant un algorithme de calcul qui considère les mots, ou bien encore les bits de chaque mot dans un ordre différent de l'ordre dans lequel ils apparaissent sur le support sur lequel ils sont implantés.

Le principe mis en oeuvre consiste à calculer une signature dont le format est utilisable directement par les circuits de traitement de l'objet portatif, alors que le nombre de bits constituant le message peut être nettement supérieur au nombre de bits correspondant au format de traitement des circuits.

Le principe mis en oeuvre illustré par la figure 2, consiste à effectuer autant d'opérations qu'il y a de blocs, et a combiner le résultat de chacune de ces opérations, de façon à obtenir une signature S possédant un nombre (s) de bits limité. Une valeur d'initialisation VI peut être appliquée à l'entrée des circuits de traitement prenant en compte l'algorithme A, et la clé secrète K est appliquée sur une autre entrée des mêmes circuits, de sorte que l'application de la clé secrète K et de la valeur d'initialisation VI procure un premier résultat intermédiaire qui est combiné avec le contenu du premier bloc B1, par exemple par l'intermédiaire d'une fonction OU EXCLUSIF. Le résultat de la transformation est réalisé grâce au OU EXCLUSIF qui est alors appliqué sur une première entrée des circuits de traitement prenant en compte l'algorithme, la seconde entrée prélevant la clé secrète K, de sorte qu'un second résultat intermédiaire est obtenu en sortie des circuits de traitement qui est combiné avec le contenu du second bloc B2 par l'intermédiaire de la même fonction de transformation, c'est-à-dire le OU EXCLUSIF.

Il en est ainsi pour chacun des blocs, chacun leur tour jusqu'au dernier Bf, dont le contenu est combiné par l'intermédiaire de la fonction OU EXCLUSIF avec un résultat antérieur Rf obtenu par les circuits de traitement, par application de l'algorithme A sur la clé secrète K et sur le résultat du OU EXCLUSIF précédent. Le résultat obtenu à partir du OU EXCLUSIF appliqué sur le contenu du bloc Bf et sur le résultat antérieur Rf, est combiné à la clé K par l'intermédiaire de l'algorithme A dans les circuits de calcul de la signature, et le résultat de cette dernière combinaison constitue la signature S.

Il est bien entendu que la valeur d'initialisation VI doit être la même que celle qui servira aux calculs ultérieurs, en vue d'une vérification.

Lorsque la signature S a été calculée, elle est alors stockée dans la mémoire de l'objet portatif, de façon à pouvoir servir de référence pour un contrôle ultérieur. Lorsque l'on veut contrôler l'intégrité du message, avant de s'en servir, il suffit alors de faire vérifier par l'objet portatif que la signature calculée est bien égale à celle qui a été stockée de façon inaccessible, autrement que par les circuits de traitement, dans la mémoire de l'objet portatif.

Une valeur d'initialisation VI peut être constituée par une valeur contenue dans l'objet portatif, par exemple par le numéro de série de celui-ci. Il peut également s'agir d'un code confidentiel, qui est entré par la personne lançant le calcul de la signature qui doit être mémorisée, et qui sera par la suite fourni aux autres utilisateurs susceptibles d'avoir à effectuer une vérification de l'intégrité du message. Enfin, la valeur d'initialisation VI peut être constituée par le contenu d'un registre mémoire particulier de l'objet portatif, dont le contenu est identique à chaque utilisation de l'objet portatif. La valeur VI peut encore être un nombre aléatoire déterminé par les circuits de traitement au moment du calcul des signatures avant leur mémorisation, et qui est par la suite mémorisé en même temps que les signatures.

Le mode de mise en oeuvre illustré par la figure 2 n'est qu'un exemple, et il est bien entendu que l'on peut utiliser l'algorithme différemment, et que l'on peut se passer de l'utilisation d'une valeur d'initialisation, ainsi que de la clé secrète K, ou bien que l'on peut utiliser des fonctions autres que la fonction OU EXCLUSIF qui a été envisagée.

Dans la mémoire d'un objet portatif destiné à la vérification de l'intégrité d'un message est contenue

au moins une signature S, et l'objet portatif comporte par ailleurs des circuits de traitement, tels qu'un processeur, et un algorithme de transformation A. L'objet portatif est agencé de façon à ce que le message à vérifier puisse être adressé au circuit de traitement de l'objet portatif. Eventuellement, l'objet portatif contient aussi une clé secrète K, et une signature est dans ce cas le résultat d'un chiffrement.

La figure 3 illustre une variante perfectionnée d'un objet portatif 1 qui contient dans sa zone de mémoire non volatile 10 et accessible par les seuls circuits de traitement 11 de l'objet, plusieurs signatures S1, S2 ... Sm, de même qu'une clé secrète unique K. Chaque signature est en fait la signature d'un message différent, et a été obtenue à partir de la même clé secrète K. On peut avoir cette configuration lorsqu'un même fournisseur de logiciels procure plusieurs logiciels à un même utilisateur. Dans ce cas, la signature de chacun des logiciels peut être stockée dans le même objet portatif.

Par ailleurs, l'algorithme A est mémorisé dans une autre zone 12 de l'objet.

Ce peut être encore le cas lorsqu'un utilisateur, qui désire se protéger, utilise un objet portatif spécifique qui contient au départ une clé secrète K, et dans lequel il fait mémoriser la signature de chacun des logiciels et/ou supports de données dont il dispose.

Dans un tel cas, où un même objet portatif contient les signatures de plusieurs messages, il faut que chaque signature soit associée à des moyens pour identifier le message d'origine, afin qu'au moment d'une vérification de signature, les circuits de traitement de l'objet portatif sachent retrouver dans sa mémoire la signature du message d'origine sensé correspondre au message à vérifier. Pour ce faire, par exemple, un numéro d'ordre, ou bien encore un identifiant différent, est associé à chaque message d'origine, au moment du précalcul d'une signature en vue de sa mémorisation. Une donnée en correspondance avec ce numéro d'ordre ou cet identifiant est mémorisée dans l'objet portatif au moment où la signature correspondante est mémorisée, de sorte que les circuits de traitement de l'objet portatif sont en mesure de faire la corrélation entre une donnée d'identification et la signature correspondante.

Le numéro d'ordre, ou l'identifiant, peut être soit déterminé par l'utilisateur qui désire effectuer le précalcul d'une signature en vue de sa mémorisation, soit par les circuits de traitement de l'objet portatif lui-même.

Il est bien entendu qu'un objet portatif ne peut pas être utilisé seul, et que dans ce cas, comme dans toutes les variantes qui seront illustrées par la suite, il doit être couplé, par l'intermédiaire de circuits de couplage et/ou d'interface, avec un système 2 plus vaste, et notamment avec le système de traitement de données dans lequel le message d'origine (logiciel ou données) va être exploité. Ce système de traitement de données est généralement une partie d'un ordinateur et comporte au moins un clavier, et des moyens d'impression et/ou de visualisation. Les circuits de couplage et/ou d'interface permettent d'établir un dialogue entre les circuits de traitement de l'objet portatif et les circuits de traitement du système plus vaste.

Dans le cas où c'est l'utilisateur qui détermine le numéro d'ordre ou l'identifiant d'un logiciel, il le rentre alors dans le système, par exemple, par l'intermédiaire du clavier du système de traitement plus large. Par contre, lorsque ce sont les circuits de traitement de l'objet portatif qui déterminent la donnée en correspondance, alors l'utilisateur est informé du numéro d'ordre ou de l'identifiant associé au message d'origine dont il vient de faire effectuer le précalcul de signature, suite à un dialogue qui s'est établi entre les circuits de traitement de l'objet portatif et ceux du système plus large, soit par les moyens de visualisation de ce système, soit par les moyens d'impression.

Quelle que soit la variante utilisée, l'utilisateur doit garder trâce, sur un support distinct, de l'identifiant ou du numéro d'ordre correspondant à un message d'origine donné, et doit l'indiquer au système, par l'intermédiaire du clavier, ou d'autres moyens d'entrée de données, au moment du calcul d'une signature pour vérification, de sorte que les circuits de traitement de l'objet portatif effectueront la comparaison de signature uniquement sur la signature sensée correspondre en mémoire.

Le calcul d'une signature, en vue de sa mémorisation dans les circuits de mémoire 10 d'un objet portatif, peut être effectué directement à partir des circuits de traitement 11 de l'objet portatif lui-même et de la clé secrète qu'il renferme éventuellement suite à sa fabrication, ce qui est très avantageux, puisque la signature peut être calculée sans jamais être divulguée à l'extérieur, car dès que les circuits de traitement ont terminé le calcul, ils effectuent la mémorisation de la signature. On peut également envisager que le premier calcul de signature, avant sa mémorisation, soit effectué par des circuits de traitement extérieurs à ceux de l'objet portatif et incorporés dans un système extérieur 2 auquel est susceptible d'être connecté l'objet portatif 1. Le système extérieur 2 est par exemple une unité de traitement du logiciel ou des données à vérifier. Ces circuits extérieurs mettent en oeuvre le même algorithme que celui contenu dans l'objet portatif. Dans un tel cas, si les calculs utilisent une clé secrète , celle-ci peut être déterminée au moment du calcul de chaque signature, puis mémorisée en même temps que cette dernière, ou bien encore la clé secrète peut être prélevée à l'intérieur de l'objet portatif, sous contrôle des circuits de traitement qu'il incorpore, puis transmise aux circuits extérieurs, en vue du calcul de la signature avant stockage de cette dernière. Cette solution présente cependant l'inconvénient qu'il faut transmettre la clé secrète à l'exté-

rieur et qu'il faut donc par la suite la faire disparaître des circuits de traitement extérieurs après calcul de la signature. Cette solution est cependant avantageuse lorsque les logiciels, de taille plus importante, nécessitent des temps de traitement assez importants pour le calcul des signatures. En effet, le temps de traitement de l'unité centrale du microprocesseur peut constituer un obstacle à l'utilisation du procédé, dans la mesure où la signature est calculée sur la base de la totalité du message constituant le logiciel et/ou les données. En effet, pour un logiciel de un mégaoctet, il est nécessaire de disposer de plus d'une heure de calcul avant d'obtenir le résultat du calcul de signature, et par conséquent le résultat de la vérification, car les circuits de traitement incorporés dans les objets portatifs usuels, tels que les cartes à microcircuits, ont des temps de traitement nettement inférieurs aux temps de traitement des ordinateurs plus puissants.

Des temps de traitement élevés lors de la vérification peuvent s'avérer absolument inacceptables pour une utilisation fréquente. C'est pourquoi, dans une variante, on propose un processus de contrôle beaucoup plus rapide que le précédent, utilisable dans tous les cas de figure. Pour ce faire, le message est découpé préalablement en un certain nombre de parties ou modules M1, M2 ... Mm, et à chaque module on associe une signature S1, S2, S3 ... Sm, et chaque signature est mémorisée dans une zone secrète différente d'un même objet portatif. Selon la taille du message pour lequel on effectue le calcul de signature S1, S2, S3 ... Sm, avant leur stockage, on choisit de faire effectuer le calcul, soit par les circuits de traitement de l'objet portatif, dans le cas où le message n'est pas de taille trop importante, ou bien, si la taille nécessite un temps de calcul trop élevé, alors on fait effectuer les calculs par des circuits de traitement plus rapides, par exemple ceux d'un ordinateur auquel est accouplé l'objet portatif pendant la phase de calcul des signatures.

Il faut cependant noter que, lors de la phase de calcul des signatures, en vue de leur stockage dans la mémoire 10 de l'objet portatif, le temps de calcul n'est pas critique, de sorte qu'afin d'éviter de divulguer la clé secrète à l'extérieur, on préfère la solution consistant à faire effectuer le calcul par les circuits 11 de traitement de l'objet portatif lui-même.

On constate donc que l'ensemble du message a servi de base à l'élaboration d'un certain nombre de signatures, de sorte que tous les bits constitutifs du message ont été pris en compte. Dans un tel cas, il est possible d'utiliser plusieurs processus pour contrôler l'intégrité du message ayant ainsi servi de base à l'élaboration de plusieurs signatures.

Une première méthode consiste à faire choisir au hasard plusieurs p modules différents parmi l'ensemble des m modules qui représentent la totalité du logiciel à contrôler. Le nombre p peut être prédéterminé et constant lors de chaque vérification, les circuits de traitement se contentant de choisir des modules différents pour une vérification, et étant agencés pour avoir cette possibilité que les modules choisis d'une vérification à l'autre soient éventuellement différents.

Afin de déterminer des modules, les circuits de traitement de l'objet portatif utilisent le même processus que celui qui a été utilisé pour déterminer les m modules ayant servi au calcul initial des m signatures. Ainsi, si lors du calcul initial le message a été découpé en modules de k bits, alors au moment de la vérification des signatures, les circuits de traitement de l'objet portatif redécoupent le message qu'ils reçoivent en modules de k bits, et choisissent aléatoirement parmi ces modules un nombre p de modules pour lesquels la vérification de la signature va être effectuée. Les circuits de traitement de l'objet portatif effectuent alors le calcul des signatures des p modules choisis et les comparent aux signatures qui sont sensées correspondre dans la mémoire de la carte.

La comparaison peut être effectuée au coup par coup, c'est-à-dire que dès que la carte a effectué le recalcul d'une signature, elle vérifie la concordance ou non de cette signature avec la signature sensée correspondre dans la mémoire, ou bien encore l'ensemble des signatures recalculées peut être stocké dans une mémoire tampon, puis la comparaison a lieu après que les p signatures aient été récalculées.

Dès lors que les circuits de traitement de l'objet portatif détectent une différence entre une signature recalculée, et la signature sensée correspondre en mémoire de l'objet portatif, alors le message est jugé non intègre, et des moyens associés au circuit de traitement, tel que des moyens d'affichage du système auquel l'objet portatif est connecté, indiquent le résultat positif ou négatif de la comparaison.

Ainsi, à titre d'exemple, si le message d'origine a été découpé en modules de huit bits, le premier module étant constitué par les huit premiers bits du message, le second module étant constitué par les huit bits suivant, et ainsi de suite, la signature enregistrée du premier module correspond au chiffrement des huit premiers bits, et la signature du second module correspond au chiffrement du neuvième au seizième bit du message initial. Si, lors de la vérification de signature, les circuits de traitement décident de vérifier la signature du deuxième module, alors les circuits de traitement prennent en compte la deuxième série de huit bits du message dont il faut vérifier l'intégrité, recalculent la signature de cette deuxième série de huit bits du message à vérifier en appliquant l'algorithme A sur cette série de huit bits, et en appliquant éventuellement une clé secrète K si elle a été utilisée lors du calcul avant mémorisation, et la signature stockée dans la mémoire de l'objet portatif, correspondant au deuxième module initial, est comparée à la signature recalculée du module sensé correspondre.

Bien entendu, si les p signatures recalculées sont correctes, bien que la vérification ne soit pas survenue sur les m signatures initialement calculées, puis stockées dans la mémoire de l'objet portatif, le système considère cependant que le logiciel peut être jugé intègre, et le résultat positif de la comparaison est indiqué.

Le nombre p de modules sur lesquels l'objet portatif doit effectuer la vérification des signatures peut être prédéterminé, les modules vérifiés pouvant par contre être différents d'une vérification à l'autre. Le nombre p de modules à vérifier est indiqué aux circuits de traitement de l'objet portatif, et il est choisi de façon que l'exhaustivité du contrôle soit suffisante pour atteindre un niveau de confiance acceptable. En effet, puisque l'on vérifie p signatures au lieu des m signatures d'origine, on réduit le temps de calcul nécessaire à la vérification de façon conséquente par rapport au temps de calcul qui avait été nécessaire pour mémoriser la totalité des signatures.

Egalement, le nombre p de modules peut ne pas être prédéterminé par avance, mais il peut être choisi de façon aléatoire par les circuits de traitement de l'objet portatif. Dans ce cas, il faut prévoir une vérification de la valeur du nombre p calculé, pour éviter que la vérification porte sur un nombre trop faible de modules, de sorte que la validité du contrôle ne serait pas suffisante. Par ailleurs, il ne faut pas que le nombre p soit trop élevé, afin que les temps de traitement demeurent néanmoins acceptables.

La figure 4 présente une courbe qui permet d'apprécier le nombre p de modules qu'il faut contrôler, en fonction du nombre total m de modules que contient le message, pour obtenir une probabilité Pr égale à 0,9, c'est-à-dire qu'elle présente le nombre de modules qu'il faut contrôler pour que l'on ait neuf chances sur dix de découvrir une altération du message, dans le cas où de plus, le message contient un nombre q de modules altérés équivalent au nombre p de modules contrôlés. On constate ainsi que pour un logiciel contenant mille modules, dont une soixantaine environ seraient altérés, il faut contrôler soixante modules pour avoir neuf chances sur dix de découvrir une altération du message.

En d'autres termes, ceci signifie que si on effectue la vérification d'un soixantaine de signatures de modules, pour un message contenant mille modules, et que moins de soixante modules ont été altérés, alors la probabilité de découvrir une altération à l'issue de ces soixantes contrôles est supérieure à 9/10, et a contrario, elle diminue si le nombre de modules altérés est supérieur.

Afin de déterminer le nombre p de modules qu'il faut vérifier, il faut donc effectuer un compromis entre le nombre q de modules susceptibles d'être altérés et le nombre total n de modules que contient le message.

En supposant que chaque module est repérable par un numéro d'ordre, comme ceci a été exposé précédemment, le choix aléatoire des p modules, sur lesquels la vérification de signature va être effectuée, peut consister à faire élaborer par les circuits de traitement de l'objet portatif p nombres aléatoires différents déterminant chacun le numéro d'ordre du module choisi.

Ainsi, en supposant que l'on désire effectuer quatre vérifications, dans un ensemble de m modules, les circuits de traitement de l'objet portatif effectueront quatre tirages au sort de nombres inférieurs ou égaux à m, différents entre eux. Si, par exemple, le calcul donne les nombres 2, 4, j, m-1, alors les circuits de traitement de l'objet portatif effectueront le calcul des signatures des seconds, quatrième, jème, m-1ème modules du message dont il faut vérifier l'intégrité, et iront les comparer aux deuxième, quatrième, jème, m-1ème signatures mémorisées dans la mémoire de l'objet portatif.

A la suite de la comparaison, les circuits de traitement de l'objet portatif agiront sur des moyens permettant d'indiquer si une différence, ou bien encore une similitude a été détectée lors des différentes comparaisons.

Dans une variante, pour déterminer quels modules devront faire l'objet d'une vérification de signature, on fait calculer aux circuits de traitement de l'objet portatif un nombre binaire (a), de longueur m, c'est-à-dire qui comporte un nombre de bits égal au nombre de modules constituant le message. Par ailleurs, on fait en sorte que le nombre binaire soit issu d'un code p parmi m, c'est-à-dire qu'il comporte un certain nombre de bits p dans un état logique déterminé, alors que les m-p bits restant sont dans un état logique complémentaire. Ainsi, par exemple, si l'état initial des bits est l'état logique "0", la génération du nombre (a) fera que p bits soient dans un état différent de l'état de repos. Chaque bit pouvant être repéré par son numéro d'ordre dans l'ensemble des m bits, on choisira d'effectuer la vérification des signatures sur les modules dont le numéro d'ordre correspond au numéro d'ordre des bits passés à "1" dans le nombre binaire aléatoire (a).

D'autres possibilités encore peuvent être envisagées par l'homme du métier sans sortir pour autant du cadre de la présente invention, tel qu'il est défini par les revendications.

Les différentes variantes qui viennent d'être exposées peuvent s'avérer suffisantes et offrent un degré de sécurité satisfaisant pour déterminer avec de bonnes probabilités qu'un message a été altéré ou non. Cependant, on peut envisager d'autres variantes qui permettent d'améliorer la sécurité de la méthode.

Ces variantes perfectionnées sont réalisées en tenant compte du fait qu'une intrusion frauduleuse consiste généralement à modifier une séquence d'instructions voisines dans le programme, ou bien

encore à introduire des séquences parasites assez localisées et servant peu souvent. C'est seulement lorsque le programme en cours de déroulement utilise ces instructions parasites que les effets dûs à leur présence peuvent s'avérer désastreux. Il peut en effet arriver que ces instructions parasites ne soient pas appelées par le programme en cours, sous certaines conditions d'utilisation, alors qu'elles le seront sous d'autres conditions.

C'est pourquoi, afin d'augmenter la probabilité de détection d'une modification locale extrêmement peu utilisée, dans une variante de mise en oeuvre, préalablement au calcul des signatures, l'algorithme mis en oeuvre dans les circuits de traitement de l'objet portatif est tel qu'il organise un découpage du message en blocs, indépendamment de leur contenu, et organise un entrelacement des blocs dans les calculs de signatures, de façon à rendre impossible toute modification cohérente du message. Un ensemble de blocs entrelacés forme un module.

On peut ainsi considérer que chaque mot binaire élémentaire, constituant le message, constitue un bloc. Dans un tel cas, si le format de travail des circuits de traitement est l'octet, chaque bloc sera constitué de huit bits.

Dans une variante, on considère qu'un bloc n'est pas constitué par un seul mot binaire, mais par exemple par plusieurs mots binaires consécutifs dans la suite du message. Ainsi, un premier bloc peut être constitué par exemple par les cent premiers mots binaires, c'est-à-dire les cent premiers mots du message dont on souhaite vérifier l'intégrité, et qui comportent chacun un nombre de bits déterminé, par exemple huit ou seize, ou tout autre valeur compatible avec les formats de travail des circuits de traitement de l'objet portatif, le second bloc étant constitué alors par les cent mots binaires suivant, et ainsi de suite jusqu'à la fin du message.

Bien entendu, il peut arriver que le nombre total de mots constituant le message ne permette pas de constituer un dernier bloc avec autant de mots binaires appartenant au message que les blocs précédents. C'est le cas lorsque le quotient du nombre total de mots constituant le message par le nombre de mots choisis pour constituer chaque bloc n'est pas un nombre entier. Dans un tel cas, le dernier bloc ne peut pas contenir un nombre de mots issu du message égal au nombre que contiennent les blocs précédents.

C'est par exemple le cas lorsqu'un message contient mille trente mots et que chaque bloc est constitué de cent mots. Il est ainsi possible de constituer dix blocs de cent mots et il reste seulement trente mots pour constituer le dernier bloc. Dans un tel cas, le dernier bloc est constitué à l'aide de ces trente mots restant auquels on ajoute par exemple soixante-dix mots de valeur binaire nulle, c'est-à-dire constitués exclusivement par des "0".

Pour éviter cet artifice, on peut faire en sorte que le nombre de mots constituant chaque bloc soit un diviseur exact du nombre de mots constituant le message, de sorte que l'ensemble des blocs ne contiendra que des mots issus du message d'origine. Cette solution présente l'inconvénient qu'il est nécessaire que les circuits de traitement comptabilisent le nombre total de mots constituant le message, préalablement à la constitution des blocs, pour déterminer quel nombre de mots devra constituer chaque bloc. Préalablement, il faut donc que les circuits de traitement sachent quel est le nombre optimal de mots pour constituer chaque bloc, et si ce nombre optimal ne constitue par un diviseur du nombre total de mots constituant le message, les circuits de traitement doivent déterminer quel est le diviseur de valeur inférieure qu'il faut employer. On conçoit donc que cette solution est un peu plus lourde et difficile à mettre en oeuvre ; elle est cependant envisageable.

La figure 5 permet d'illustrer la façon de comprendre comment les blocs sont répartis à l'intérieur du message, et comment les circuits de traitement effectuent le calcul des premières signatures, ou bien encore les circuits de traitement de l'objet portatif, lorsqu'ils exécutent les calculs de vérification, les entrelacent.

Sur la figure 5, on a représenté un ensemble m blocs numérotés de b1 à bm. Chaque bloc peut être constitué, comme décrit précédemment, par un seul mot binaire, ou par plusieurs mots binaires associés les uns aux autres. En associant un nombre déterminé de blocs entre eux, on obtient un module qui peut, par la suite, servir de base au calcul d'une signature, comme on l'a décrit auparavant.

Si le message P est découpé en m blocs numérotés de B1 à Bm, il y a possibilité de constituer m/n modules, de n blocs chacun tels que le module Mi de rang i, soit constitué par les blocs de rang i, i + n, i + 2n, et ainsi de suite, jusqu'à i + rn avec $1 \leqq i \leqq n$ et $0 \leqq r \leqq m/n - 1$.

Ainsi, le premier module serait constitué par les blocs B1, B1 + n, B1 + 2n, ... , B1 + rn accolés les uns aux autres.

La détermination du nombre n doit être faite de façon que les temps de calcul et de vérification de signatures ne soient pas trop longs d'une part, et d'autre part, pour que chaque module contienne, autant de possible, exclusivement des informations issues du message d'origine, ou du message à vérifier.

En effet, si n n'est pas un diviseur exact du nombre de blocs, certains modules pourraient contenir des informations autres que les informations relatives aux messages sur lesquels il faut effectuer le calcul des signatures. Il faudrait en effet compléter certains modules, par exemple avec des "0" ou des "1" binaires.

C'est pourquoi, de préférence, le nombre n de blocs constituant chaque module est choisi de façon

que n soit un diviseur du nombre total de blocs constituant le message.

Indépendamment du nombre de mots constituant chaque bloc, ou du nombre de blocs constituant chaque module, on voit que l'imbrication des blocs dans les modules et que les signatures sont totalement indépendantes de la structure même du message, ce qui est particulièrement important dans le cas où le message est un logiciel sur lequel on veut effectuer des vérifications d'intégrité. En effet, la modification d'une partie du message, en raison de cette imbrication, est susceptible d'apparaître au travers de plusieurs signatures, de sorte que l'on augmente la probabilité de détecter les modifications, puisque par rapport à leur état d'origine, plusieurs modules sont susceptibles de subir des modifications. Il y a donc, en quelque sorte, une diffusion des modifications dans l'ensemble ou dans une partie non négligeable des modules. Cette disposition permet donc de réduire le nombre de calculs de signatures de vérification, par rapport à la disposition selon laquelle on considérait que chaque module était constitué par un certain nombre de blocs se suivant dans l'ensemble du message.

Une autre façon de constituer les modules, illustrée par la figure 6 consiste, d'une façon semblable à ce qui a été fait précédemment, à découper l'ensemble du message en blocs, B1, B2, B3 ... Bm, possédant chacun un nombre déterminé de bits ou de mots. Le bloc B1 est par exemple constitué par les k premiers mots binaires du message, le bloc B2 est constitué par les k mots suivant, et ainsi de suite jusqu'à la fin. Là encore le nombre k est choisi, par exemple, pour être un diviseur du nombre de mots constituant le message, de façon que le dernier mot constitué lors de l'arrangement en vue du calcul des signatures avant mémorisation dans l'objet portatif soit constitué exclusivement de mots appartenant au message d'origine, et que l'on ne soit pas obligé par un artifice de finir le bloc avec des informations non significatives.

Chaque bloc comporte donc un nombre déterminé de bits, chacun de ces bits étant repérable par son rang à l'intérieur du bloc, et la constitution d'un module consiste à associer un ou plusieurs bits de rangs déterminés d'un bloc avec le ou les bits de mêmes rangs dans les autres blocs, puis le processus de calcul des signatures peut avoir lieu en utilisant les modules ainsi constitués.

Ainsi, en supposant que l'on prenne un bit dans chaque bloc pour constituer les modules, un premier module serait constitué par le premier bit du premier bloc, par le premier bit du second bloc, et ainsi de suite jusqu'au premier bit du dernier bloc constituant le message. Un second module serait constitué par le second bit du premier bloc, le second bit du second bloc, et ainsi de suite.

Chaque module est donc constitué par une chaîne d'informations transversale du message considéré, et il devient dès lors très difficile qu'une modification cohérente d'une partie du message passe inaperçue lors du recalcul des signatures, quel que soit le nombre de signatures recalculées et comparées par rapport au nombre initial de signatures. En effet, généralement les instructions s'il s'agit d'un logiciel, ou bien encore des données, sont écrites successivement de façon longitudinale, et la moindre modification cohérente du programme ou des données se traduirait par une chaîne transversale différente quasiment impossible à maîtriser simultanément. Par ailleurs, cette solution permet de réduire encore plus le nombre de signatures qu'il faut recalculer et comparer aux signatures correspondantes d'origine lors de la vérification du message.

Plutôt que de choisir une chaîne transversale, telle qu'elle a été définie précédemment, on peut constituer un module à l'aide d'une chaîne d'informations pseudo-transversale qui prend un nombre déterminé de bits dans chaque bloc, au moment du calcul de chaque signature avant leur mémorisation, de façon aléatoire. Ainsi, il est possible de prendre le premier bit du premier bloc et de l'associer au dernier bit du second bloc, puis de l'associer à un bit de rang différent du troisième et ainsi de suite. Bien entendu, une telle association nécessite l'utilisation d'un nombre aléatoire de référence, par exemple tiré au moment du calcul de la signature avant mémorisation. Ce nombre aléatoire de référence est pris en compte par les circuits de traitement, pour déterminer quelle suite de bits il faut considérer, et doit être mémorisé dans les circuits de mémoire de l'objet portatif, de façon à ce qu'au moment d'une vérification, les circuits de traitement puissent savoir comment ils doivent refaire la répartition pour constituer des modules à partir du message sur lequel la vérification doit porter.

En outre, tout autre type de variante pour l'entrelacement des blocs afin de constituer des modules, ou pour l'entrelacement des mots, voire même des bits, afin de constituer des blocs, peut être envisagée. En particulier, plutôt que de suivre une suite logique pour l'entrelacement des mots ou des blocs entre eux, on peut envisager que l'entrelacement des mots, afin de constituer un bloc, ou des blocs afin de constituer un module, puisse se faire de façon aléatoire. A cet effet, par exemple, les circuits de traitement, avant de constituer les modules, en vue du calcul de leur signature pour mémorisation, devront déterminer comment sont reconstitués les modules, et il faudra mémoriser les paramètres utilisés pour constituer les modules, afin de pouvoir faire les vérifications ultérieures. Ainsi, on peut considérer que les circuits de traitement élaborent une suite de n nombres aléatoires, n correspondant au nombre de blocs qui constituera chaque module, le message contenant un nombre m de modules, et la suite de nombres aléatoires V1, V2, V3 ... Vn, permet de déterminer, à

partir d'un bloc qui constituera le premier bloc d'un module, quels sont les autres blocs constitutifs de ce module.

Dans ce cas, également, il faut garder trâce de la suite de nombres aléatoires qui ont été utilisés pour pouvoir reconstituer des modules à partir du message sur lequel il faut effectuer la vérification.

Bien que l'emploi d'une clé secrète soit préférable, afin de sécuriser la vérification, il existe cependant des cas où cet emploi n'est pas nécessaire, que le message soit découpé en modules ou non et serve à l'élaboration d'une ou de plusieurs signatures distinctes. Elle procure seulement une plus grande sécurité, car elle évite que deux objets portatifs différents, utilisés pour le calcul des signatures d'un même message, enferment des signatures identiques, puisque la clé secrète contenue dans chacun des objets portatif est différente. Elle permet donc de limiter les risques de fraude au cas où un fraudeur observerait ce qui se passe avec un message tel qu'un logiciel, et chercherait à tromper les moyens de vérification de signatures. L'emploi d'une clé secrète est important notamment lorsque le logiciel doit être transféré d'une personne à une autre par exemple. Cependant, lorsque le calcul de la ou des signatures d'un message est requis par un utilisateur final, en vue de leur mémorisation, pour une vérification ultérieure, il n'est pas nécessaire que l'objet portatif servant à la vérification contienne une clé secrète. La signature du message peut être obtenue par une simple transformation des données qu'il contient, prises dans leur ensemble, ou bien encore prises par modules distincts, à l'aide d'un algorithme, de sorte que chaque signature est une simple image des données qui ont servi à la calculer. En effet, dans le cas où c'est un utilisateur final qui souhaite pouvoir vérifier l'intégrité de son support de données, au fur et à mesure de ses utilisations, il n'y a aucun intérêt à ce qu'il cherche à tromper le système. Dans un tel cas, une signature peut résulter d'une simple compression d'informations.

Le calcul d'une signature portant sur un message ou une partie de message serait seulement fonction de ce message ou de cette partie de message auquel on ferait subir l'algorithme considéré.

Un objet portatif pour le calcul des signatures en vue de leur vérification ne comprendra donc plus, comme c'est le cas sur la figure 3 une zone 10 de mémoire incorporant la clé secrète K, mais par contre il pourrait contenir une ou plusieurs zones 10 de mémoire contenant chacune une signature S1 , S2, Sm, de même qu'il comprendrait un processeur avec des circuits 11 de traitement pour la mise en oeuvre d'un algorithme A.

En conséquence, la génération d'une signature, telle qu'elle a été décrite en regard de la figure 2, ne tiendrait plus compte de l'emploi de la clé secrète K lors de chaque opération intermédiaire nécessitée par le calcul de la signature.

Un système pour la mise en oeuvre de l'invention comporte un objet portatif 1 comprenant au moins une zone mémoire 10 pour la mémorisation d'au moins une signature du logiciel, et des circuits de traitement 11 mémorisant un algorithme 12, pour au moins effectuer l'opération de recalcul des signatures, après qu'au moins une signature originale ait été inscrite. Le fait que le recalcul soit effectué à l'intérieur des circuits de traitement de l'objet portatif permet d'éviter à un observateur extérieur de voir la valeur de la signature recalculée, même si cet observateur a eu connaissance de la signature initialement calculée, en vue de son introduction dans la mémoire de l'objet portatif, lorsque ladite signature mémorisée avait été calculée par des circuits de traitement extérieurs à l'objet portatif, par exemple pour éviter d'avoir des temps de calcul trop importants.

Les circuits de traitement de l'objet portatif contiennent donc un algorithme ou programme de chiffrement A, pour au moins transformer un message M dont il convient de vérifier l'intégrité, et la mémoire peut en outre contenir une clé K, dans le cas où on veut conférer une plus grande sécurité encore. En outre, dans le cas où la mémoire de l'objet portatif contient un grand nombre de signatures appartenant chacune à un module différent d'un message initial, il convient que les circuits de traitement de l'objet portatif soient agencés de manière telle qu'ils peuvent effectuer la vérification sur un nombre limité de modules par rapport au nombre initial, afin de réduire les temps de calcul. Egalement, les circuits de traitement doivent être en mesure de reconstituer les modules du message à vérifier, de la même façon qu'ils ont été constitués lors du calcul des signatures et de leur mémorisation.

Bien entendu, la zone de mémoire dans laquelle les signatures sont mémorisées est une zone de mémoire non volatile.

Par ailleurs, comme indiqué précédemment, l'objet portatif 1 seul ne peut pas fonctionner, et il faut l'associer à d'autres moyens 2 pour constituer un système de mise en oeuvre du procédé de l'invention. Il faut en particulier constituer des circuits d'interface entre l'objet portatif, et l'ordinateur, ou le dispositif de traitement du logiciel ou des données dont il convient de vérifier l'intégrité. C'est par l'intermédiaire de ce dispositif de traitement ou de cet ordinateur, que les informations à vérifier, de même que les informations constituant le message d'origine sont exploitées dans l'objet portatif, suite à un dialogue s'établissant entre les circuits de traitement de l'objet portatif et les circuits de traitement du dispositif de traitement associé. Entre autre, le clavier qui existe généralement sur les dispositifs de traitement, ou d'autres moyens d'introduction de données (souris, écran tactile, etc) peuvent être utilisés pour établir le dialogue avec l'objet portatif, notamment lorsqu'il s'agit de rentrer des clés

confidentielles d'accès, ou bien encore des identifiants correspondant aux messages dont il convient de vérifier l'intégrité par rapport à des messages d'origine.

Les circuits d'interface peuvent être directement incorporés dans le dispositif de traitement ou l'ordinateur, ou bien ils peuvent être placés à l'extérieur, et connectés par l'intermédiaire d'une liaison. Bien entendu, un connecteur est prévu entre l'objet portatif et les circuits d'interface ou de couplage.

Lorsque le système est utilisé pour une vérification, dans le cas où une seule signature portant sur la totalité du message original a été mémorisée, les circuits de traitement de l'objet portatif prélèvent la totalité du message dont il convient de vérifier l'intégrité, et calculent la signature de ce message à vérifier, en mettant en oeuvre l'algorithme qu'ils contiennent, puis vérifient la concordance ou non entre la signature stockée et la signature recalculée. Ceci est applicable dans le cas de messages de faible longueur.

Par contre, lorsque le message initial a été découpé en plusieurs modules, car sa longueur est relativement importante, et qu'en conséquence plusieurs signatures ont été mémorisées dans la mémoire de l'objet portatif, alors les circuits de traitement de ce dernier déterminent, dans la mesure où ceci n'est pas prédéterminé, un nombre p de modules devant faire l'objet d'une vérification de signatures, de même que leur numéro d'ordre. Ils reconstituent alors les modules devant faire l'objet d'une vérification de signatures, à partir de leur numéro d'ordre, de la même façon que les modules qui sont sensés correspondre dans le message original avaient été constitués lors du précalcul de toutes les signatures.

De préférence, afin d'éviter qu'un fraudeur puisse savoir sur quelle partie du message le calcul des signatures de vérification est effectué, l'ensemble du message est prélevé par les circuits de traitement de l'objet portatif, et c'est seulement à l'intérieur de celuici que s'effectue le tri. Bien entendu, dans la mesure où le message peut avoir une longueur très importante, nettement supérieure à la capacité de mémoire de l'objet portatif, les circuits de traitement peuvent se contenter de lire au passage les données constitutives du message à vérifier, et de ne tenir compte que des données pouvant servir de base aux signatures à vérifier.

Egalement, comme il a été évoqué précédemment, une même carte peut contenir les signatures associées à plusieurs logiciels. Afin de pouvoir distinguer ces signatures, on peut prévoir dans l'objet portatif une zone de contrôle contenant des informations relatives à l'identité de chaque logiciel dont au moins une signature a été mémorisée, cette zone de contrôle indiquant en outre aux circuits de traitement à quelles adresses en mémoire se situent les signatures relatives à un message d'origine donné. Il peut s'agir d'un numéro d'ordre, ou bien encore de tout autre type d'information permettant de repérer quel message doit faire l'objet d'une comparaison. Dans un tel cas, au moment d'une vérification, il est prévu que le système demande à l'utilisateur le numéro ou l'identité du message devant faire l'objet de la vérification.

Enfin, dans une variante perfectionnée, on prévoit que dans le cas où le message est modifié de façon volontaire en vue d'une mise à jour, alors une mise à jour de chaque signature correspondante mémorisée dans l'objet portatif, puisse avoir lieu. Dans ce cas, sous contrôle de l'utilisateur, une réinscription complète des nouvelles signatures correspondant au message modifié peut avoir lieu, soit dans une autre zone de la mémoire de l'objet portatif, soit dans la même zone, qui est alors par exemple de type EEPROM, c'est-à-dire qu'elle est effaçable et reprogrammable électriquement sous contrôle des circuits de l'objet portatif. Ceci est totalement à la portée de l'homme de l'art, puisque des objets portatifs du type cartes à mémoire et microcircuits électroniques incorporent généralement des mémoires de ce type, et il peut être prévu au niveau des bornes de contact entre ces objets portatifs et un sytème extérieur, en plus des bornes nécessaires à l'alimentation et au transfert des données, une borne nécessaire à la programmation ou à l'effacement de certaines zones de mémoire. Dans d'autres cas, les tensions de programmation sont fournies par l'objet portatif lui-même. Cependant, dans chaque cas, l'effacement et la réinscription de nouvelles signatures en mémoire s'effectuent de façon sélective, et n'affectent que les zones qui doivent être modifiées.

L'invention est donc particulièrement avantageuse, car elle permet d'une façon simple, sûre et relativement peu coûteuse de s'assurer de l'intégrité d'un message constitué par un logiciel et/ou par des données mémorisés sur un support informatique. Il peut s'agir de messages qui ont été chargés à partir d'un original et qui ont été modifiés, soit suite à une intrusion sur le site même où elles se trouvent, soit à distance, par l'intermédiaire d'une ligne de transmission.

Il est bien entendu que des modifications peuvent être apportées au procédé et au système pour sa mise en oeuvre, sans qu'elles sortent pour autant du cadre de la présente invention, tel qu'il est défini par les revendications.

Revendications

1. Procédé pour vérifier l'intégrité, par rapport à un message original, d'un message subséquent contenant des informations, consistant à faire calculer par des moyens de traitement une signature à partir du message original (M) auquel on applique un algorithme (A), caractérisé en ce qu'il comprend les étapes consistant à :
   - diviser le message original en un nombre

total (m) de modules (M1, M2... Mm) ;
- utiliser les dits moyens de traitement pour appliquer le dit algorithme (A) à chaque module du message original pour calculer une signature respective (S1, S2,... Sm) de chaque module ;
- utiliser un objet portatif électronique possédant des circuits de traitement (11) et des moyens de mémorisation incluant une zone de mémoire protégée non volatile (10) accessible seulement par les dits circuits de traitement, et mémoriser les dites signatures dans la zone de mémoire protégée et le dit algorithme dans les moyens de mémorisation ;

et vérifier les messages subséquents dans l'objet portatif par les étapes consistant à :
- sélectionner, dans les messages subséquents, un nombre partiel (p) de modules inférieur au nombre total (m) de modules, le dit nombre partiel étant tel que la probabilité de détecter une altération dans le message subséquent, en considérant seulement le dit nombre partiel de modules, a une valeur déterminée ;
- utiliser les circuits de traitement pour appliquer le dit algorithme à chaque module sélectionné de façon à calculer une signature respective ;
- sélectionner dans la zone de mémoire protégée les signatures des modules du message original qui correspondent aux modules sélectionnés du message subséquent et comparer ces signatures avec celles du message subséquent ; et
- faire envoyer par l'objet portatif une réponse indiquant si l'intégrité entre les messages original et subséquent est confirmée, c'est-à-dire si les dites signatures du message original correspondent respectivement à celles du message subséquent.

2. Procédé selon la revendication 1 caractérisé en ce que les modules sont repérés par un numéro et, pour déterminer le numéro des modules sur lesquels va porter la vérification de signatures, les circuits de traitement de l'objet portatif effectuent (p) tirages successifs de nombres aléatoires différents, chaque nombre aléatoire déterminant le numéro d'un module sur lequel doit porter la vérification.

3. Procédé selon la revendication 2 caractérisé en ce que, pour déterminer le numéro des modules sur lesquels doit porter la vérification de signatures, parmi les (m) modules que contient le message d'origine, les circuits de traitement de l'objet portatif effectuent un tirage d'un nombre binaire (a) de longueur (m) exprimée en bits de sorte que la longueur du nombre aléatoire tiré est directement représentative du nombre de modules que contient le message d'origine, et en ce que la valeur du nombre aléatoire est issue d'un code (p) parmi (m), c'est-à-dire que parmi les (m) bits que contient le nombre aléatoire, il en existe (p) ayant une certaine valeur binaire (1 ou 0), alors que les (m - p) bits restant sont à la valeur complémentaire, et en ce que chaque bit du nombre aléatoire pouvant être repéré par un numéro d'ordre différent, le numéro des modules sur lesquels doit porter les vérifications de signatures est déterminé par le numéro d'ordre des (p) bits pris parmi les (m) dans le nombre aléatoire.

4. Procédé selon la revendication 1 caractérisé en ce qu'un message est constitué d'une succession de bits repérables par leur numéro d'ordre ou leur adresse en fonction de la position qu'ils occupent dans le message, et en ce que les circuits de traitement élaborent les différents modules (M1, M2, ... Mm), à partir du message d'origine, afin de calculer la signature de chacun de ces modules, avant d'entraîner leur mémorisation par les moyens de mémorisation de l'objet portatif, de la façon suivante :

- chaque module est constitué en prélevant un cartain nombre de bits du message selon une règle prédéterminée et/ou une règle élaborée par les circuits de traitement de l'objet portatif, prenant en compte par exemple des éléments aléatoires, et la règle ayant servi à constituer chaque module à partir du message original est conservée dans l'objet portatif, de façon à reconstituer à partir d'un message à certifier les modules de ce message selon la même règle que celle qui a été utilisée pour la constitution des modules à partir du message d'origine.

5. Procédé selon la revendication 1 caractérisé en ce que le message étant organisé sous forme de mots binaires, contenant chacun un nombre déterminé de bits, une information constituant une instruction ou une partie d'instruction d'un logiciel, ou une donnée ou une partie de donnée, dont l'intégrité doit être préservée, pour le bon déroulement d'un programme mettant en oeuvre ces instructions et/ou données, la constitution des modules (M1, M2, ... Mm), en vue du calcul de leur signature d'au moins certains d'entre eux, en vue de la vérification de l'intégrité du message, consiste à faire effectuer par les circuits de traitement devant réaliser le calcul des signatures considérées, d'une part la constitution d'un certain nombre (n) de blocs (B1, B2, ... Bn), regroupant chacun un certain nombre d'informations, de sorte que chaque information ou partie

d'information contenue dans un bloc peut être repérée par la position des bits qui constituent cette information ou partie d'information dans le bloc considéré, et en ce qu'il consiste, pour constituer un module déterminé, à faire prélever par les circuits de traitement, selon une règle déterminée, au moins un bit dans chacun des blocs.

6. Procédé selon la revendication 5, caractérisé en ce que le regroupement des informations, de façon à constituer des blocs, s'effectue en prélevant les informations dans l'ordre dans lequel elles apparaissent à la lecture du message.

7. Dispositif pour vérifier l'intégrité, par rapport à un message original, d'un message subséquent contenant des informations, agencé pour faire calculer par des moyens de traitement une signature (S) à partir du message original (M) auquel on applique un algorithme (A), caractérisé en ce qu'il comprend un objet portatif comprenant des moyens de connexion à un dispositif (2), tel qu'un ordinateur, susceptible d'exécuter le message, ledit objet portatif comprenant également des circuits de traitement (11) et des moyens de mémorisation incluant une zone de mémoire protégée non volatile (10) accessible seulement par les circuits de traitement, dans lequel :

- le message original (M) est divisé en un nombre total (m) de modules (M1, M2, ... Mm) et la zone de mémoire protégée mémorise une pluralité de signatures (S1, S2, ... Sm), chaque signature étant le résultat de l'application de l'algorithme (A) à un module prédéterminé du message original ;

l'objet portatif contenant l'algorithme (A) et étant agencé pour :

- sélectionner, dans le message subséquent, un nombre partiel (p) de modules inférieur au nombre total (m) de modules, le dit nombre partiel étant tel que la probabilité de détecter une altération dans le message subséquent, en considérant seulement le nombre partiel de modules, a une valeur déterminée ;
- utiliser les circuits de traitement pour appliquer le dit algorithme à chaque module sélectionné de façon à calculer une signature respective ;
- sélectionner dans la zone de mémoire protégée les signatures des modules du message original qui correspondent aux modules sélectionnés du message subséquent et comparer ces signatures avec celles du message subséquent ; et
- faire envoyer par l'objet portatif une réponse indiquant si l'intégrité entre les messages original et subséquent est confirmée, c'est-à-dire si les dites signatures du message original correspondent respectivement à celles du message subséquent.

Patentansprüche

1. Verfahren zum Überprüfen der Integrität einer Informationen enthaltenden folgenden Nachricht in bezug auf eine ursprüngliche Nachricht, das darin besteht, mit Verarbeitungsmitteln ausgehend von der ursprünglichen Nachricht (M), auf die ein Algorithmus (A) angewendet wird, eine Kennung zu berechnen, dadurch gekennzeichnet, daß es die Schritte enthält, die darin bestehen, daß:
   - die ursprüngliche Nachricht in eine Gesamtmenge (m) von Modulen (M1, M2, ... Mm) unterteilt wird;
   - die Verarbeitungsmittel dazu verwendet werden, den Algorithmus (A) auf jeden Modul der ursprünglichen Nachricht anzuwenden, um eine jeweilige Kennung (S1, S2, ... Sm) jedes Moduls zu berechnen;
   - ein tragbarer elektronischer Gegenstand verwendet wird, der Verarbeitungsschaltungen (11) sowie Speichermittel enthält, die eine nichtflüchtige, geschützte Speicherzone (10) enthalten, zu der nur die Verarbeitungsschaltungen Zugang haben, und die Kennungen in der geschützten Speicherzone und der Algorithmus in den Speichermitteln gespeichert werden;

   die folgenden Nachrichten im tragbaren Gegenstand durch die Schritte überprüft werden, die darin bestehen, daß:
   - in den folgenden Nachrichten eine Teilmenge (p) von Modulen gewählt wird, die kleiner als die Gesamtmenge (m) von Modulen ist, wobei die Teilmenge so beschaffen ist, daß die Wahrscheinlichkeit, bei ausschließlicher Betrachtung der Teilmenge von Modulen in der folgenden Nachricht eine Änderung festzustellen, einen bestimmten Wert beisitzt;
   - die Verarbeitungsschaltungen dazu verwendet werden, den Algorithmus auf jeden gewählten Modul in der Weise anzuwenden, daß eine entsprechende Kennung berechnet wird;
   - in der geschützten Speicherzone die Kennungen derjenigen Module der ursprünglichen Nachricht gewählt werden, die den gewählten Modulen der folgenden Nachricht entsprechen, und diese Kennungen mit denjenigen der folgenden Nachricht verglichen werden; und
   - mittels des tragbaren Gegenstandes eine Antwort abgeschickt wird, die angibt, ob die

Integrität zwischen der ursprünglichen und der folgenden Nachricht bestätigt wird, d.h. ob die Kennungen der ursprünglichen Nachricht jeweils denjenigen der folgenden Nachricht entsprechen.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß die Module durch eine Nummer gekennzeichnet sind und daß die Verarbeitungsschaltungen des tragbaren Gegenstandes für die Bestimmung der Nummer derjenigen Module, auf die sich die Überprüfung der Kennungen erstrecken wird, (p) aufeinanderfolgende Ziehungen von verschiedenen Zufallszahlen ausführen, wobei jede Zufallszahl die Nummer eines Moduls bestimmt, auf den sich die Überprüfung erstrecken soll.

3. Verfahren nach Anspruch 2, dadurch gekennzeichnet, daß für die Bestimmung der Nummer derjenigen Module unter den zu der ursprünglichen Nachricht gehörenden (m) Modulen, auf die sich die Überprüfung der Kennungen erstrecken soll, die Verarbeitungsschaltungen des tragbaren Gegenstandes eine Ziehung einer Binärzahl (a) der Länge (m) ausführen, die durch Bits in der Weise ausgedrückt wird, daß die Länge der gezogenen Zufallszahl direkt die Anzahl der Module repräsentiert, die die ursprüngliche Nachricht enthält, und daß der Wert der Zufallszahl aus einem (p)-aus-(m)-Code abgeleitet wird, d.h., daß unter den (m) Bits, die die Zufallszahl enthält, (p) Bits vorhanden sind, die einen bestimmten Binärwert (1 oder 0) besitzen, während die (m-p) verbleibenden Bits den komplementären Wert besitzen, und daß, da jedes Bit der Zufallszahl durch eine andere Ordnungsnummer gekennzeichnet werden kann, die jeweilige Nummer der Module, auf die sich die Überprüfungen der Kennungen erstrecken sollen, durch die jeweilige Ordnungsnummer der (p) Bits bestimmt ist, die aus den (m) Bits in der Zufallszahl entnommen werden.

4. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß eine Nachricht aus einer Folge von Bits gebildet ist, die in Abhängigkeit von der Position, die sie in der Nachricht einnehmen, durch ihre Ordnungsnummer oder durch ihre Adresse gekennzeichnet werden können, und daß die Verarbeitungsschaltungen vor der Speicherung der verschiedenen Module (M1, M2, ...Mm) durch die Speichermittel des tragbaren Gegenstandes diese Module ausgehend von der ursprünglichen Nachricht auf die folgende Weise verarbeiten, um die Kennung jedes dieser Module zu berechnen:

- jeder Modul wird gebildet durch die Entnahme einer bestimmten Anzahl von Bits der Nachricht gemäß einer vorgegebenen Regel und/oder einer durch die Verarbeitungsschaltungen des tragbaren Gegenstandes erzeugten Regel, die beispielsweise die Zufallselemente berücksichtigt, wobei die Regel, die der Bildung jedes Moduls ausgehend von der ursprünglichen Nachricht gedient hat, im tragbaren Gegenstand gehalten wird, so daß ausgehend von einer zu bestätigenden Nachricht die Module dieser Nachricht gemäß derselben Regel wie diejenige, die für die Bildung der Module ausgehend von der ursprünglichen Nachricht verwendet worden ist, wiederhergestellt werden.

5. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß dann, wenn die Nachricht in Form von binären Worten organisiert ist, wovon jedes eine bestimmte Anzahl von Bits enthält, und wenn eine Information einen Befehl oder einen Teil des Befehls einer Software oder eine Dateneinheit oder einen Teil einer Dateneinheit bildet, deren Integrität für den guten Ablauf eines diese Befehle und/oder Dateneinheiten verwendenden Programms beibehalten werden soll, die Bildung der Module (M1, M2, ...Mm) im Hinblick auf die Berechnung der Kennung wenigstens von einigen von ihnen zur Überprüfung der Integrität der Nachricht darin besteht, daß die Verarbeitungsschaltungen, die die Berechnung der betrachteten Kennungen ausführen müssen, einerseits die Bildung einer bestimmten Anzahl (n) von Blöcken (B1, B2, ...Bn) ausführen, in welchen jeweils eine bestimmte Anzahl von Informationen zusammengefaßt ist, derart, daß jede Information oder jeder Teil einer Information, der in einem Block enthalten ist, durch die Position der diese Information oder diesen Teil einer Information bildenden Bits gekennzeichnet werden kann, und daß es darin besteht, daß für die Bildung eines bestimmten Moduls durch die Verarbeitungsschaltungen gemäß einer vorgegebenen Regel wenigstens ein Bit aus jedem der Blöcke entnommen wird, um einen bestimmten Modul zu bilden.

6. Verfahren nach Anspruch 5, dadurch gekennzeichnet, daß die Zusammenfassung der Informationen in der Weise, daß die Blöcke gebildet werden, durch Entnehmen der Informationen in derjenigen Reihenfolge erfolgt, in der sie beim Lesen der Nachricht auftreten.

7. Vorrichtung zum Überprüfen der Integrität einer Informationen enthaltenden folgenden Nachricht in bezug auf eine ursprüngliche Nachricht, wobei die Vorrichtung so beschaffen ist, daß sie durch Verarbeitungsmittel ausgehend von einer ursprünglichen Nachricht (M), auf die ein Algorithmus (A) angewendet wird, eine Kennung (S) be-

rechnet, dadurch gekennzeichnet, daß sie einen tragbaren Gegenstand enthält, der seinerseits Mittel für den Anschluß an eine Vorrichtung (2) wie etwa einen Rechner, der die Nachricht verarbeiten kann, enthält, wobei der tragbare Gegenstand außerdem Verarbeitungsschaltungen (11) und Speichermittel enthält, die ihrerseits eine nichtflüchtige, geschützte Speicherzone (10) enthalten, zu der nur die Verarbeitungsschaltungen Zugang haben, wobei in der Vorrichtung:

- die ursprüngliche Nachricht (M) in eine Gesamtmenge (m) von Modulen (M1, M2,... Mm) unterteilt wird und die geschützte Speicherzone mehrere Kennungen (S1, S2, ... Sm) speichert, wobei jede Kennung das Ergebnis der Anwendung des Algorithmus (A) auf einen vorgegebenen Modul der ursprünglichen Nachricht ist;

wobei der tragbare Gegenstand den Algorithmus (A) enthält und so beschaffen ist, daß er:

- in der folgenden Nachricht eine Teilmenge (p) von Modulen, die kleiner als die Gesamtmenge (m) von Modulen ist, auswählt, wobei die Teilmenge derart ist, daß die Wahrscheinlichkeit, bei Betrachtung lediglich der Teilmenge der Module die Erfassung einer Veränderung in der folgenden Nachricht einen bestimmten Wert besitzt;
- die Verarbeitungsschaltungen dazu verwendet, den Algorithmus auf jeden ausgewählten Modul anzuwenden, um eine jeweilige Kennung zu berechnen;
- in der geschützten Speicherzone die Kennungen derjenigen Module der ursprünglichen Nachricht auswählt, die den gewählten Modulen der folgenden Nachricht entsprechen, und diese Kennungen mit denjenigen der folgenden Nachricht vergleicht; und
- durch den tragbaren Gegenstand eine Antwort abschickt, die angibt, ob die Integrität zwischen der ursprünglichen und der folgenden Nachricht bestätigt wird, d.h. ob die Kennungen der ursprünglichen Nachricht jeweils denjenigen der folgenden Nachricht entsprechen.

Claims

1. Process for checking the integrity, against an original message, of a subsequent message containing information, consisting of causing processing means to calculate a signature from the original message (M) to which an algorithm (A) is applied, characterized in that it comprises the steps consisting of:
   - dividing the original message into a total number (m) of modules (M1, M2..Mm);
   - using said processing means to apply said algorithm (A) to each module of the original message to calculate a respective signature (S1, S2....Sm) of each module;
   - using an electronic portable object having processing circuits (11) and memorizing means including a protected non-volatile memory area (10) accessible only by said processing circuits, and memorizing said signatures in the protected memory area and said algorithm in the memorizing means;

     and checking the subsequent messages in the portable object by the steps consisting of:
   - selecting, in the subsequent messages, a partial number (p) of modules less than the total number (m) of modules , said partial number being such that the probability of detecting an alteration in a subsequent message, by considering only the partial number of modules, has a given value;
   - using the processing circuits to apply said algorithm to each selected module so as to calculate a respective signature;
   - selecting, in the protected memory area, the signatures of the modules of the original message which correspond to the selected modules of the subsequent message and comparing these signatures with those of the subsequent message; and
   - causing the portable object to send a response indicating whether the integrity between the original and subsequent messages is confirmed, i.e. whether said signatures of the original message correspond respectively to those of the subsequent message.

2. Process according to claim 1, characterized in that the modules are located by a number and, to determine the number of modules on which the signature check will be made, the processing circuits of the portable object make (p) successive choices of different random numbers, each random number determining the number of a module to be checked.

3. Process according to claim 2, characterized in that, to determine the number of modules on which the signature check will be made, from the (m) modules contained in the original message, the processing circuits of the portable object choose a binary number (a) of length (m) expressed in bits such that the length of the random number chosen is directly representative of the number of modules contained in the original message, and in that the value of the random number

comes from a code (p) out of (m), i.e. of the (m) bits contained in the random number, there are (p) that have a certain binary value (1 or 0) while the remaining (m - p) bits have the complementary value, and that each bit of the random number can be located by a different serial number; the number of modules to be signature-checked is determined by the serial number of the (p) bits taken from the (m) bits in the random number.

4. Process according to claim 1, characterized in that a message is constituted of a sequence of bits locatable by their serial numbers or addresses as a function of the positions they occupy in the message, and that the processing circuits create the different modules (M1, M2,...Mm) from the original message in order to calculate the signature of each these modules before bringing about its memorizing by the processing circuits of the portable object, as follows:

- each module is constituted by taking a certain number of message bits according to a predetermined rule and/or a rule created by the processing circuits of the portable object, taking into account random elements for example, and the rule, having served to constitute each module from the original message is kept in the portable object so as to reconstitute, from a message to be checked, the modules of this message according to the same rule used for constituting the modules from the original message.

5. Process according to claim 1, characterized in that the message is organized in the form of binary words, each containing a given number of bits, a piece of information constituting a software instruction or part of an instruction, or a datum or part of a datum, the integrity of which is to be preserved, for correct running of a program using these instructions and/or data, the constitution of modules (M1, M2,...Mm) with a view to calculating the signature of at least certain of them, with a view to checking the integrity of the message, consists of causing the processing circuits that are to calculate the signatures considered, firstly to constitute a certain number (n) of blocks (B1, B2,...Bn) each containing a certain number of pieces of information such that each piece of information or part of a piece contained in a block can be located by the positions of the bits which constitute this piece of information or part of a piece in the block considered, and in that it consists, for constituting a given module, of causing the processing circuits to take at least one bit from each of the blocks according to a given rule.

6. Process according to claim 5, characterized in that the grouping of the information in order to form blocks, is carried out by taking the information in the order in which it appears when the message is read.

7. Device for checking the integrity, against an original message, of a subsequent message containing information, arranged to cause processing means to calculate a signature (S) from the original message (M) to which an algorithm (A) is applied, characterized in that it comprises a portable object comprising means for connection to a device (2), such as a computer, able to execute the message, said portable object also comprising processing circuits (11) and memorizing means including a protected non-volatile memory area (10) accessible only by the processing circuits, in which:-

- the original message (M) is divided into a total number (m) of modules (M1, M2,...Mm) and the protected memory area memorizes a plurality of signatures (S1, S2,...Sm), each signature being the result of applying the algorithm (A) to a given module of the original message;

the portable object containing the algorithm (A) being arranged to:

- select, in the subsequent message, a partial number (p) of modules less than the total number (m) of modules, said partial number being such that the probability of detecting an alteration in the subsequent message, by considering only the partial number of modules, has a given value;
- use the processing circuits to apply said algorithm to each selected module so as to calculate a respective signature;
- select, in the protected memory area, the signatures of the modules of the original message which correspond to the selected modules of the subsequent message and compare these signatures to those of the subsequent message; and
- cause the portable object to send a response indicating whether the integrity between the original and subsequent messages is confirmed, i.e. whether said signatures of the original message correspond respectively to those of the subsequent message.

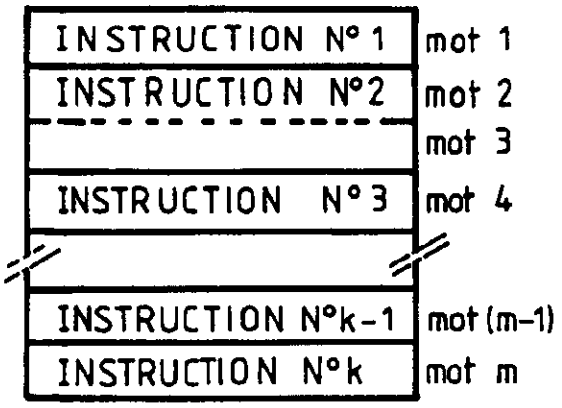

FIG.1A

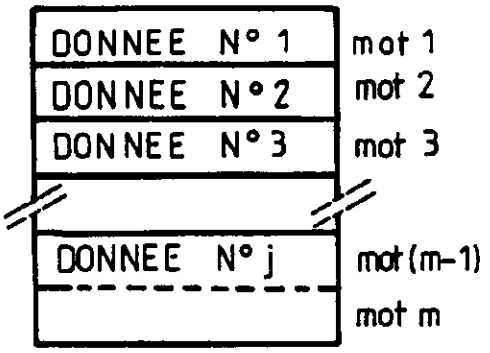

FIG.1B

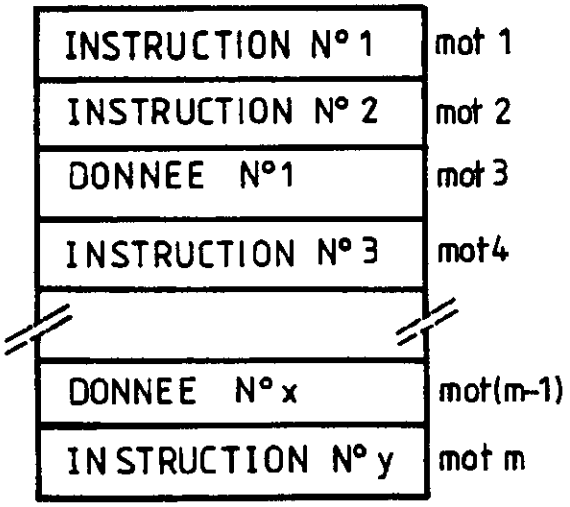

FIG.1C

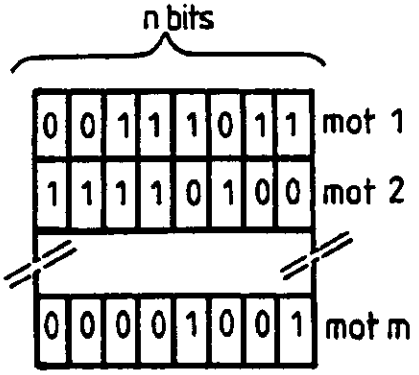

FIG.1D

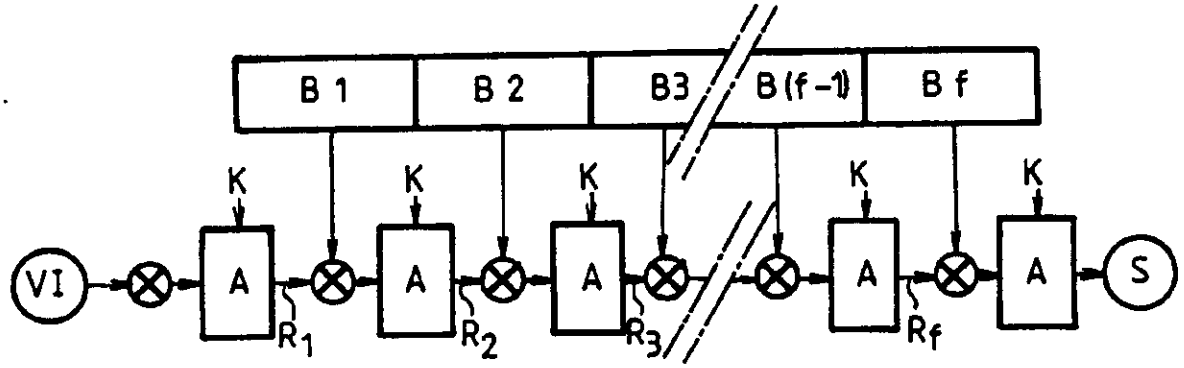

FIG.2

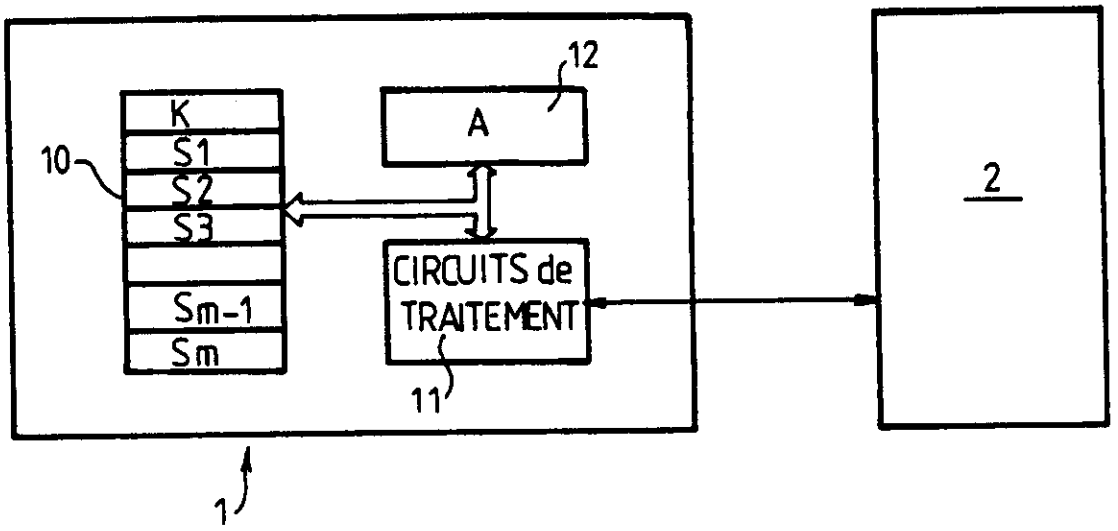
12
A
K
S1
S2
S3
Sm-1
Sm
10
CIRCUITS de TRAITEMENT
11
2
1

FIG.3

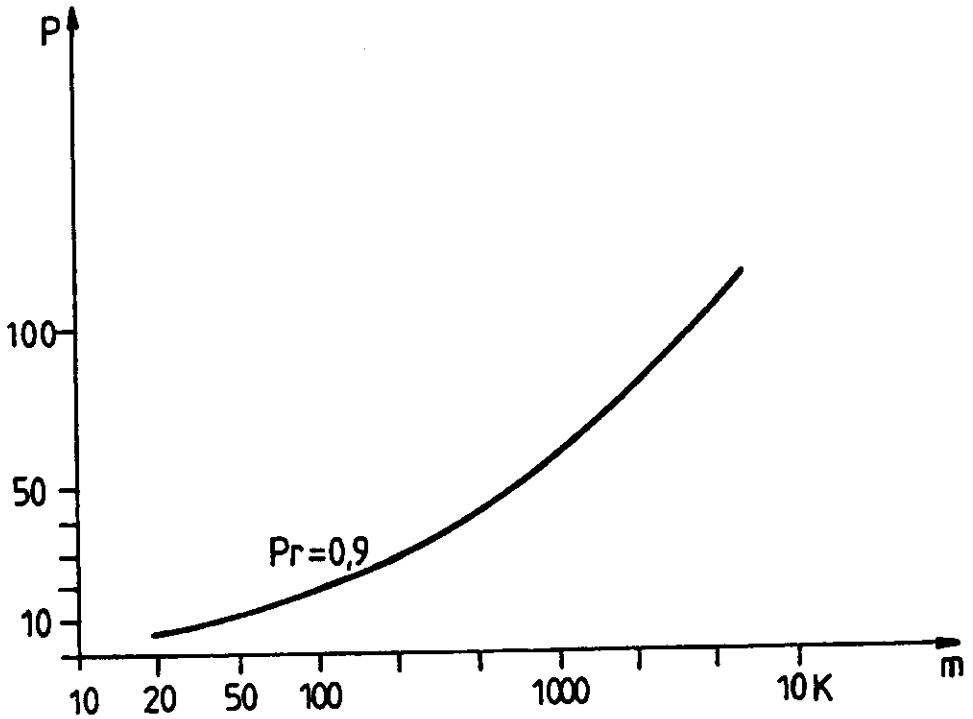
P
100
50
10
Pr=0,9
10
20
50
100
1000
10K
m

FIG.4

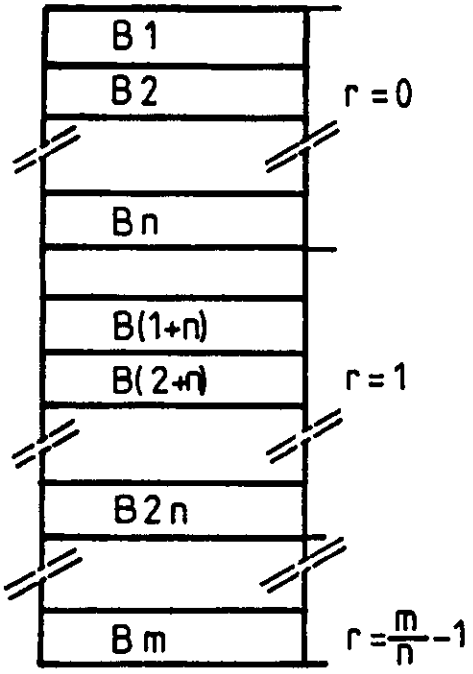

FIG.5

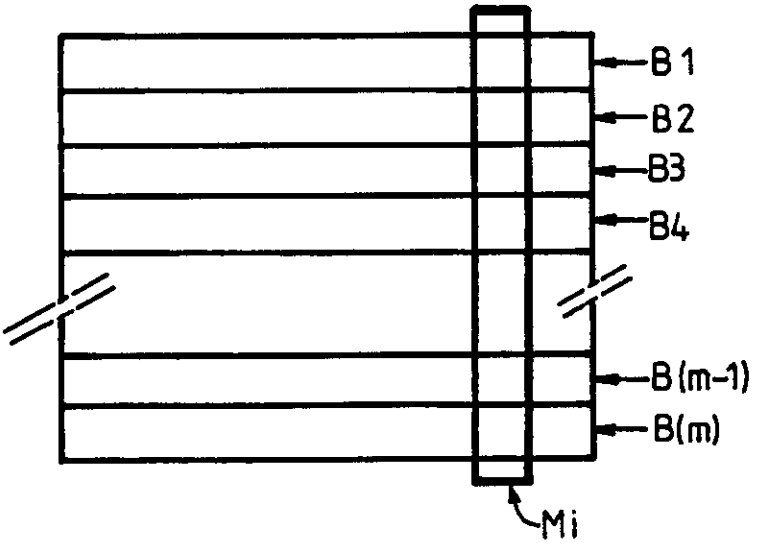

FIG.6

Patents Form 54/77

Patents Act 1977
(Rule 80 and Schedule 4)

THE PATENT OFFICE
19 OCT 1995
RECEIVED BY HAND

The Patent Office

54/77

20OCT95 E150444-2 D02807
P54/7700 35.00

Bull
O/N 19328
£35.00

Filing a translation in connection with a European patent or a European patent application

(See the notes on the back of this form)

The Patent Office

Cardiff Road
Newport
Gwent NP9 1RH

RECEIVED IN
24 OCT 1995
EUROPEAN TRANSLATIONS

1. Your reference — SEM/NL/19328

2. European patent number or publication number of application *(or International publication number (see note (e))* — 0 402 210

3. Full name and address of the or of each applicant for or proprietor of the European patent (UK) — BULL CP8
68 route de Versailles, B.P. 45
F-78430 Louveciennes/FR

   Patents ADP number *(if you know it)*

4. What kind of translated document listed at note (c) are you sending with this form? — 1(i)

   (Answer by writing 1(i), 1(ii), 1(iii) or 2)

5. Date when the European patent (UK) was granted or amended *(See note (f))* — 30-08-1995

6. Full name, address and postcode in the United Kingdom to which all correspondence relating to this form and translation should be sent — BARON & WARREN
18 South End
Kensington
London W8 5BU

   Patents ADP number *(if you know it)* — 281001

7. Do you want the address in part 6 above to be the address for service recorded on the Register or to replace the address for service currently on the Register?
   (If so then write 'YES') — YES

8. Signature — Baron & Warren — Date 19-10-1995

   AGENTS FOR THE PROPRIETORS

9. Name and daytime telephone number of person to contact in the United Kingdom — SUSAN E. MURGATROYD
0171-937-0294.

dd JS 54/77-27

IN THE MATTER OF European Patent
Application no. 90401476.8
Publication no. 0 402 210
in the name of BULL CP8

DECLARATION

I, Gillian Elizabeth HARGREAVES, a Member of the Institute of Translation and Interpreting, of 25 Station Road, Digswell, Welwyn, Hertfordshire AL6 0DU, do hereby declare that I am conversant with the English and French languages and am a competent translator thereof.

I further declare that to the best of my knowledge and belief the following translation of the text of the description of the above application, as accepted for grant, is a true and faithful rendering of the French.

Signed this 12th day of October, 1995

G E Hargreaves

The invention relates to a system for checking the integrity of a piece of software or data, and to a system for implementing this process. It makes it possible to check that information such as data that have to remain constant from one use to another of a data processing system, and that are memorised on a storage medium, in precise locations in that medium, or information constituting the operating instructions for a computer program, also known as software, have not been altered intentionally or accidentally between successive uses. For software to function correctly, its instructions (or any data required between two successive uses) must not have been modified haphazardly.

Indeed, since computers are in almost general use, any user can gain access to application programs, or to the operating systems of the computers.

Because of this possibility of general access, it sometimes happens that users make keying errors that result in an alteration of the program or the data necessary for the operating of the data processing system, or that certain ill-intentioned persons deliberately modify the structure of the programs, or the content of the data in a data processing system, so as to disrupt the operation of the system. In the latter case, the problem is even more crucial than a modification made as a result of an error, since the intentional modification may consist in introducing into the program, or into the data, extraneous elements such as instructions that, when followed by the program, lead to self-modification thereof, which can even cause its complete destruction, by a knock-on effect.

Furthermore, most software is now copy-protected and, for this purpose, contains means that make it possible, when a copy is made, for the copy, or even the original version, to be contaminated and/or to degenerate through successive uses. This can cause a problem when a user has purchased in good faith a copy made fraudulently and quickly sees the software become unusable.

This type of software or data contamination leads to the progressive deterioration of the software and is much more difficult to detect than contamination leading to immediate software malfunctioning. Indeed, this latter type of contamination,

entailing immediate malfunctioning, can be noticed very quickly, since it generally gives aberrant processing results.

In contrast, contamination leading to knock-on deterioration, particularly when done with harmful intent, is not necessarily detectable just by reading the results, since it may be introduced in such a way that the results produced on initial uses are correct or, at least, seem to be, detectable errors becoming apparent after a number of uses.

Finally, after correct software has been loaded with a view to its execution, particularly when the system is operating in a network, a last type of contamination can consist in causing extraneous instructions to be inserted or substituted for some of the normal operating instructions of the software. This can be done remotely, for example via transmission lines.

In this case, even if the user who loaded the software knows that the original software he loaded is correct, he cannot necessarily detect the intrusion and the modification made externally and, as a result, he launches the program which will immediately give aberrant processing results.

Instructions or data implemented in a data processing system are encoded in the form of binary words, each comprising a given number of bits. A usual form is the eight-bit byte, that is, a word contains eight bits, and can have a logic state "0" or "1". Depending on the type of instructions or data, it may be necessary to use several words for any one instruction or any one datum, and a software program is constituted by a succession of these words. The alterations mentioned before can consist either in adding binary words, i.e. extraneous instructions or data, or in modifying the state of one or more bits of certain words in the original program.

A first known solution for detecting alterations, which may prove satisfactory when an instruction or datum has been modified unintentionally, consists in signing the software, that is, one or more binary words following the instructions or data constitute the signature of the software and/or data. To this end, the original

program may be considered to be a message M subject to a transformation S = f(M), the result S of which constitutes a signature which is placed at a given location in the software. For example, the signature can be constituted by the last word of the software.

In order to check the signature when the software is subsequently loaded for use, the signature of the loaded software is recalculated and compared with the signature memorised on the storage medium. When they match, this means that the original has not been altered. Such a process is known, for example, from EP-A-280 035.

However, in the case of intentional fraud, a competent person intending to commit fraud who knows the function or algorithm used to calculate the signature can arrange for the memorised signature to be modified on each illicit intrusion and consequently, at the next use, for the recalculated signature to match the signature most recently stored on the storage medium of the software or data. Thus someone responsible for checking the conformity of a signature will find a match when using the message-conversion algorithm, and will not discover the intrusion.

Furthermore, this method is difficult to apply where the software is on a large scale, since each calculation of a signature for checking purposes requires a certain length of time during which the software is not used for the function for which it is intended.

The aim of the invention is to remedy these disadvantages, by proposing a process and a system making it possible to check quickly and reliably, in all circumstances, that a piece of software or data are in conformity and have not been altered, intentionally or otherwise, between two uses.

The invention is defined by the attached independent claims.

It concerns a process for checking the integrity, against an original message, of a subsequent message containing information, consisting in causing processing means to calculate a signature from the original message to which an algorithm is applied, characterised in that it comprises the steps consisting in:

means to calculate a signature from the original message to which an algorithm is applied, characterised in that it comprises the steps consisting in:

- dividing the original message into a total number of modules;
- using said processing means to apply said algorithm to each module of the original message to calculate a respective signature of each module;
- using an electronic portable object having processing circuits and memorising means including a protected non-volatile memory area accessible only to said processing circuits, and memorising said signatures in the protected memory area and said algorithm in the memorising means;

and checking the subsequent messages in the portable object by the steps consisting in:

- selecting, in the subsequent messages, a partial number of modules less than the total number of modules, said partial number being such that the probability of detecting an alteration in the subsequent message, by considering only said partial number of modules, has a given value;
- using the processing circuits to apply said algorithm to each selected module so as to calculate a respective signature;
- selecting, in the protected memory area, the signatures of the modules of the original message that correspond to the selected modules of the subsequent message and comparing these signatures with those of the subsequent message; and
- causing the portable object to send a response indicating whether the integrity between the original and subsequent messages is confirmed, i.e. whether said signatures of the original message correspond respectively to those of the subsequent message.

The invention is therefore particularly advantageous since it can be implemented at any time during the life of the software or data. Indeed, the memorisation of signatures within a portable object, which can advantageously be a device of the memory-card type with electronic microcircuits, can be performed at any time, by any user who wants to protect himself. This can be done by the designer, in which case the portable object is supplied with the software; it can be performed by a user who wants to check that the software remains uncorrupted during use. In this case

the user obtains a specific blank portable object and runs the program for calculating the signatures he wants, so that they can be memorised, then arbitrarily, using the portable object, he can occasionally check the integrity of the software.

Furthermore, given that the calculation of the signatures with a view to comparing them with the signatures initially stored is performed inside the portable object and since the conformity of the signatures is checked by extracting signatures memorised in an area accessible only to the processing circuits of the portable object, it is impossible for a person intending to commit fraud to deceive the system. It is the circuits of the portable object that recalculate each signature and send them to the comparison circuits, which are accessible only under the control of the processing circuits. Consequently, if the message is corrupted, it is impossible to send a false signature to the address of the comparison circuits, so that any modification or intrusions is detected immediately.

Moreover, the fact that the checking of the message involves not the total number m of modules, but a partial number p less than m, makes it possible to reduce the calculation time required for the checking, with advantageous effect. In so far as the calculation is performed in a portable object, that is, a device having calculation means necessarily limited by the volume available, this reduction in calculation time is valuable.

Other characteristics and advantages of the invention will become apparent from the following description, given with regard to Figures 1 to 6 attached illustrating the principle, and certain embodiments of the present invention.

Figures 1A to 1D illustrate the structure, known in itself, of a series of pieces of information, to which the present invention may apply.

Figure 1A illustrates the typical diagram of a software structure. A software program is composed of a series of instructions, numbered 1 to k in the example, encoded in binary form, using words that can each be located by their addresses. In the example illustrated, the software comprises m binary words, where m can be equal

to or greater than k. Depending on the type of instructions, the encoding of any one instruction can affect several binary words. This has been illustrated in Figure 1A where instruction no. 2 is encoded on two words, words 2 and 3. In a structured data processing system each word generally comprises a given number n of bits, which is generally the size of a byte, that is, eight bits, or a multiple of a byte. Of course, this is not exhaustive and the invention could just as well apply to any other software structure in which the format of the words could vary from one to another.

Software having this structure, with words of the same length, that is, n bits, if taken in a given order, constitutes a message of a total length, expressed in bits, of L = m x n. In general, the total length of the message consists of the sum of all the bits constituting each of the words used for the sequence of instructions.

The invention also applies to checking the integrity of data sequences, as illustrated in Figure 1B. Indeed, it may be necessary to check the integrity of data that may be used repetitively since they are, for example, necessary for running a program.

The data can be independent of the program, as illustrated in Figure 1B showing a set of m words of n bits containing a number j of data.

Like instructions, data can be encoded on several binary words, which explains why the number j of data can be different from the number n of words containing these data, that is, why j can be less than or equal to m.

Figure 1C illustrates a particular example of software in which instructions and data are mixed (x data and y instructions).

It should be noted that, in general, an instruction can be considered to be a particular type of datum.

A piece of software is generally resident in a particular storage medium, for example in the central memory of a computer, or on the hard disk containing the programs of a microcomputer. It can also be used directly from the original medium, such as

intentional or otherwise, results in the modification of one or other of the words mentioned previously.

Figure 1D shows that a word is in fact constituted by a set of bits that can take the value "0" or "1", and the modification of a word results in the change of state of at least one of the bits. Another modification can consist in superimposing extraneous instructions or data on the original instructions or data. This can arise when the software is loaded on a storage medium other than its original medium, for example when the software is installed on a hard disk in order to be used from that hard disk.

When an instruction or a datum, whether extraneous or modified, is read, uncontrollable phenomena can occur.

The present invention consists in considering that the series of instructions or data, which will be denoted in the rest of the description by either of these terms indiscriminately, or by the term information, since the invention applies indiscriminately to checking the integrity of software and/or data, constitutes a message of length L expressed in bits, the value of L being equal to the number of bits constituting the message.

A first solution, applicable directly to short messages, consists in precalculating an electronic signature for the message, using an algorithm A for converting this message, which algorithm can also use at least one secret key, then storing the signature obtained S in the electronic memory of a microcomputer card possessing the algorithm used for precalculating the signature, and the portable object further comprises a memory area accessible only to the processing circuits it incorporates, containing the secret key where this was used when precalculating the signature.

When the integrity of a message is checked the user connects the portable object to the system containing the software, and launches a checking program that causes the processing circuits of the portable object to recalculate another signature involving the series of bits constituting the message to be checked, using the algorithm it contains and, where appropriate, the secret key.

involving the series of bits constituting the message to be checked, using the algorithm it contains and, where appropriate, the secret key.

The recalculated signature is then compared, using the processing circuits of the portable object, with the signature initially calculated and stored in the memory of the portable object, and accessible only to the processing circuits of the latter. If the message used as a basis for calculating the first signature is identical to that used for checking, that is, where the software and/or data have not been altered, then the signature recorded in the memory of the portable object will be identical to the recalculated signature, which can be indicated to the person conducting the check.

Another point of interest in the invention will be immediately apparent: since the signature calculated for checking is calculated by the processing circuits of the portable object and is compared within the portable object by the same processing circuits, it becomes impossible to simulate a false signature, since it would not be taken into account by the processing circuits, in contrast with what happened in the prior art when a person intending to commit fraud could associate with software and/or data a false signature corresponding to the reworked software and/or data.

Figure 2 illustrates the principle of one embodiment of the present invention, with a view to calculating the signature to be stored in the memory of the portable object used subsequently for checking, using a secret key K.

The message M of length L is for example divided into elementary blocks B1, B2, ... Bf, each of which contains a number of bits compatible with the working format of the processing circuits calculating the signature. For example, it could be envisaged that each block contains one bit, but with current processing circuits each block is more likely to be constituted by a number of bits that is a multiple of a byte. The first block B1 can for example be constituted by the first word of the message serving as a basis for the signature, the second block B2 can be constituted by the second word of the message, and so on until the last block Bf which is constituted by the last word of the message. Of course, it is quite possible to envisage complicating the work of a person intending to commit fraud by using a calculation

algorithm that considers the words, or else the bits of each word, in a different order from the one in which they appeared on the storage medium on which they are installed.

The principle implemented consists in calculating a signature, the format of which can be used directly by the processing circuits of the portable object, while the number of bits constituting the message can be significantly greater than the number of bits corresponding to the processing format of the circuits.

The principle implemented, illustrated in Figure 2, consists in performing as many operations as there are blocks, and combining the result of each of these operations, so as to obtain a signature S having a limited number (s) of bits. An initialisation value VI can be applied to the input of the processing circuits involving the algorithm A, and the secret key K is applied to another input of the same circuits, such that the application of the secret key K and the initialisation value VI brings a first intermediate result, which is combined with the content of the first block B1, for example via an EXCLUSIVE OR function. The result of the conversion is produced by means of the EXCLUSIVE OR, which is then applied to a first input of the processing circuits involving the algorithm, the second input taking the secret key K, such that a second intermediate result is obtained at the output of the processing circuits which is combined with the content of the second block B2 through the same conversion function, i.e. the EXCLUSIVE OR.

The same applies to each of the blocks in turn, up to the last one Bf, the content of which is combined through the EXCLUSIVE OR function with the previous result Rf obtained by the processing circuits, by applying the algorithm A to the secret key K and to the result of the preceding EXCLUSIVE OR. The result obtained from the EXCLUSIVE OR applied to the content of the block Bf and to the previous result Rf is combined with the key K through the algorithm A in the signature calculation circuits, and the result of the latter combination constitutes the signature S.

Naturally, the initialisation value VI must be the same as the one that will be used for subsequent calculations, with a view to checking.

When the signature S has been calculated it is then stored in the memory of the portable object, so as to be capable of use as a reference in a subsequent check. When the integrity of the message is to be checked before use, it is then sufficient to get the portable object to check that the signature calculated is indeed equal to the one that has been stored, so that it is inaccessible other than to the processing circuits, in the memory of the portable object.

An initialisation value VI can be constituted by a value contained in the portable object, for example its serial number. It can also be a confidential code, which is entered by the person launching the calculation of the signature that has to be memorised and will subsequently be supplied to other users who may have to check the integrity of the message. Finally, the initialisation value VI can be constituted by the content of a particular memory register of the portable object, the content of which is identical each time the portable object is used. The value VI can also be a random number determined by the processing circuits when the signatures are calculated before being memorised, and which is subsequently memorised at the same time as the signatures.

The embodiment illustrated in Figure 2 is only an example and, naturally, the algorithm can be used in a different way, and both the use of an initialisation value and of the secret key K can be dispensed with, or else functions other than the EXCLUSIVE OR function envisaged can be used.

In the memory of a portable object intended for checking the integrity of a message there is at least one signature S, and the portable object also comprises processing circuits such as a processor and a conversion algorithm A. The portable object is arranged such that the message to be checked can be sent to the processing circuit of the portable object. If necessary, the portable object also contains a secret key K, and in this case a signature is the result of encoding.

Figure 3 illustrates an improved variant of a portable object 1 that contains several signatures S1, S2, ... Sm, and also a unique secret key K in its non-volatile memory area 10, accessible only to the processing circuits 11 of the object. Each signature

is in fact the signature of a different message, and has been obtained from the same secret key K. This configuration can obtain when the same software supplier provides several pieces of software to the same user. In this case the signature of each of the pieces of software can be stored in the same portable object.

Furthermore, the algorithm A is memorised in another area 12 of the object.

This may also be the case when a user who wants protection uses a specific portable object that initially contains a secret key K, and in which he memorises the signatures of each of the pieces of software and/or data storage media available to him.

In such a case, where the same portable object contains the signatures of several messages, each signature has to be associated with means for identifying the original message so that when a signature is checked the processing circuits of the portable object know where in memory to find the original message signature supposed to correspond to the message to be checked. To do this, for example, a serial number or a different identifier is associated with each original message when a signature is precalculated with a view to being memorised. A datum corresponding to this serial number or this identifier is memorised in the portable object at the time that the corresponding signature is memorised, such that the processing circuits of the portable object are in a position to correlate an identification datum with the corresponding signature.

The serial number or identifer can be determined either by the user who wants to precalculate a signature with a view to memorising, or by the processing circuits of the portable object itself.

Naturally, a portable object cannot be used alone and, in this case, as in all variants that will be illustrated subsequently, it must be coupled via coupling and/or interface circuits with a more extensive system 2, especially with the data processing system in which the original message (software or data) is to be exploited. This data processing system is generally part of a computer and comprises at least a

keyboard and printing and/or display means. The coupling and/or interface circuits allow a dialogue to be established between the processing circuits of the portable object and the processing circuits of the more extensive system.

Where it is the user who determines the serial number or identifier of a piece of software, he then enters it in the system, for example via the keyboard of the more extensive processing system. On the other hand, when it is the processing circuits of the portable object that determine the corresponding datum, then the user is informed of the serial number or identifier associated with the original message for which the signature has just been precalculated, following a dialogue that is established between the processing circuits of the portable object and those of the more extensive system, either by the display means of this system or by the printing means.

Whatever the variant used, the user has to keep a record, on a separate storage medium, of the identifier or serial number corresponding to a given original message, and has to indicate it to the system via the keyboard, or other data input means, when a signature is calculated for checking, such that the processing circuits of the portable object will perform the signature comparison solely on the signature supposed to correspond in the memory.

The calculation of a signature, with a view to its memorisation in the memory circuits 10 of a portable object, can be effected directly from the processing circuits 11 of the portable object itself and from the secret key it may contain following manufacture, which is very advantageous, since the signature can be calculated without ever being disclosed externally, for as soon as the processing circuits have finished their calculation they memorise the signature. It can also be envisaged that the first signature calculation, before memorisation, is effected by processing circuits outside the portable object and incorporated in an external system 2 to which the portable object 1 may be connected. The external system 2 is for example a unit for processing the software or data to be checked. These external circuits use the same algorithm as that contained in the portable object. In such a case, if the calculations use a secret key, it may be determined when each signature is

calculated, then memorised at the same time as the latter, or else the secret key can be taken from within the portable object, under the control of the processing circuits it incorporates, then transmitted to the external circuits with a view to calculating the signature before the latter is stored. This solution does however have the disadvantage that the secret key has to be transmitted externally and that it therefore has to be deleted subsequently from the external processing circuits after the signature has been calculated. This solution is however advantageous when voluminous software requires fairly long processing times for calculating signatures. Indeed, the processing time of the central unit of the microprocessor may constitute an obstacle to the use of the process, in so far as the signature is calculated on the basis of the whole message constituting the software and/or data. Indeed, for software of one megabyte it takes more than an hour of calculation to obtain the result of the signature calculation, and hence the result of the check, since the processing circuits incorporated in usual types of portable object, such as microcircuit cards, have processing times distinctly inferior to the processing times of the most powerful computers.

Long processing times during checking may prove absolutely unacceptable for frequent use. For this reason, one variant proposes a checking process that is much faster than the preceding one, and can be used in all cases. For this, the message is first divided into a number of portions or modules M1, M2, ... Mm, and a signature S1, S2, S3, ... Sm is associated with each module, and each signature is memorised in a different secret area of the same portable object. Depending on the size of the message for which the signature S1, S2, S3, ... Sm is calculated before storage, it is decided whether to have the calculation effected by the processing circuits of the portable object, where the message is not too long, or else, if the size requires too long a processing time, then the calculations are effected by faster processing circuits, for example those of a computer with which the portable object is coupled during the signature calculation phase.

It should however be noted that, in the phase when signatures are calculated with a view to being stored in the memory 10 of the portable object, the calculation time is not critical so that, to avoid disclosing the secret key externally, the solution in

which the calculation is performed by the processing circuits 11 of the portable object itself is preferred.

It will be noted then that the whole message has served as a basis for preparing a number of signatures, such that all the bits constituting the message have been taken into account. In such a case it is possible to use several processes to check the integrity of the message that has served as the basis for preparing several signatures.

A first method consists in causing several p different modules to be chosen at random from among all the m modules that represent the whole of the software to be checked. The number p can be predetermined and constant at each check, and the processing circuits will confine themselves to choosing different modules for a check, being arranged so as to allow for the eventuality of the modules chosen being different from one check to another.

In order to determine the modules, the processing circuits of the portable object use the same process as that used to determine the m modules used for the initial calculation of the m signatures. Thus, if at the time of the initial calculation the message has been divided into modules of k bits, then when the signatures are checked the processing circuits of the portable object again divide the message they receive into modules of k bits, and randomly choose from these modules a number p of modules for which the signature will be checked. The processing circuits of the portable object then calculate the signatures of the p modules chosen and compare them with the signatures that are supposed to correspond in the memory of the card.

The comparison can be made as and when the need arises, that is, when the card has recalculated a signature it checks whether or not this signature matches the signature that is supposed to correspond in the memory, or else all the recalculated signatures can be stored in a buffer memory, then the comparison is made after the p signatures have been recalculated.

As soon as the processing circuits of the portable object detect a difference between a recalculated signature and the signature supposed to correspond in the memory of the portable object, then the message is deemed to be corrupted, and means associated with the processing circuit, such as display means of the system to which the portable object is connected, indicate the positive or negative result of the comparison.

Thus, by way of example, if the original message was divided into eight-bit modules, the first module being constituted by the first eight bits of the message, the second module being constituted by the next eight bits, and so on, the signature recorded for the first module corresponds to the encoding of the first eight bits and the signature for the second module corresponds to the encoding of the ninth to the sixteenth bit of the initial message. If, when the signature is checked, the processing circuits decide to check the signature of the second module, then the processing circuits will take into account the second series of eight bits of the message, the integrity of which has to be checked, recalculate the signature of this second series of eight bits of the message to be checked by applying the algorithm A to this series of eight bits and, if appropriate, by applying a secret key K if it was used at the time of the calculation before memorisation, and the signature stored in the memory of the portable object, corresponding to the second initial module, is compared with the recalculated signature of the module that is supposed to correspond.

Of course, if the p recalculated signatures are correct, although the check is not made on the m signatures initially calculated, then stored in the memory of the portable object, the system nevertheless considers that the software can be deemed not to be corrupted, and the positive result of the comparison is indicated.

The number p of modules for which the portable object has to check the signatures can be predetermined, while the checked modules can differ from one check to another. The number p of modules to be checked is indicated to the processing circuits of the portable object, and it is chosen such that the check is sufficiently thorough to achieve an acceptable level of confidence. Indeed, since p signatures

are checked instead of the m original signatures, the calculation time needed for checking is reduced consistently, in comparison with the calculation time that had been necessary for memorising all the signatures.

Moreover, the number p of modules may not be predetermined in advance, but can be chosen at random by the processing circuits of the portable object. In this case the value of the number p calculated needs to be checked, to avoid the check involving too small a number of modules, such that the validity of the check would not be sufficient. Furthermore, the number p must not be too high, so that processing times nevertheless remain acceptable.

Figure 4 is a curve showing the number p of modules that have to be checked, depending on the total number m of modules contained in the message, to obtain a probability Pr equal to 0.9, i.e. it shows the number of modules that have to be checked in order for there to be nine chances in ten of discovering that the message has been altered in the case where, furthermore, the message contains a module q of altered modules equivalent to the number p of checked modules. It will be observed, then, that for a piece of software containing a thousand modules, around sixty of which were altered, sixty modules will have to be checked in order to have nine out of ten chances of discovering an alteration to a message.

In other words, this means that if around sixty module signatures are checked for a message containing a thousand modules, and if less than sixty modules have been altered, then the probability of discovering an alteration at the end of this sixty checks is greater than 9/10 and, vice versa, it diminishes if the number of altered modules is greater.

In order to determine the number p of modules that have to be checked, a compromise must therefore be made between the number q of modules likely to be altered and the total number n of modules contained in the message.

Assuming that each module can be located by a serial number, as has been stated previously, the random choice of the p modules on which the signature check is to

be made can consist in having the processing circuits of the portable object prepare p different random numbers each determining the serial number of the selected module.

Thus, assuming that four checks are to be made on a set of m modules, the processing circuits of the portable object will four times randomly draw numbers less than or equal to m, and different from each other. If, for example, the calculation gives the numbers 2, 4, j, m-1, then the processing circuits of the portable object will calculate the signatures of the second, fourth, jth, m-1st modules of the message, the integrity of which needs to be checked, and will compare them with the second, fourth, jth, m-1st signatures memorised in the memory of the portable object.

Following the comparison, the processing circuits of the portable object will act on the means for indicating whether a difference or a similarity has been detected in the various comparisons.

In a variant, to determine which modules will have to be signature-checked, the processing circuits of the portable object are caused to calculate a binary number (a), of a length m, that is, one comprising a number of bits equal to the number of modules constituting the message. Furthermore, the process ensures that the binary number comes from a code p among m, that is, it comprises a certain number of bits p in a given logic state, while the m-p bits remaining are in a complementary logic state. For example, if the initial state of the bits is the logic state "0", the generation of the number (a) will cause p bits to be in a state different from the rest state. Since each bit can be located by its serial number in the set of m bits, a choice will be made to signature-check the modules, the serial numbers of which correspond to the numbers to the serial numbers of the bits that have changed to "1" in the random binary number (a).

Yet other possibilities can be envisaged by the person skilled in the art without thereby leaving the scope of the present invention, as defined by the claims.

The different variants that have just been presented may prove sufficient and offer a satisfactory degree of security for determining with a high probability whether or not a message has been altered. However, other variants can be envisaged that make it possible to improve the security of the method.

These improved variants are made by taking account of the fact that a fraudulent intrusion generally consists in modifying a series of sequential instructions in the program, or of introducing fairly localised and infrequently-used extraneous sequences. It is only when the program uses these extraneous instructions as it is running that the effects of their presence can prove disastrous. These extraneous instructions may indeed not be called up by the program as it is running under certain conditions of use, while they are under other conditions.

For this reason, in order to increase the probability of detecting an extremely infrequently used local modification, in a variant embodiment, before the signatures are calculated, the algorithm used in the processing circuits of the portable object is such that it organises a division of the message into blocks, irrespective of their content, and organises an interlacing of the blocks in the signature calculations, so as to make any consistent modification of the message impossible. A set of interlaced blocks forms a module.

Thus it may be considered that each elementary binary word constituting the message constitutes a block. In such a case, if the working format of the processing circuits is the byte, each block will be constituted by eight bits.

In a variant, a block is considered not to be constituted by a single binary word but, for example, by several consecutive binary words in the sequence of the message. A first block may thus be constituted for example by the first hundred binary words, that is, the first hundred words of the message, the integrity of which is to be checked, which words each contain a given number of bits, for example eight or sixteen, or any other value compatible with the working formats of the processing circuits of the portable object, the second block then being constituted by the next hundred binary words, and so on until the end of the message.

Of course, it can happen that the total number of words constituting the message does not allow a last block to be constituted with as many binary words belonging to the message as the preceding blocks. This is the case when the quotient of the total number of words constituting the message and the number of words chosen to constitute each block is not a whole number. In such a case the last block cannot contain a number of words coming from the message that is equal to the number contained in the preceding blocks.

This is the case, for example, when a message contains one thousand and thirty words and each block is constituted by one hundred words. Ten blocks of a hundred words can therefore be constituted and only thirty words remain to constitute the last block. In such a case the last block is constituted using these thirty remaining words to which, for example seventy words of zero binary value, that is, containing exclusively "0"s, are added.

To avoid this artifice, one can act such that the number of words constituting each block is an exact divisor of the number of words constituting the message, so that all the blocks will contain only words coming from the original message. This solution has the disadvantage that the processing circuits count the total number of words in the message before constituting the blocks, in order to determine the number of words that have to constitute each block. The processing circuits therefore have to know in advance the optimum number of words in each block, and if this optimum number does not consist of a divisor of the total number of words constituting the message, the processing circuits have to determine what divisor of a lower value should be employed. It will be seen therefore that this solution is rather more cumbersome and difficult to implement; it is however capable of being envisaged.

Figure 5 makes it possible to illustrate how to understand how the blocks are distributed within the message, and how they are interlaced by the processing circuits calculating the first signatures, or the processing circuits of the portable object when they execute the checking calculations.

Figure 5 shows a set m of blocks numbered from b1 to bm. As described above, each block can be constituted by a single binary word, or by several binary words associated with one another. By associating a given number of blocks with each other, a module is produced that will subsequently serve as a basis for calculating a signature, as described above.

If the message P is divided into m blocks numbered from B1 to Bm, it is possible to constitute m/n modules, each of n blocks, such that the module Mi ranking i is constituted by the blocks ranking i, i + n, i + 2n, and so on as far as i + rn with $1 \leq i \leq n$ and $0 \leq r \leq m/n - 1$.

Thus the first module will be constituted by the blocks B1, B1 + n, B1 + 2n, ..., B1 + rn one after another.

The number n must be determined so that the times for calculating and checking signatures are not too long on the one hand, and on the other hand so that, where possible, each module contains exclusively information coming from the original message, or the message to be checked.

If n is not an exact divisor of the number of blocks, certain modules could contain information other than the information relating to the messages on which the signatures have to be calculated. Indeed, certain modules would have to be supplemented, for example with binary "0"s or "1"s.

This is why the number n of blocks constituting each module is preferably chosen such that n is a divisor of the total number of blocks constituting the message.

Irrespective of the number of words constituting each block, or the number of blocks constituting each module, it will be seen that the overlapping of the blocks in the modules and that the signatures are totally independent of the actual structure of the message, which is particularly important where the message is a piece of software that is to be checked for integrity. Indeed, a change made to part of the message because of this overlapping may appear across several signatures, such that the

probability of detecting modifications is increased since several modules might be altered from their original state. This means that the modifications are, to a certain extent, diffused through all or a significant part of the modules. This arrangement therefore allows the number of signature-checking calculations to be reduced in comparison with the arrangement in which each module was considered to be constituted by a certain number of blocks succeeding each other in the whole message.

Another way of constituting the modules, illustrated in Figure 6, consists, rather as before, in dividing the whole message into blocks B1, B2, B3, ... Bm, each having a given number of bits or words. The block B1 is for example constituting by the first k binary words of the message, the block B2 is constituted by the next k words, and so on until the end. There too the number k is chosen, for example, so as to be a divisor of the number of words in the message, such that the last word constituted at the time of the arrangement with a view to calculating the signatures, before memorisation in the portable object, is constituted exclusively of words belonging to the original message, and that there is no an artifice requiring the block to be completed with meaningless information.

Each block therefore comprises a given number of bits, each of these bits being locatable by its rank within the block, and a module is constituted by associating one or more bits of a given rank in a block with the bit(s) of the same rank in the other blocks, then the signatures can be calculated by using the modules constituted in this way.

Thus, assuming that a bit is taken from each block in order to constitute the modules, a first module would be constituted by the first bit of the first block, the first bit of the second block, and so on up to the first bit of the last block constituting the message. A second module would be constituted by the second bit of the first block, the second bit of the second block, and so on.

Each module is thus constituted by a chain of information across the message in question, and it then becomes very difficult for a consistent modification of part of

the message to go unnoticed when the signatures are recalculated, whatever the number of signatures recalculated and compared with the initial number of signatures. Indeed, the instructions in the case of software, or the data, are generally written in succession longitudinally, and the slightest consistent modification of the program or data would result in a different transverse chain that would be almost impossible to handle simultaneously. Furthermore, this solution allows the number of signatures that have to be recalculated and compared with the corresponding original signatures, when the message is checked, to be reduced even further.

Rather than choosing a transverse chain, as defined above, a module can be constituted using a pseudo-transverse chain of information that takes a given number of bits at random in each block, when each signature is calculated before memorisation. Thus the first bit of the first block can be associated with the last bit of the second block, then associated with a bit of a different rank in the third, and so on. Of course, such an association requires the use of a random reference number, picked when the signature is calculated before memorisation, for example. This random reference number is used by the processing circuits to determine the series of bits to be considered, and must be memorised in the memory circuits of the portable object, such that, at the time of checking, the processing circuits know how they have to perform the distribution again in order to constitute the modules from the message that is to be checked.

Furthermore, any other type of variant for interlacing blocks in order to constitute modules, or for interlacing words or even bits to make up blocks, can be envisaged. In particular, rather than following a logical sequence when interlacing words or blocks, it may be envisaged that the interlacing of words to make up a block or blocks, in order to constitute a module, can take place at random. To this effect, for example, before the processing circuits constitute the modules with a view to calculating their signatures for memorisation, they will have to determine how the modules are reconstituted, and the parameters used for constituting the modules will have to be memorised so that subsequent checks can be conducted. Thus, it may be considered that the processing circuits prepare a series of n random numbers,

n corresponding to the number of blocks that will constitute each module, the message containing a number m of modules, and the series of random numbers V1, V2, V2, ... Vn makes it possible to determine, from a block that will constitute the first block of a module, which other blocks will constitute this module.

In this case it is also necessary to keep a record of the series of random numbers that have been used so that the modules can be reconstituted on the basis of the message that has to be checked.

Although the use of a secret key is preferable, in order to make the check fail-safe, there are however cases when this is not necessary, whether or not the message is divided into modules and serves for the preparation of one or more distinct signatures. It simply gives greater security, since it prevents two different portable objects, used for calculating signatures for the same message, from containing identical signatures, since the secret key contained in each of the portable objects is different. This limits the risk of fraud in the case where a person intending to commit fraud observes what is happening with a message such as a piece of software, and attempts to deceive the signature-checking means. The use of a secret key is specially important when the software has to be transferred from one person to another for example. However, when the calculation of the signature(s) of a message is required by an end user, with a view to memorising them, for subsequent checking, it is not necessary for the portable object used for checking to contain a secret key. The signature of the message can be obtained by simply converting the data it contains, taken as a whole or in distinct modules, using an algorithm, so that each signature is a simple image of the data used in order to calculate it. Indeed, where it is an end user who wants to be able to check the integrity of his data storage medium, as and when he uses it, he has no interest in seeking to deceive the system. In such a case, a signature can result from a simple compression of information.

The calculation of a signature related to a message or part of a message would be a function only of that message or that part of the message subjected to the algorithm in question.

A portable object for calculating signatures with a view to checking them will therefore not, as is the case in Figure 3, comprise a memory area 10 incorporating the secret key K, but on the other hand it could contain one or more memory areas 10 each containing a signature S1, S2, Sm, in the same way that it would comprise a processor with processing circuits 11 for implementing an algorithm A.

Accordingly, the generation of a signature as described in relation to Figure 2, will no longer involve the use of the secret key K at the time of each intermediate operation required by calculating the signature.

A system for implementing the invention comprises a portable object 1 comprising at least one memory area 10 for memorising at least one software signature, and processing circuits 11 memorising an algorithm 12, in order at least to perform the operation of recalculating the signatures after at least one original signature has been recorded. The fact that the recalculation takes place inside the processing circuits of the portable object makes it possible to prevent an outside observer from seeing the value of the recalculated signature, even if this observer knew the signature initially calculated, with a view to introducing it into the memory of the portable object, when said memorised signature had been calculated by processing circuits outside the portable object, to avoid having very lengthy calculation times, for example.

The processing circuits of the portable object therefore contain an algorithm or encoding program A, in order at least to convert a message M, the integrity of which has to be checked, and the memory can also contain a key K where even greater security is to be conferred. Furthermore, where the memory of the portable object contains a large number of signatures each belonging to a different module of an initial message, the processing circuits of the portable object should be arranged so that they can perform the check on a number of modules less than the initial number, to reduce calculation times. The processing circuits must also be in a position to reconstitute the modules of the message to be checked, in the same way that they were constituted when the signatures were calculated and memorised.

Of course, the memory area in which the signatures are memorised is a non-volatile memory area.

Furthermore, as indicated above, the portable object 1 cannot function alone, and needs to be associated with other means 2 for constituting a system for implementing the invention. In particular, interface circuits must be constituted between the portable object and the computer, or the processing device for the software or data, the integrity of which is to be checked. It is via this processing device or this computer that the information to be checked, and the information constituting the original message, are handled in the portable object, following a dialogue established between the processing circuits of the portable object and the processing circuits of the associated processing device. Among other things, the keyboard generally associated with processing devices, or other data-entry means (mouse, touch screen, etc.) can be used for establishing the dialogue with the portable object, especially when there is a question of entering confidential access keys, or identifiers corresponding to the message, the integrity of which should be checked against the original messages.

The interface circuits can be directly incorporated into the processing device or the computer, or they can be located outside, and connected via a link. Of course, a connector is provided between the portable object and the interface or coupling circuits.

When the system is used for checking, in the case where a single signature related to the whole of the original message has been memorised, the processing circuits of the portable object take the whole message, the integrity of which is to be checked, and calculate the signature of this message to be checked, by implementing the algorithm that they contain, then check whether or not the stored signature matches the recalculated signature. This applies to short messages.

In contrast, when the initial message has been divided into several modules, because it is relatively long, and consequently several signatures have been memorised in the memory of the portable object, then the processing circuits of the

latter will determine, in so far as it has not already been determined, a number p of modules that have to be signature-checked, and also their serial numbers. They then reconstitute the modules to be signature-checked on the basis of their serial numbers, in the same way as the modules that are supposed to correspond in the original message had been constituted when all the signatures were precalculated.

Preferably, in order to prevent a person intending to commit fraud from finding out the part of the message for which signatures have been calculated for checking, the whole message is sampled by the processing circuits of the portable object, and it is only inside the latter that sorting takes place. Of course, since the message can be very long, significantly greater than the memory capacity of the portable object, the processing circuits can confine themselves to reading the data constituting the message to be checked, as they pass, and using only the data that may serve as a basis for the signatures to be checked.

Moreover, as has been mentioned before, any one card can contain the signatures associated with several pieces of software. To be able to distinguish these signatures, a control area can be provided in the portable object, containing information as to the identity of each piece of software for which at least one signature has been memorised, this control area also indicating to the processing circuits the memory addresses where the signatures relating to a given original message are located. This can be a serial number, or any other type of information making it possible to identify which message has to be subjected to a comparison. In such a case, at the time of checking, provision is made for the system to ask the user for the number or identity of the message to be checked.

Finally, in an improved variant it is provided that, where the message is intentionally modified for updating purposes, each corresponding signature memorised in the portable object can be updated. In this case, under the control of the user, a complete re-recording of the new signatures corresponding to the modified message can take place, either in another memory area of the portable object or in the same area, which is then for example of the EEPROM type, that is, it is electrically erasable and reprogrammable under the control of the circuits of the portable object.

This is fully within the scope of the person skilled in the art, since portable objects of the memory-card type with electronic microcircuits generally incorporate memories of this type and a terminal can be provided as necessary for programming or erasing certain memory areas, at the level of the terminals connecting these portable objects with an external system, in addition to the terminals needed for power supply and data transfer. In other cases the programming voltages are supplied by the portable object itself. However, in each case, the erasing and re-recording of new signatures in the memory take place selectively, and affect only the areas to be modified.

The invention is therefore particularly advantageous, since it allows the integrity of a message constituted by a piece of software and/or data memorised on a computer storage medium to be ensured in a simple, safe and relatively inexpensive way. This can involve messages that have been loaded from an original and have been modified, either following an intrusion into the actual site where they are located, or remotely via a transmission line.

Naturally, modifications can be made to the process and the system for implementing it without thereby leaving the scope of the present invention, as defined by the claims.

CLAIMS

1. A process for checking the integrity, against an original message, of a subsequent message containing information, consisting of causing processing means to calculate a signature from the original message (M) to which an algorithm (A) is applied, characterised in that it comprises the steps consisting of:
   - dividing the original message into a total number (m) of modules (M1, M2, ... Mm);
   - using said processing means to apply said algorithm (A) to each module of the original message to calculate a respective signature (S1, S2, ... Sm) of each module;
   - using an electronic portable object having processing circuits (11) and memorising means including a protected non-volatile memory area (10) accessible only by said processing circuits, and memorising said signatures in the protected memory area and said algorithm in the memorising means;

   and checking the subsequent messages in the portable object by the steps consisting of:
   - selecting, in the subsequent messages, a partial number (p) of modules less than the total number (m) of modules, said partial number being such that the probability of detecting an alteration in the subsequent message, by considering only said partial number of modules, has a given value;
   - using the processing circuits to apply said algorithm to each selected module so as to calculate a respective signature;
   - selecting, in the protected memory area, the signatures of the modules of the original message that correspond to the selected modules of the subsequent message and comparing these signatures with those of the subsequent message; and
   - causing the portable object to send a response indicating whether the integrity between the original and subsequent messages is confirmed, i.e. whether said signatures of the original message correspond respectively to those of the subsequent message.

2. A process according to Claim 1, characterised in that the modules are located by a number and, to determine the number of modules on which the signature check will be made, the processing circuits of the portable object make (p) successive choices of different random numbers, each random number determining the number of a module to be checked.

3. A process according to Claim 2, characterised in that, to determine the number of modules on which the signature check must be made, from the (m) modules contained in the original message, the processing circuits of the portable object choose a binary number (a) of length (m) expressed in bits such that the length of the random number chosen is directly representative of the number of modules contained in the original message, and in that the value of the random number comes from a code (p) out of (m), i.e. of the (m) bits contained in the random number, there are (p) that have a certain binary value (1 or 0) while the remaining (m - p) bits have the complementary value, and that each bit of the random number can be located by a different serial number; the number of modules to be signature-checked is determined by the serial number of the (p) bits taken from the (m) bits in the random number.

4. A process according to Claim 1, characterised in that a message is constituted by a sequence of bits locatable by their serial numbers or addresses as a function of the positions they occupy in the message, and that the processing circuits create the different modules (M1, M2, ... Mm) from the original message in order to calculate the signature of each of these modules before bringing about its memorising by the processing circuits of the portable object, as follows:
   - each module is constituted by taking a certain number of message bits according to a predetermined rule and/or a rule created by the processing circuits of the portable object, taking into account random elements for example, and the rule, having served to constitute each module from the original message, is kept in the portable object so as to reconstitute, from a message to be checked, the modules of this message according to the same rule used for constituting the modules from the original message.

5. A process according to Claim 1, characterised in that the message is organised in the form of binary words, each containing a given number of bits, a piece of information constituting a software instruction or part of an instruction, or a datum or part of a datum, the integrity of which must be preserved, for the correct running of a program using these instructions and/or data, the constituting of modules (M1, M2, ... Mm) with a view to calculating the signature of at least some of them, with a view to checking the integrity of the message, consists of causing the processing circuits that have to calculate the signatures considered, first to constitute a certain number (n) of blocks (B1, B2, ... Bn) each containing a certain number of pieces of information such that each piece of information or part of a piece contained in a block can be located by the positions of the bits that constitute this piece of information or part of a piece in the block considered, and in that it consists, for constituting a given module, of causing the processing circuits to take at least one bit from each of the blocks according to a given rule.

6. A process according to Claim 5, characterised in that the grouping of the information in order to form blocks is carried out by taking the information in the order in which it appears when the message is read.

7. A device for checking the integrity, against an original message, of a subsequent message containing information, arranged to cause processing means to calculate a signature (S) from the original message (M) to which an algorithm (A) is applied, characterised in that it comprises a portable object comprising means for connection to a device (2), such as a computer, able to execute the message, said portable object also comprising processing circuits (11) and memorising means including a protected non-volatile memory area (10) accessible only by the processing circuits, in which:-

- the original message (M) is divided into a total number (m) of modules (M1, M2, ... Mm) and the protected memory area memorises a plurality of signatures (S1, S2, ... Sm), each signature being the result of applying the algorithm (A) to a given module of the original message;

the portable object containing the algorithm (A) and being arranged so as to:

- select, in the subsequent message, a partial number (p) of modules less than the total number (m) of modules, said partial number being such that the probability of detecting an alteration in the subsequent message, by considering only the partial number of modules, has a given value;
- use the processing circuits to apply said algorithm to each selected module so as to calculate a respective signature;
- select, in the protected memory area, the signatures of the modules of the original message that correspond to the selected modules of the subsequent message and compare these signatures with those of the subsequent message; and
- cause the portable object to send a response indicating whether the integrity between the original and subsequent messages is confirmed, i.e. whether said signatures of the original message correspond respectively to those of the subsequent message.

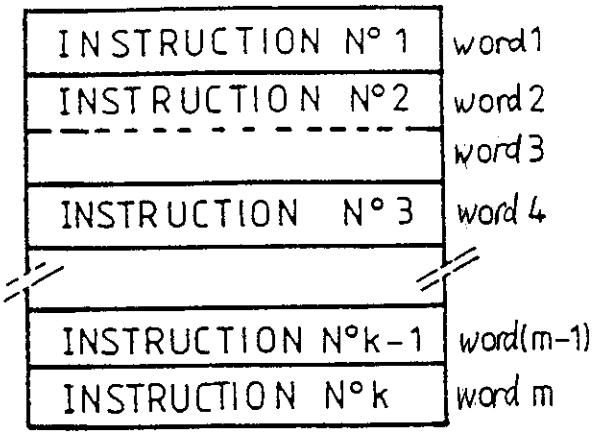

FIG.1A

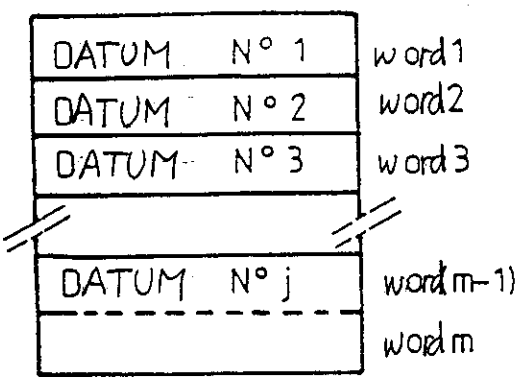

FIG.1B

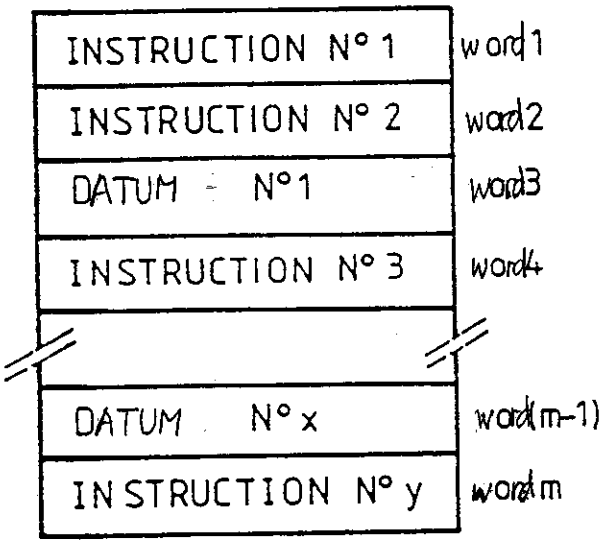

FIG.1C

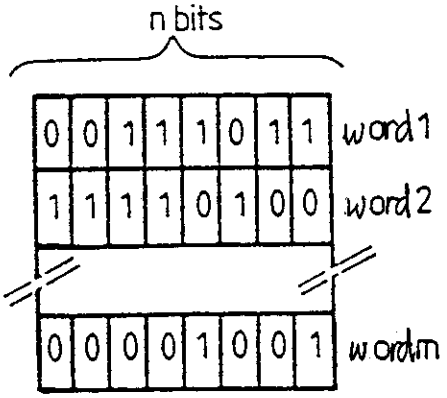

FIG.1D

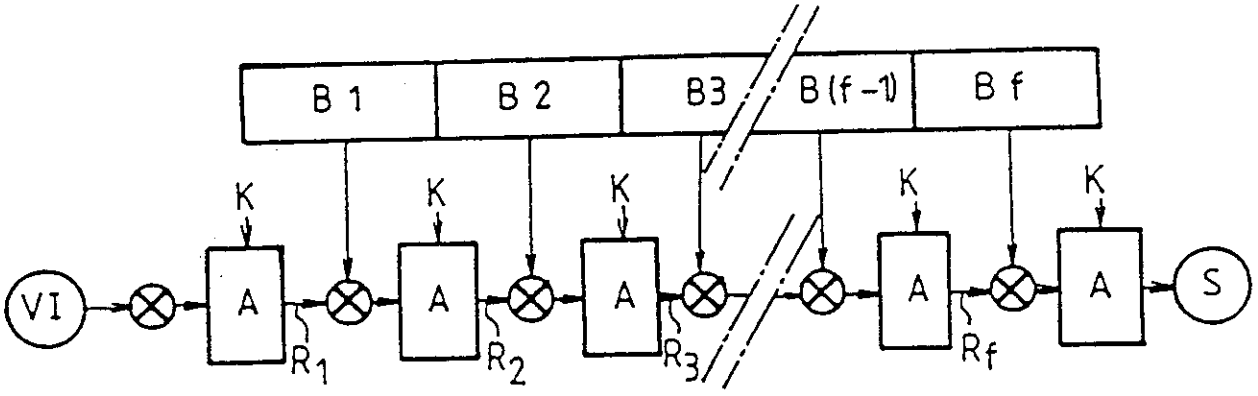

FIG.2

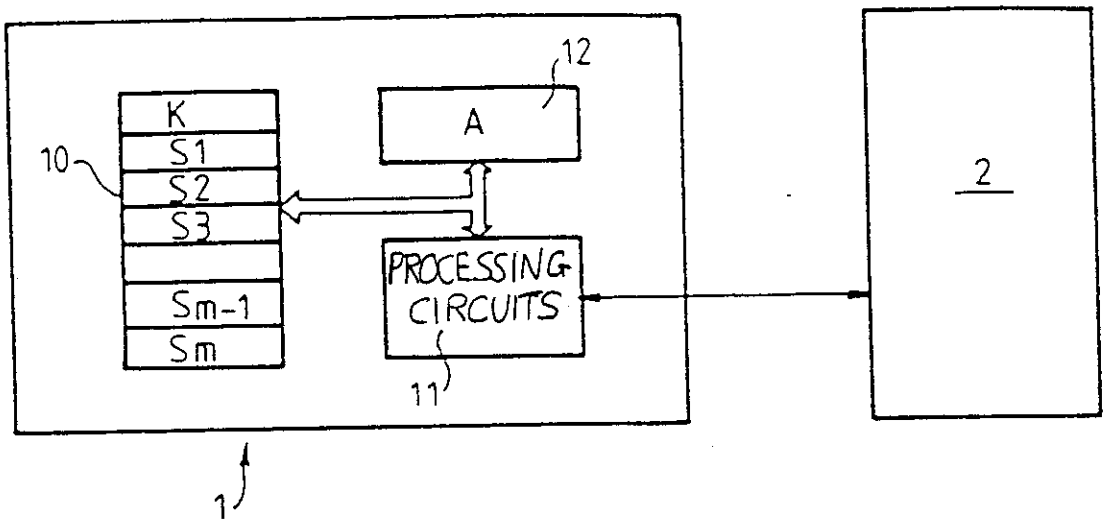
12
K
S1
S2
S3
Sm-1
Sm
10
A
PROCESSING CIRCUITS
11
2
1

FIG.3

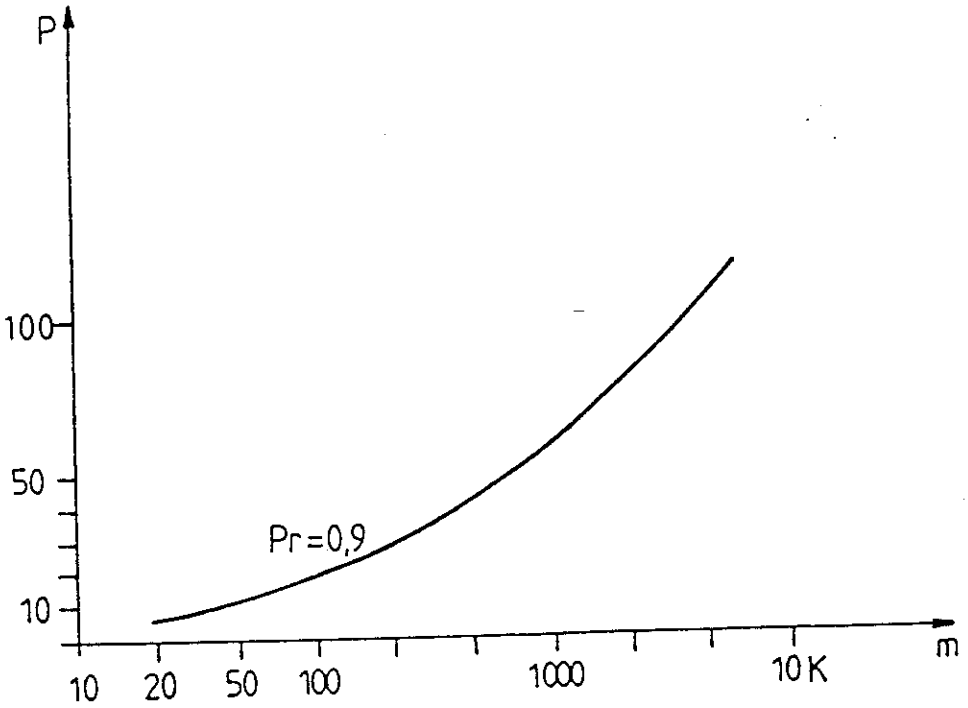
P
100
50
10
Pr=0,9
10
20
50
100
1000
10K
m

FIG.4

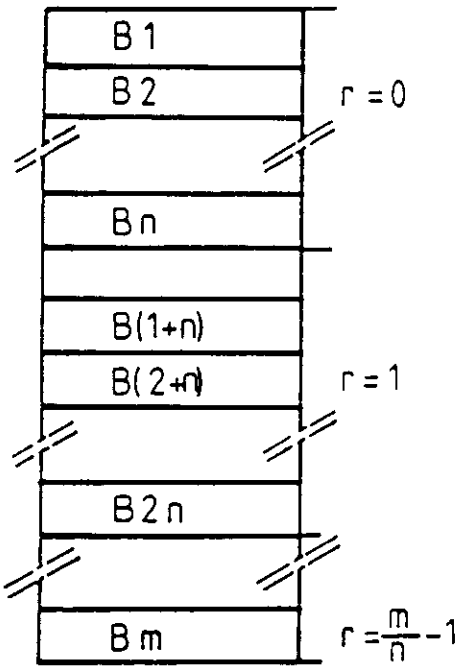

FIG.5

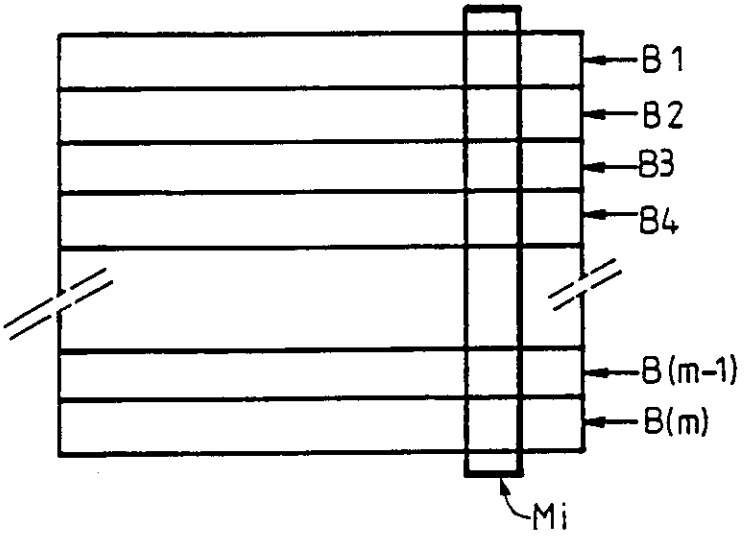

FIG.6

REGISTER ENTRY FOR EP0402210

European Application No EP90401476.8 filing date 01.06.1990

Application in French

Priority claimed:
06.06.1989 in France - doc: 8907429

Designated States BE CH DE DK ES FR GB GR IT LI LU NL SE AT

Title METHOD FOR VERIFYING THE INTEGRITY OF SOFTWARE OR DATA AND SYSTEM FOR IMPLEMENTING THIS METHOD.

Applicant/Proprietor
BULL CP8, Rue Eugène Hénaff BP 45, F-78190 Trappes, France
[ADP No. 50542315001]

Inventors
ANDRÉ OISEL, 5, Les Nouveaux Horizons, F-78990 Elancourt, France
[ADP No. 58082918001]

MICHEL UGON, 6, rue des Cépages, F-78310 Maurepas, France
[ADP No. 51267615001]

Classified to
G06F

Address for Service
BARON & WARREN, 18 South End, Kensington, LONDON, W8 5BU, United Kingdom
[ADP No. 00000281001]

EPO Representative
YVES DEBAY, BULL S.A. Industrial Property Department P.C.: HQ 8M006 B.P. 193.16, 121 avenue de Malakoff, F-75764 Paris Cédex 16, France
[ADP No. 50765312001]

Publication No EP0402210 dated 12.12.1990 and granted by EPO 30.08.1995.
Publication in French

Examination requested 30.04.1991

Patent Granted with effect from 30.08.1995 (Section 25(1)) with title METHOD FOR VERIFYING THE INTEGRITY OF SOFTWARE OR DATA AND SYSTEM FOR IMPLEMENTING THIS METHOD.. Translation filed 19.10.1995

27.01.1995 Notification from EPO of change of EPO Representative details from YVES DEBAY, BULL S.A. Industrial Property Department P.C.: HQ 8M006 B.P. 193.16, 121 avenue de Malakoff, F-75764 Paris Cédex 16, France
[ADP No. 50765312001]

to
BERNARD EDOUARD CORLU, Bull S.A. Direction de la Propriété Intellectuelle Bull S.A. Poste Courrier LV/59C18 68, Route de Versailles, F-78430 Louveciennes - B.P. 45, France
[ADP No. 62558424001]

Entry Type 25.14 Staff ID. RD06 Auth ID. EPT

17.03.1995 BARON & WARREN, 18 South End, Kensington, LONDON, W8 5BU, United Kingdom [ADP No. 00000281001]
registered as address for service
Entry Type 8.11 Staff ID. DD1 Auth ID. AA

12.05.1995 Notification from EPO of change of Applicant/Proprietor details from
BULL CP8, Rue Eugène Hénaff BP 45, F-78190 Trappes, France [ADP No. 50542315001]
to
BULL CP8, 68 route de Versailles, B.P. 45, F-78430 Louveciennes, France [ADP No. 50542315002]
Entry Type 25.14 Staff ID. RD06 Auth ID. EPT

09.06.1995 Notification from EPO of change of EPO Representative details from
BERNARD EDOUARD CORLU, Bull S.A. Direction de la Propriété Intellectuelle Bull S.A. Poste Courrier LV/59C18 68, Route de Versailles, F-78430 Louveciennes - B.P. 45, France [ADP No. 62558424001]
to
BERNARD EDOUARD CORLU, Direction de la Propriété Intellectuelle BULL SA, Poste courrier: LV59C18, 68 route de Versailles, F-78430 Louveciennes, France [ADP No. 61145033001]
Entry Type 25.14 Staff ID. RD06 Auth ID. EPT

02.11.1995 Patent Granted with effect from 30.08.1995 (Section 25(1)) with title METHOD FOR VERIFYING THE INTEGRITY OF SOFTWARE OR DATA AND SYSTEM FOR IMPLEMENTING THIS METHOD. Translation filed 19.10.1995
Entry Type 2.2 Staff ID. AMB1 Auth ID. F54

**** END OF REGISTER ENTRY ****

RENEWAL DETAILS

PUBLICATION NUMBER EP0402210

PROPRIETOR(S)

BULL CP8, 68 route de Versailles, B.P. 45, F-78430 Louveciennes, France

DATE FILED 01.06.1990

DATE GRANTED 30.08.1995

DATE NEXT RENEWAL DUE 01.06.1997

DATE NOT IN FORCE

DATE OF LAST RENEWAL 09.05.1996

YEAR OF LAST RENEWAL 07

STATUS PATENT IN FORCE

**** END OF REPORT ****