

12

DEMANDE DE BREVET D'INVENTION

A1

22 Date de dépôt : 29.02.00.

30 Priorité :

43 Date de mise à la disposition du public de la
demande : 31.08.01 Bulletin 01/35.

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

60 Références à d'autres documents nationaux
apparentés :

71 Demandeur(s) : *SAFELOGIC Société à responsabilité
limitée — FR.*

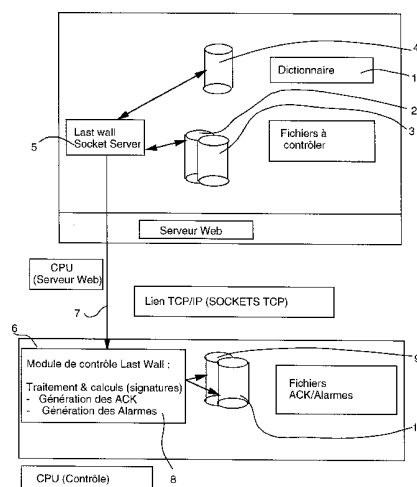
72 Inventeur(s) : POMEREU DE NICOLAS.

73 Titulaire(s) :

74 Mandataire(s) : BREESE MAJEROWICZ SIMONNOT.

54 PROCÉDE DE CONTROLE D'INTEGRITE DE FICHIERS INFORMATIQUES ET EQUIPEMENT POUR LA MISE
EN OEUVRE DE CE PROCÉDE.

57 Procédé de contrôle d'intégrité de fichiers informatiques d'un poste client comportant une étape d'enregistrement sur le poste client d'une valeur de hachage de référence associée à chaque fichier et des procédures de comparaison de ladite valeur de hachage enregistrée avec une valeur de hachage calculée périodiquement au moment du contrôle du fichier caractérisé en ce que les valeurs de hachage de référence sont enregistrées dans un dictionnaire local signé par un moyen cryptographique, le dictionnaire comprenant une liste de séquences formées par une adresse associée au fichier à contrôler et par la valeur de hachage prédéterminée, et en ce que la procédure de comparaison est réalisée par un équipement distant communiquant avec le poste client par le protocole TCP/ IP.



Procédé de contrôle d'intégrité de fichiers informatiques et équipement pour la mise en œuvre de ce procédé.

5 La présente invention concerne le domaine du contrôle d'intégrité de fichiers informatiques.

 On connaît, dans l'état de la technique, différentes solutions de contrôle d'intégrité de fichiers et de supervision des tentatives d'intrusions. Le brevet
10 US5961644 décrit par exemple un procédé pour tester l'intégrité de fichiers et le déclenchement d'une alarme consistant à simuler des attaques et à vérifier si l'attaque a été détectée.

 Le brevet PCT WO9940702 concerne un procédé
15 permettant de signer et de vérifier des données. La signature des données consiste à diviser un ensemble de données en paquets, à hacher ces données dans chacun des paquets de manière à produire un bloc de hachage comprenant des valeurs de hachage et à appliquer une signature
20 numérique audit bloc de hachage. Le procédé de vérification consiste à recevoir un paquet comprenant des données, un bloc de hachage et une signature numérique, à vérifier la signature numérique, à hacher les données de manière à produire des valeurs de hachage et à comparer les valeurs
25 de hachage aux valeurs du bloc de hachage.

 Le brevet US5944821 propose un autre procédé pour vérifier l'intégrité de données consistant à enregistrer dans une table locale une valeur de hachage et de comparer localement cette valeur.

30 Le but de la présente invention est de proposer un procédé permettant de vérifier en permanence et automatiquement l'intégrité des données d'un poste client, notamment d'un serveur de production, sans pour autant

perturber le fonctionnement du poste client par une consommation intempestive des ressources locales.

5 A cet effet, l'invention concerne selon son
acceptation la plus générale un procédé de contrôle
d'intégrité de fichiers informatiques d'un poste client
comportant une étape d'enregistrement sur le poste client
d'une valeur de hachage de référence pour chacun desdits
10 fichiers et des procédures de comparaison de ladite valeur
de hachage enregistrée avec une valeur de hachage calculée
périodiquement au moment du contrôle du fichier caractérisé
en ce que les valeurs de hachage de référence sont
enregistrées dans un dictionnaire local signé par un moyen
15 cryptographique, le dictionnaire comprenant une liste de
séquences formées par l'adresse du fichier à contrôler
associé à la valeur de hachage prédéterminée, et en ce que
la procédure de comparaison est réalisée par un équipement
distant communiquant avec le poste client par le protocole
TCP/IP, la procédure comportant une étape d'ouverture d'un
20 socket TCP/IP, une étape de lecture et de contrôle de la
signature dudit dictionnaire, une étape de comparaison de
la signature de référence de chaque fichier du dictionnaire
avec une signature du fichier correspond à l'adresse
associée, calculée dynamiquement, et une étape de
25 notification d'un message d'acquiescement ou d'alarme en
fonction du résultat respectivement positif ou négatif de
l'étape de comparaison, ledit message étant signé.

Avantageusement, le procédé comporte une étape
de signature sur le poste client du dictionnaire local un
moyen cryptographique, et en ce que la procédure de
30 contrôle comporte une étape de lecture et de contrôle de la
signature dudit dictionnaire.

Selon un mode de mise en œuvre préféré, la
procédure de contrôle est périodiquement interrompue.

De préférence, le procédé selon l'invention comporte une étape d'ouverture d'un premier socket TCP/IP pour les données relatives à la lecture du dictionnaire, et d'un deuxième socket TCP/IP pour les données relatives au balayage des fichiers dudit dictionnaire et au calcul des signatures.

Selon un mode de mise en œuvre particulier, il comporte une étape de génération d'une bi-clé cryptographique K_{RU1} , K_{PU1} pour la signature du dictionnaire et la signature de chaque fichier du dictionnaire. Avantageusement, il comporte en outre une étape de génération d'une bi-clé cryptographique K_{RUC} , K_{PUC} pour la signature des message d'acquiescement ou d'alarme.

Selon une variante particulière, il comporte une étape de propagation de la bi-clé cryptographique K_{RU1} , K_{PU1} entre une pluralité de postes clients.

Avantageusement, les bi-clés cryptographiques K_{RU1} , K_{PU1} et K_{RUC} , K_{PUC} sont identiques.

Selon un mode de réalisation préféré, l'étape d'enregistrement sur le poste client d'une valeur de hachage de référence associée à chaque fichier consiste à calculer pour ledit fichier un peigne par une extraction de données dudit fichier et à calculer la valeur de hachage dudit peigne, l'étape de comparaison consistant à comparer la signature de référence associée à chaque peigne avec une signature du peigne correspond à l'adresse associée au fichier à contrôler, calculée dynamiquement.

L'invention concerne également un équipement pour le contrôle d'intégrité de fichiers informatiques d'un poste client selon le procédé objet de la revendication 1 caractérisé en ce qu'il est constitué par une station de contrôle comportant des moyens de communication avec au moins un poste client par un réseau informatique, une mémoire l'enregistrement des données transitoires résultant

de l'étape de comparaison des signatures et des moyens pour la signature automatique des messages d'acquittement ou d'alerte.

5 L'invention sera mieux comprise à la lecture de la description d'un mode de réalisation non limitatif qui suit, se référant à un exemple de mise en œuvre de l'invention. Elle est illustrée par la figure unique représentant l'architecture d'une infrastructure pour la mise en œuvre du procédé.

10 La description qui suit concerne l'utilisation et l'administration d'un équipement pour la mise en œuvre de l'invention en mode client-serveur.

15 Le procédé est destiné à contrôler en permanence les signatures de fichiers d'un serveur de réseau, Intranet ou Internet. L'objectif est de solliciter les ressources du serveur contrôlé le moins possible, par le déport des opérations consommatrices en calcul/CPU d'un serveur contrôlé vers une machine dédiée, c'est-à-dire l'équipement de contrôle.

20 Le poste client (1) est constitué par un micro-ordinateur avec une mémoire, par exemple un disque dur, sur lequel sont enregistrés les fichiers (2, 3) à contrôler et une base de données (4) comprenant les signatures de référence et les adresses des fichiers (2, 3) à contrôler.

25 Le poste client (1) comprend par ailleurs un module de communication TCP/IP (5) permettant d'ouvrir des sessions sécurisées avec un poste contrôleur (6) par l'intermédiaire d'un réseau (7) intranet ou internet.

30 Ce poste contrôleur (6) comporte un calculateur pour les traitements du procédé de contrôle d'intégrité selon l'invention, ainsi que des fichiers (9, 10) pour mémoriser les résultats des contrôles.

Le procédé selon l'invention permet de réduire fortement le recours du calculateur du poste client lors du

process de contrôle. Ainsi, les ressources systèmes du serveur contrôlé sont très peu pénalisées.

Du point de vue des ressources systèmes, le procédé utilise le calculateur du poste client comme indiqué dans le tableau ci-dessous :

Type de consommation	Type d'opération	Partie de l'opération
I/O	N°1	Lecture du Dictionnaire Signé.
I/O	N°2	Lecture de chaque fichier à contrôler.
I/O	N°3	Ecriture du fichier Log
CPU	N°1	Calcul final de la signature cryptographique du Dictionnaire Signé.
CPU	N°2	Calcul final de la valeur hash du fichier à contrôler
CPU	N°3	Calcul final de la signature du fichier de log.

Le procédé selon l'invention utilise des techniques de cryptographie pour contrôler l'intégrité des fichiers :

- Chaque Administrateur dispose d'une paire de clés privée/publique RSA ou DSA. Nous verrons plus bas les différentes utilisations de la paire de clés.

- L'intégrité d'un fichier est contrôlée en calculant sa valeur de hachage ou « hash » avec l'algorithme SHA-1 et en comparant cette valeur avec une valeur stockée dans un fichier appelé le « Dictionnaire Signé ».

- Le « Dictionnaire Signé » contient :

- La liste des adresses des fichiers à contrôler à partir d'un Dictionnaire Texte.
- Accolé à chaque nom de fichier : la valeur « hash » SHA-1.

5 • Lorsque l'Administrateur génère un Dictionnaire Signé, le système lui demande la clé privée associée (protégée par une passphrase). Ce principe permet de signer le Dictionnaire Signé avec la clé privée de l'Administrateur.

10 • L'Administrateur lance le contrôle d'un Dictionnaire Signé. Le process de contrôle effectue les étapes :

- Contrôle de la signature du dictionnaire en utilisant la clé publique de l'Administrateur.

15 • Contrôle de l'intégrité de chaque fichier du dictionnaire avec calcul dynamique de son SHA-1 et comparaison avec la valeur stockée.

- Si le contrôle est OK : Génération d'un ACK (acquittance) signé, avec la clé privée de l'Administrateur.

20

- Si une anomalie quelconque est détectée : le process émet un NACK (message d'Alarme) sous forme de fichier Texte ou HTML et d'émission d'e mail(s).

- L'Administrateur a la possibilité à tout moment :

25

- D'arrêter le processus de contrôle d'un Dictionnaire Signé.
- D'arrêter tous les process de contrôle, c'est à dire de stopper Lastwall.

30 **Cas d'émission d'une Alarme**

Le procédé émet une Alarme (fichier Texte/HTML et e mail(s)) signée pour les tentatives d'attaque suivantes :

- Modification du contenu ou suppression du Dictionnaire Signé.
- Modification du contenu ou suppression d'un des fichiers à contrôler.

5 **gen_keys : Génération d'une paire de clés**

La génération d'une paire de clés privée/publique servira :

- A signer les dictionnaires textes (Administrateur) .
- 10 • A contrôler la validité de signature d'un dictionnaire (process Lastwall).

La génération d'au moins une paire de clés est obligatoire. On a les caractéristiques :

- Chaque paire de clés est associée à un E
15 mail.

- Les clés sont stockées dans le répertoire clw_keys_dir défini dans le fichier lastwall.ini.

- Les clés privées sont chiffrées avec une « passphrase » avant le stockage sur disque. Il est
20 indispensable de rappeler cette passphrase pour toute opération qui nécessite l'utilisation de la clé privée.

- Une interface demande la génération d'une valeur « seed » ou d'initialisation. Cette valeur est
25 utilisée pour initialiser un générateur de nombres pseudo aléatoires.

- Si la passphrase n'est pas passée en paramètre, elle est demandée, avec confirmation, par une interface graphique.

30 Lors de la *première utilisation* sur une machine, gen_keys lance un programme destiné à collecter des nombres aléatoires.

a) Génération des nombres aléatoires sous Unix.
Le programme demande de frapper lentement sur la touche <entrée> 128 fois : c'est la différence de temps entre les

frappes qui sert à fabriquer la séquence de nombres aléatoires. Veuillez espacer suffisamment longtemps (au moins 1 seconde) vos frappes sur la touche<entrée> afin de fabriquer une séquence de nombres réellement aléatoires.

5 b) Génération des nombres aléatoires sous Windows NT. Une fenêtre s'affiche avec un indicateur de progression. Il suffit d'appuyer au hasard et lentement des touches pendant quelques secondes sur le clavier pour créer la séquence aléatoire.

10 **Création d'un Dictionnaire Texte**

add_files permet de rajouter facilement des fichiers à un « Dictionnaire Texte ». Le dictionnaire texte est la source qui permet de générer un « Dictionnaire Signé ».

15 Un Dictionnaire Texte est donc composé de noms de fichiers, avec les caractéristiques :

- Les noms de fichiers sont complets et incluent unité et directories.
- Chaque nom de fichier occupe une ligne.

20 Voici un exemple de Dictionnaire Texte sous Windows NT:

```
c:\aroot\lastwall\exemple1.txt
c:\aroot\lastwall\exemple2.txt
c:\aroot\lastwall\exemple3.txt
```

25 La création du Dictionnaire texte c:\aroot\lastwall\lwdic_text.txt est effectué par la commande :

```
add_files c:\aroot\lastwall\lwdic_source.txt
c:\aroot\lastwall\lwdic_text.txt
```

30 A p r è s e x é c u t i o n ,
c:\aroot\lastwall\lwdic_text.txt contient :

```
c:\aroot\lastwall\exemple1.txt
c:\aroot\lastwall\exemple2.txt
c:\aroot\lastwall\exemple3.txt
```

gen_signed_dic permet de générer un

5 Dictionnaire Signé à partir :

- D'un Dictionnaire Texte précédemment créé.
- D'une clé privée Lastwall, identifiée par le couple :
- E mail identifiant la clé privée.
- 10 • Passphrase de protection par chiffrement de la clé privée.

Un Dictionnaire Signé contient la liste des fichiers du Dictionnaire Texte, avec les caractéristiques suivantes :

- 15 • A chaque fichier est accolée sa valeur SHA-1 sous forme hexadécimale.

- Le Dictionnaire est « signé » avec la clé privée correspondant à un e mail email@domain.net Cela signifie que le Dictionnaire Signé contient une zone qui permet à un process qui connaît la clé publique de :

20 1) Vérifier que le Dictionnaire a bien été signé par email@domain.net .

2) Vérifier que le contenu du Dictionnaire n'a pas été altéré par un tiers (intégrité).

25 Usage :

```
gen_signed_dic <option switches> <text dictionnary> <signed
dictionnary> <error file> <signing e mail> [passphrase]
```

Avec :

- option switches : <+d[r]|-d>

30 <+d[r]|-d> : expand directories [r]=recurse | do not
expand directories

- text dictionnary : full name of input text dictionnary
- signed dictionnary: full name of signed dictionnary to create
- error file : file containing reporting errors
- signing e mail : e mail corresponding to a LW private key
- passphrase : e mail private key passphrase

Exemple :

Voici un exemple de Dictionnaire Signé correspondant au Dictionnaire Texte décrit dans le script précédent (gen_signed_dic) :

```
<!--//BEGIN LW SIGNED MESSAGE-->
// Lastwall (R) Copyright (C) SafeLogic 2000
// c:\aroot\lastwall\lwdic_sign.txt Signed Dictionary
Creation at 17/2/2000 19:33:20
//
//BEGIN LW FIELDS
key_id=info@safelogic.com
key_algo=MD5/RSA/PKCS#1
crypto_algo=Blowfish/CBC/PKCS5Padding
crypto_iv=8215A2C54127E18A
//END LW FIELDS
//
LW_FILE:
c:\aroot\lastwall\exemple1.txt=ABD77AAC92E4133F1205FD1AC56E
DD85F4509BB9 LW_LENGTH: 19
LW_FILE:
c:\aroot\lastwall\exemple2.txt=ABD77AAC92E4133F1205FD1AC56E
DD85F4509BB9 LW_LENGTH: 19
LW_FILE:
c:\aroot\lastwall\exemple3.txt=8CC1FA4C1FA603725982DD3D8BCB
F26409681EA7 LW_LENGTH: 19
<!--//BEGIN LW SIGNATURE
```

```

19DFB548BAEA0579A5B4E23309135BCC524C9765AAB30EAB7CBF2226B29
4E25E4E13A2310CECB94C403B9262C568D3B814ECC95ADFA9821DFDDB3B
7628F8BEC9AFA1BEE8FCE1B4422F505C663BD1FDEBA6D638424282569DD
7A82E7B1A247EDBF2DA640AF87502593E0D22A8F923B9559C5575069E5F
F1C55C9D663D534AC4DC
//END LW SIGNATURE-->

```

Note : il s'agit d'un Dictionnaire Signé non chiffré (configuration par défaut).

Le script `disp_crypt` permet d'afficher les Dictionnaires Signés qui sont cryptés au moment de la génération.

1.1 `disp_crypt` : affichage d'un Dictionnaire Signé crypté
Si le Dictionnaire Signé a été crypté à la génération, `disp_crypt`, permet de l'afficher en mode décrypté. Il faut lancer le process en passant comme paramètres :

- Le nom du Dictionnaire Signé.
- Le nom de l'e mail de signature/chiffrement.
- La passphrase associée à l'e mail de signature/chiffrement.

Usage :

```

disp_crypt <signed dictionary> <signing e mail>
[passphrase]

```

Avec :

- signed dic. : signed dictionary to display
- signing e mail : e mail corresponding to a LW private key
- passphrase : e mail private key passphrase

Note :

`disp_crypt` n'est à utiliser que si le paramètre `clw_crypt_sign_dic` de votre fichier d'initialisation est valorisé à true.

1.2 `check_dic` : lancement d'un process de contrôle

check_dic permet de lancer un process de contrôle en une fois ou continu d'un dictionnaire, en précisant :

- Une série de switchs pour indiquer :
- Contrôle en boucle ou en un seul passage.
- 5 • Alerte immédiate à la première erreur ou alerte à la fin du parcours du Dictionnaire Signé.
- Affichage ou non du contrôle sur la console.
- Trace de l'activité ou non dans un fichier de trace.
- L'adresse IP du serveur contenant le Dictionnaire Signé
- 10 - Version « Mono-Server » ou utiliser la valeur « localhost » qui indique le host local.
- Version « Full » : utiliser l'adresse IP de la machine du Socket Server.
- Le nom du Dictionnaire Signé.
- 15 • L'adresse IP d'un serveur de Logging qui contient le Fichier de Logging Dynamique.
- Version « Mono-Server » : utiliser la valeur localhost qui indique le host local.
- Version « Full » : utiliser l'adresse IP de la machine
- 20 du Socket Server.
- Un fichier de « Logging Dynamique » Texte ou HTML qui est rafraîchi en permanence.
- Un intervalle en secondes, c'est à dire un mode « sleep » du process entre deux contrôles du Dictionnaire.
- 25 • Le nom complet, avec le chemin d'accès, du fichier d'initialisation à utiliser. Cette option permet d'utiliser des « clones » du fichier d'initialisation par défaut, avec des valeurs différentes pour chaque dictionnaire à contrôler.
- 30 La valeur « default » permet d'utiliser le nom de fichier ini par défaut. (lastwall.ini).
- L'e mail correspondant à la clé privée de signature.

- La passphrase correspondant à la clé privée de signature du process/dictionnaire.

Usage:

```
check_dic <option switches> <host> <signed dictionary>
<logging host>
<logging file> <interval> <ini file> <signing e mail>
[passphrase]
```

5

Avec :

```
- option switches    : <-l|+l> <-c|+c> <-s|+s> <-f|+f>
  +l|-l : continuous checks in loop mode | one pass check
  +c|-c : continue | stop when file integrity error
```

10

detected

```
  +s|-s : trace check on screen | do not trace check on
screen
```

```
  +f|-f : trace activity in file | do not trace activity in
file
```

15

```
- host                : localhost or IP address of Signed
Dic. location
```

```
- signed dictionary: Signed Dictionary containing the
files to check
```

20

```
- logging host        : localhost or IP adress of logging
file location
```

```
- logging file        : full name of logging file
```

```
- interval            : interval in seconds between two
```

Dictionary Check

```
- ini file            : special ini file to use or "default"
```

25

for normal usage

```
- signing e mail      : e mail corresponding to a LW private
key
```

```
- passphrase          : e mail private key passphrase
```

Notes :

1) Paramètres restreints

En version « Mono-Server », les paramètres suivants sont restreints :

- host : Seule la valeur localhost est autorisée.
- 5 - logging host : Seule la valeur localhost est autorisée.

2) Signature des e mails

Les e mails d'Alerte sont signés. Il est donc possible de vérifier leur authenticité avec le script :

```
check_sign <signed_file >
```

10 Avec :

- signed file : file or e mail content signed by Lastwall
- Pour vérifier l'authenticité d'un e mail, il suffit de copier son contenu (corps) dans un fichier et d'appliquer check_sign au fichier créé.

15 check_sign affiche true si la signature de l'e mail est correcte.

3) Cas de déclenchement d'une Alarme

Une Alerte se déclenche (terminal et e mail) dans les cas suivants :

- 20 • Modification ou suppression d'un Dictionnaire Signé.
- Modification ou suppression d'un des fichiers référencés à contrôler dans un Dictionnaire Signé.

1.3 check_stop : arrêt d'un process de contrôle

L'arrêt d'un process de contrôle doit passer par check_stop afin de signer et authentifier la demande d'arrêt.

- 25 • Une Alarme console sera déclenchée.
- Un e mail d'Alarme sera envoyé aux destinataires spécifiés dans check_dic.

check_stop permet de préciser :

- 30 • L'adresse IP du serveur qui contient le Dictionnaire Signé dont il faut arrêter le process (mode client/serveur).
- Version « Mono-Server »: utiliser la valeur localhost qui indique le host local.

- Version « Full » : utiliser l'adresse IP de la machine du Socket Server.
- L'arrêt des contrôles pour tous les Dictionnaires Signés, ou pour un Dictionnaire précis.

5 Usage :

```
check_stop <option switches> <l|r> <host> <signed
dictionary|all> <signing e mail> [passphrase]
```

Avec :

- option switches : <-s|+s> <-f|+f>
- 10 +s|-s : trace stop on screen | do not trace stop on screen
- +f|-f : trace activity in file | do not trace activity in file
- l|r : (l)ocal stop or (r)emote Socket
- 15 Server stop
- host : localhost or IP address of Signed Dic. location
- signed dic.|all : check to stop: signed dictionary | all dictionaries
- 20 - signing e mail : e mail corresponding to a LW private key
- passphrase : e mail private key passphrase
- 1) Paramètres restreints
- En version « Mono-Server », les paramètres suivants sont
- 25 restreints :
- l|r : Seule la valeur l est autorisée.
- host : Seule la valeur localhost est autorisée.

30 Le procédé selon une variante de réalisation se décompose en deux procédures distinctes dans leur rôle un process de contrôle :

- Sous-process « client » : chargé d'effectuer les opérations de calcul consommatrices en CPU sur une machine dédiée.
- Sous-process « serveur » : il tourne en tâche de fond sur le serveur dont on veut surveiller les fichiers et communique avec le sous-process client pour les opérations de I/O sur les fichiers.

Nous appellerons Monitor un sous-process « client » et Socket Server un sous-process « serveur ».

Le tableau suivant décrit les rôles des deux composants de Lastwall Client-Serveur dans le cas d'un process de contrôle :

Composant	Rôle / Activité
Monitor	• Etablit un lien persistant et synchrone en mode Sockets TCP/IP avec Socket Server. • Lit les contenus des fichiers sur une Socket TCP/IP. • Contrôle de signature du Dictionnaire Signé. • Calcul de la valeur « hash » (ou de hachage) de chaque fichier à contrôler. • Ecriture et signature du fichier de Log.
Socket Server	• Etablit un lien persistant et synchrone en mode Sockets TCP/IP avec Monitor • Ecrit les contenus des fichiers à surveiller sur une Socket TCP/IP.

Le tableau suivant décrit les caractéristiques systèmes de chaque composant :

Composant	Rôle système & consommation
Monitor	• « Calculateur » cryptographique. • Forte consommation de CPU.
Socket	• Diffusion de fichiers (broadcast).

Server	•Fonctionnement en mode multi-thread. •Consommation de CPU très réduite.
--------	---

Selon un exemple de mise en œuvre, une machine de contrôle avec Monitor est reliée par une Socket TCP/IP à un serveur à contrôler :

5 1.3.1 Instances multiples de Monitor

Dans l'exemple ci-dessous, une machine dédiée sous Linux surveille à la fois :

- Un serveur sous Windows NT.
- Un serveur sous Linux.

10 Deux instances indépendantes de Monitor s'exécutent sous la machine de contrôle. Il n'y a pas de limite théorique au nombre de serveurs que peut surveiller une machine de contrôle qui exécute des process Monitor.

1.3.2 Multi-threading de Socket Server

15 Dans ce cas, plusieurs Socket Server s'exécutent de façon multi-threadée sous le serveur à surveiller :

- Il n'y a qu'un seul processus Socket Server de lancé.
- Chaque client Monitor lance un sous-thread « fils » du processus père Socket Server.

20 •Cette configuration fonctionne avec un ou plusieurs processeurs clients de surveillance.

Le mode multi-threading permet de « discriminer » des process de surveillances sur un serveur, en particulier de jouer sur les fréquences de contrôle.

25 •1) un premier process lastwall qui surveillera très régulièrement et avec une pause rapide de quelques secondes les fichiers systèmes vitaux de /etc ou c:\wint\system32 pour éviter les attaques très graves sur root avec un rootkit.

30 •2) Un deuxième process, avec une fréquence d'activité moins élevée, sera consacré à la surveillance des programmes Web (CGI, ASP, Servlet, etc.) les plus importants.

- 3) Un troisième process avec un temps de pose beaucoup plus élevé (plusieurs minutes ou même heures) surveillera les fichiers applicatifs moins sensibles de type pages HTML de second niveau ou images non vitales, etc.

5 1.4 Mode opératoire en Client-Serveur

Nous distinguerons quatre étapes successives dans le mode opératoire :

- 1) Préparation des process de contrôle.
- 2) Lancement du Socket Server
- 10 •3) Lancement et arrêt des process de contrôle.
- 4) Arrêt du Socket Server.

1.4.1 Préparation des process de contrôle

1.4.1.1 Scripts

15 La préparation de process est constituée des scripts et étapes suivantes décrits dans le document [lw_doc_admin_gene]:

Script	Etape / description du script
gen_userid	Génération d'un couple userid password
gen_keys	Génération d'une paire de clés
add_files	Création d'un Dictionnaire Texte
gen_signed_dic	Génération d'un Dictionnaire Signé
disp_crypt	affichage d'un Dictionnaire Signé crypté

20 Il est indispensable de transférer sur la (ou les) machine(s) de monitoring les jeux de clés privées/publiques de Lastwall générées sur le Serveur à contrôler.

Il existe un et un seul jeu de clés Lastwall à utiliser :

1) Le jeu de clés privées/publiques de références est celui qui correspond au serveur à surveiller (machine avec Socket Server).

25

2) Il faut transférer ce jeu de clés sur toutes les machines de monitoring (machines avec Monitor).

3) S'il existe plusieurs serveurs à surveiller : il faut créer un jeu de clés unique sur un des serveurs et le transférer sur les autres.

Il existe deux clés associées à un e mail et à une passphrase :

- Une clé publique
- Une clé privée chiffrée (avec une clé privée dérivée de la valeur de la passphrase).

Le transfert du jeu de clés entre ordinateurs ne pose pas de problème de confidentialité car la clé privée est chiffrée avec un algorithme symétrique fort et une clé longue de 128 bits.

Les nomenclatures des deux fichiers-clé publique et clé privée sont :

```
<clw_keys_dir>RSA_<e_mail_de_la_clé>.pkf
<clw_keys_dir>RSA_<e_mail_de_la_clé>.skf
```

Avec :

<clw_keys_dir> :

directory de localisation des clés correspondant au champ clw_keys_dir de clw_base_unix.ini ou clw_base_wnt.ini.

<e_mail_de_la_clé> :

Paramètre <identity e mail> du script de génération de clé gen_keys.

Rappel des valeurs par défaut de clw_keys_dir :

- Unix : /home/lastwall/keys/
- Windows NT : c:\aroot\lastwall\keys\

Exemple :

Si l'e mail email@domain.net sert d'e mail d'identité du jeu de clés, on aura par défaut les deux fichiers clés :

OS	Fichiers clés pour email@domain.net
----	-------------------------------------

Unix	/home/lastwall/keys/RSA_email@domain.net.pkf /home/lastwall/keys/RSA_email@domain.net.skf
Windows NT	c:\aroot\lastwall\keys\RSA_email@domain.net.pkf c:\aroot\lastwall\keys\RSA_email@domain.net.skf

1.4.2 start_server : lancement du Socket Server

Usage :

```
start_server [+v]
```

5

Avec :

- +v : verbose mode

Notes :

10 Le mode +v (verbose on) permet d'afficher en détail l'activité de start_server. En version 1.01, +v est surtout un mode de debug.

Après le lancement, le Socket Server affiche les lignes suivantes et se met en attente des connexions des clients

Monitor :

15

```
Lastwall (R) V1.00A 23/11/1999 (jdk 1.1.8) Copyright (C)
SafeLogic, 1999
http://www.safelogic.com
Lastwall Socket Server - ** multi thread version **
```

20

Rappel : Il ne faut lancer qu'une seule fois start_server, même si plusieurs machines se connectent en mode Monitor.

1.4.3 Lancement et arrêt des process de contrôle

La troisième et quatrième étape se déroulent entièrement sur la ou les machine(s) de contrôle qui exécutent les process clients Lastwall Monitor.

Il s'agit des deux scripts :

Script	Etape / description du script
check_dic	Lancement d'un process de contrôle
check_stop	Arrêt d'un processus de contrôle en client-serveur.

Les modes opératoires sont identiques à ceux de Lastwall Mono-Server et décrits dans [lw_doc_admin_gene]. Se référer à cette documentation pour la syntaxe et le fonctionnement de gen_userid et check_stop.

1.4.3.1 check_dic : lancement d'un process Monitor de contrôle

Usage:

Cf. [lw_doc_admin_gene].

Equivalent Web :

Cf. [lw_doc_admin_gene].

Notes :

Seul change le deuxième paramètre en mode client-serveur :

host	: localhost or IP address of Signed Dic.
------	--

location

Il prend la valeur de l'adresse IP de la machine où est lancé le Socket Server.

Exemple : 192.168.0.1

Le logging s'effectue sur la machine de contrôle, i.e la machine de lancement de Monitor et du script check_userid. En version 1.01A, le paramètre numéro 4 logging host prend toujours la valeur « localhost » :

- logging host : localhost or IP adress of logging
file location

1.4.3.2 check_stop : arrêt d'un process Monitor de contrôle

Usage:

5 Cf. [lw_doc_admin_gene].

Equivalent Web :

Cf. [lw_doc_admin_gene].

Notes :

Seul change le troisième paramètre en mode client-serveur
10 :

- host : localhost or IP address of Signed
Dic. location

Il prend la valeur de l'adresse IP de la machine dont on
veut stopper proprement le Socket Server.

15 Exemple : 192.168.0.1

1.4.4 check_stop : arrêt du Socket Server

L'arrêt du Socket Server sur une machine se fait en
utilisant le process check_dic de façon particulière.

20 Usage :

check_stop <option switches> "r" <host> "all" <signing e
mail> [passphrase]

Avec :

- option switches : <-s|+s> <-f|+f>

25 +s|-s : trace stop on screen | do not trace stop on
screen

+f|-f : trace activity in file | do not trace activity in
file

- r : (r)emote Socket Server stop

30 - host : IP address of Signed Dic. location

- all : check to stop all dictionnaires

- signing e mail : e mail corresponding to a LW private key

- passphrase : e mail private key passphrase

Exemple :

- 5 L'arrêt du Socket Server de la machine 192.168.0.1 se fait avec la commande :

```
check_stop +s+f r 192.168.0.1 all email@domain.net
```

Notes :

10

Tous les process Monitor de toutes les machines connectées seront automatiquement arrêtés si un Socket Server est arrêté.

2 INTERFACE HTML D'ADMINISTRATION

2.1 Objectifs

L'interface HTML a pour but d'offrir un mode opératoire graphique pour chaque script décrit dans [lw_doc_admin_gene] et cette documentation.

Par exemple, le script gen_userid de génération du dictionnaire signé sera remplacé par une page HTML avec des champs à remplir et un CGI associé.

Les traitements de base appelés sont les mêmes que pour les scripts ; seule change l'interface.

L'interface HTML est utilisable pour l'administration et l'exécution de Lastwall en mode mono-serveur et en mode client-serveur.

L'interface HTML offre deux avantages importants par rapport au mode « scripting » :

- 1) Possibilité de déporter via un réseau TCP/IP non sécurisé l'administration sur tout hardware qui supporte l'utilisation d'un navigateur en HTML 3.2.
- 2) Mise en œuvre d'une interface graphique.

2.2 Configuration de lastwall.ini

Le fichier lastwall.ini comporte des paramètres spécifiques à l'interface HTML.

Se référer à [lw_doc_admin_gene] pour la localisation de lastwall.ini sur votre disque dur. Voici la liste des paramètres de lastwall.ini à mettre à jour pour utiliser l'interface HTML :

Paramètre	Rôle	Valeur par défaut
-----------	------	-------------------

clw_screen	Largeur de l'interface HTML de l'Administrateur : 800 ou 600 pixels.	800
Directories et fichiers virtuels (HTTP)		
clw_cgi- bin_dir	Adresse Web des CGI en Java	/cgi-bin/
clw_servlet_d ir	Adresse Web des Servlet Java	/servlet/
clw_alarm_way	Adresse Web du Signal sonore d'Alarme	/lastwall/ ringout.wa v
clw_img_alias _dir	Adresse Web du dossier images des pictos	/lastwall/ images
Paramètres SMTP/POP3		
clw_email	E mail du WebMaster	A modifier
	Gestion des Process Lastwall	
clw_elapsed_m ax	Durée maximum autorisée en secondes d'inactivité du process Lastwall.	600

2.3 Configuration des droits de l'utilisateur Serveur Web
le Serveur Web doit avoir les droits suivants sur les
directories correspondants aux paramètres suivants

5 (notation des droits en « mode Unix ») :

Paramètre de directory physique	Valeur par défaut Unix	Valeur par défaut Windows NT	Dro its
clw_swap_di r	/home/lastwa ll/swap/	c:\aroot\lastwa ll\swap\	+rw -

clw_keys_dir	/home/lastwall/keys/	c:\aroot\lastwall\keys\	+rw -
clw_random_bin_dir	/home/lastwall/swap/	c:\aroot\lastwall\swap\	+rw -
clw_trace_dir	/home/lastwall/swap/	c:\aroot\lastwall\swap\	+rw -

Note Unix : Il est préférable d'attribuer par un chown ces directories à l'utilisateur serveur Web.

- 5 Le Server Web doit aussi avoir un droit en lecture et écriture sur le fichier lastwall.ini, si vous souhaitez le modifier en mode HTML.

Fichier	Localisation par défaut Unix	Localisation par défaut Windows NT	Droits
lastwall.ini	/com/safelogic/servlet/	c:\com\safelogic\servlet\	+rw -

2.4 Mapping entre Serveur HTTP et directories physiques

- 10 Si on prend comme références :

- Pour le stockage des objets HTML le répertoire /home/lastwall/html/ ou c:\aroot\lastwall\html\ défini par défaut à l'installation (Cf. votre Documentation d'Installation Lastwall)

- 15 • Les valeurs par défaut des champs clw_img_alias_dir et clw_alarm_wav.

On doit alors avoir le mapping suivants sur le Serveur HTTP sous Unix ou Windows NT :

Directories Unix	Alias http	Commentaire
/home/lastwall/html/	/lastwall/	Répertoire de stockage du

		fichier son du paramètre : clw_alarm_wav
/home/lastwall/html/images/	/lastwall/images/	Répertoire de stockage des images du paramètre : clw_img_alias_dir

Directories Windows NT	Alias http	Commentaire
c:\aroot\lastwall\html\	/lastwall/	Répertoire de stockage du fichier son du paramètre : clw_alarm_wav
c:\aroot\lastwall\html\images\	/lastwall/images/	Répertoire de stockage des images du paramètre : clw_img_alias_dir

Utilisation de l'interface HTML

Les URLs d'appel de l'interface HTML sont :

OS	URL d'appel de l'interface HTML
Unix	http://localhost/lastwall/cgiu_lwmain2.htm
Windows NT	http://localhost/lastwall/cgiw_lwmain2.htm

5

L'interface HTML est auto-documentée. Voici une capture d'écran du menu général :

A chaque script décrit dans [lw_doc_admin_gene] correspond une ligne d'un sous-menu d'accès HTML.

On a la table de correspondance :

Script	Sous - Menu & Ligne d'accès
gen_userid	<i>Pas de correspondance pour des raisons de sécurité.</i>
gen_keys	Clés Création
add_files	Dictionnaire Texte Création auto.
gen_signed_dic	Dictionnaire Signé Génération
disp_crypt	Dictionnaire Signé Affichage
check_dic	Contrôles Lancement
check_stop	Contrôles Arrêt

Une variante de mise en œuvre préférée de l'invention consiste à échanger entre le client et le serveur non pas l'intégralité des fichiers à vérifier, mais un extrait seulement de ces fichiers.

L'objectif visé par la réalisation de cette variante mettant en œuvre le calcul d'un peigne est de réduire le flot de données transférées entre le client (« Monitor ») et le serveur Lastwall (« Socket Server »).

Pour cela on n'utilisera qu'une partie des données de chacun des fichiers à signer et à contrôler.

Cette sélection sera faite par le client avant envoi vers le serveur.

Ce module doit prendre en compte les aspects suivants :

- Assurer une faible consommation des ressources du client.
- Etre rapide.
- Préserver la sécurité (séquence de données non prévisible).
- Permettre de sauvegarder et de « transporter » aisément le peigne.

Comme on laisse des zones « de blanc » dans le fichier (parties non sélectionnées), un attaquant pourrait modifier ces parties sans être détecté. Il est donc primordial que les parties non sélectionnées ne soient pas prévisibles.

Définition d'un peigne

On souhaite transporter moins de données concernant le fichier à signer mais on ne peut pas faire de compression sinon on chargerait la machine serveur (celle qui exécute le Socket Server) en calcul, ce que l'on veut justement éviter par déportation en fonctionnant en client / serveur.

On va donc se contenter d'accéder au fichier et de lire seulement quelques parties de son contenu. On doit cependant à chaque lecture extraire le même contenu si l'on veut que la signature soit juste.

On va utiliser un « peigne » pour lire ce fichier, c'est-à-dire certaines valeurs de début de séquence de lecture et une taille de bloc lu fixe. On extraira ainsi toujours les mêmes données du fichier.

Ce qui va constituer un peigne a été évoqué précédemment :

- Une séquence de positions de départ pour les lectures ;
- Une taille de bloc fixe de lecture à partir de ces positions ;
- Le taux de données extraites (pourcentage).

Les différents calculs à effectuer pour déterminer un peigne concernent tout d'abord, la taille du buffer contenant les données extraites :

5 $Taille_totale_extraits = Taille_globale_fichier \times$
Taux_requis / 100 ;

Pour déterminer la séquence de lecture, on va ensuite avoir besoin des valeurs suivantes :

$Nombre_lectures_requis = Taille_totale_extraits /$
Taille_bloc ;

10 $Nombre_lectures_possible = Taille_globale_fichier /$
Taille_bloc ;

On devra donc effectuer *Nombre_lectures_requis* tirages dans un ensemble à *Nombre_lectures_possible* éléments.

15 Ces tirages devront être effectuées aléatoirement et les valeurs ne seront pas triées afin de garantir une meilleure sécurité (données moins prévisibles).

CLASSES JAVA DU MODULE « PEIGNE »

Le peigne : CSaComb

20 Pour pouvoir aisément utiliser et manipuler le peigne, nous avons créé une classe peigne : CSaComb. Elle sert au stockage des informations d'un peigne.

On va donc y retrouver des fonctions set (regroupées en fait sous forme d'un constructeur) et des fonctions get pour les attributs, ainsi que des méthodes permettant de manipuler un peigne sous forme de String (constructeur appelant setFromValue() et méthode toString() surchargée).

Le générateur de peigne : CSaCombGenerator

30 Il s'agit de l'équivalent d'une « Factory » qui permet de créer un nouveau peigne pour un fichier.

Le constructeur permet d'initialiser le générateur de nombres aléatoires pour le fichier spécifié.

Notons que le générateur n'est valide que pour le fichier pour lequel il est créé.

Le générateur permet de créer un nouveau peigne grâce à la méthode `getComb()`, celle-ci appelant
5 `generateSequence()` pour créer la séquence de positions aléatoires.

Le lecteur : `CSaCombReader`

Ce lecteur permet de lire un fichier à l'aide d'un peigne. C'est à ce niveau que va donc se passer
10 l'extraction.

Cette classe et ses méthodes de lectures ne doivent être utilisées qu'avec un peigne généré pour le fichier en question par le générateur.

Il existe deux modes de lecture :

15 Lecture de tous les extraits et renvoie d'un buffer contenant le tout ;

Lecture extrait par extrait (streaming).

L'intérêt est de pouvoir adapter la lecture à l'utilisation que l'on va en faire. En effet, dans le cas
20 d'une émission de ces données (vocation première de ce module), on va pouvoir lire et envoyer en continu plutôt que de tout lire puis d'envoyer. Ceci accélérera le processus et évitera une bufferisation.

Outre le constructeur, on dispose donc d'une
25 méthode de lecture globale, `readAll()`, et d'un jeu de méthodes pour la lecture en continu :

`prepareStream()` pour préparer le flot ;

`isAvailable()` pour s'assurer qu'il y a encore au moins une lecture à faire ;

30 `streamingReadNext()` pour lire le prochain extrait.

Enfin la méthode `close()` doit être appelée dans tous les cas pour fermer le Reader.

Cette classe s'appuie sur la classe RandomAccessFile pour effectuer ses lectures mais n'en hérite pas.

Emulateur de test : CSaCombTest

5 Cette classe permet de tester le module. Il suffit de l'appeler et la syntaxe correcte s'affiche, ce qui vous permettra de l'utiliser.

10 L'algorithme utilisé pour cette conversion est un algorithme de type « Horner » qui va effectuer des décalages successifs dans les registres que constituent les octets, calculer la puissance de deux correspondante et la sommer au résultat qui sera retourné.

REVENDEICATIONS

1 - Procédé de contrôle d'intégrité de fichiers informatiques d'un poste client comportant une étape
5 d'enregistrement sur le poste client d'une valeur de hachage de référence associée à chaque fichier et des procédures de comparaison de ladite valeur de hachage enregistrée avec une valeur de hachage calculée périodiquement au moment du contrôle du fichier caractérisé
10 en ce que les valeurs de hachage de référence sont enregistrées dans un dictionnaire local signé par un moyen cryptographique, le dictionnaire comprenant une liste de séquences formées par une adresse associée au fichier à contrôler et par la valeur de hachage prédéterminée, et en
15 ce que la procédure de comparaison est réalisée par un équipement distant communiquant avec le poste client par le protocole TCP/IP, la procédure comportant une étape d'ouverture d'un socket TCP/IP, une étape de lecture et de contrôle de la signature dudit dictionnaire, une étape de comparaison de la signature de référence associée à chaque
20 fichier du dictionnaire avec une signature du fichier correspond à l'adresse associée, calculée dynamiquement, et une étape de notification d'un message d'acquiescement ou d'alarme en fonction du résultat respectivement positif ou
25 négatif de l'étape de comparaison, ledit message étant signé.

2 - Procédé de contrôle d'intégrité de fichiers informatiques d'un poste client selon la revendication 1
30 caractérisé en ce qu'il comporte une étape de signature sur le poste client du dictionnaire local un moyen cryptographique, et en ce que la procédure de contrôle comporte une étape de lecture et de contrôle de la signature dudit dictionnaire,

3 - Procédé de contrôle d'intégrité de fichiers informatiques d'un poste client selon la revendication 1 ou 2 caractérisé en ce que la procédure de contrôle est périodiquement interrompue.

4 - Procédé de contrôle d'intégrité de fichiers informatiques d'un poste client selon l'une quelconque des revendications précédentes caractérisé en ce qu'il comporte une étape d'ouverture d'un premier socket TCP/IP pour les données relatives à la lecture du dictionnaire, et d'un deuxième socket TCP/IP pour les données relatives au balayage des fichiers dudit dictionnaire et au calcul des signatures.

5 - Procédé de contrôle d'intégrité de fichiers informatiques d'un poste client selon l'une quelconque des revendications précédentes caractérisé en ce qu'il comporte une étape de génération d'une bi-clé cryptographique K_{RU1} , K_{PU1} pour la signature du dictionnaire et la signature de chaque fichier du dictionnaire.

6 - Procédé de contrôle d'intégrité de fichiers informatiques d'un poste client selon l'une quelconque des revendications précédentes caractérisé en ce qu'il comporte une étape de génération d'une bi-clé cryptographique K_{RUC} , K_{PUC} pour la signature des message d'acquiescement ou d'alarme.

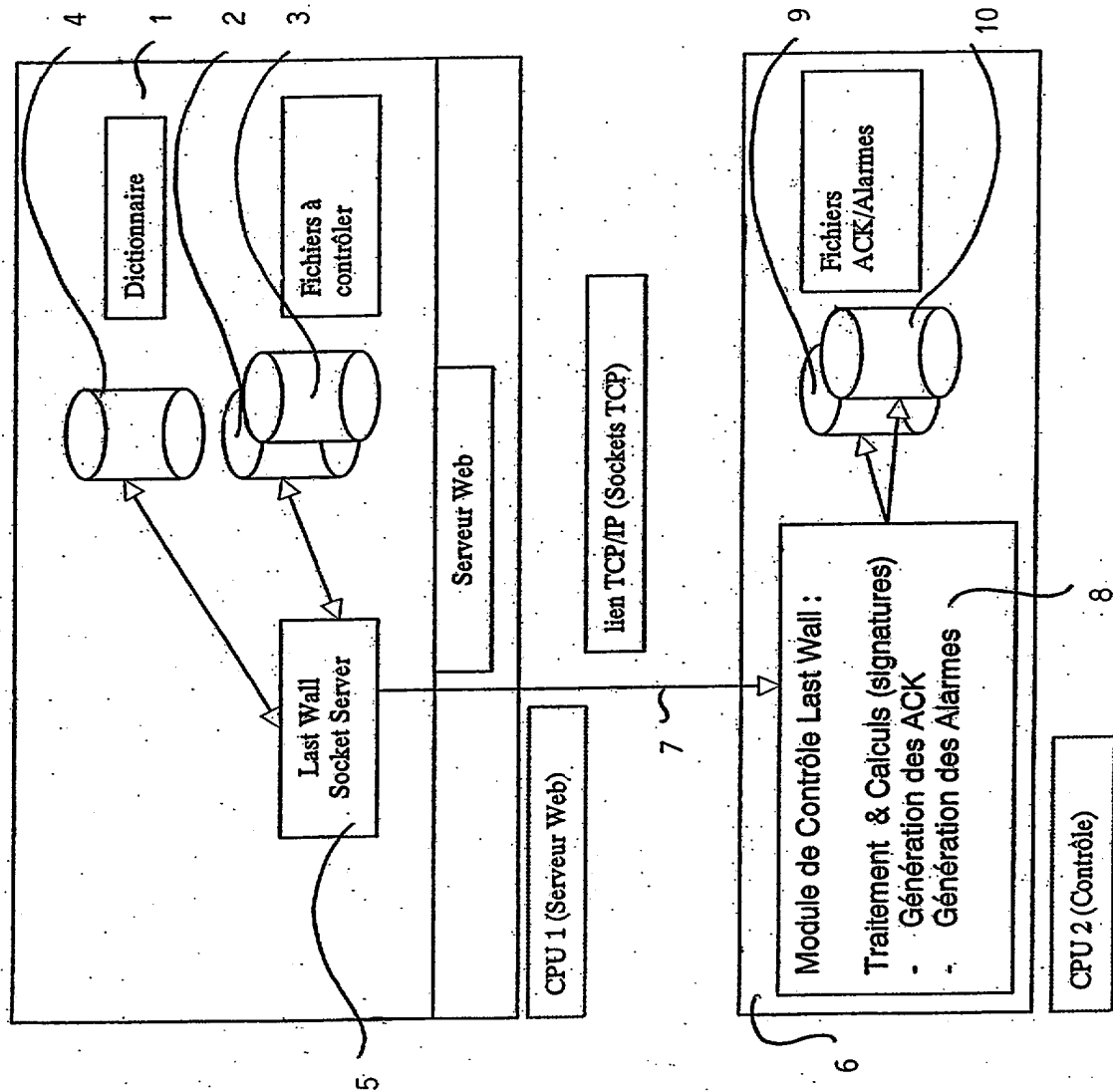
7 - Procédé de contrôle d'intégrité de fichiers informatiques d'un poste client selon l'une quelconque des revendications précédentes caractérisé en ce qu'il comporte une étape de propagation de la bi-clé cryptographique K_{RU1} ,
5 K_{PU1} entre une pluralité de postes clients.

8 - Procédé de contrôle d'intégrité de fichiers informatiques d'un poste client selon l'une quelconque des revendications précédentes caractérisé en ce que les bi-clés cryptographiques K_{RU1} , K_{PU1} et K_{RUC} , K_{PUC} sont identiques.
10

9 - Procédé de contrôle d'intégrité de fichiers informatiques d'un poste client selon l'une quelconque des revendications précédentes caractérisé en ce que l'étape d'enregistrement sur le poste client d'une valeur de hachage de référence associée à chaque fichier consiste à calculer pour ledit fichier un peigne par une extraction de données dudit fichier et à calculer la valeur de hachage dudit peigne, l'étape de comparaison consistant à comparer
15 la signature de référence associée à chaque peigne avec une signature du peigne correspond à l'adresse associée au fichier à contrôler, calculée dynamiquement.
20

10 - Equipement pour le contrôle d'intégrité de fichiers informatiques d'un poste client selon le procédé objet de la revendication 1 caractérisé en ce qu'il est constitué par une station de contrôle comportant des moyens de communication avec au moins un poste client par un réseau informatique, une mémoire l'enregistrement des données transitoires résultant de l'étape de comparaison des signatures et des moyens pour la signature automatique des messages d'acquiescement ou d'alerte.
25
30

Fig. 1



RAPPORT DE RECHERCHE PRÉLIMINAIRE

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

2805691

N° d'enregistrement
national

FA 589268
FR 0002564

DOCUMENTS CONSIDÉRÉS COMME PERTINENTS		Revendication(s) concernée(s)	Classement attribué à l'invention par l'INPI
Catégorie	Citation du document avec indication, en cas de besoin, des parties pertinentes		
A	EP 0 940 945 A (AT & T CORP) 8 septembre 1999 (1999-09-08) * abrégé * * colonne 2, ligne 56 - colonne 12, ligne 23 * * figures 1-7 * ---	1-5,10	H04L9/14
A	FR 2 767 208 A (MICROSOFT CORP) 12 février 1999 (1999-02-12) * page 13, ligne 24 - page 31, ligne 32 * * figure 2 * ---	1-3,5,10	
A	US 5 530 757 A (KRAWCZYK HUGO M) 25 juin 1996 (1996-06-25) * colonne 3, ligne 65 - colonne 6, ligne 36 * * figures 1-3 * ---	1,3,9,10	
A	US 5 919 257 A (TROSTLE JONATHAN) 6 juillet 1999 (1999-07-06) * abrégé * * colonne 3, ligne 53 - colonne 7, ligne 50 * ---	1,3,10	
A	WO 99 13415 A (KONINKL PHILIPS ELECTRONICS NV ; PHILIPS AB (SE)) 18 mars 1999 (1999-03-18) -----		
			DOMAINES TECHNIQUES RECHERCHÉS (Int.CL.7)
			G06F H04L
Date d'achèvement de la recherche		Examineur	
30 novembre 2000		Jacobs, P	
CATÉGORIE DES DOCUMENTS CITÉS			
X : particulièrement pertinent à lui seul Y : particulièrement pertinent en combinaison avec un autre document de la même catégorie A : arrière-plan technologique O : divulgation non-écrite P : document intercalaire		T : théorie ou principe à la base de l'invention E : document de brevet bénéficiant d'une date antérieure à la date de dépôt et qui n'a été publié qu'à cette date de dépôt ou qu'à une date postérieure. D : cité dans la demande L : cité pour d'autres raisons & : membre de la même famille, document correspondant	