MASTER EN SCIENCES INFORMATIQUES

Application de la cryptographie à la sécurisation d'un réseau informatique à l'aide d'un processeur de sécurité

Hubin, Joël

Award date:
1990

Awarding institution:
Universite de Namur

Facultés
Universitaires
N. D. de la Paix
NAMUR

Institut d'informatique

Application de la cryptographie à la sécurisation d'un réseau informatique à l'aide d'un processeur de sécurité

par Joël HUBIN

Promoteur
Professeur J. RAMAEKERS

Mémoire présenté en vue de l'obtention du grade de Licencié et Maître en Informatique

Année académique 1989 - 1990

Avant de présenter le travail qui clôturera ma Licence et Maîtrise en informatique, je tiens à remercier chaleureusement tous mes professeurs et tous ceux qui m'ont appris quelque chose.

Entre tous, je tiens à remercier plus spécialement ceux qui m'ont aidé au cours de cette année académique :

- Monsieur le professeur Ramaekers, Monsieur Ferreira et Monsieur Quisquater qui m'ont offert cinq des mois les plus enrichissants de ma "carrière d'étudiant" ;

- tout le personnel du Centre de Technologie Informatique de Fontenay qui m'a toujours accueilli simplement et sympathiquement ;

- plus particulièrement Didier Labat pour sa disponibilité et ses explications pratiques ; Christophe Louis qui a toujours su me guider tout en m'écoutant ; Robert Naciri pour ses remarques mathématiques judicieuses ; Monsieur Vincensini pour la "logistique" ;

- Mademoiselle Claire Bawin qui a accepté de relire mon travail.

Je remercie enfin mes parents, mes meilleurs professeurs.

Merci Geneviève.

Résumé — Abstract

De nos jours, la banalisation de la micro-informatique et l'incessante expansion des réseaux rendent les systèmes informatiques de plus en plus vulnérables et de plus en plus difficiles à protéger. Bien que les différentes fonctions à mettre en place pour combattre la criminalité informatique soient bien identifiées, leur mise en oeuvre reste complexe. Dans le cadre de notre stage, nous avons participé au développement d'un serveur de sécurité. Il s'agit d'un ensemble homogène, tant matériel que logiciel, destiné à sécuriser un réseau en fournissant des services tels que le contrôle d'accès, l'authentification d'utilisateurs et de stations de travail, la confidentialité et l'intégrité de données et de programmes ainsi qu'une gestion de clés puissante. Après un résumé des notions essentielles de la cryptologie et une introduction aux cartes à microprocesseur que nous avons utilisées, nous exposerons les solutions adoptées pour réaliser une maquette de ce serveur de sécurité.

Nowadays, because of the widespread of micro-computers and because of the expansion of networking, computer systems are becoming more and more difficult to protect. The different functions which can be used to fight computer crime are well known but complexe to realize in real environment. During our training period, we have participated to the development of a security server. It is an homogenous device, both hardware and software, providing, in networks, security services like access control, users and workstations authentication, datas and programs confidentiality and integrity, and strong key management. After a brief presentation of essential notions of cryptology and an introduction to the smart card we had used, we will expose the solutions choosen to realize a prototype of such a security server.

Sommaire

Liste des figures

Avant-propos

La sécurité sous toutes ses formes n'est pas un problème nouveau, même si la médiatisation dont elle fait l'objet depuis quelques années tend à nous le faire croire. En effet, de tous temps l'homme a cherché à protéger ce qu'il possédait face aux agressions extérieures. Mais l'avènement des techniques modernes, l'importance croissante de l'information dans notre style de vie et l'utilisation de plus en plus répandue des ordinateurs et des télécommunications posent le problème de la sécurité sous un jour nouveau.

Il était en effet (relativement) facile de protéger les informations en les enfermant dans un coffre, de s'assurer du transport en toute discrétion d'un document en le faisant parvenir à son destinataire de la main à la main dans une enveloppe scellée, de se faire connaître de son interlocuteur en lui serrant la main ou de confirmer la validité d'un document en apposant sa signature au bas d'une feuille. Mais de nos jours, l'information ne peut plus rester dans un endroit restreint et facilement contrôlable. Le courrier et l'argent sont électroniques, la signature est digitale et les poignées de mains doivent se faire d'un côté à l'autre de l'océan (il faut avoir le bras long !). Il faut donc réactualiser les anciennes techniques pour utiliser leur équivalent électronique.

C'est dans ce contexte que nous nous proposons de développer notre mémoire de fin d'étude.

Le chapitre 1 sera consacré à une brève étude factuelle du phénomène de la fraude informatique, point de départ de tous les mécanismes que nous détaillerons dans la suite.

Le chapitre 2 présentera le principal moyen de protection logique des systèmes informatiques : la cryptographie. Nous passerons en revue les outils de base que nous aurons à utiliser.

Le chapitre 3, quant à lui, montrera quelques utilisations simples et classiques des outils cryptographiques.

Le but du chapitre 4 est double. D'une part, il est destiné à présenter, de façon générale, la valeur ajoutée d'un outil de sécurité

prometteur : la carte à microprocesseur. D'autre part, cette présentation se basera sur un exemple concret, la carte Philips masque DES-D2, que nous manipulerons dans les chapitres suivants.

Dans le chapitre 5, nous présenterons globalement le projet auquel nous avons participé lors de notre stage de fin d'étude. Nous avons effectué ce stage au Centre de Technologie Informatique de la société T.R.T. (Philips France), au sein de l'équipe SAPPHIRE (Security Advanced Project for Philips In RSA Environment). Le projet consistait à spécifier et développer la maquette d'un serveur de sécurité.

Nous ferons ensuite, dans le chapitre 6, une analyse plus détaillée d'une partie essentielle de ce serveur : les protocoles de base destinés à gérer, de façon dynamique et sécurisée, les ressources du serveur.

Logiquement, le chapitre 6 aurait pu être suivi de la modularisation plus poussée, des spécifications externes et internes, nécessaires à l'implémentation, ainsi que du "coding" de ces protocoles. Comme nous considérons que cette partie n'aurait apporté aucun élément fondamental supplémentaire, et pour des raisons évidentes de ... sécurité, nous passerons cette partie sous silence.

Nous rappellerons enfin le fil conducteur de ce mémoire, ce qui nous permettra de conclure notre travail.

1. Sécurité informatique : l'escalade

1.1 Risques et pertes : classification

1.1.1 L'APSAIRD

En matière de risque, les compagnies d'assurance sont souvent les organismes qui disposent des meilleurs chiffres. Pour obtenir des données statistiques, nous nous sommes donc tourné vers l'APSAIRD (Assemblée Plénière des Sociétés d'Assurance contre l'Incendie et les Risques Divers), société d'assurance française regroupant de nombreux spécialistes de l'informatique. Chaque année, l'APSAIRD publie un tableau résumant les pertes dues à l'informatique en France. Ce tableau à deux entrées se base sur une classification des risques et des pertes que nous allons examiner brièvement.

1.1.2 Les types de risque

L'APSAIRD identifie dix types de risque différents.

1. Les *risques matériels accidentels* qui peuvent détruire ou endommager sérieusement le matériel informatique au sens large. Dans cette classe, on regroupe ainsi les incendies, les explosions, les bris de machine et les catastrophes naturelles.
2. Le *vol et le sabotage de matériel* de plus en plus fréquent.
3. Les *pannes et dysfonctionnement de matériel et logiciel de base.*
4. Les *erreurs de saisie, de transmission et d'utilisation des informations.*
5. Les *erreurs d'exploitation* qui regroupent les oublis divers et les erreurs de manipulation de système, matériel ou langage informatique.
6. Les *erreurs de conception et de réalisation.*

7. La *fraude et le sabotage immatériel*, de plus en plus à la mode dans les manchettes des quotidiens.

8. L'*indiscrétion et le détournement d'informations* qui peut aller jusqu'à l'espionnage industriel.

9. Le *détournement de logiciel* qui se développe de jour en jour avec la banalisation de la micro-informatique.

10. La *grève et le départ de personnel stratégique*.

Les rubriques que nous venons de voir peuvent également être rassemblées en trois classes moins fines mais plus tangibles :

- les sinistres dus à la **malveillance** (risques non fonctionnels) ;
- les sinistres **accidentels** (les coups du sort) ;
- les sinistres causés par les **erreurs** (risques fonctionnels).

1.1.3 Les types de pertes

L'APSAIRD distingue :

1. Les *dommages matériels et annexes* qui englobent les coûts de réparation ou de remplacement du matériel informatique endommagé ou volé.

2. Les *pertes d'exploitation* : pertes d'intérêts bancaires, de chiffre d'affaire, de clientèle.

3. Les *pertes de fonds* qui correspondent à la perte de biens financiers.

4. Les *pertes de biens*.

5. Les *frais supplémentaires* qui regroupent les frais de mise en oeuvre des moyens de sécurité tels que les ressources et le personnel de secours, les frais de reconstruction.

1.2 Le risque informatique : analyse factuelle

La figure 1 reprend, de façon synthétique (tous les types de pertes confondus) les tableaux publiés par l'APSAIRD pour les années 1984 à 1988. Ce tableau nous permettra de mettre en évidence deux tendances en matière de pertes dues à l'informatique. Remarquons dès à présent que ces chiffres ne sont très certainement qu'une borne inférieure. Certaines études tendent à montrer qu'il ne s'agit que de la partie visible de l'iceberg : en effet, nonante pour cent des pertes seraient cachées par les victimes désireuses, semble-t-il, de cacher leurs faiblesses.

	1984	1985	1986	1987	1988
Risques matériels	870	950	1080	1176	1270
Vol, sabotage de matériel	60	60	65	70	80
Pannes et dysfonctionnement de matériel et logiciel de base	900	900	1025	971	1020
Erreurs de saisie, de transmission et d'utilisation des informations	1140	970	800	773	750
Erreurs d'exploitation	300	390	320	321	350
Erreurs de conception et réalisation	540	600	620	702	800
Fraude, sabotage immatériel	1050	1300	1700	2000	2300
Indiscrétion, détournement d'informations	180	250	310	381	480
Détournement de logiciels	750	750	1200	1414	990
Grèves, départ de personnel stratégique	90	100	90	100	90
Divers	60	50	60	0	0
TOTAL	5940	6320	7270	7908	8130
Malveillance	2040	2360	3275	3865	3850
Accidents	1860	1950	2195	2247	2380
Erreurs	1980	1960	1740	1796	1900

Les chiffres sont exprimés en millions de francs français.

Figure 1 : Pertes dues à l'informatique en France (APSAIRD 1984 à 1988)

Nous pouvons déjà constater l'importance relative du montant total des pertes pour 1988 : 8 milliards de francs français, soit environ 50 milliards de francs belges ou 500 milliards de francs belges si l'on considère qu'un dixième seulement des sinistres est avoué par les victimes.

En dehors de toute considération purement financière, il est à noter que les conséquences d'une perte due à l'informatique sont fatales à des nombreuses entreprises de faible taille.

Les graphiques de la figure 2, obtenus directement à partir de notre tableau précédent, montre que le total des pertes dues à

l'informatique n'a cessé d'augmenter depuis 1984. Remarquons cependant que cette croissance s'affaiblit en 1988.

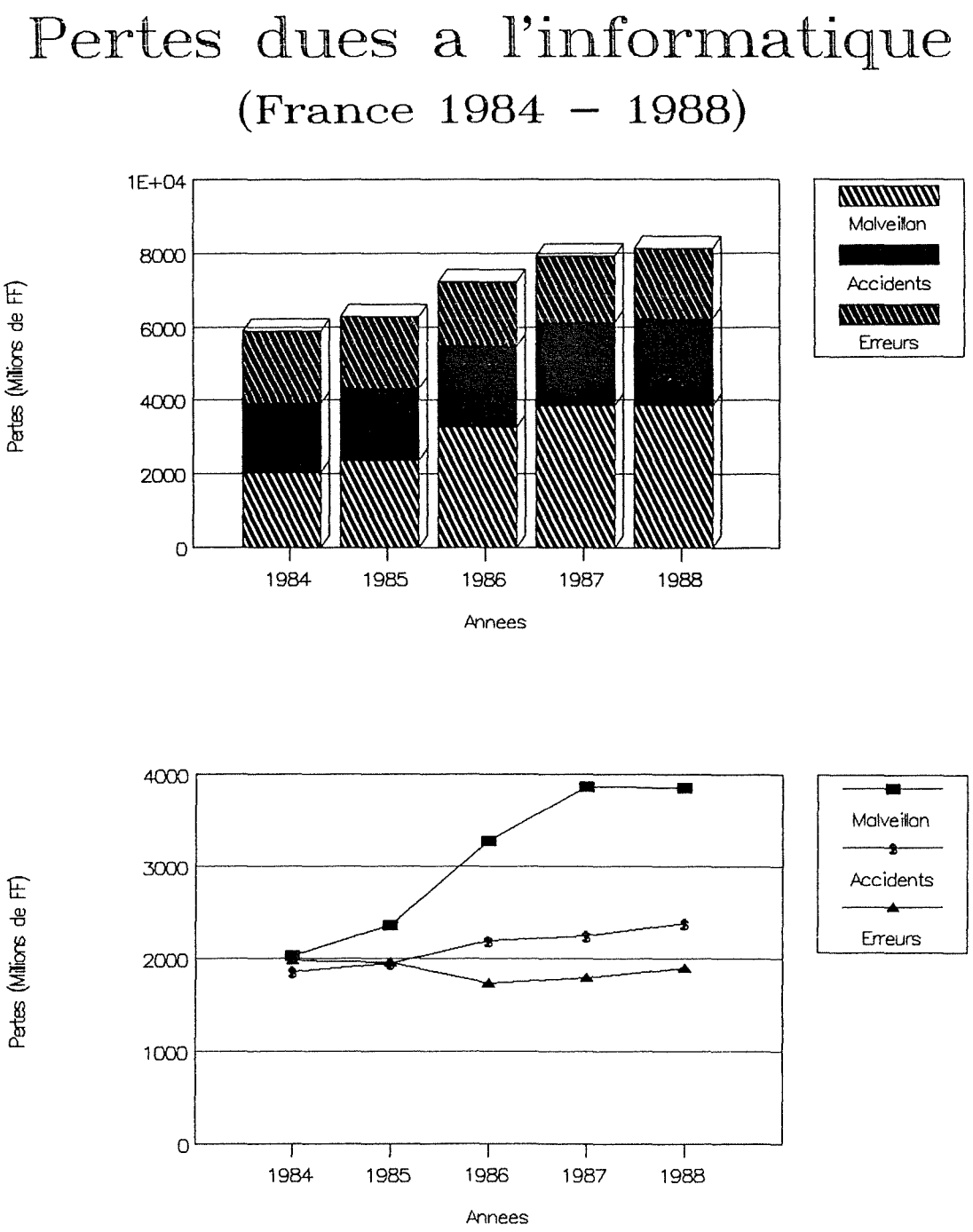
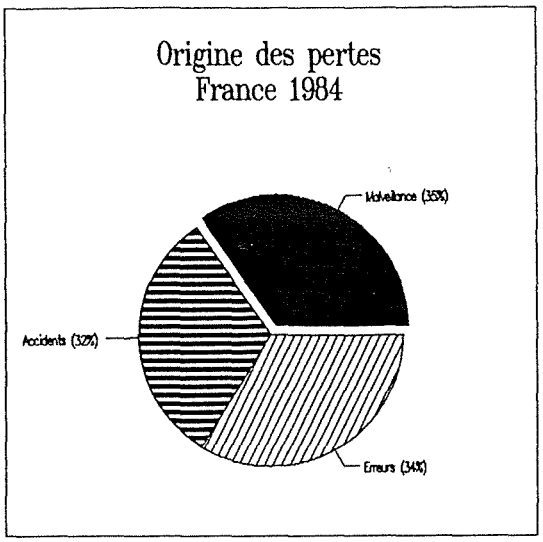
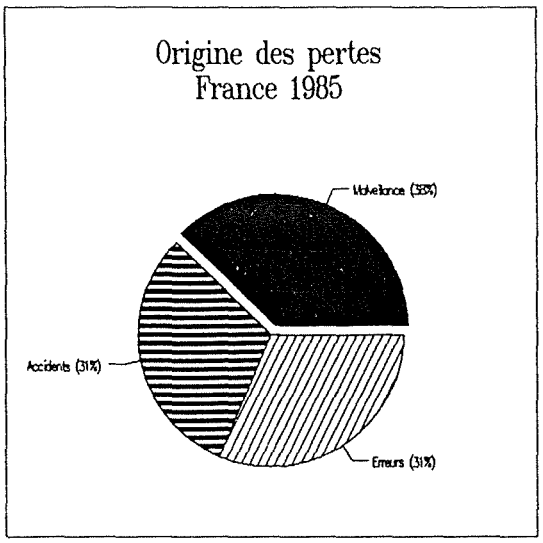
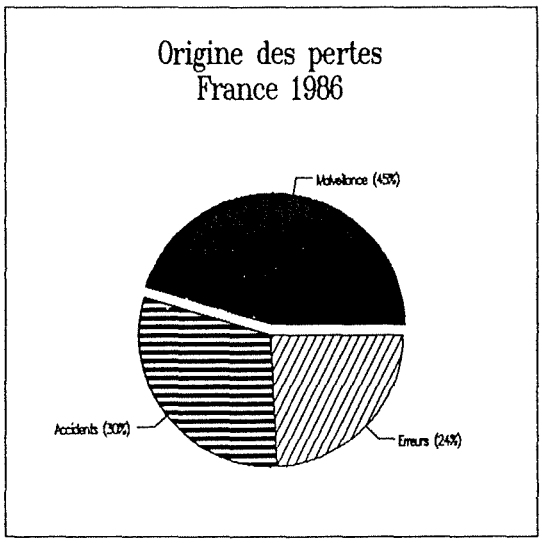
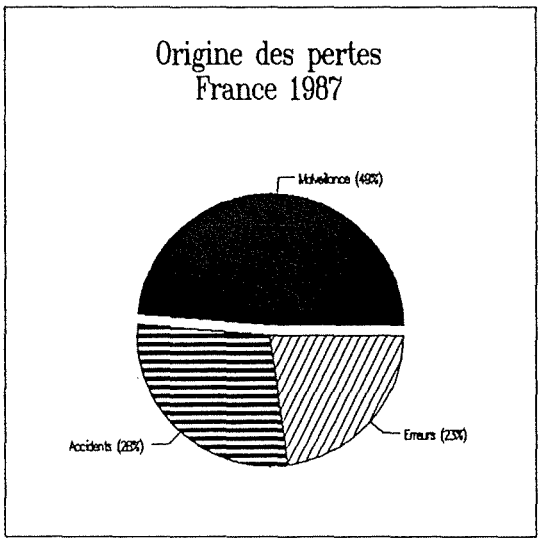


Figure 2 : Pertes dues à l'informatique (France 1984 - 1988)









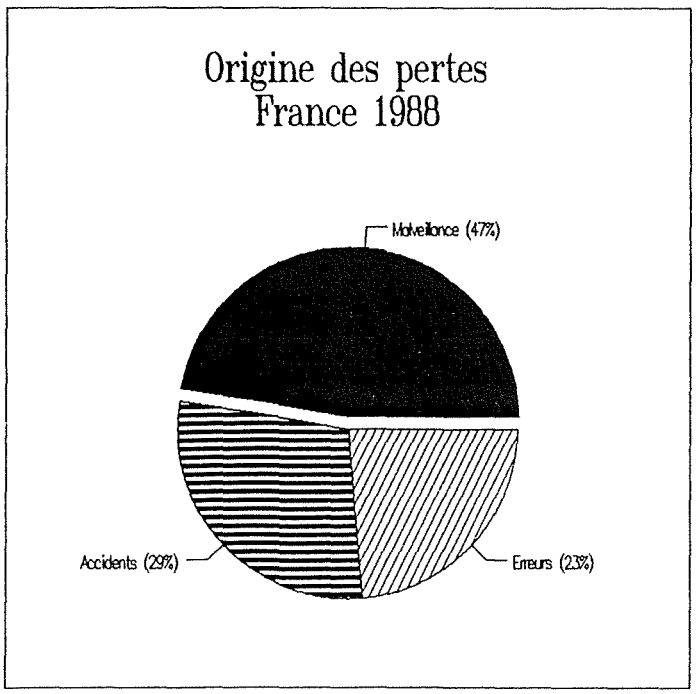


Figure 3 : Origine des pertes dues à l'informatique (France 1984 - 1988)

On peut également constater que la fraude est la grande responsable de cette ascension dans les milliards. L'importance de la malveillance est également soulignée par les graphiques de la figure 3. En 1984, la malveillance représentait le tiers des pertes pour atteindre presque la moitié en 1987. En 1988, on distingue une faible diminution, probablement due à la mise en place de plus en plus généralisée de la sécurité informatique.

Ces chiffres tendent alors à nous mettre en garde : les pertes liées à l'informatique sont de plus en plus la conséquence d'actes d'individus mal intentionnés, alors que les pertes accidentelles sont bien gérées par des protections classiques semblables à celles que nous allons rappeler brièvement.

1.3 Sécurité : des solutions bien connues

Nous nous contenterons ici de citer quelques méthodes qui permettent d'éviter ou de minimiser les pertes dues à des risques classiques.

1. Etude détaillée de l'environnement physique (architecture, infrastructure, choix de l'emplacement géographique) ;
2. Contrôle d'accès physique et logique aux ressources informatiques (badge, gardien, mot de passe ...) ;
3. L'entretien du matériel ;
4. La redondance des moyens (systèmes tolérant aux pannes) ;
5. Les contrats ;
6. Les plans de "disaster recovery".

1.4 La malveillance : escalade des moyens

Nous l'avons vu, la malveillance prend une part de plus en plus importante et inquiétante dans les estimations des pertes dues à

l'informatique. De plus, elle prend des formes sans cesse plus complexes et sournoises. On assiste ainsi à une véritable escalade dans le raffinement des moyens utilisés par les fraudeurs qui nécessitera, comme nous le verrons par la suite, des moyens de protection toujours plus évolués.

Les premiers systèmes informatiques n'étaient pas conçus pour se protéger si bien que les premiers fraudeurs, bien souvent inoffensifs, n'eurent aucun mal à agir. Ensuite, les protection faibles, mais néanmoins toujours utilisées car faciles à mettre en oeuvre, ont fait leur apparition ; nous pensons par exemple au mot de passe. Bien vite, les fraudeurs ont découvert des moyens plus ou moins complexes de déjouer de telles protections : essai de mots de passe classiques, espionnage ou chevaux de Troie. Ensuite, l'informatique a subi deux grandes révolutions : l'ouverture au monde par la banalisation de la micro-informatique et l'expansion des réseaux. Ces changements ont amené avec eux des fraudes plus complexes, plus anonymes et aussi plus lucratives comme l'espionnage, le vol d'information ou le piratage de logiciels ou autres ressources informatiques. Et à l'heure actuelle, la fraude devient encore plus vicieuse avec l'apparition du sabotage et du chantage appuyés par les bombes logiques, les virus et vers informatiques.

Nous nous concentrerons donc, dans les chapitres à venir, sur l'escalade des solutions que l'on peut développer pour répondre à cette escalade du banditisme (terrorisme !) informatique.

2. La cryptographie comme outil sécuritaire

2.1 Introduction

On peut trouver dans [LEGLU,sd] et [KRAYE,87] les définitions suivantes :

- la **cryptographie** consiste à faire subir des transformations aux textes des messages au moyen d'un jeu de clés, de manière à les rendre illisibles et à les restituer en clair par le déchiffrement ;

- un **cryptosystème** est un procédé mathématique pour coder ou transformer d'une façon unique un message écrit en clair en un message dit chiffré afin qu'il soit rendu inintelligible pour ceux à qui il n'est pas destiné ;

- le **cryptogramme (ciphertext)** est le message écrit en caractères secrets ;

- le **chiffrement (encipherment)** est l'opération qui consiste à transformer un texte clair en cryptogramme ;

- le **déchiffrement (decipherment)** est l'opération qui consiste à rétablir en clair un texte chiffré à l'aide de la clé ;

- le **décryptage** est l'opération qui consiste à traduire des messages chiffrés dont on ne possède pas la clé ;

- la **cryptanalyse** est l'art de décrypter les chiffres ;

- la **cryptologie** est la science du chiffre dans ses deux aspects : la cryptographie et la cryptanalyse.

2.2 Notions de base – Définitions

2.2.1 Mécanismes principaux

La cryptographie est basée sur deux mécanismes fondamentaux que nous allons rappeler brièvement.

a) Chiffrement par substitution

Ce mécanisme de chiffrement consiste à remplacer les caractères du texte clair par d'autres caractères selon une loi bien définie ; cette loi peut être exprimée sous forme d'une clé (elle-même constituée d'une suite de caractères).

b) Chiffrement par transposition

Ce mécanisme de chiffrement consiste à faire subir une permutation aux caractères du texte clair.

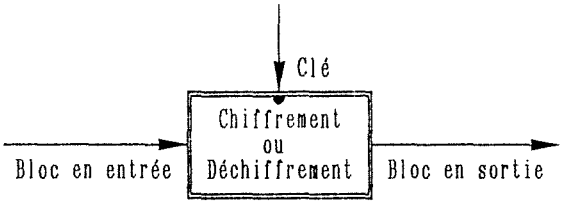
c) Chiffrement produit

Les deux méthodes précédentes, employées séparément, sont cependant sensibles aux attaques et un message est rapidement décrypté. Pour s'en convaincre, il suffit de recourir à des méthodes de cryptanalyse simples. On peut cependant, selon Shannon, combiner ces deux mécanismes pour obtenir des méthodes de **chiffrement produit** moins sensibles à des attaques rapides. L'exemple type de cette méthode de chiffrement est le D.E.S. (Data Encryption Standard) que nous examinerons plus tard.

2.2.2 Classification

a) Chiffrement par bloc

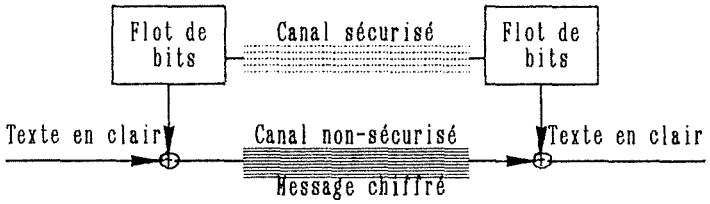
Un **algorithme de chiffrement par bloc** traite, en entrée, des chaînes de symboles de longueur fixe (un **bloc**) et rend en sortie une autre chaîne de symboles de longueur également fixée (pas obligatoirement la même, bien que ce soit souvent le cas pour des raisons de simplicité). Le nombre de symboles formant un bloc est la **taille du bloc**. Un avantage de cette méthode de chiffrement est sa rapidité dans la plupart des cas.



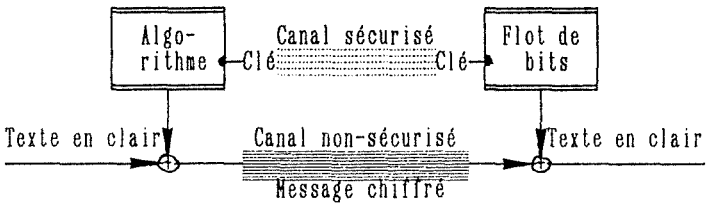
Pour renforcer la sécurité d'un chiffrement par bloc, on a défini plusieurs modes d'utilisation. Une norme internationale de l'ISO peut être trouvée dans [ISO 8372,87]. D'autres organismes ont également normalisé ces modes : on peut se référer à [ANSI X3.106,83] et [FIPSP,81].

b) Chiffrement par flot

Un **chiffrement par flot (stream cipher)** utilise un générateur de suite de bits. On parle alors parfois de **Générateur d'Octets Chiffrant (GOC)**. Cette suite de bits est ensuite combinée avec le message pour produire le cryptogramme. Si la génération du flot initial de bits est aléatoire, tout ce flot est utilisé comme clé de chiffrement. Un exemple bien connu de ce type de chiffrement est le **système de Vernam (one-time tape)**. Par contre, ce flot peut être obtenu grâce à un algorithme cryptographiquement solide paramétré par une clé. Ce système est moins "solide" mais il permet du moins une diffusion de la clé plus aisée.



ou, avec un algorithme :



c) Chiffrement symétrique ou asymétrique

Ces notions seront explicitées en détail dans la suite de l'exposé.

d) Protocoles à apport nul de connaissances

Il est possible de synthétiser, au risque de les caricaturer, les protocoles à apport nul de connaissance en disant qu'il sont destinés à prouver à quelqu'un que l'on possède un secret sans jamais devoir lui montrer. Ce type de protocole, assez récent, ouvre ainsi de nouveaux horizons dans la sécurisation des systèmes : plus besoin d'éléments secrets, plus de gestion de clé complexe ... . Les protocoles GQ (Guillou - Quisquater) et FS (Fiat - Shamir) sont deux protocoles de ce type ([GUILL,87], [GUILL,88.2]).

2.2.3 Puissance d'un cryptosystème

a) Types d'attaque

Lorsque l'on essaie de trouver une approximation de la puissance d'un cryptosystème, pour être sûr de trouver une borne inférieure, on pose l'hypothèse que le fraudeur connaît l'algorithme de chiffrement. Il est bien évident que si l'on parvient à "casser" un chiffrement sans connaître son algorithme au préalable, celui-ci ne peut être d'aucune utilité.

[DAVIE,84] distingue trois types d'attaque selon ce que possède le fraudeur :

- **ciphertext-only attack** : c'est la position la plus défavorable pour le décrypteur qui, dans ce cas, ne possède aucune paire (texte clair,texte chiffré correspondant). On suppose également que les textes en clair ne contiennent aucune redondance ;

- **known-plaintext attack** : il s'agit d'une position plus favorable pour la cryptanalyse ; on possède au moins une paire (texte clair,texte chiffré correspondant) et il suffit donc de trouver une clé qui fasse correspondre texte et cryptogramme ; si le texte est suffisamment long, on peut ainsi identifier la clé avec certitude [DAVIE,84] ;

- **chosen-plaintext attack** : c'est, pour le cryptanalyste, la situation la plus agréable ; il peut choisir un texte utile pour accélérer la recherche de la clé (une suite de zéros, par exemple).

On peut citer une anecdote : pendant la deuxième guerre mondiale, pour décrypter les codes secrets ennemis, les alliés ont expressément

bombardé une ville. Dès lors, il leur a suffi d'écouter les communications ennemies dans les heures qui suivirent et de tenter une recherche de clé à partir de mots probables tels que "bombardement" ou le nom même de la ville !

Toujours dans un souci de recherche de borne inférieure à la difficulté de décryptage, on considère l'attaque la plus favorable pour le fraudeur.

Pour effectuer une cryptanalyse, plusieurs types d'attaque sont envisageables selon la méthode utilisée :

- une **attaque exhaustive** consiste à essayer systématiquement toutes les solutions possibles (toutes les clés permises, tous les messages possibles ...) ;

- une **attaque par analyse de fréquence** consiste à utiliser le fait que dans le langage courant, certaines lettres ou certains groupes de mots ont une probabilité d'apparition élevée ;

- une **attaque déterministe** consiste à écrire le système de chiffrement sous forme d'équations et à trouver algébriquement une solution au déchiffrement.

b) Qualités requises pour un cryptosystème

On voit donc que, pour des raisons de commodité, on limite la taille de la clé. Or le décryptage consiste à découvrir, parmi le nombre de clés possibles, celle qui est la bonne. Une façon de procéder est l'**essai exhaustif** des clés.

Shannon a énormément contribué au développement de la théorie de la cryptologie ([DAVIE,84], p. 41). Il a, entre autres, défini deux types de systèmes de chiffrement :

- les **systèmes inconditionnellement sûrs (unconditionally secure)** qui résistent à la cryptanalyse même si la puissance de calcul était illimitée (exemple : one-time tape) ;

- les **systèmes pratiquement sûrs (computationally secure)** qui résistent à la cryptanalyse avec la puissance de calcul disponible dans l'état actuel de la technique.

Pour savoir si un cryptosystème est pratiquement sûr, deux solutions sont possibles : soit on étudie les méthodes de cryptanalyse possibles, soit on cherche à obtenir un algorithme de chiffrement tel que le casser équivaut à trouver une solution d'un problème connu difficile à résoudre (NP-Complétude en théorie de la calculabilité) [KRAYE,87].

2.2.4 Notations

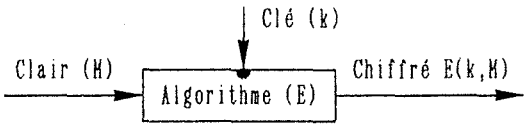
Un système cryptographique est constitué de trois parties [GIRAU,86] qui sont :

1. Une fonction E(k , M) de chiffrement (Encryption)
   où k est la clé de chiffrement
   M est le message (un bloc du message) à chiffrer.
2. Une fonction D(k', M) de déchiffrement (Decryption)
   où k' est la clé de déchiffrement associée à k
   M est le message à déchiffrer.
3. Une méthode permettant de produire des couples (k , k') de **clés associées** c'est-à-dire telles que D(k' , E(k , M)) = M. L'ensemble des clés possibles est appelé l'**espace des clés.**

D'autres notations sont parfois utilisées. Considérons l'exemple suivant : nous voulons indiquer qu'un message M est chiffré avec la clé k. On peut employer les différentes notations suivantes :

- E(k , M)
- E_k (M)
- $\{ M \}^k$

On peut également utiliser la représentation graphique suivante :



2.3 Cryptosystèmes symétriques

2.3.1 Introduction

Un **cryptosystème symétrique** ou **cryptosystème à clé secrète** est un cryptosystème basé sur un algorithme (secret ou connu de tous) pour lequel la clé qui est utilisée au chiffrement aussi bien qu'au déchiffrement (symétrique !) doit impérativement rester secrète vis-à-vis d'un tiers. Ainsi, toute personne qui possède la clé secrète est capable de chiffrer et déchiffrer un message. On peut schématiser la situation de la façon suivante :

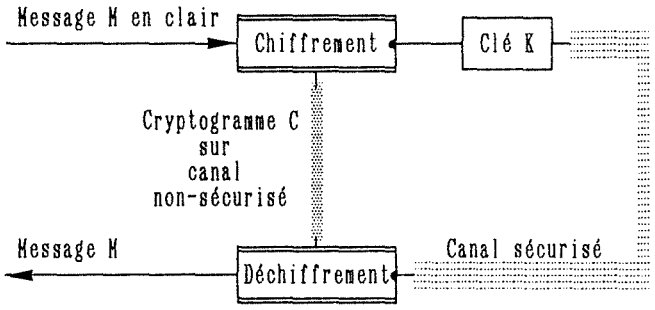


Figure 4 : Schéma de chiffrement symétrique.

Mathématiquement, on a d'après nos notations $k = k'$ et la connaissance de E implique la connaissance de D.

2.3.2 Exemple 1 : cryptosystème sûr de Vernam

On peut voir que le point faible d'un cryptosystème, c'est la façon dont les redondances du texte en clair apparaissent encore dans le cryptogramme correspondant. Pour obtenir une méthode de chiffrement la plus sûre possible, il faudrait donc que, quelles que soient les redondances du texte de base, le flot de caractères sortant du cryptosystème soit le plus aléatoire possible. On peut obtenir cette qualité en utilisant le **chiffrement de Vernam** :

- on produit une suite aléatoire de caractères de longueur égale à la longueur du message à chiffrer ; cette suite sera la clé de chiffrement ;

- on "additionne" chaque caractère du texte en clair à son homologue de la clé ; le cryptogramme ainsi obtenu est donc aussi aléatoire que peut l'être la clé ;

- on transmet ce texte par voie normale et la clé par voie sécurisée.

On est parvenu à démontrer que ce cryptosystème est inconditionnellement sûr en termes de théorie de l'information, c'est-à-dire que, quelle que soit la puissance de calcul, on ne peut le décrypter.

Ce système est malheureusement difficile à utiliser car la distribution de la clé n'est pas aisée : elle est aussi longue que le texte à chiffrer. Or, pour que le système soit vraiment sûr, il faut que la clé soit abandonnée après chaque utilisation ; de la vient son nom : **One-Time Tape** ou **One-Time Pad**. Il est cependant utilisé là où un maximum de sécurité est requis (sur le fameux téléphone rouge, par exemple).

2.3.3 Exemple 2 : le DES (Data Encryption Standard)

a) Historique

En 1974, le NBS (National Bureau of Standard) fait un appel aux propositions pour un algorithme de chiffrement universel (et donc public et protégé seulement par le secret de la clé) et performant (débit de l'ordre du Mbit/s) permettant de protéger le transfert et le stockage des données dans les milieux commerciaux et industriels. IBM propose alors son algorithme **Lucifer**, inventé dans les années 70. En 1974, le NBS publia une proposition appelée DES (Data Encryption Standard), une version de Lucifer remaniée par IBM et la NSA (National Security Agency).

b) Principe de fonctionnement

Le DES est un algorithme qui accepte une clé binaire, obligatoirement secrète, de 56 bits et un bloc de texte de 64 bits.

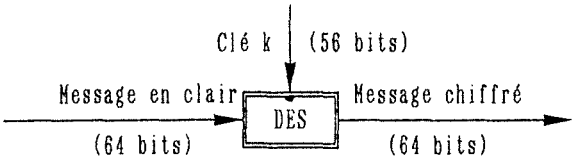


Figure 5 : Schéma externe du DES.

Le principe du DES est l'exécution de 16 itérations. Chaque itération consiste à effectuer une substitution et une permutation, l'ensemble dépendant d'une clé intermédiaire (il y en a 16, une par itération) calculée à partir de la clé de 56 bits.

Comme nous l'avons déjà signalé, l'utilisation répétée d'alternance substitution-permutation (chiffrement produit) donne au bout du compte un algorithme cryptographiquement solide (théorie de Shannon).

c) Propriétés

1) Complémentation du DES

Au cours des recherches effectuées pour "casser" le DES, on a trouvé une propriété qui permet de réduire de 50% l'effort de recherche lors d'une attaque par texte choisi. On a, en effet :

Si $C = DES_k (P)$

Alors $\underline{C} = DES_{\underline{k}} (\underline{P})$

où $\underline{C}$, $\underline{k}$, $\underline{P}$ sont les complémentaires de C, k, P.

2) Clés faibles

Une autre régularité dans l'algorithme DES est due à la façon dont les bits sont sélectionnés pour former les clés intermédiaires. En effet, si la clé est formée soit uniquement de zéros, soit uniquement de uns, la clé intermédiaire sera toujours la même à chaque itération. Dans ce cas, les clés de chiffrement (K1 à K16) seront identiques aux clés de déchiffrement (K16 à K1) : et en conséquence, chiffrement et déchiffrement sont des opérations identiques. On a donc 4 **clés faibles (weak keys)** bien connues et à éviter :

01 01 01 01 01 01 01 01
FE FE FE FE FE FE FE FE
1F 1F 1F 1F 0E 0E 0E 0E
E0 E0 E0 E0 F1 F1 F1 F1

3) Clés semi-faibles

En plus des 4 clés faibles, on connaît 6 paires de **clés semi-faibles (semi-weak keys)**. Lorsque l'on effectue un chiffrement avec la première clé de la paire suivi d'un chiffrement avec la seconde, on retrouve le texte en clair. Ces paires de clés sont :

01 FE 01 FE 01 FE 01 FE
FE 01 FE 01 FE 01 FE 01

1F E0 1F E0 0E F1 0E F1
E0 1F E0 1F F1 0E F1 0E

01 E0 01 E0 01 F1 01 F1
E0 01 E0 01 F1 01 F1 01

1F FE 1F FE 0E FE 0E FE
FE 1F FE 1F FE 0E FE 0E

01 1F 01 1F 01 0E 01 0E
1F 01 1F 01 0E 01 0E 01

E0 FE E0 FE F1 FE F1 FE
FE E0 FE E0 FE F1 FE F1

d) Evaluation/Perspectives d'avenir

Le DES est controversé car la NSA qui a parrainé sa naissance refusait de le certifier après 1988. De plus, plusieurs spécialistes de la cryptographie estiment possible la fabrication d'une machine à base de processeurs en parallèle (Coûteuse ! De l'ordre de 20 millions de dollars en 1979) permettant de faire une recherche exhaustive de clé en un temps acceptable (voir [RIVES,88]). Les autorités militaires américaines sont aussi parfois accusées d'avoir volontairement bridé le DES en limitant la taille de la clé à 56 bits, alors que celle à 128 (comme prévu dans l'algorithme Lucifer initial) n'aurait apporté que de petites modifications mais aurait accru la solidité [LEGLU,sd].

Certains mathématiciens se sont également attaqués à la lourde tâche de "casser du DES". Jusqu'à présent, ils y sont parvenus, mais seulement pour un DES à 8 étages [RONAL,89] !

On peut trouver une analyse des propriétés du DES dans [DAVIO,83], ainsi qu'une étude plus récente de la "solidité" du DES dans [SMID,88].

2.4 Cryptosystèmes asymétriques

2.4.1 Introduction

Un **cryptosystème asymétrique** ou **cryptosystème à clé publique** ou **cryptosystème à clé révélée** est un cryptosystème basé sur un algorithme (secret ou connu de tous) pour lequel deux clés distinctes sont disponibles.

L'une d'elles est secrète et l'autre peut être dévoilée à tout le monde. Ainsi, tout le monde pourra chiffrer/déchiffrer (selon le cas) un message avec la clé publique et seul le possesseur de la clé secrète pourra le déchiffrer/chiffrer (respectivement). On peut schématiser une telle situation de la manière suivante :

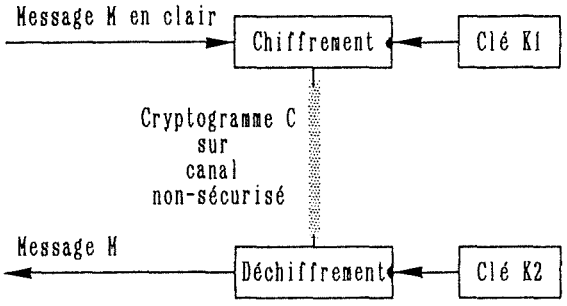


Figure 6 : Schéma de chiffrement asymétrique.

Mathématiquement, on obtient, d'après nos notations, k différent de k' ainsi que E de D. Et connaissant la clé publique, il est pratiquement impossible de trouver la clé secrète.

2.4.2 Fonction à sens unique

a) Introduction - Définitions

Une fonction f est appelée **fonction à sens unique (one-way function)** si pour tout x de son domaine de définition, il est facile de calculer la valeur y=f(x), alors que le calcul de x tel que f(x)=y pour tout y de l'image de f est pratiquement infaisable (computationally infeasible).

Cette fonction est donc non-inversible d'un point de vue pratique ; elle ne l'est pas du point de vue mathématique où une **fonction non-inversible** est définie comme une fonction telle que l'inverse d'un point de l'image n'est pas unique.

Une fonction f à sens unique est appelée **fonction cryptographique à sens unique** si elle est paramétrée par un élément secret (une clé).

Une fonction à sens unique ne doit contenir aucune régularité qui pourrait être exploitée pour l'inverser. Par exemple, on pourrait utiliser un ensemble de paires (x,f(x)) pour extrapoler la fonction. Ainsi donc, une bonne fonction à sens unique devrait générer des valeurs aléatoires, tout comme un bon cryptosystème. La conception d'une bonne fonction à sens

unique est donc une tâche aussi ardue que la conception de méthodes sûres de chiffrement.

b) Exemple 1

On remarque immédiatement la similitude existant entre fonction à sens unique et algorithme de chiffrement. En effet, tant que la clé reste secrète, il est impossible de retrouver le clair à partir d'un chiffré. Un algorithme de chiffrement constitue donc une fonction cryptographique à sens unique.

c) Exemple 2

On peut utiliser la fonction exponentielle avec modulo comme fonction à sens unique. En effet, il est relativement facile de calculer $y=a^x \bmod p$ connaissant a, x et p. Il existe à cet effet des méthodes rapides de calcul de l'exponentielle.

d) Introduction aux chiffrements à clé publique

Pour certaines utilisations, on introduit volontairement une "**trappe**" dans une fonction à sens unique, c'est-à-dire un moyen de rendre facile le calcul de l'inverse de la fonction à condition de connaître un "secret". Ainsi, il est facile, pour tout le monde, de calculer f(x) pour un x quelconque. Mais par contre, seul le détenteur du "secret" peut aisément calculer x à partir de f(x). Ce mécanisme, appelé "**trapdoor**" dans la littérature anglaise, est à la base de plusieurs cryptosystèmes à clé révélée.

2.4.3 Exemple : le RSA

a) Historique

En 1976, Diffie et Hellman publièrent un article [DIFFI,76] ouvrant la voie à la création de systèmes cryptographiques "révolutionnaires" en ce sens qu'ils pouvaient être paramétrés par deux clés liées (une pour l'émetteur et une pour le récepteur) dont une seulement devait rester secrète.

Parmi les systèmes de chiffrement basés sur cette idée originale, l'un des premiers publié fut le RSA, du nom de ses auteurs Rivest, Shamir et Adleman, que l'on retrouve dans [RIVES,78]. "Il présente le double avantage de jouir d'une description élégante et d'avoir résisté jusqu'à

présent aux attaques de toutes sortes dont il a été l'objet" ([GIRAU,86], p. 37).

b) Principe de fonctionnement

L'algorithme RSA est basé sur le problème NP-complet de la décomposition d'un nombre en facteurs premiers. Ce problème est réputé difficile - pratiquement insoluble - pour un nombre très grand ; en revanche, la transformation inverse, qui consiste à multiplier deux nombres premiers, est relativement simple à effectuer.

Des introductions au RSA peuvent être trouvées dans de nombreuses publications (voir, par exemple, [INDJO,89], [DROR,89], [RIVES,88]). Les ouvrages de référence [DAVIE,84] (pp. 237-246) et [MEYER,82] (pp. 33-48) traitent également le sujet. Mais LA référence pour le RSA est évidemment l'article de ses inventeurs : [RIVES,78].

c) Résumé de la méthode

1. Choisir aléatoirement deux grands nombres premiers p et q.

2. Calculer n = p.q.

3. Choisir aléatoirement un grand nombre d, premier avec
   phi(n) = (p-1).(q-1) .

4. Calculer e tel que e.d = 1 (mod phi(n))
   (e,n) constitue la clé publique
   (d,n) constitue la clé secrète.

5. Coder le message sous forme d'une suite M = (M_1, M_2, ..., M_n) d'entiers M_i tels que 0 <= M_i <= n-1.

6. Considérer :
   E : x --> x^e comme fonction de chiffrement, et
   D : x --> x^d comme fonction de déchiffrement,

on a bien D (E (x)) = D (x^e) = $(x^e)^d$ = $x^{e \cdot d}$ = x (mod n)
Pour chaque bloc, le chiffrement est obtenu grâce à la fonction E(M_i) et le déchiffrement grâce à la fonction D(M_i) définies ci-dessus.

L'implémentation de l'algorithme RSA est loin d'être triviale. Elle nécessite la génération de grands nombres premiers et surtout l'exponentiation modulaire sur des grands entiers. Ce qui pose des

problèmes très ardus d'optimisation, de validité et de test au cours de l'implémentation. On peut lire à ce propos [JUNG,87].

Connu de l'émetteur	Public	Connu du récepteur
n e	n e	Premiers p et q n = p.q Clés e et d tq e.d = 1 (mod phi(n)) où phi(n) = (p-1).(q-1)
Message M $C = M^e \pmod n$	Chiffré C	Déchiffré $M = C^d \pmod n$

Figure 7 : Quantités utilisées pour le RSA.

d) Evaluation/Perspectives d'avenir

La seule façon de "certifier" un cryptosystème, c'est d'essayer de le casser par tous les moyens possibles. Au bout de nombreux essais infructueux, on peut dire que c'est un "bon" cryptosystème.

Pour essayer de "casser" le RSA, trois méthodes existent :

- factoriser le nombre public n,
- calculer phi(n) sans factoriser n,
- déterminer d sans factoriser n ni calculer phi(n).

Plusieurs mathématiciens, dont les plus connus sont Fermat et Legendre, ont cherché des méthodes efficaces de factorisation de grands nombres. Un de ces algorithmes est dû à Richard Schroeppel, mais malgré sa rapidité, le tableau suivant nous montre qu'il n'est d'aucune utilité pratique pour décrypter un message chiffré par RSA.

Chiffres	Nbre d'opérations	Temps d'exécution
50	1.4×10^{10}	3.9 heures
75	9.0×10^{12}	104 jours
100	2.3×10^{15}	74 ans
200	1.2×10^{23}	3.8×10^{8} années
300	1.5×10^{28}	4.9×10^{15} années
500	1.3×10^{38}	4.2×10^{25} années

Ce tableau donne le nombre d'opérations et le temps nécessaire à les effectuer pour factoriser n à l'aide de l'algorithme de Schroeppel, en fonction du nombre de chiffres décimaux qui constituent n. On suppose que chaque opération demande une microseconde.

Figure 8 : Temps d'exécution pour la factorisation.

On peut encore ajouter que la factorisation d'un nombre de 256 bits prend quelques heures sur un Cray XMP. Actuellement, un bon niveau de sécurité est obtenu en choisissant p et q de l'ordre de 256 bits chacun.

Quant aux deux autres méthodes, on peut montrer qu'elles ne sont pas plus faciles que la factorisation de n ([RIVES,78]).

De plus, le RSA résiste jusqu'à présent aux autres "tortures" que les cryptanalystes ne manquent pas de lui faire subir ([GIRAU,86]).

e) Utilisation

Les performances du système RSA ne sont pas aussi bonnes que celles du DES en tant qu'algorithme de chiffrement/déchiffrement. En effet, un calcul RSA avec des clés de 512 bits coûte environ une seconde sur la plupart des microprocesseurs 32 bits du marché. Ce qui explique l'utilisation fréquemment combinée du RSA avec d'autres algorithmes, tel que le DES, dans la sécurisation des systèmes d'information.

2.5 Résumé du chapitre

Dans ce chapitre destiné à donner quelques notions de base nécessaires à la bonne compréhension de la cryptographie, après avoir explicité quelques définitions, nous avons énoncé les caractéristiques et les qualités nécessaires pour disposer d'un bon algorithme de chiffrement.

Ensuite, suivant la classification basée sur la distinction entre cryptosystème symétrique et cryptosystème asymétrique, nous avons regardé un exemple de chaque catégorie. Les exemples choisis ont été les classiques DES et RSA. Pour chacun d'eux, nous avons donné le principe de fonctionnement ainsi que les faiblesses et les perspectives d'avenir.

3. Utilisation des outils cryptographiques

3.1 Introduction

Pour pouvoir sécuriser quoi que se soit, il faut tout d'abord bien cerner les menaces auxquelles il faudra faire face. On distingue ainsi :

- **attaque passive** : le fraudeur se contente de prendre connaissance des données ;
- **attaque active** : le fraudeur modifie les données.

Ces deux types d'attaque ont amené à définir deux qualités concernant l'information :

- **la confidentialité** : seule une personne autorisée peut en prendre connaissance ;
- l'**intégrité** : seule une personne autorisée peut l'émettre ou la modifier.

Il faut remarquer que sous le terme "modification" de nombreuses variantes se cachent. En effet, si l'on considère un message par exemple, on peut envisager :

- de modifier son contenu,
- de modifier son auteur/destinataire,
- de le dupliquer,
- de le subtiliser ...

Comme nous venons de le voir, la notion de sécurité est une notion très large. Il faut identifier les **menaces** et, pour chacune d'entre elles, trouver et mettre en oeuvre un moyen de la contrer.

Nous envisagerons ici :

- le **chiffrement** pour assurer la confidentialité ;

- l'**identification**, c'est-à-dire le mécanisme permettant de s'assurer de l'identité de son interlocuteur ;

- l'**authentification** qui permet de s'assurer que l'on communique bien avec l'interlocuteur que l'on croit ou que l'on utilise ou lit un document qui vient d'une source autorisée ;

- la **signature digitale** qui permet d'authentifier un document vis-à-vis d'une tierce personne.

3.2 Protocoles de base

3.2.1 Notations

Pour faciliter l'écriture et la lecture des protocoles que nous allons développer dans la suite, nous utiliserons la notation suivante.

A : ... signifie "A effectue les opérations ..."
A —> B : ... signifie "A envoie ... à B"
B <— A : ... signifie "B reçoit ... de A"
?...? signifie un test
!...! exprime ce que l'on peut déduire du test ou du dernier message reçu

3.2.2 Identification

a) Définition

Pour pouvoir envisager la sécurisation d'un système, la première chose qu'il faut être capable de faire avant tout, c'est **identifier** correctement la personne qui veut utiliser le système. Il faut ainsi pouvoir vérifier si une personne qui se présente au système (terminal, distributeur automatique de billets, porte d'accès ...) est bien la personne qu'elle prétend être.

Pour cela, dans la vie courante, nous disposons de moyens bien connus qui sont très mal maîtrisés par les systèmes électroniques. En effet, à n'importe quel moment, nous sommes capables de prouver notre identité en exhibant notre carte d'identité ou passeport. Mais comment pourrait-on "montrer sa carte" à un sytème informatique ?

Plusieurs méthodes sont envisageables et sont classées en 4 catégories :

1. Identification grâce à un élément connu de la personne seule (ex.: un mot de passe) ;

2. Identification grâce à un élément que seule la personne possède (ex.: une carte d'identité) ;

3. Identification grâce à une caractéristique physique de la personne (ex.: empreintes digitales) ;

4. Identification grâce au résultat d'une action involontaire de la personne (ex.: signature manuscrite).

Remarquons que les points 3 et 4 sont souvent assimilés. Maintenant, examinons quelques exemples représentatifs de chaque classe.

b) Identification par les connaissances

1) Mot de passe

Le **mot de passe** et le **PIN** (pour **Personal Identification Number**) représentent sans doute le moyen le plus répandu de vérification d'identité par les connaissances. La plupart du temps, sa longueur est insuffisante pour identifier à coup sûr un utilisateur. Un PIN de 4 chiffres, comme celui utilisé dans les distributeurs automatiques de billets (DAB) ne peut servir d'identifiant (au sens strict du terme) que d'au plus 10^4 utilisateurs, ce qui serait un peu faible comme clientèle d'un réseau de guichets automatiques ! Ainsi, un mot de passe n'est utilisé que pour vérifier si une personne "ment" lorsqu'elle décline son identité, et ceci grâce à une dépendance mémorisée entre mot de passe et identité lors de la distribution de ce mot de passe au client habilité. Le risque que quelqu'un entre le PIN correct alors qu'il n'est pas censé le connaître est ici de 10^{-4}. On calcule ainsi la longueur d'un PIN en fonction du rapport Chance de trouver le PIN / Perte encourue en cas de fraude. Nous n'épiloguerons pas ici sur les faiblesses bien connues du mot de passe.

2) Questionnaire

A chaque demande d'accès, l'utilisateur se voit poser des questions auxquelles lui seul doit être en mesure de répondre (couleur des yeux de la grand-mère maternelle, joueur de basket-ball préféré, ...). Au

besoin, on peut introduire des "mensonges" dans la réponse type attendue par le système.

Un avantage de ce **système de questionnaire** est qu'il est facile de se souvenir des réponses que l'on doit fournir pour s'identifier correctement.

c) Identification par la propriété

1) Cartes à piste(s) magnétique(s)

Les **cartes magnétiques** normalisées sont souvent utilisées en duo avec un PIN. Celui-ci est chiffré et enregistré sur la piste avec l'identité du porteur accrédité. Lors d'un processus d'identification, l'autorité lit la carte, déchiffre le PIN et compare avec le PIN fourni par l'utilisateur. On peut également utiliser, comme dans le cas d'une liste de mots de passe, une fonction à sens unique pour effectuer une comparaison entre élément stocké (supposé correct, car sa modification n'apporterait rien) et élément fourni. Ce mécanisme permet l'utilisation "off-line" d'un processus d'identification, c'est-à-dire sans devoir consulter en direct une liste centralisée des PIN de chaque utilisateur.

2) Cartes à microprocesseur

Une carte à piste magnétique présente un défaut non négligeable : il est relativement facile de la copier. On peut en tout cas acheter l'équipement nécessaire sans problème. Par contre, il est virtuellement impossible de copier ni même de lire une **carte à microprocesseur**. De plus, une telle carte possède des fonctions dynamiques, que nous examinerons dans un chapitre ultérieur, qui lui confèrent un plus considérable quant à la sécurité.

3) Calculette

Dans un système utilisant ce que l'on appelle une "**calculette**", l'autorité, lors d'une demande d'accès, fournit à l'utilisateur certains paramètres ; puis elle demande de calculer une fonction de ces paramètres. L'utilisateur accrédité peut alors donner le résultat requis grâce à l'utilisation d'un petit module de calcul (style calculatrice de poche) qui lui a été fourni au préalable. Auparavant, l'utilisateur peut se voir commander d'introduire un PIN dans sa calculette pour pouvoir l'utiliser.

d) Identification par les caractéristiques personnelles

1) La signature manuscrite

Les **signatures manuscrites** sont effectuées si fréquemment qu'elles deviennent une action réflexe dont certaines caractéristiques sont pratiquement impossibles à reproduire (vitesse d'exécution, force d'appui, rapport de dimension, angles). On fournit donc des modèles de signature au système de vérification qui calcule et mémorise des paramètres caractéristiques de ces exemplaires. Une demande d'accès ultérieure est alors accompagnée de signatures que le système peut comparer avec ses connaissances.

2) Les empreintes digitales

Les systèmes informatiques peuvent utiliser la technique bien connue de toutes les polices du monde, découverte en 1897 par Sir Edward Henry. Les **empreintes digitales** de chaque individu diffèrent par des caractéristiques que l'on peut répertorier (courbes, boucles, lignes qui se terminent ou en rejoignent d'autres...) et qui sont appelée "minutiae". En repérant un certain nombre de ces minutiae, il est alors possible d'identifier avec une grande probabilité un individu.

3) La reconnaissance de la parole

Ce système complexe présente certains désavantages. Il faut prendre le temps de prononcer une phrase caractéristique (souvent plusieurs fois !) et la voix peut être affectée par l'environnement ou l'état physique de la personne (état de santé, émotion provoquée parfois par un essai préalable refusé par l'autorité ...).

4) La rétine de l'oeil

L'utilisateur doit, pour se faire reconnaitre, regarder dans un binoculaire pour permettre au système d'analyser les vaisseaux sanguins de la rétine dont l'agencement est particulier à chaque individu. Le système peut alors comparer le résultat lu aux données qui lui ont été fournies au préalable, lors de l'acceptation de l'utilisateur.

3.2.3 Authentification

a) Introduction

L'**authentification** est le mécanisme qui permet de s'assurer qu'un document (message émis sur un réseau, fichier stocké sur disque, programme à installer, carte à piste ou à microprocesseur...) a bien été émis par une personne autorisée, sans aucune modification illicite,c'est-à-dire qu'il est authentique. L'authentification d'un interlocuteur consiste donc à authentifier son identité. L'authentification est à la fraude ce que les codes détecteurs/correcteurs d'erreurs sont à l'erreur de transmission. En effet, il s'agit dans les deux cas de vérifier si le document n'a subi aucune altération. La différence principale, c'est que si l'on veut frauduleusement changer le contenu d'un document, il est toujours possible de recalculer le code détecteur. Il faut donc rendre impossible cette manipulation. Nous allons donc examiner quelques techniques réalisant cette tâche.

b) Méthodes d'authentification

1) Authentification sans élément secret

En se basant sur l'idée des codes détecteurs, on peut élaborer un système élémentaire permettant de s'assurer de la non modification d'un document : il s'agit tout simplement de calculer un paramètre caractéristique du texte (un code détecteur par exemple), d'envoyer le document sur le canal non sécurisé et de faire parvenir par un moyen sûr ce paramètre. Ainsi, le destinataire pourra recalculer ce paramètre sur le document reçu et le comparer avec celui qu'il a reçu. On a :

(1) A --> B : M (B reçoit M')
(2) A --> B : C=f(M) (par canal sécurisé)
(3) B : ?f(M')=C? !M'=M!.

2) Authentification par chiffrement

Le moyen le plus simple qui vient à l'esprit pour être sûr qu'un document n'a pas été altéré après son émission par une source autorisée, c'est de chiffrer ce document avec une clé connue de l'émetteur (A) et du recepteur (B) autorisé. Ainsi, toute manipulation du cryptogramme par un fraudeur donnerait un texte déchiffré n'ayant plus aucun sens. On a :

(1) A --> B : $E_k(M)$ où k (secrète) connue de A et B
(2) B : $D_k(E_k(M)) = M$

3) Authentification par chiffrement et code détecteur

Cette technique est employée lorsque l'on désire envoyer un message entièrement chiffré. Elle consiste à calculer un paramètre caractéristique du texte clair (ce que l'on appelle un **condensé**) et à envoyer l'ensemble Message-Paramètre sous forme chiffrée. Ce paramètre est souvent appelé **Manipulation Detection Code (MDC)**. On a ainsi :

(1) A --> B : E_k(M , MDC(M)) où k (secrète) connue de A et B

4) Authentification avec fonction cryptographique à sens unique

Cette technique est employée lorsque l'on ne désire pas (ou que l'on ne peut pas) envoyer un message entièrement chiffré. Elle consiste à calculer un paramètre caractéristique du texte clair et à envoyer le texte en clair suivi de ce paramètre appelé dans ce cas **Message Authentication Code (MAC)** ou, parfois, **certificat (authenticator)**. L'élément essentiel à cette technique est bien évidemment le calcul du MAC qui doit pouvoir être paramétré par une clé propre à l'émetteur et au récepteur (sinon il s'agirait d'un simple code détecteur d'erreur !) et qui ne peut être inversible (sinon il serait simple, pour le fraudeur, de trouver un message ayant le même MAC mais lui étant plus favorable). On peut schématiser de la façon suivante :

(1) A --> B : (M , MAC_k(M))
où k (secrète) connue de A et B

On peut, en utilisant le DES, calculer un MAC de la façon suivante :

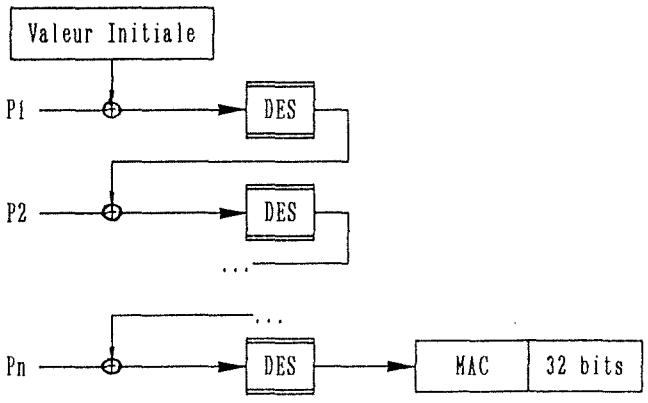


Figure 9 : Schéma de calcul de MAC à base de DES.

On peut remarquer que le calcul d'un **MAC DES** revient à chiffrer le message en mode CBC avec le DES comme algorithme de chiffrement par bloc [ISO 8372,87]. Ainsi, lorsque l'on calcule un MAC DES de cette façon, il n'est pas plus "coûteux" de fournir le texte chiffré, le MAC DES étant constitué des 32 bits de gauche du dernier bloc chiffré.

D'autres protocoles d'authentification seront développés dans la suite. Nous verrons en particulier des protocoles d'authentification d'utilisateur à l'aide de leur carte à microprocesseur.

c) Problèmes posés par l'authentification

1) Problème du rejeu

En général, les protocoles permettent bien d'identifier un document, mais ils sont cependant encore sensibles à une forme de modification : la duplication. On peut très bien en effet simuler un protocole d'authentification sans aucune connaissance en émettant tout simplement les mêmes messages que ceux entendus précédemment. On peut ainsi mystifier un interlocuteur en prenant l'identité d'un autre. Ce genre de problème peut être évité en ajoutant au message, pour le rendre unique, un aléa ou bien la date et l'heure.

2) Problème de dispute

Les protocoles développés ci-dessus permettent à deux interlocuteurs d'être sûrs de la personne avec laquelle ils dialoguent. Mais que se passerait-il si l'un des deux interlocuteurs niait avoir envoyé un message (**problème de non répudiation**) ou s'il forgeait lui-même un message pour prétendre qu'il provient de son interlocuteur ? Aucune personne extérieure ne pourrait arbitrer le conflit (sauf dans le protocole d'authentification avec cryptosystème asymétrique). Il faut pour cela inventer des protocoles d'authentification vis-à-vis d'un tiers. C'est le but de la **signature digitale** qui prouve qu'une personne, et elle seule, était capable d'envoyer un certain document et ce à tout agent intérieur ou extérieur au réseau.

3) Problèmes liés aux fonctions de condensation

Il faut signaler qu'il est très difficile de trouver une bonne fonction de condensation. On en trouve "dix par jour et on les casse le lendemain". De plus, le condensé étant par définition de courte taille, il

fragilise généralement le système. Nous pensons principalement au paradoxe des anniversaires, décrit dans de nombreuses références, qui ouvre la porte à des attaques classiques, du type de l'attaque de Yuval ([DAVIE,84], pp. 116 et 278, [CAMPA,88] par exemple).

3.2.4 Signature digitale

a) Introduction

Comme nous l'avons vu précédemment, le mécanisme d'authentification est insuffisant en cas de dispute au sein des interlocuteurs. Il faut donc trouver des mécanismes logiques (**signature digitale**) permettant de remplacer une signature manuscrite. Pour donner à un texte la force associée à un contrat valable, il faut qu'il soit signé de façon telle que l'ensemble constitué du texte et de sa signature ne puisse avoir été produit que par le seul signataire [MOLLI,83]. De plus, la vérification de la signature devrait pouvoir être effectuée par tout le monde.

b) Schémas de signature digitale

1) Signature par chiffrement asymétrique

En chiffrant simplement le texte avec un cryptosystème asymétrique, on est assuré que seul la personne possédant la clé secrète correspondante a pu émettre le message ([DAVIE,84] p. 267).

(1) $A \dashrightarrow B : E_{skA}(M)$

(2) $B : D_{pkA}(E_{skA}(M)) = M$

2) Signature avec chiffrement asymétrique

Dans la technique ci-dessus, n'importe qui peut lire le message envoyé (clé publique !). Si l'on veut empêcher cette situation, il suffit de chiffrer une deuxième fois avec la clé publique du destinataire. Cette opération assure que lui seul pourra recevoir le document ([DAVIE,84] p. 270 et [MOLLI,83]).

(1) $A \dashrightarrow B : E_{pkB}(E_{skA}(M)) = M'$

(2) $B : D_{skB} (M') = E_{skA}(M) ; D_{pkA}(E_{skA} (M)) = M$

3) Signature par chiffrement symétrique

Lorsque l'on utilise un chiffrement symétrique pour effectuer une signature, il est évidemment nécessaire que la clé soit inconnue des interlocuteurs. A cet effet, on utilise des modules "Tamper resistant" (physiquement sécurisés) ou des cartes à microprocesseur auxquelles nous consacrerons le quatrième chapitre.

mk (Master Key) est cette clé contenue dans le module sécurisé

(1) A : kA = Random ; pkA = E_{mk}(kA , A)

(2) A --> B : pkA , M , E_{kA}(M)

(3) B : D_{mk}(pkA) ; ?D_{kA}(E_{kA}(M)) = M? !Message bien signé!

où A représente l'identité de A

Ce système fonctionne bien pour un petit groupe contrôlé d'utilisateurs, mais non pour un public général ([DAVIE,84] p. 273).

Dans un schéma de ce type, la signature est contrôlée par une autorité extérieure. Ainsi, par analogie à la vie courante, où dans ce cas on utilise un notaire, on parle de service de **notarisation.** Les schémas du même type que celui développé ci-dessus est alors appelé schéma de **signature avec notarisation.**

4) Signature par condensé

Dans les techniques vues précédemment, pour effectuer une signature, tout le message à signer était chiffré. Il est cependant possible, comme dans le problème de l'authentification, de ne chiffrer qu'une partie du message à signer (un condensé) et de l'envoyer en fin de document (comme un certificat, mais ici il s'agira d'une signature : un certificat pour un tiers). On définit ainsi la **signature digitale avec ombre** qui permet à elle seule de retrouver tout le texte signé et la **signature digitale avec empreinte** qui accompagne le texte signé. Rappelons que l'utilisation d'un condensé ouvre la porte à une attaque basée sur le paradoxe des anniversaires.

On peut, par exemple, utiliser le schéma suivant ([DAVIE,84] p. 276) :

(1) A : C = Condensé de M ; S = E_{skA}(C)

(2) A --> B : M , S

(3) B : C = Condensé de M ; C' = D_{pkA}(S) ; ?C=C'? !Signé!

On peut trouver d'autres schémas de signature digitale dans de nombreuses publications. Citons par exemple le schéma de signature de Elgamal [ELGAM,85] ; le schéma de Ong Schnorr Shamir ([DAVIE,84] p. 274) ; la méthode de Rabin, citée dans [MEYER,82].

3.3 Gestion des clés

3.3.1 Introduction

Comme nous l'avons vu dans tout ce qui précède, la sécurité d'un système cryptographique a été, pour des raisons de facilité, concentrée dans le secret de la clé. Il faut donc pouvoir assurer la confidentialité de ces clés ; cela serait facile si elles devaient rester dans un endroit restreint et contrôlé. Mais le but d'une clé, c'est évidemment de voyager d'un interlocuteur à l'autre, pour que chacun puisse se comprendre. Il faut donc pouvoir générer des clés qui ne soient pas trop faibles ; il faut ensuite pouvoir les acheminer en toute sécurité vers leur destinataire légitime et là, les stocker de façon sûre. C'est le problème, crucial en sécurité, de la **gestion des clés (key management)**.

3.3.2 Concepts de base

Les problèmes de chiffrement et déchiffrement sont relativement bien connus et maîtrisés et sont ainsi réalisés à faible coût (tout est réalisé par ordinateur). Par contre, la gestion des clés nécessite l'emploi d'un canal sûr, ce qui était souvent réalisé, à grands frais, par des transports physiques (emploi de coursiers ...). Pour réduire ces frais, les utilisateurs de la cryptographie ont donc essayé de trouver des moyens électroniques sécurisés pour effectuer ces transports. Pour cela, les clés sont envoyées elles-mêmes chiffrées au moyen d'autres clés (appelées alors **Key-Encrypting Keys** par opposition aux **Data-Encrypting Keys**), reportant ainsi le problème de la sécurité sur ces nouvelles clés.

3.3.3 Problème de stockage des clés

Le nombre de clés qui doivent être stockées pose problème. En effet, supposons que n utilisateurs d'un réseau veuillent communiquer entre eux (deux par deux) de façon sûre. Cette hypothèse nécessite le stockage

de (n-1) clés par utilisateur (en supposant qu'une clé serve aussi bien au chiffrement qu'au déchiffrement), donnant ainsi un total de n(n-1)/2 clés, le nombre devient vite énorme lorsque n augmente. On a donc recours à une autorité, chargée de la gestion des clés. Cette autorité aura pour mission de stocker les n clés des utilisateurs du réseau (ces clés sont alors appelées **Terminal Keys**), et les utilisateurs ne devront plus en stocker qu'une (leur Terminal Key). Pour dialoguer entre elles, deux stations peuvent alors demander (dynamiquement) à l'autorité une clé qui ne sera utilisée que pour un dialogue (une **session**), ce qui permet de changer régulièrement de clé (alors appelée **Session Key**) et de minimiser les effets d'une attaque cryptographique réussie. Une section spéciale sera, dans la suite, consacrée à l'examen de quelques protocoles d'échange de clé de session.

Lorsque, au sein d'un même réseau, il n'y a qu'une seule autorité, on parle de **réseau à domaine simple** ; s'il y en a plusieurs, on parle de **réseau multi-domaine**.

Le stockage des clés pose aussi des problèmes de sécurité, tout comme leur transport. On utilise alors une clé pour chiffrer les clés de chiffrement avant de les stocker (ce qui empêche toute manipulation illicite). Cette "super clé", utilisée pour en protéger beaucoup d'autres, est appelée **Master Key**. On parle ainsi de **Host Master Key (clé-maître de l'autorité)** et de **Terminal Master Key (clé-maître d'un utilisateur)**.

Ainsi, en chiffrant une quantité non secrète à l'aide d'une clé secrète (la clé maître) on peut obtenir une clé également secrète. Ce mécanisme est appelé **diversification** de clé car différentes clés sont obtenues à partir d'une seule clé appelée également **clé de diversification**. Ce mécanisme peut être schématisé de la façon suivante :

Clé maître ou clé de diversification (secrète)	Paramètre de diversification (non secrèt)	Clé diversifié
Kd	P1 P2 .. Pn	K1 = E_{Kd} (P1) K2 = E_{Kd} (P2) ... Kn = E_{Kd} (Pn)

Figure 10 : Exemple de diversification de clés.

Ce mécanisme permet de ne plus stocker de façon sécurisée qu'une seule clé, alors que l'on peut en calculer et en utiliser plusieurs.

Ces clés maîtres sont la "clé de voûte" du système de sécurité. Leur connaissance par une personne mal intentionnée peut corrompre totalement toute la sécurité du système. De même, lorsque l'on doit utiliser une clé, il est obligatoire de la recalculer au moyen de la clé maître correspondante. La clé que l'on va utiliser (que l'on appellera désormais **Working Key**) doit donc, à un moment de sa vie, exister en clair, ce qui la rend vulnérable à ce moment-là. On utilise alors ce que l'on appelle un **Cryptographic Facility**, ou un **cryptomodule**, qui est une région sécurisée physiquement dans laquelle on place les éléments critiques à la sécurité (clés maîtres, clé de travail et, souvent, l'algorithme de chiffrement / déchiffrement). L'accès à cette région est alors soumis à un contrôle draconien grâce à un accès par un petit nombre de primitives (opérations) ([MEYER,82]).

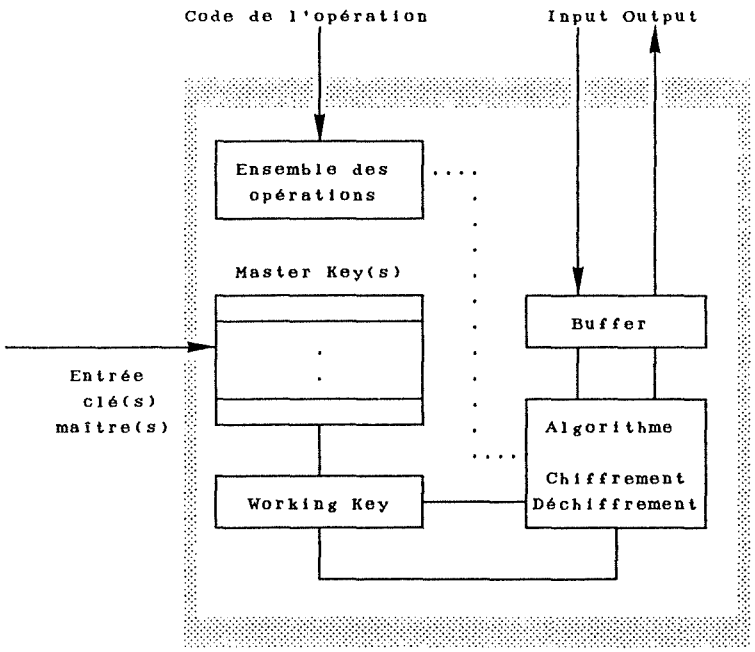


Figure 11 : Représentation d'un cryptomodule.

Des opérations permises pourraient être :

- chiffrement de données
- déchiffrement de données
- charger une clé de travail
- charger une clé maître
- ...

Ainsi, la sécurité du système est basée sur l'hypothèse suivante : il doit être impossible de retrouver des clés en clair en dehors de cette zone sécurisée physiquement.

3.3.4 Génération de clés

La **génération des clés**, comme celle des mots de passe, doit être, si non aléatoire, au moins non prévisible. L'emploi de prénoms (féminins), dates de naissance ou autres numéros de téléphone est évidemment à proscrire ! On utilise donc d'autres méthodes donnant des résultats plus imprévisibles. On utilise pour cela divers moyens de générer des nombres aléatoires imprévisibles.

3.3.5 Distribution de clés

La distribution de clés peut être envisagée selon différents angles selon que la clé à transmettre est une clé publique ou privée et selon que l'algorithme utilisé pour cette transmission est symétrique ou asymétrique. Nous présenterons ci-dessous quelques techniques classiques.

a) Distribution publique de clé : méthode SEEK

[OMURA,88] [NEWMA,87] [DAVIE,84] p. 231

Le système de **gestion de clé SEEK (Secure Electronic Exchange of Keys)** est basé sur la difficulté de calculer un logarithme dans le champ fini des entiers modulo un nombre premier p de grande taille. Cette méthode de distribution de clé secrète par échange de messages non secrets a été décrite par Diffie et Hellman. Examinons-la plus en détail.

	Connu de X	Connu de tous	Connu de Y
Initialement	x	a et p	y
Transmis	ax --> ay	ax et ay	<-- ay ax
Calculé	Z = (ay)x = axy		Z = (ax)y = axy

où x,y sont des random de 512 bits
a est une constante publique de 512 bits
p est un nombre premier de 512 bits
les exponentiations sont effectuées modulo p
Z est donc un nombre de 512 bits

Figure 12 : Schéma de gestion de clé SEEK.

Ainsi, X et Y sont en possession d'un nombre commun sans que quiconque d'autre soit capable de le recalculer, même en connaissant les nombres qui ont transité sur le canal. Ce nombre commun Z peut alors être utilisé comme clé(s) de chiffrement.

b) Distribution de clé de session avec chiffrement symétrique

Ce protocole est basé sur l'article [NEEDH,78] de Needham et Schroeder. Remarquons qu'une amélioration, basée sur l'utilisation de "time stamp" a été apportée à ce protocole dans [DENNI,81].

Supposons que, pour entrer en relation avec B, A demande à une **autorité** AS (pour **Authentication Server**) de lui fournir une clé qu'il utilisera pendant le dialogue prochain avec B. Nous allons examiner un protocole qui permet cette distribution de **clé de session** skAB tout en assurant la confidentialité et l'intégrité du message.

L'Authentication Server est censé connaître les clés kA et kB (Terminal Key) de A et B (les adresses de A et B seront notées A et B dans ce qui suit).

(1) A --> AS : A, B, I_{A1}

où I_{A1} est un identificateur de transaction unique (rejeu)

(2) AS : retrouve kA et kB et génère skAB

(3) AS --> A : E_{kA}(I_{A1}, B, skAB, E_{kB}(skAB, A))

(4) A : D_{kA}(E_{kA}(I_{A1}, B, skAB, E_{kB}(skAB, A)))

= I_{A1}, B, skAB, E_{kB}(skAB, A)

!A authentifie AS! !Seul A peut déchiffrer!

(5) A --> B : E_{kB}(skAB, A)

(6) B : D_{kB}(E_{kB}(skAB, A)) = skAB, A

!Seul B peut déchiffrer!

(7) B --> A : E_{skAB}(IB)

(8) A --> B : E_{skAB}(IB-1) !B authentifie A!

c) Distribution de clé de session avec chiffrement asymétrique

Ce protocole est également basé sur l'article [NEEDH,78] de Needham et Schroeder.

Même situation que ci-dessus mais l'on dispose d'un chiffrement à clé publique.

(1) A --> AS : A, B
(2) AS : peut retrouver pkB
(3) AS --> A : E_{skAS}(pkB, B)
(4) A : D_{pkAS}(E_{skAS}(pkB, B)) = pkB, B !A authentifie AS!
(5) A --> B : E_{pkB}(I_A, A)
(6) B : D_{skB}(E_{pkB}(I_A, A)) = I_A, A
(7) B --> AS : B, A
(8) AS : peut retrouver pkA
(9) AS --> B : E_{skAS}(pkA, A)
(10) B : D_{pkAS}(E_{skAS}(pkA, A)) = pkA, A
(11) B --> A : E_{pkA}(I_A, I_B)
(12) A : D_{skA}(E_{pkA}(I_A, I_B)) = I_A, I_B !A authentifie B!
(13) A --> B : E_{pkB}(I_B)
(14) B : DskB(E_{pkB}(I_B)) = I_B

Comme l'on peut voir, la gestion de clés est un problème complexe. De nombreuses références bibliographiques traitent du sujet et certaines normes existent comme, par exemple, [ISO 8732,88].

3.4 Résumé du chapitre

Nous avons commencé ce chapitre par l'explicitation de deux caractéristiques et qualités fondamentales que l'on peut exiger pour un document, pour de l'information transmise. Il s'agit de la confidentialité et de l'intégrité.

Ensuite, nous avons montré comment les outils cryptographiques examinés dans le chapitre précédent pouvaient être utilisés pour réaliser des protocoles de base assurant intégrité et/ou confidentialité. Ces protocoles sont l'identification, l'authentification et la signature digitale. L'explicitation de méthodes possibles nous a permis de mettre en évidence certains problèmes comme le rejeu, la dispute et les problèmes liés aux fonctions de condensation.

Enfin, nous avons mis l'accent sur un problème fondamental lorsque l'on utilise la cryptographie : le problème de gestion des clés. Nous

avons signalé quelques protocoles plus évolués destinés à résoudre ce problèmes. De plus, nous avons introduit la notion de cryptomodule sur laquelle nous reviendrons par la suite.

4. La carte à microprocesseur

4.1 Introduction

Dans la vie courante, nous utilisons de plus en plus souvent des cartes plastiques. Nous les utilisons pour téléphoner, pour obtenir de l'argent liquide à toute heure, pour payer diverses marchandises, pour pénétrer dans certains lieux ou pour garantir certains chèques. Mais il existe plusieurs types de cartes ; c'est pourquoi, avant de parler de ce qui est actuellement considéré comme le nec plus ultra en la matière - la carte à microprocesseur - nous allons établir une classification de ces cartes (basée sur [AXIS,89], pp. 15-21). A cet effet, nous avons séparé les cartes en deux grandes catégories. D'une part, nous distinguons les **cartes "passives"**, qui ne sont utilisables que pour mémoriser de l'information. D'autre part, nous avons les **cartes "actives"** qui, outre leur capacité de mémorisation, peuvent également accomplir certaines actions que nous examinerons dans le présent chapitre.

4.1.1 Les cartes passives

a) Les cartes plastiques simples

Il s'agit ici d'un simple support plastique, aux dimensions standard, qui porte le logo de l'émetteur et qui permet, par embossage (pressage de caractères en relief), de mémoriser un nombre réduit de caractères. La lecture de ces renseignements se fait ensuite de façon visuelle ou par lecteur mécanique (appelé de façon imagée "fer à repasser").

b) Les cartes à codage optique

Ces cartes utilisent la technique des hologrammes pour mémoriser une quantité énorme d'informations (de l'ordre de 25 Mbytes par centimètre carré). La technologie optique employée rend invisible l'information, ce qui assure déjà un minimum de sécurité.

c) Les cartes laser

Les **cartes laser**, encore appelées **cartes Drexler** du nom de leur inventeur, utilisent la technologie désormais bien connue du "compact disc". Le principal avantage de ces cartes est bien évidemment leur capacité de stockage qui pourrait atteindre 40 Mbytes.

d) Les cartes magnétiques

Les cartes à piste(s) magnétique(s) sont de nos jours les plus utilisées. Il s'agit d'une carte plastique simple à laquelle sont ajoutées deux ou trois pistes magnétiques normalisées par l'I.S.O. (International Standard Organization). Ces cartes se caractérisent par :

- un faible coût de fabrication ;
- une technologie (trop) bien connue et bien maîtrisée ;
- leur portabilité (risque de vol) ;
- leur facilité à se copier ;
- leur taille mémoire limitée (quelques dizaines de bytes par piste).

e) Les cartes à mémoire simple

Ces cartes sont constituées d'un support plastique traditionnel dans lequel on place une mémoire de type mémoire d'ordinateur. Cette mémoire permet de stocker de 256 bits à plusieurs Kbytes.

4.1.2 Les cartes actives

a) Les cartes à logique câblée

Il s'agit ici d'une carte à mémoire simple munie, en plus, d'une logique ayant pour fonction de contrôler les accès à la mémoire. Cette logique étant câblée, il est impossible de la programmer, ce qui restreint la carte à un seul type d'application. Malgré tout, l'adjonction de cette logique rend la carte plus "intelligente", ce qui lui permet d'être utilisée dans des applications demandant un peu plus de sécurité. Les cartes de ce type sont principalement utilisées comme outil de prépaiement. Les cartes de téléphone en France sont l'exemple le plus connu.

b) Les cartes à microprocesseur

La **carte à microprocesseur**, encore appelée **carte à microcircuit**, **carte à microcalculateur**, **smart card** ou **carte à puce** est également obtenue en ajoutant une logique à une carte à mémoire simple. Le trait distinctif de la carte à logique câblée est que cette logique se présente sous la forme d'un microprocesseur qui gère tous les accès à la mémoire. C'est ce type de carte que nous présenterons plus en détail dans la suite de ce chapitre.

c) Les super smart card

On découvre dans diverses publications ([MYERS,89] ou [AXIS,89] p. 19) ce que devient une carte à puce si l'on pousse encore l'intégration de divers éléments. On peut ainsi augmenter la capacité mémoire et ajouter un petit clavier et un écran à cristaux liquides pour transformer les cartes à microprocesseur en véritables ordinateurs sécurisés de poche. L'intégration de tous ces éléments sur une carte plastique classique offre ainsi des perspectives de sécurité encore accrues.

Les caractéristiques de chaque type de carte sont reprises à la figure 13.

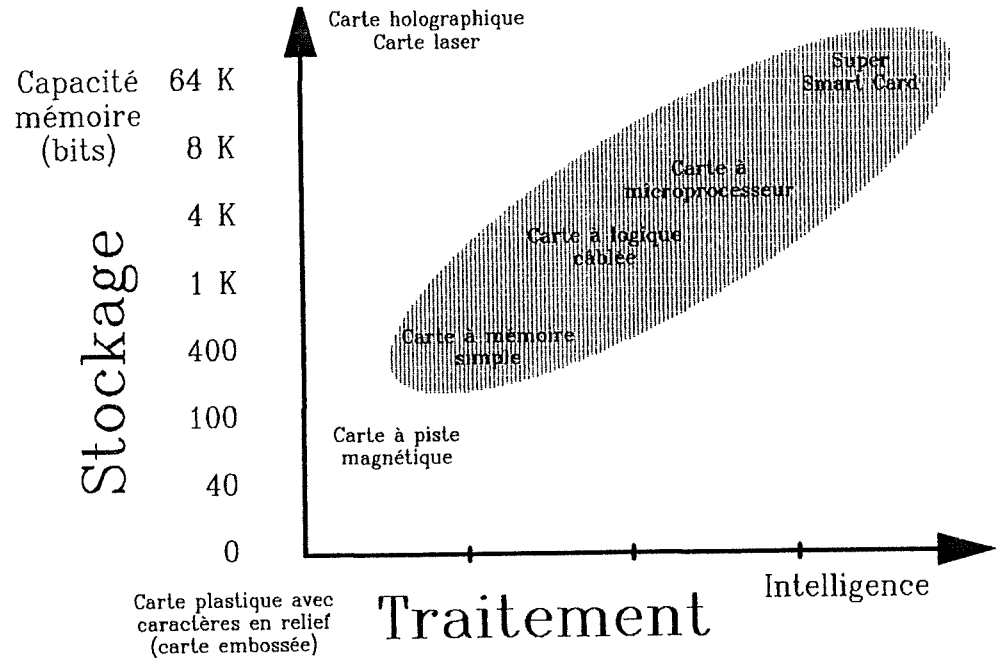


Figure 13 : Caractéristiques des "cartes à mémoire".

4.2 La carte à microprocesseur : facteurs de sécurité

4.2.1 Présentation

a) Historique

Nous nous contenterons ici de citer quelques grandes étapes du développement des cartes à puce.

- 1974 : invention de Roland MORENO (dépôt de plusieurs brevets) ;
- 1978 : démarrage des développements industriels ;
- 1984 : premiers tests de Blois, Caen et Lyon ;
- 1986 : généralisation de la carte à microprocesseur dans le système bancaire français ; généralisation de la carte de téléphone ;
- 1988 : VISA teste la "Super Smart Card".

b) Principe de fonctionnement

Une carte à microprocesseur se présente sous la forme d'une carte plastique standard, portant le logo de l'organisme émetteur et certaines informations concernant le propriétaire (voir figure 14). Dans cette carte, on scelle (en noyant dans de la glu) un microcircuit électronique qui comporte 4 éléments principaux (voir figure 15).

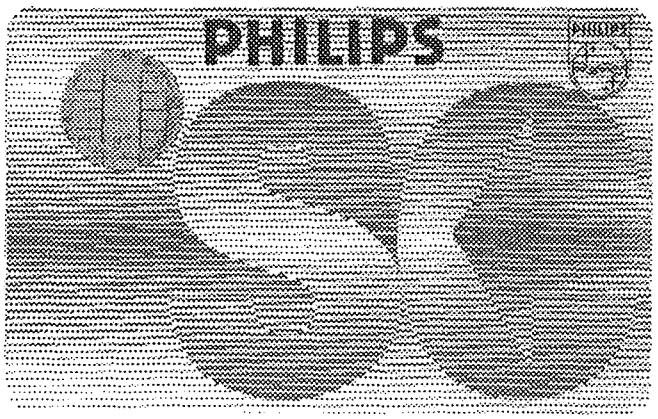


Figure 14 : Carte à microprocesseur.

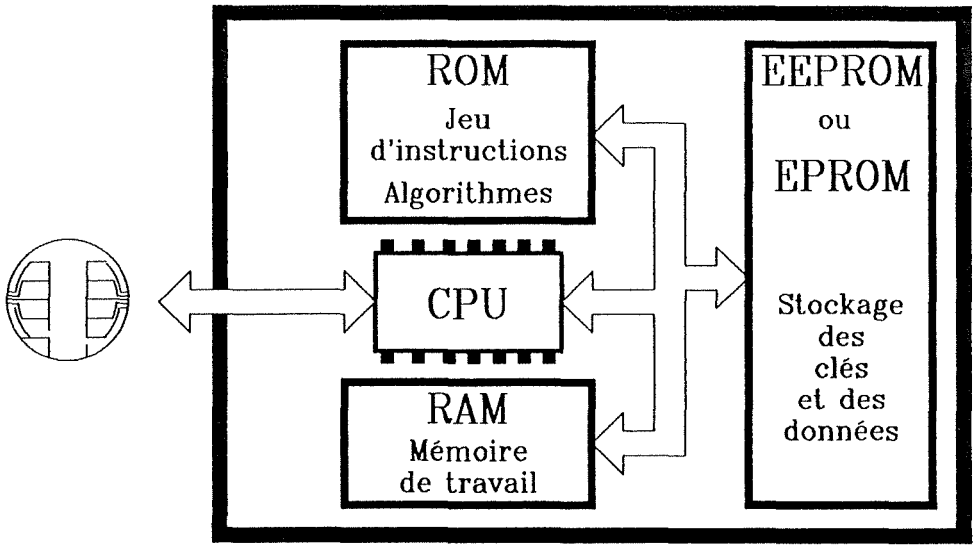


Figure 15 : Carte à microprocesseur - Architecture interne.

1. Un CPU (Central Processing Unit) qui est généralement basé sur un microprocesseur 8 bits de type Motorola 6805, Intel 8048 ou TS 1834 de Thomson ;

2. Une mémoire de type ROM (Read Only Memory de 1,6 à 6 Kbytes) destinée à recevoir le programme qui gèrera les échanges d'information entre le monde extérieur et le microprocesseur. Ce programme, véritable petit système d'exploitation de la carte, est appelé **masque** de la carte ;

3. Une mémoire de type RAM (Random Access Memory de 32 à 160 bytes) destinée à stocker les résultats intermédiaires des calculs effectués par le processeur (mémoire de travail) ;

4. Une mémoire de stockage de type EPROM (Erasable Programmable Read Only Memory de 1 à 8 Kbytes) ou EEPROM (Electrically Erasable Programmable Read Only Memory d'environ 2 Kbytes) destinée à recevoir les données relatives au propriétaire ou à l'application.

Le CPU est relié au monde extérieur, via un contact externe visible sur la carte que l'on appelle **vignette**, par 8 contacts dont deux sont réservés pour usages futurs. Les six autres sont utilisés pour

l'alimentation, la tension de programmation, la masse, la remise à zéro, le signal d'horloge et une ligne série pour les entrées/sorties.

c) Normes

La fabrication des cartes à microprocesseur est régie par trois normes principales ([GUEZ,88] pp. 173-175, [GUILL,89]).

La norme ISO 7816/1 définit les caractéristiques physiques de la carte (sensibilité aux Ultra-Violets, aux rayons X, à l'électricité statique, au champ magnétique ainsi que résistance mécanique).

La norme ISO 7816/2 donne la dimension, le nombre, l'emplacement et l'attribution des contacts externes (normalisation de la vignette en quelque sorte).

La norme 7816/3 spécifie les signaux électroniques et les protocoles d'échange entre le microcircuit et le monde extérieur (valeur des tensions et des courants, synchronisation des signaux, protocoles de transmission synchrone et asynchrone).

4.2.2 Sécurité physique

La sécurité dont jouit la carte à puce face aux attaques physiques est due principalement à la taille du composant. En effet, pour pouvoir pirater la carte, il faut tout d'abord accéder au processeur et cela n'est pas chose aisée lorsque l'on est dans des dimensions de l'ordre du millimètre carré pour un circuit intégré. De plus, l'architecture du circuit n'est pas une architecture classique, mais elle contient toute une série de "pièges" et de "leurres" destinés à rendre encore plus difficile une investigation par un fraudeur. Il faut ensuite pouvoir espionner le fonctionnement de la carte et pour cela disposer d'un matériel rare et coûteux : un microscope électronique avec micromanipulateurs complexes et effet stroboscopique ! Ainsi, les moyens techniques et la compétence nécessaires rendent la fraude vaine et/ou inutile de par son coût [GOUDE,89].

4.2.3 Sécurité logique

Comme une attaque physique est exclue, le moyen le plus simple pour accéder aux informations stockées dans la puce, ce sont les six

contacts externes. On se contente ainsi d'attaquer le circuit de façon logique. Mais ce type d'attaque est également voué à l'échec, car tous les dialogues passent par le processeur qui opère alors comme un gendarme en autorisant ou non les requêtes qu'il reçoit, selon ce que lui dicte le masque de la carte. Le processeur est un point de passage obligé entre l'extérieur et la mémoire, et il n'autorise qu'un nombre réduit d'accès aux différentes parties de la mémoire selon les circonstances (mot mémoire auquel on veut accéder, type d'accès, état de la carte, porteur identifié ou non). Ainsi, le processeur possède le moyen de modifier son comportement en fonction des événements. C'est pour cette raison que l'on parle parfois de **MAM** (**Microcalculateur Autoprogrammable Monolithique**) pour désigner la "puce".

4.2.4 Cycle de vie

Une caractéristique essentielle qui distingue la carte à puce est également sa sécurité assurée à toutes les étapes de sa vie. En effet, les cartes à puce sont protégées par une clé dès la sortie d'usine des "chips". Nous allons examiner le cycle de vie d'une carte à microprocesseur.

A chaque étape du cycle de vie d'une carte, on gère le processeur de façon telle que les autorisations d'accès à la mémoire soient transférées du fabricant à l'utilisateur de façon sûre.

Le cycle de vie d'une carte, résumé par la figure 16, se caractérise par cinq étapes que nous allons examiner plus en détail.

a) Phase de fabrication

Tout d'abord, il faut fabriquer les composants (microcircuits et vignettes) et les cartes que l'on dotera d'un logo spécifique.

Dès que les composants sont fabriqués, ils sont testés et protégés par l'écriture d'une **clé de production** dans leur mémoire. Désormais, toute action nécessitera la présentation de cette clé (et cela avant même que le composant soit placé sur une carte !). Cette clé est enregistrée dans un composant particulier qui servira plus tard à débloquer les autres cartes.

b) Phase d'encartage

Les composants sont intégrés dans les cartes et l'ensemble est à nouveau testé.

c) Phase de pré-personnalisation

Les cartes sont regroupées en lot. A l'aide de la clé de

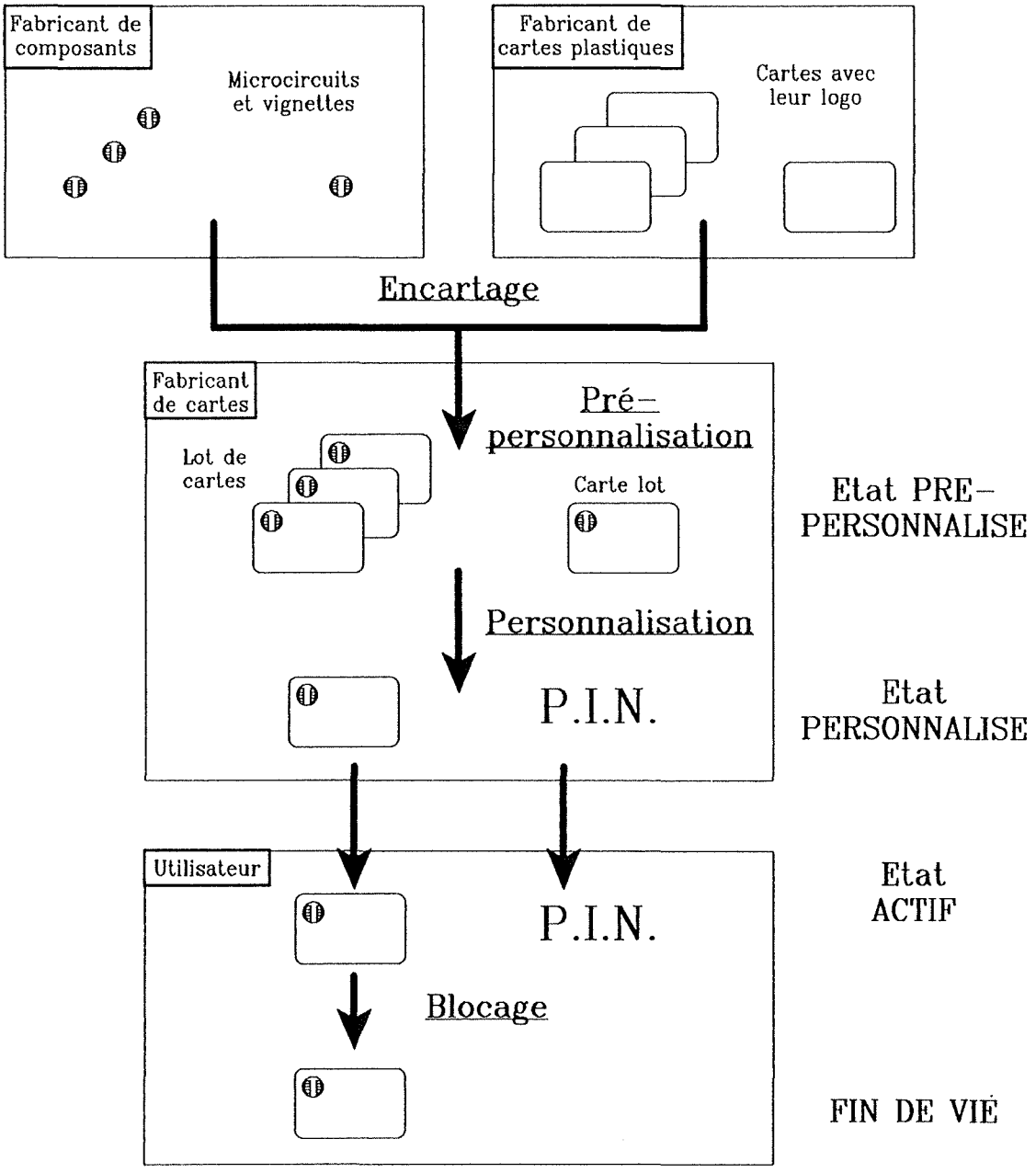


Figure 16 : Cycle de vie d'une carte à microprocesseur.

production, les cartes sont débloquées pour pouvoir les doter d'un numéro d'identification. Chaque lot de carte peut ensuite être fourni avec une carte

particulière, la **carte lot**, qui contient la clé qui sera utilisée ensuite pour débloquer les cartes du lot associé.

d) Phase de personnalisation

Durant cette phase, les cartes sont débloquées grâce à la carte lot et divers renseignements nécessaires lors de l'utilisation active de la carte sont stockés. Il s'agit principalement des clés de contrôle d'accès qui remplaceront la clé de production dans les accès futurs, du PIN qui permettra d'utiliser la carte et de divers renseignements concernant le futur utilisateur de la carte.

e) Vie active, blocage et fin de vie

Une fois la carte personnalisée, on peut la donner, accompagnée de son PIN, à son propriétaire qui pourra dès lors l'utiliser. Durant sa vie, une carte peut décider de se bloquer après trois présentations de PIN erroné. Dans ce cas, il est nécessaire de retourner auprès de l'autorité qui a délivré la carte et qui est seule capable de la débloquer.

La carte termine sa vie dans deux cas principaux : lorsque sa mémoire (ou certaines parties de celle-ci) est remplie ou lorsqu'un verrou particulier invalide la carte. Une carte peut, par exemple, s'auto-verrouiller si elle détecte une attaque entre sa fabrication et sa réception par son propriétaire.

f) Conclusion

Tout ce que nous venons d'exposer nous montre que la carte à microprocesseur peut être considérée comme un cryptomodule portable. En effet, la carte répond à toutes les caractéristiques nécessaires à un bon cryptomodule :

- possibilité de stocker clés et algorithmes ;
- possibilité d'exécuter ces algorithmes ;
- sécurisation tant logique que physique des deux points précédents.

La portabilité de ce cryptomodule ouvre de nouvelles perspectives d'utilisation : porte-clés sécurisé (pour la transmission de clés), porte-monnaie inviolable, carte d'identification inviolable.

4.3 Carte DES-D2 de Philips

4.3.1 Présentation

Après avoir montré de façon générale les capacités statiques de sécurisation (sécurité physique et logique, cycle de vie) des cartes à microcircuit, nous allons examiner les capacités de sécurité dynamique de ce type de carte sur un exemple bien particulier - la carte DES-D2 de Philips - car c'est cette carte que nous utiliserons dans la suite de nos travaux.

4.3.2 Caractéristiques

La carte D2 est basée sur l'architecture physique classique que nous avons présentée (MAM). Dans la suite de notre exposé, nous ne nous attarderons que sur certaines spécificités du masque D2 qui nous seront utiles dans les deux prochains chapitres.

a) Mémoire

Le microcircuit possède :

- 4 Kbytes d'EPROM (pour les données) ;
- 3 Kbytes de ROM (pour le masque) ;
- 76 bytes de RAM (pour les résultats intermédiaires).

La mémoire EPROM est divisée en 1024 mots de 32 bits. La topologie de la mémoire EPROM est schématisée à la figure 17. On peut y déceler, entre autres choses :

1. La zone de stockage de la clé de production.
2. La zone de stockage des **clés de base** de la carte.
3. Une zone réservée au stockage d'algorithmes optionnels supplémentaires. Il s'agit en fait de la possibilité d'étendre le jeu d'instructions disponible d'origine sur la carte par ajout de fonctions personnalisées (possibilité de **coding optionnel**).
4. La zone de stockage des PIN permettant d'utiliser la carte. Le PIN peut être modifié un certain nombre de fois.

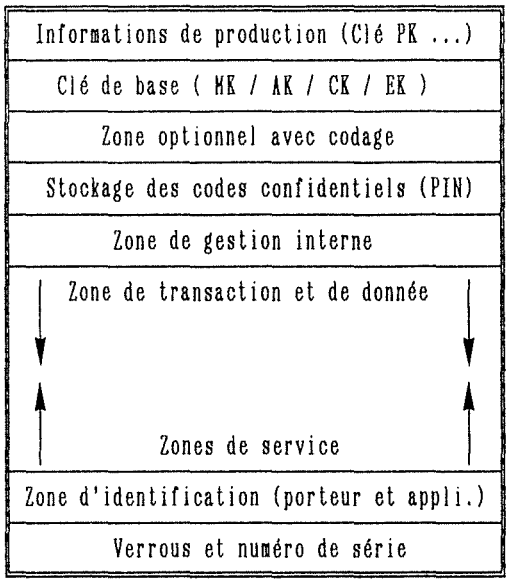


Figure 17 : Structure de la mémoire EPROM d'une carte DES D2.

5. Une file de mots destinés au stockage des données applicatives.

6. Une pile de zones (blocs de mots) appelés **zone de service**. Chaque zone de service possède, en plus des clés de base, deux clés propres appelées **clés de service**. Cette précaution permet de disposer de clés, d'espaces de stockage et de règles d'accès bien distincts pour différentes applications et de renforcer ainsi la sécurité entre les différentes utilisations de la carte.

La partie utile d'un mot mémoire, comme on peut le voir à la figure 18, est limitée à 28 bits par la présence, dans chaque mot, de quatre bits utilisés par le masque pour gérer les règles d'accès au mot. Ces bits sont dès lors appelés **bits système**. Leur signification est résumée à la figure 19.

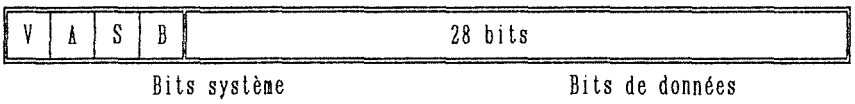


Figure 18 : Format d'un mot mémoire de la carte DES.

Bit	Signification	Règles d'accès
V = 0	Mot validé	définies par ASB
V = 1	Mot non validé	lecture libre / écriture selon ASB
A = 0	Zone de service	selon règles définies dans le header
A = 1	Zone de transaction	accès après présentation du PIN
S = 0	Mot secret	accessible uniquement par le CPU
S = 1	Mot non secret	accessible par l'extérieure (et le CPU)
B = 0	Début de bloc (header)	
B = 1	Intérieur de bloc	

Figure 19 : Signification des bits système de la carte DES.

b) Clés

Les clés principales définies dans une carte DES-D2 sont :

- la clé de production (PK) ;
- la clé maître (MK) ;
- la clé d'ouverture de zones (AK) ;
- la clé de certification (CK) ;
- la clé d'échange (EK) ;
- deux clés de zone de service (SK1 et SK2).

Leur emploi est résumé à la figure 20.

Fonctions	Clés utilisables	
	de base	de service
Sécurisation entre production et personnalisation	PK	
Certification : - authentification - lecture certifiée / chiffrée	CK	SK1
Recyclage d'une carte après 3 erreurs sur PIN	MK + PIN	
Ecriture protégée : - ouverture des zones de service - écriture des clés de service	MK AK	
Mise à jour protégée des données dans une Z de S		SK1
Transfert de clés temporaires sous forme chiffrée	EK	SK2
Chiffrement / Déchiffrement / diversification	Clés temporaires EK	SK2
	Clés temporaires	

Figure 20 : Résumé des clés d'une carte DES.

4.3.3 Commandes

Les phases que parcourt une carte sont caractérisées, entre autres choses, par l'ensemble des fonctions accessibles. La figure 21 reprend les fonctions d'origine de la carte D2 en spécifiant la période à laquelle elles peuvent être utilisées.

Fonction	Perso	Active	Fin	Fonction	Perso	Active	Fin
Alternate reset of blocked card		X		Protected write under MK		X	
Certify word with CK	X	X	x	Protected write under PK	X		
Certify word with SK1		X	x	Protected update under SK1		X	
Compare MAC result		X		Read in card	X	X	x
Compute MAC		X		Read in service zone	X	X	x
Decryption with temporary key		X		Request MAC counting value		X	x
Diversify key		X		Request result	X	X	X
Encrypted modification of PIN		X		Reset	X	X	X
Encrypted presentation of PIN		X		Reset of blocked card		X	
Encrypted reset of blocked card		X		Search on argument in memory	X	X	x
Encryption with temporary key		X		Search on argument with mask	X	X	x
Generate random		X		Start MAC		X	
Generate temporary key with EK		X		Unlock card for personalisation	X		
Generate temporary key with SK2		X		Write locks and certify	X	X	X
PIN modification		X		Write word in card	X	X	
Presentation of PIN		X		Write word in service zone		X	
Protected write under AK		X					

Figure 21 : Fonctions de la carte DES D2.

4.3.4 Fonctions dynamiques de sécurité

Voici maintenant des exemples d'utilisation de la carte D2 qui nous permettront de montrer son apport de sécurité dans quelques cas concrets. Cet exposé pourra également constituer un bon exercice de mise en pratique des fonctions énoncées dans la section précédente. Par ailleurs, ces protocoles seront utilisés dans les deux chapitres suivants.

a) Identification

Il s'agit de s'assurer que la personne qui détient la carte est bien son propriétaire légitime.

Avec les cartes à puce et contrairement aux cartes magnétiques, l'identification du propriétaire est effectuée par la carte elle-même, et non par le terminal. Le schéma classique d'identification est ainsi résumé par la figure 22. Le PIN est introduit au clavier et transmis à la carte qui peut alors elle-même le comparer avec le PIN qui a été précédemment stocké dans la mémoire protégée du composant.

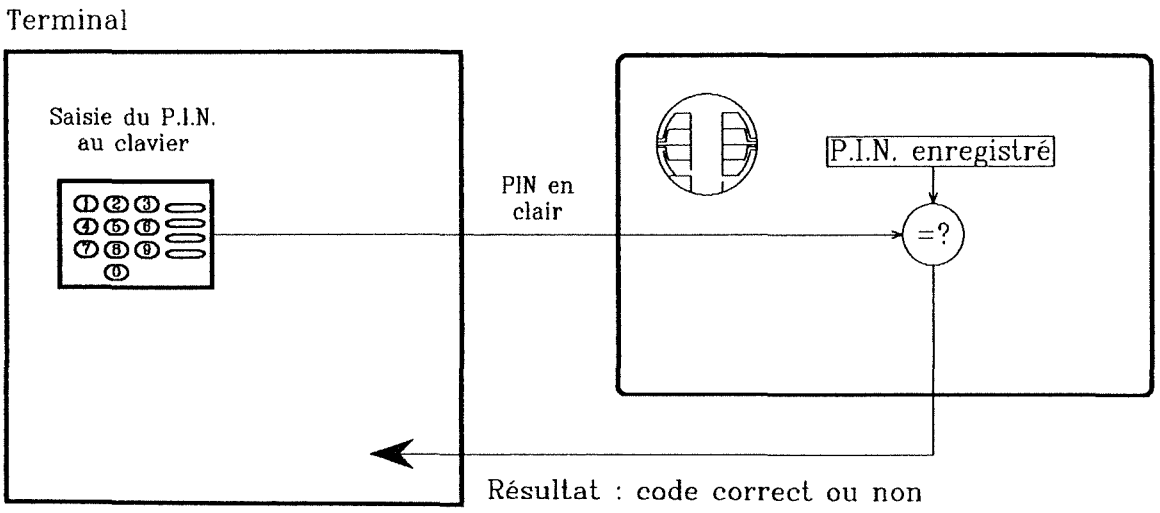


Figure 22 : Identification classique.

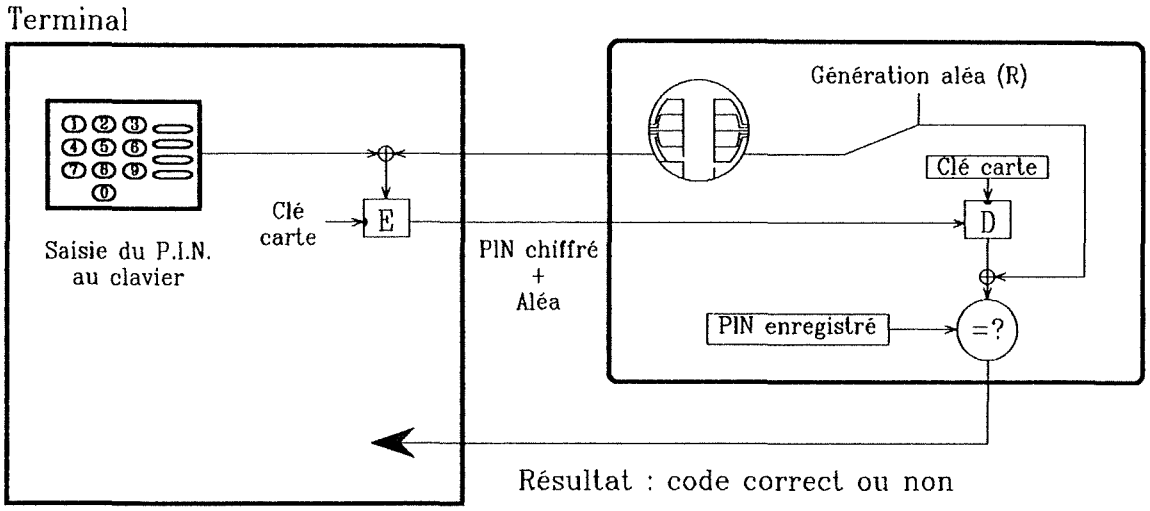


Figure 23 : Identification avec algorithme de chiffrement.

L'intégration d'un algorithme de chiffrement réversible dans le masque de la carte D2 permet un schéma d'identification encore plus sûr que le précédent (voir figure 23). Un aléa R est généré par la carte et transmis au terminal qui effectue un OU-Exclusif de R avec le PIN saisi au clavier. Le résultat est ensuite chiffré avec une clé partagée par la carte et le terminal, et transmis à la carte. Celle-ci peut alors déchiffrer ce qu'elle vient de recevoir, extraire le PIN qu'on lui a envoyé et comparer avec le

PIN dont elle dispose dans sa mémoire protégée. Ce schéma possède deux avantages. D'une part, le PIN ne transite pas en clair et l'utilisation d'un aléa empêche le rejeu. D'autre part, ce protocole permet à la carte d'authentifier le terminal grâce à la clé que les deux interlocuteurs se partagent.

b) Authentification

Dans le jargon lié à la carte à puce, une authentification consiste à prouver à une entité (un terminal par exemple) qu'une carte n'a pas été falsifiée et a bien été émise par une source autorisée. Un protocole d'authentification classique de carte est résumé à la figure 24. Un terminal génère un aléa R qu'il envoie à la carte qui peut le ré-envoyer après l'avoir chiffré avec une clé qu'elle partage avec le terminal. Celui-ci peut alors déchiffrer ce qu'il reçoit et comparer le résultat avec l'aléa d'origine. Il n'y aura égalité que si la carte dispose de la clé nécessaire, ce qui prouve son authenticité.

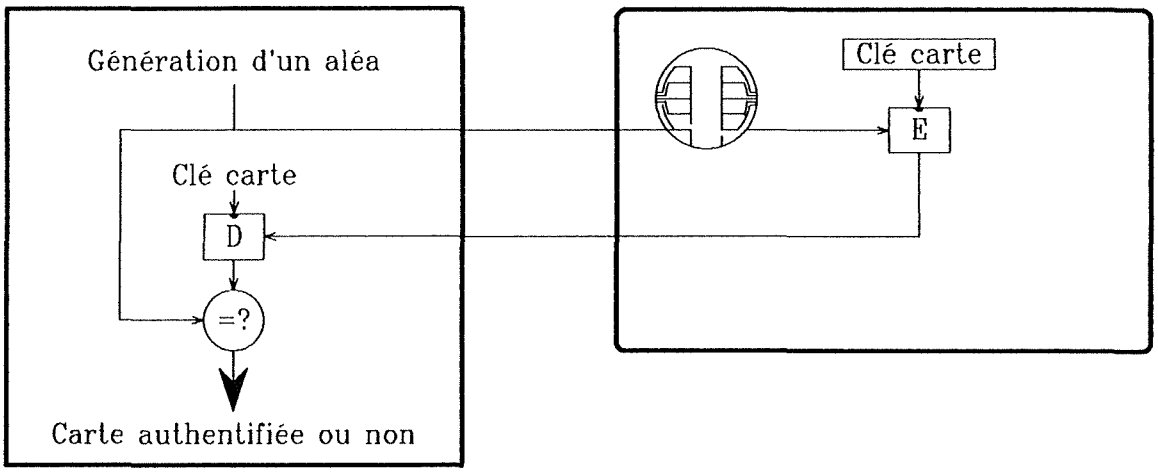


Figure 24 : Authentification d'une carte à microprocesseur.

c) Certification

La certification consiste à vérifier qu'une donnée se trouve bien stockée à un endroit particulier de la carte. Si cette donnée est le solde d'un compte par exemple, il est important de vérifier s'il a été correctement inscrit dans la carte. Un protocole possible de certification est montré à la figure 25. Il est semblable au protocole d'authentification décrit dans la section précédente mais, dans ce cas-ci, c'est le contenu d'un mot mémoire donné qui est chiffré et comparé. De plus, la comparaison est effectuée sur

les quantités chiffrées plutôt que sur les quantités en clair. Cette technique, qui aurait pu être appliquée dans le protocole d'authentification, permet d'implémenter ce type de protocole en utilisant un algorithme cryptographique non réversible. C'est le cas dans les cartes à microprocesseurs CP8 de Bull qui disposent de l'algorithme secret Télépass qui, dans sa première version, est non réversible.

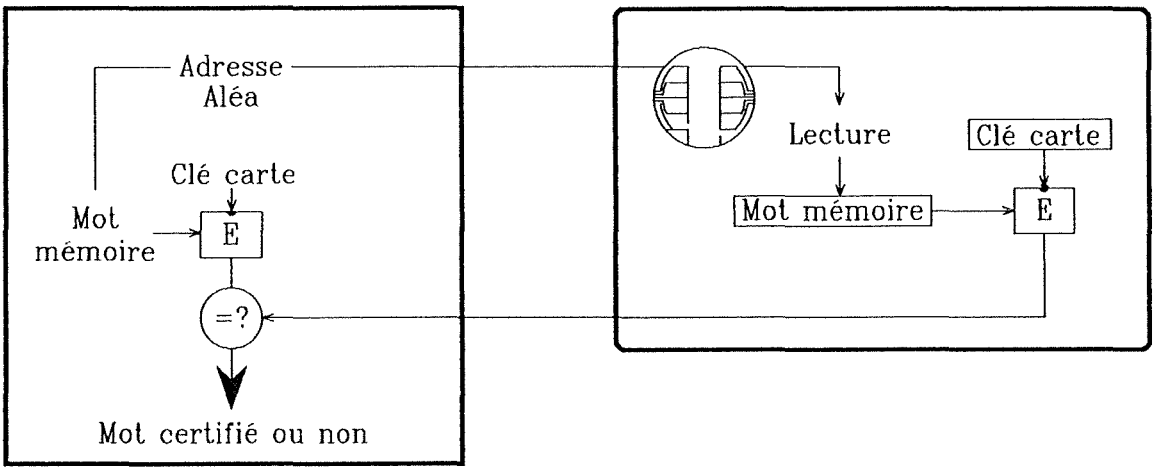


Figure 25 : Certification d'un mot mémoire.

d) Scellement de message - Signature digitale

La carte DES-D2 permet, comme nous l'avons schématisé à la figure 26, d'assurer l'intégrité de documents transmis. En effet, elle dispose, dans ses primitives, du moyen de calculer un MAC sur le document grâce à son algorithme DES incorporé. De plus, comme la clé qui sert au calcul du MAC peut rester secrète même aux yeux du propriétaire de la dite carte, il est possible de créer une carte capable de calculer un MAC et une autre capable de le vérifier. Cette précaution permet de réaliser une signature digitale notarisée du document.

e) Chiffrement

La carte DES-D2 possède un algorithme DES intégré qui lui permet, comme nous le montre la figure 27, de chiffrer un document.

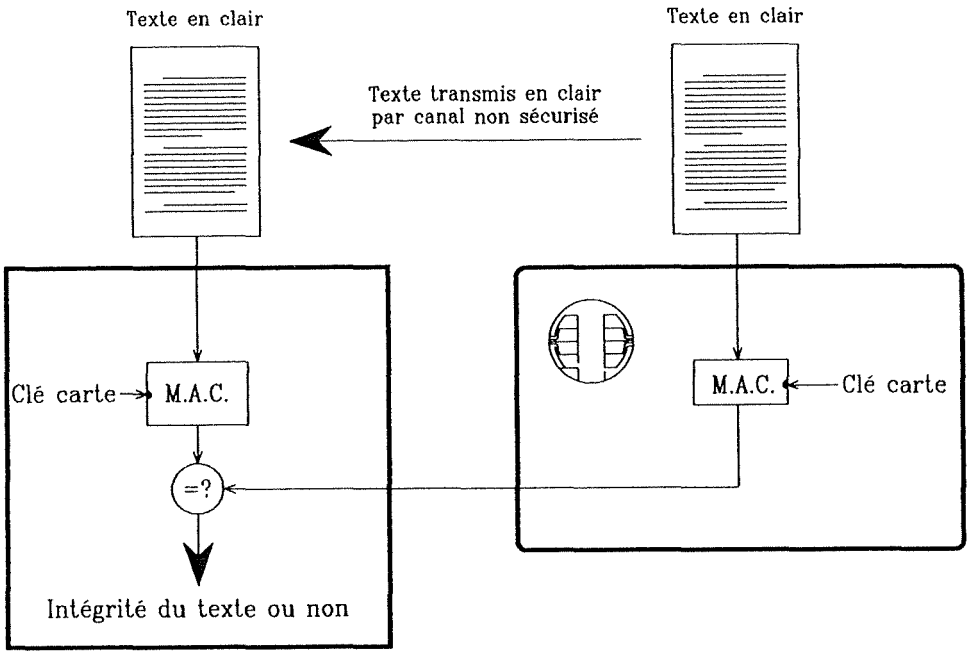


Figure 26 : Scellement de message avec une carte à microprocesseur.

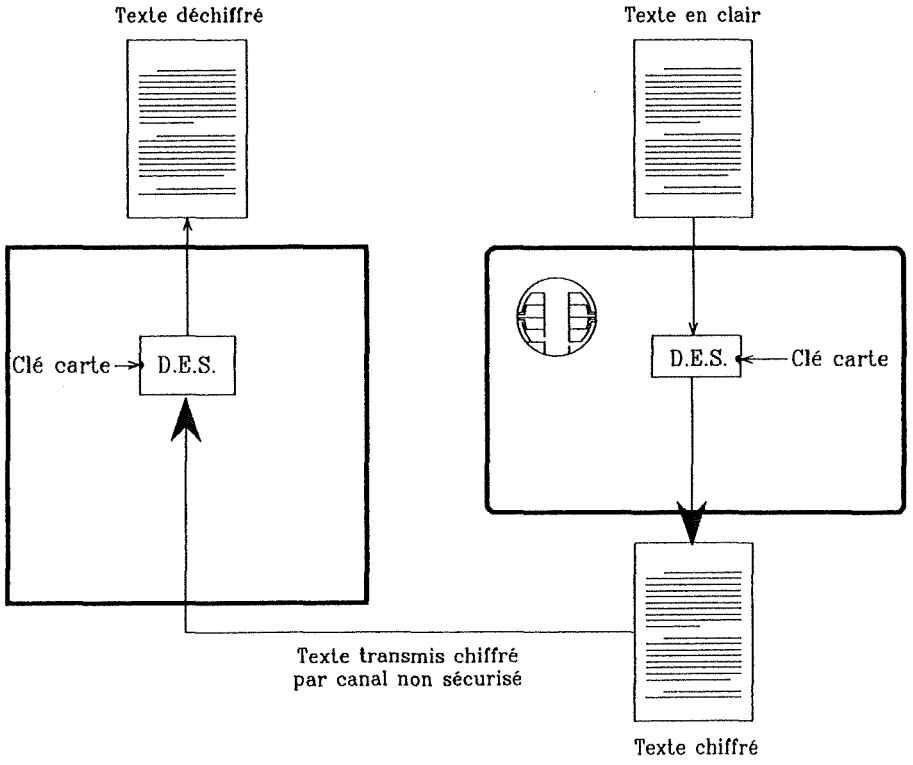


Figure 27 : Chiffrement de message avec une carte à microprocesseur.

4.4 Problèmes non (encore) résolus par la carte

4.4.1 Capacités

Les capacités des cartes à microprocesseur sont encore, à l'heure actuelle, relativement limitées. Nous pensons ici à la capacité des mémoires utilisées et à la rapidité des cryptosystèmes intégrés. Ces derniers sont soit limités du point de vue de leur utilisation (non inversibles, par exemple), soit limités du point de vue de leur débit. De plus, l'intégration d'algorithmes plus complexes, comme le RSA ou les protocoles à apport nul de connaissances, pose encore d'énormes problèmes [FERRE,89a]. Il est par exemple possible d'intégrer le RSA dans une carte à puce (de type D2) mais personne n'a encore réussi à optimiser sa taille mémoire, ce qui a comme conséquence fâcheuse qu'il occupe alors entièrement l'espace mémoire de la carte, la privant de ce fait d'une grande partie de ses capacités [SECINF,89].

4.4.2 Rigidité du masque

Le masque est, rappelons-le, le "système d'exploitation" de la carte. Il est microcodé et placé en mémoire ROM. Cette situation a deux conséquences sur le masque : d'une part, sa taille est limitée par la taille de la ROM disponible et, d'autre part, changer de masque n'est pas une opération aisée (sans compter le coût de développement du nouveau masque). Cela se traduit par des cartes aux possibilités qui se veulent générales et qui sont rigides et encore relativement restreintes. Les fabricants essaient cependant de remédier à ce problème de diverses façons ; la possibilité de coding optionnel de la carte D2 en est un exemple.

4.4.3 Nécessité d'un "répondant"

Comme on peut le voir dans les différentes figures résumant les capacités dynamiques de sécurité des cartes, la carte dialogue toujours avec une seconde entité. Cette entité dispose, selon les cas, d'un clavier, d'un algorithme de chiffrement et des clés associées. La sécurité maximale de l'ensemble étant conditionnée par la sécurité du plus faible de ses composants, il faut donc utiliser une entité aussi sûre que la carte à microprocesseur. Cette entité, qui est le "répondant" de la carte doit donc

être un cryptomodule. On utilise alors ce que l'on appelle un **processeur de sécurité associé** (**PSA**) qui peut être lui-même une carte à puce ou uniquement une puce de carte incorporée dans les circuits du terminal. Dans ce dernier cas, on parle parfois de MCSD, pour **Micro Card Security Device**, ou plus spécifiquement de **PSMC**, pour **Philips Security Micro Controler**.

Cependant, si ce "répondant" est utilisé pour de nombreux dialogues avec différentes cartes, ses capacités de stockage et de calcul doivent être supérieures à celles d'une simple carte. Il faut donc souvent utiliser un processeur de sécurité associé beaucoup plus puissant qu'une carte.

4.5 Résumé du chapitre

Ce chapitre, consacré à la carte à microprocesseur, a débuté par un rappel succinct des différentes cartes utilisables de nos jours. Nous avons ainsi pu différencier les cartes à mémoire simple des cartes à mémoire dites "intelligentes", parmi lesquelles se range la carte à microprocesseur. Ensuite, nous avons présenté la technologie des cartes à microprocesseur, ce qui nous a permis de mettre en évidence sa capacité de sécurisation statique et de l'assimiler dès lors à un cryptomodule portable.

La suite du chapitre était destinée à la présentation de la carte DES-D2 de Philips. Nous avons ainsi décrit brièvement son fonctionnement interne et ses capacités dynamiques de sécurisation. Cela nous a donné l'occasion de montrer des protocoles classiques - tels que l'identification, l'authentification, la certification de message - dont la connaissance sera utile dans les chapitres suivants.

Enfin, après l'exposé des qualités de la carte à microprocesseur quant à son apport non négligeable au niveau sécurité, nous avons mis l'accent sur les problèmes qu'elle ne résout pas encore, ainsi que sur les problèmes nouveaux que son utilisation induit. Ceci nous a donc permis de montrer la nécessité, dans certaines circonstances, d'un dispositif encore plus "puissant".

5. Serveur de sécurité

5.1 Introduction

Nous avons vu dans les chapitres précédents, qu'il est impossible de concevoir un dispositif de sécurité informatique basé uniquement sur de la protection logique des données. Pour assurer un niveau de sécurité acceptable, il est nécessaire de protéger physiquement certaines données, certains algorithmes, et ce problème n'est pas toujours entièrement résolu par l'usage d'une carte à microprocesseur.

Nous allons maintenant présenter un dispositif matériel, un processeur de sécurité qui, sous certains aspects, ressemble à une "super carte à microprocesseur". Ce matériel, une fois utilisé avec des cartes à puce et piloté par un ensemble logiciel adéquat, sera capable de fournir à son propriétaire un niveau élevé de sécurité. Par la suite, nous parlerons de **système SPHINX**, car comme ce dernier, notre dispositif est un gardien tranquille habile à poser des questions, des énigmes, difficiles ([LAROU,68], p. 877).

5.2 Philosophie / But

5.2.1 But

Le dispositif Sphinx a toujours été développé dans l'optique de fournir les bases solides nécessaires pour faire tourner des **applications sécuritaires de haut niveau**. Il est en effet impératif, en informatique comme partout ailleurs, que la sécurité soit pensée dès le départ, aux plus bas niveaux, sous peine de bâtir un bel édifice sur des fondations inadéquates qui le feront s'écrouler à la moindre sollicitation. Une sécurité sérieuse ne peut pas être obtenue par une "couche" que l'on rajoute à un système existant ; cette stratification ne fournirait qu'un dangereux sentiment de sécurité à l'utilisateur.

5.2.2 Application sécuritaire de haut niveau

Nous allons maintenant expliciter ce que nous entendons par le terme d'"application sécuritaire de haut niveau".

Pour la sécurité logique, la cryptographie fournit les briques de base nécessaires. Ces briques sont assemblées pour fournir les protocoles de base que sont l'identification, l'authentification et la signature digitale. Les protocoles de base fournissent des moyens de sécurisation entre deux entités, principalement. Mais la tendance actuelle à l'ouverture des systèmes informatiques implique la sécurisation d'un nombre sans cesse croissant d'entités. Il est donc nécessaire de créer et d'utiliser des protocoles de sécurité toujours plus complexes bien que toujours basés sur les protocoles de base. Ces protocoles plus complexes sont, par exemple, des protocoles de gestion de clé et des protocoles de notarisation dont nous avons parlé précédemment. Dans la section suivante, nous allons donner un exemple de protocole de haut niveau.

5.2.3 Kerberos : un service d'authentification pour réseaux ouverts ([STEIN,88])

Le but du système Kerberos ([STEIN,88]) est de distribuer, de façon sécurisée, des tickets infalsifiables permettant l'accès à des serveurs se trouvant sur un réseau ouvert. Ce système est basé sur le modèle de Needham et Schroeder ([NEEDH,78]) que nous avons déjà présenté.

Le système Kerberos fonctionne en cinq ou six phases, qui sont schématisées à la figure 28, et que nous allons examiner plus en détail.

a) Entités échangées

Au cours des dialogues, deux entités essentielles sont manipulées : il s'agit du **ticket** et de l'**authentifieur**.

Un ticket est un "pattern" qui donne le droit à un client c d'accéder à un serveur s. Il a la forme suivante :

$\{ s , c , adr , TS , vie , K_{c,s} \} K_s$ noté $\mathbb{T}_{c,s}$

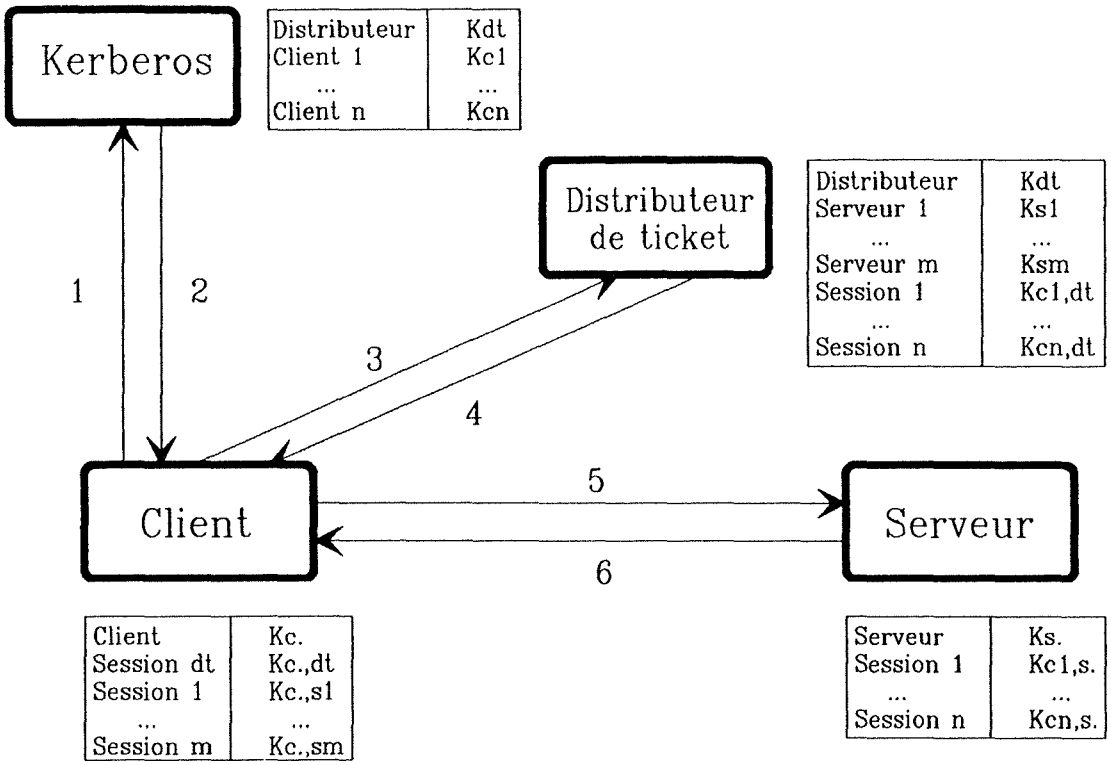


Figure 28 : Protocole Kerberos.

Un authentifieur, quant à lui, permet au client c de se faire reconnaître du serveur s et a la forme suivante :

$\{ c , adr , TS \} K_{s,c}$ noté $\mathbb{A}_{c,s}$

Conventions de notation

s = identité du serveur
c = identité du client
adr = adresse du client dans le réseau
TS = Time Stamp
= heure de création du pattern
vie = durée de vie du ticket
$K_{x,y}$ = clé de session utilisée entre x et y. La génération d'une clé de session se fait de manière (pseudo-) aléatoire
K_x = clé de x
$\{ abc \} K_x$ = abc chiffré avec la clé de x

b) Obtention initiale d'un ticket

Avant d'obtenir un ticket d'accès à un serveur, un client doit d'abord obtenir le droit d'accéder à un distributeur de tickets. Ce dernier étant également un serveur, il faut obtenir un ticket spécial pour accéder au distributeur de tickets. Pour cela, le client doit s'adresser au système

Kerberos qui, s'il reconnaît le client, fournira le ticket nécessaire. Nous voyons donc se dessiner les deux premières phases du protocole :

Phase 1 : Client --> Kerberos : c , dt
où dt est l'identité du distributeur de ticket

Si Kerberos connaît le client (i.e. s'il se trouve dans ses listes), il connaît également sa clé secrète et peut donc passer à la phase suivante.

Phase 2 : Kerberos --> Client : { $K_{c,at}$, $\mathbb{T}_{at,c}$ } K_c
où K_c est la clé que se partagent le client et Kerberos.

c) Obtention d'un ticket pour un serveur

Muni de son ticket d'accès au distributeur de tickets et d'une clé de session, le client peut maintenant se construire un authentifieur et demander un ticket pour n'importe quel autre serveur.

Phase 3 : Client --> Distributeur : s , $\mathbb{T}_{at,c}$, $\mathbb{A}_{c,at}$

Le distributeur de ticket peut vérifier l'identité du client et son droit d'accès (ticket) puis, si ce contrôle est positif, il pourra poursuivre. On peut ainsi vérifier :

- le Time Stamp, pour éviter le rejeu ;
- la date de validité du ticket ;
- la validité du ticket ;
- la correspondance entre l'authentifieur et le ticket.

Phase 4 : Distributeur --> Client : { $\mathbb{T}_{s,c}$, $K_{c,s}$ } $K_{c,at}$

d) Dialogue avec le serveur

Maintenant que le client possède une clé de session pour dialoguer avec le serveur désiré ainsi que le ticket d'accès, il peut envoyer ses requêtes accompagnées de son authentifieur et de ce ticket.

Phase 5 : Client --> Serveur : $\mathbb{A}_{c,s}$, $\mathbb{T}_{c,s}$, requête

Ensuite, si le serveur reconnaît le client, il peut envoyer un accusé de réception et/ou une réponse à la requête. L'accusé de réception sert également à rendre mutuelle l'authentification.

Phase 6 : Serveur --> Client : { TS+1 } $K_{c,s}$, réponse

On retrouve de nombreux exemples de protocoles de haut niveau dans la littérature. Citons comme exemples : [COHEN,85], [ESTRI,89] et [TSUDI,89].

5.3 Environnement retenu

Décrivons maintenant notre cadre de travail ainsi que les différentes entités qui formeront notre système Sphinx.

Le Sphinx a été créé, dans une première étape de test, au Centre de Technologie Informatique (C.T.I.) de Fontenay-aux-Roses (France). Il a été développé pour faire tourner des protocoles propres à sécuriser des réseaux locaux connectés entre eux par des réseaux longue distance et reliés à des Mainframes. On retrouve actuellement cette topologie dans la plupart des réseaux bancaires où la sécurité est évidemment un souci primordial.

5.3.1 Réseau local (Ethernet/Lan Manager)

La typologie du réseau sur lequel le Sphinx a été développé est représenté à la figure 29.

Les stations de travail sont des micro-ordinateurs de type PC compatible à base de processeur Intel 80386. Le système d'exploitation utilisé est OS/2 de Microsoft. Le réseau utilise la norme IEEE 802.3 (Ethernet) gérée par une partie du système d'OS/2 appelé Lan Manager.

Chaque PC peut ou doit posséder un lecteur de carte à microprocesseur de type DES-D2 Philips.

5.3.2 Processeur de sécurité (sous VRTX)

Le coeur du Sphinx est un dispositif matériel appelé **processeur de sécurité**. Ce dispositif est relié à un PC via un double ligne Ethernet à haut débit (10 Mbps théorique).

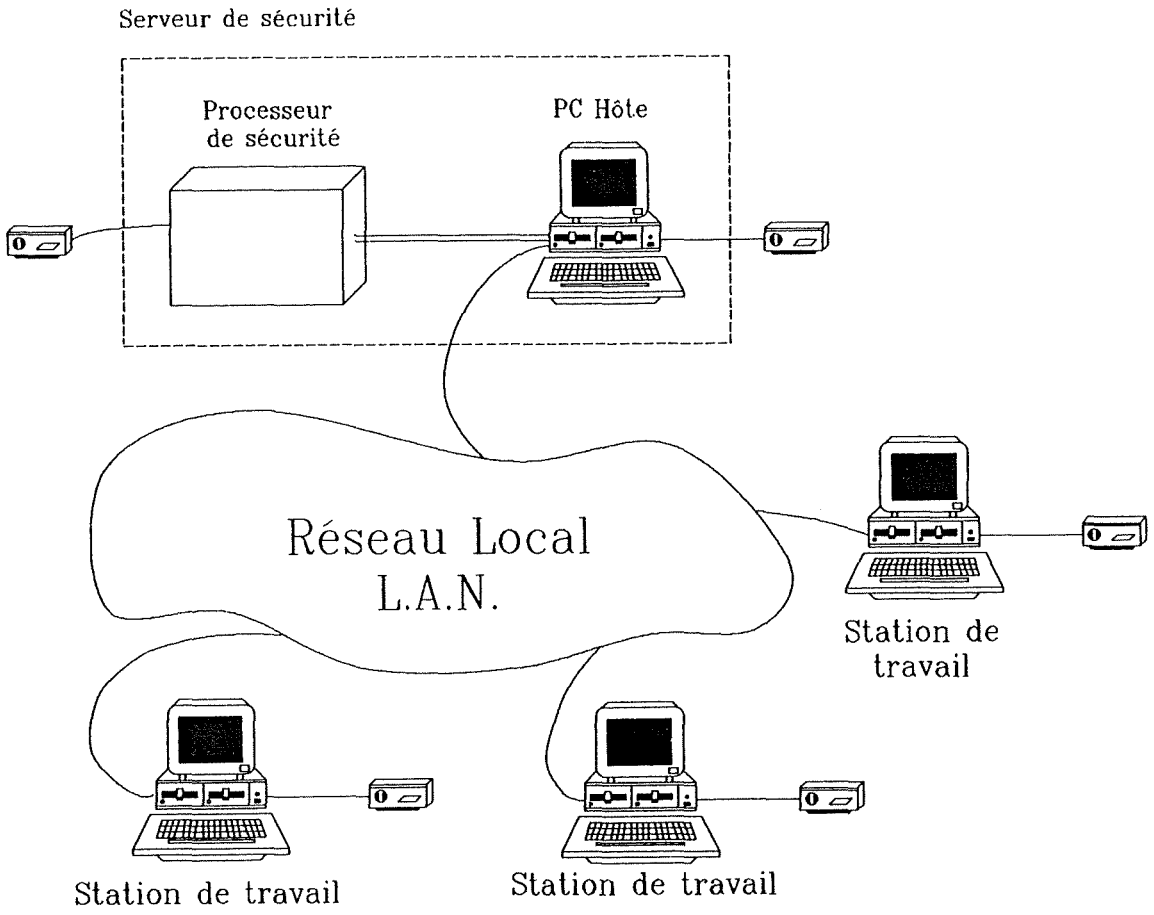


Figure 29 : Réseau de développement du Sphinx.

Le processeur de sécurité, comme on peut le voir à la figure 30, est basé sur une architecture à microprocesseur classique et comprend les éléments suivants.

1. Un microprocesseur de type Intel 80286. L'utilisation d'un 80386 SX ou d'un 80386 est envisageable en fonction des besoins ultérieurs.

2. De la mémoire de type RAM (Random Access Memory), EPROM (Erasable Programmable Read Only Memory) et EEPROM (Electrically Erasable Programmable Read Only Memory). Un "panachage" de 1 Mbyte de ces différentes mémoires est utilisé par le microprocesseur.

3. Une horloge temps réel.

4. Une batterie de sauvegarde rechargeable capable d'alimenter la mémoire RAM et l'horloge lors d'une panne d'alimentation. Un câblage particulier permet de remettre à zéro une partie de la mémoire RAM.

5. Un circuit Ethernet pour assurer la liaison avec le PC hôte.

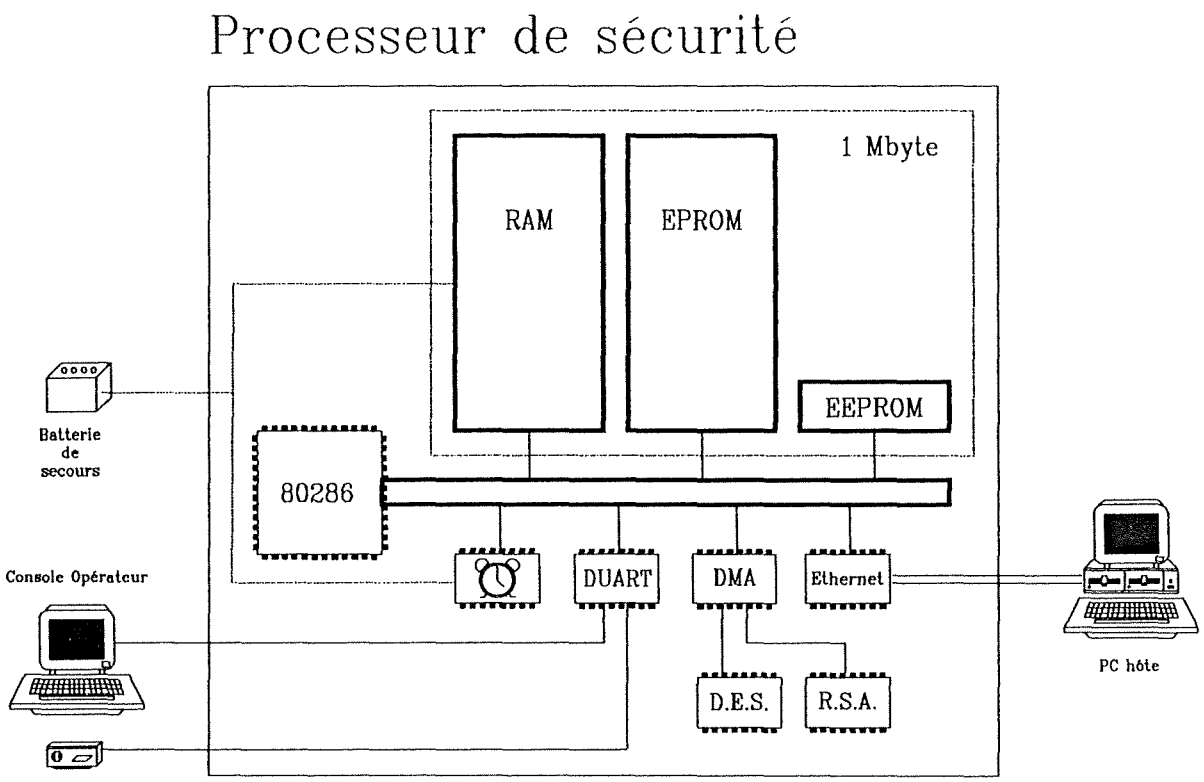


Figure 30 : Architecture du processeur de sécurité.

6. Un processeur cryptographique DES dont le débit de chiffrement peut atteindre 8 Mbps.

7. Un processeur cryptographique RSA de type Cylink capable d'effectuer des opérations arithmétiques sur des nombres de 512 bits.

8. Un circuit DMA (Direct Memory Access) qui est utilisé par les deux processeurs cryptographiques pour décharger le processeur principal lors des accès à la mémoire.

9. Un circuit de communication asynchrone permettant la connexion directe, au processeur de sécurité, d'un lecteur de cartes à micro-calculateur et d'une console opérateur. Cette console est utilisée, lors de la phase de test, pour simuler des attaques.

Dès qu'il est raccordé au PC hôte, le processeur de sécurité agit comme un esclave et répond aux requêtes envoyées.

5.3.3 Notion de serveur de sécurité

Le processeur de sécurité que nous venons d'examiner ne dispose pas de mémoire secondaire ni de dispositif d'entrée/sortie convivial. C'est pour cette raison qu'il est connecté à un PC hôte qui lui fournit ces ressources manquantes. L'ensemble formé par le processeur de sécurité et le PC hôte qui l'accueille est ainsi capable de fournir des services de sécurisation à son propriétaire. C'est pourquoi cet ensemble est appelé **serveur de sécurité.**

Ces services sont :

1. Assurer la gestion des utilisateurs autorisés sur le serveur de sécurité (identification, authentification, ...) ainsi que la gestion des applications sécuritaires que l'on peut utiliser dans le serveur de sécurité (contrôle d'intégrité, chargement sécurisé, contrôle des droits d'accès ...).

2. Tous les services imaginables rendus par des protocoles de base ou des applications sécuritaires de haut niveau que l'on peut ensuite implémenter sur le serveur de sécurité. Ce dernier veille alors à la bonne utilisation de ces applications comme nous l'avons vu au point précédent.

5.4 Architecture logicielle globale

5.4.1 Découpe en niveaux

L'architecture globale du logiciel destiné à sécuriser un réseau à l'aide du Sphinx est schématisée à la figure 31. Ce schéma fait apparaître un double découpage. Le premier, vertical, est dû à la localisation "géographique" des différents éléments logiciels dans le réseau. Le second, horizontal, est obtenu par une modularisation de l'ensemble du logiciel selon une hiérarchie basée sur une relation de type "utilise" ([AVL,89]).

5.4.2 Niveau Hardware

Le niveau hardware n'est repris dans l'architecture logicielle que pour situer "géographiquement" les différents modules. Rappelons qu'il s'agit de micro-ordinateurs de type PC compatible basés sur un microprocesseur 80386 et d'un circuit spécialisé basé sur un processeur 80286 avec processeurs cryptographiques spécialisés.

5.4.3 Niveau système d'exploitation

a) Sur les PC : OS/2

1) Présentation

Le système d'exploitation utilisé sur les stations de travail est OS/2 de Microsoft dans sa version 1.1 (avec l'interface graphique Presentation Manager). C'est un système multitâche, mono-utilisateur, qui fonctionne sur les PC compatibles basés sur microprocesseur 80286 et 80386 (bien que les ressources de ce dernier soient sous-exploitées par la version 1.1 d'OS/2). Le système n'est pas multiposte mais il est cependant multi-sessions. Chaque session correspond à une système virtuel (Ecran - Clavier - Souris) et peut se dérouler dans une fenêtre de type "Windows".

2) Utilisation de la mémoire

Grâce à l'utilisation qu'il fait du mode protégé du processeur, OS/2 peut accéder directement à 16 Mo. De plus, il implémente un système de mémoire virtuelle (géré par swapping) qui lui permet d'accéder à 1 Go. Les mécanismes de protection de la mémoire fournis par les processeurs 80x86 sont en outre utilisés.

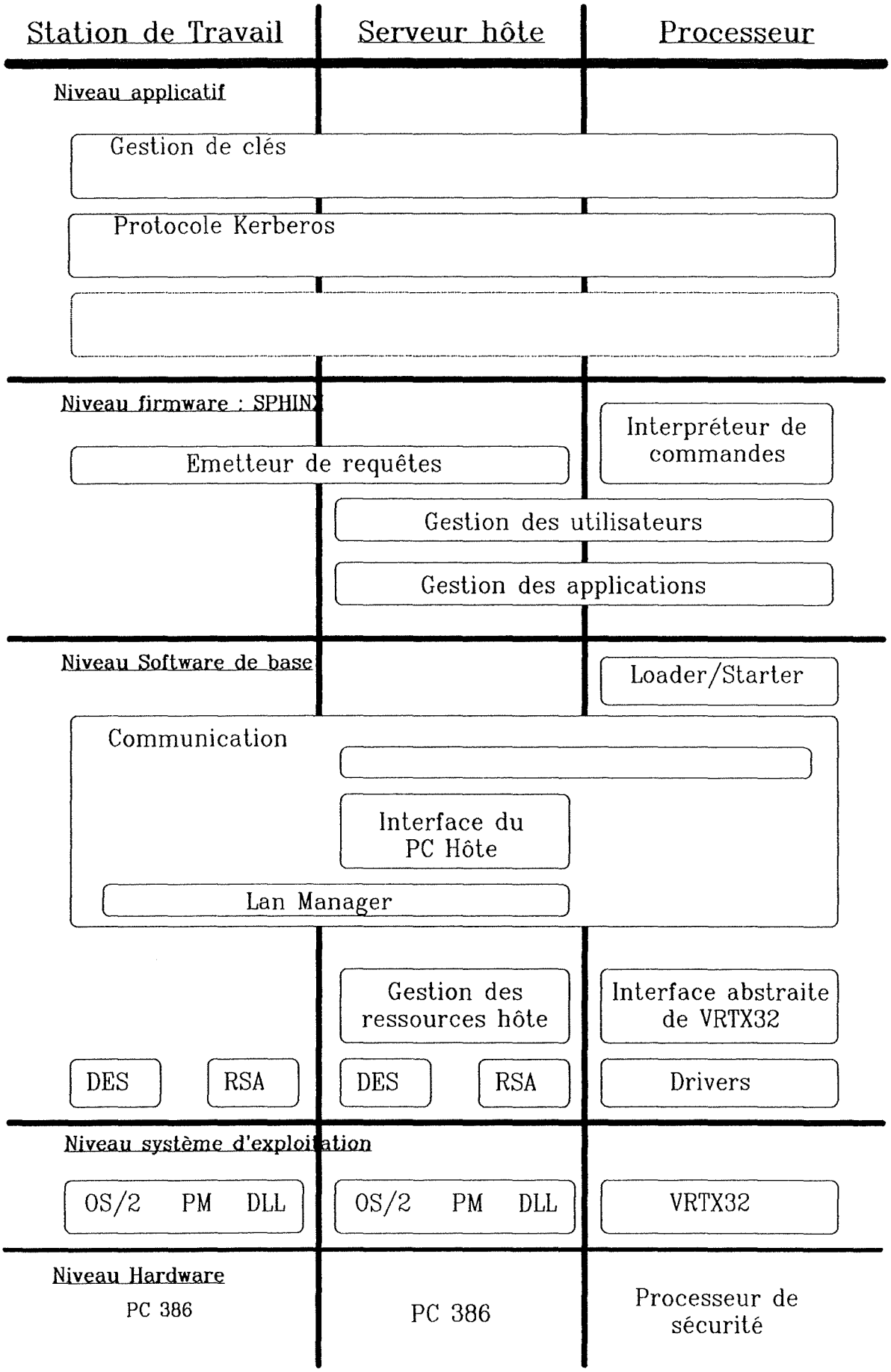


Figure 31 : Architecture logicielle du système Sphinx.

3) Multitâche

La seconde caractéristique intéressante d'OS/2 est sa capacité à exécuter simultanément (en apparence du moins) plusieurs applications. Ce principe de multitâche mis en oeuvre par OS/2 repose sur le partage du processeur entre les applications (time-slicing). Des mécanismes de synchronisation et de communication inter-applications sont fournis dans la bibliothèque système.

4) Gestion des fichiers

La gestion des fichiers est, du moins jusqu'à la version 1.1, identique à celle utilisée par DOS. Ceci assure une totale compatibilité entre les fichiers des deux systèmes mais implique les mêmes inconvénients : nom de fichier de taille limitée, fragmentation du support, faible protection.

b) Sur le processeur de sécurité

Le système d'exploitation utilisé sur le processeur de sécurité est VRTX32 (Versatile Real-Time eXecutive), l'un des systèmes temps réel le plus répandu sur processeurs de la famille iAPX86. VRTX32 possède toutes les caractéristiques des systèmes temps réel :

- multitâche ;
- "scheduling" basé sur des priorités ;
- communication et synchronisation inter-tâches ;
- allocation dynamique de mémoire ;
- "time-slicing" optionnel ;
- temps de réponse faible.

De plus, VRTX32 :

- requiert peu de mémoire, ce qui lui permet de fonctionner sur iAPX 86 ;
- est extensible (on peut lui ajouter des routines) ;
- peut se placer n'importe où en mémoire centrale (relogeable).

5.4.4 Niveau Software de base

a) Algorithmes cryptographiques

Une implémentation logicielle d'algorithmes cryptographiques des types DES et RSA peut être nécessaire sur les différents PC du réseau. Cette implémentation est évidemment inutile sur le processeur de sécurité.

b) Drivers

Différents "drivers" ont dû être développés sur le serveur de sécurité. Ces drivers pilotent les deux processeurs cryptographiques ainsi que les circuits DMA et Ethernet. L'accès aux ressources du "chip" DES et du "chip" RSA se fait par l'intermédiaire d'une ouverture de session. Les drivers sont capables d'ouvrir simultanément plusieurs sessions.

c) Gestion des ressources hôte

Ce module implémente les différentes fonctions de gestion des ressources du PC hôte utilisées par le processeur de sécurité. Ces fonctions gèrent principalement :

1. les bases de données ou les fichiers nécessaires au fonctionnement du processeur de sécurité ;
2. les programmes exécutables qui sont stockés sur le PC hôte avant d'être téléchargés sur le processeur de sécurité ;
3. le lecteur de carte à puce obligatoirement connecté au PC hôte.

d) Interface abstraite de VRTX32

Les services fournis par VRTX32 sont destinés à être accessibles aux programmeurs du processeur de sécurité (au plus bas niveau) mais aussi aux programmeurs d'applications sécuritaires de haut niveau. Ainsi, pour des raisons de simplicité et de sécurité, ces programmeurs n'ont accès au système d'exploitation que via un nombre restreint de procédures (gestion de sémaphore, allocation dynamique de mémoire, suspension sur une file d'attente, lecture/écriture sur une file d'attente ...) qui se retrouvent dans le module d'interface abstraite de VRTX32. De plus, cette technique permet de changer aisément de système d'exploitation dans le processeur de sécurité sans devoir répercuter ce changement par des modifications coûteuses aux niveaux supérieurs.

e) Niveau communication

Le module de communication est subdivisé en trois modules. Le premier gère la communication entre les stations de travail et le PC hôte. Le deuxième gère, quant à lui, la communication entre le PC hôte et le processeur de sécurité. Le troisième, enfin, est utilisé comme passerelle entre les deux modules précédents. L'ensemble fournit donc les outils nécessaires pour permettre un dialogue entre les trois intervenants.

1) Communication Station de Travail - PC Hôte

La communication entre les stations de travail et le PC hôte se fait grâce aux mécanismes de communication inter-processus fournis par OS/2. Ces mécanismes, appelés "Named Pipes", peuvent également être utilisés entre des processus situés sur des PC différents, reliés par un réseau, grâce au LAN MANAGER de Microsoft. Il s'agit du logiciel de communication réseau fourni en option avec OS/2.

2) Communication PC Hôte - Processeur de sécurité

Deux niveaux de contrôle d'erreur et de flux sont implémentés. Le premier se fait au niveau des trames qui transitent sur la ligne Ethernet. Le second est effectué au niveau du message. Un **message** est une suite particulière de trames. C'est également l'entité qui est manipulée au niveau du programmeur d'applications sur le processeur de sécurité. Ainsi, l'on dispose de primitives qui permettent l'envoi et la réception des messages. Les messages contiennent principalement des requêtes d'exécution d'une primitive, accompagnée des paramètres nécessaires.

3) Interface du PC Hôte

Ce module est utilisé à la fois comme passerelle et comme filtre entre les deux lignes gérées par les modules décrits précédemment. Son rôle est de regarder les messages qui transitent sur ces deux lignes. Si un message lui parvient sur une ligne, il regarde si le destinataire est sur l'autre ligne. Dans l'affirmative, il réinjecte le message tel quel sur l'autre ligne, vers son destinataire légitime. Mais il se peut que le message lui soit destiné en propre. A ce moment, le message contient une demande d'exécution d'une primitive appartenant au module de gestion des ressources du PC Hôte. Ce dernier fait alors appel à la fonction adéquate.

f) Loader/Starter

Ce module offre les primitives de base nécessaires à la gestion des programmes que l'on désire exécuter sur le processeur de sécurité. On peut ainsi télécharger dans la mémoire du processeur de sécurité des fichiers exécutables d'un format particulier à partir du disque du PC hôte. On peut décharger la mémoire d'un exécutable et l'on peut, bien sûr, lancer l'exécution d'un programme téléchargé.

Un fichier exécutable peut être de trois types :

1) Procédure

Le code d'un tel exécutable n'est pas partageable entre plusieurs applications. L'idée est ici la même que dans tout langage de programmation : on peut lancer une procédure avec certains paramètres , on reçoit les résultats et, ensuite, on continue où l'on était dans l'application.

2) Tâche utilisateur

Une tâche est un morceau de programme qui s'exécute de manière autonome et qui possède sa propre file d'attente. Le dialogue avec une tâche se fait donc via des messages postés sur sa file d'attente. La réception des paramètres résultats, suite à l'activation d'une tâche, se fait en asynchrone vis-à-vis du déroulement de l'application appelante. Des mécanismes de synchronisation et de suspension sont, pour gérer cela, fournis dans le module d'interface abstraite de VRTX32.

3) Tâche système

Une tâche système est semblable à une tâche utilisateur mais elle se distingue de cette dernière par le fait que son chargement en mémoire est "statique". Le téléchargement d'une tâche système est effectué automatiquement au démarrage du système et son déchargement peut être fait au "shutdown" du système. Un utilisateur habituel n'a donc pas les droits nécessaires pour procéder à des manipulations sur une telle tâche.

Par exemple, les drivers des processeurs cryptographiques peuvent être implémentés sous forme de tâche système. Ceci permet de modifier l'architecture du processeur de sécurité (changement de processeur cryptographique, dans ce cas) et de modifier aisément le driver correspondant.

5.4.5 Niveau protocoles de base

Les protocoles de sécurité de base ont pour but de permettre l'exécution des applications sécuritaires de haut niveau dans les meilleures conditions de sécurité possibles. Pour cela, trois modules ont été développés.

1. Gestion des utilisateurs ;
2. Gestion des applications ;
3. Interpréteur de commandes.

Grâce à ces trois modules, le gestionnaire du réseau peut adapter son serveur de sécurité à ses besoins et le personnaliser pour les utilisateurs du réseau. Ainsi, seul un utilisateur habilité pourra exécuter un protocole de sécurité qui aura été mis à sa disposition par l'administrateur du réseau. De plus, tout le monde aura la certitude que l'application de sécurisation qu'il utilise est intègre.

C'est cet ensemble, formé par le serveur de sécurité et géré par les protocoles de sécurité de base, que nous appellerons désormais de façon plus précise **système Sphinx.**

Les protocoles de sécurité de base feront l'objet d'une analyse plus détaillée dans le chapitre suivant.

5.4.6 Niveau applicatif

Tous les modules que nous avons présentés précédemment sont mis à la disposition du programmeur du Sphinx. Un acheteur du système pourra ainsi demander à Sphinx de protéger son réseau en lui "apprenant" des protocoles de gestion de clés, des protocoles de notarisation, des protocoles de style Kerberos ou autres. Cet apprentissage se fait en développant, chez le vendeur du système, un nouveau logiciel et en l'installant dans le Sphinx.

5.5 Cycle de vie du processeur de sécurité

Dans ce paragraphe, nous allons examiner le cycle de vie du système Sphinx. Ce cycle est divisé en quatre étapes :

1. Construction du système ;
2. vente du système et service après-vente ;
3. vie active du système ;
4. mort ou régénération du système.

Cette découpe résulte du type de carte à micro-calculateur utilisé ainsi que de la technique mise en oeuvre pour la protection physique du processeur de sécurité.

5.5.1 Cycle de vie lié à la carte à puce

Le cycle de vie du processeur de sécurité comporte beaucoup de similitudes avec le cycle de vie d'une carte à micro-calculateur. On parlera ainsi de phases de personnalisation, de mise en route ... . Deux raisons expliquent cette ressemblance. Tout d'abord, d'un point de vue matériel, le processeur de sécurité est basé sur une architecture semblable à celle d'une carte (microprocesseur, mémoire à accès protégé, liaison externe). Ensuite, l'emploi de cartes pour sécuriser la gestion du processeur de sécurité demande une synchronisation entre les différentes phases que subissent les cartes et le processeur de sécurité.

Voyons plus en détail ces différentes phases.

a) Schéma général

Les différents traitements que subit le processeur de sécurité lors de ses changements d'état sont repris à la figure 32. A chaque étape, le serveur de sécurité mémorise l'état dans lequel il se trouve ainsi que les renseignements supplémentaires qu'il acquiert à propos de ses futurs utilisateurs. Tout comme une carte à puce, le processeur de sécurité admet ou non certains ordres selon qu'il se trouve dans tel ou tel état. Examinons quels sont ces états.

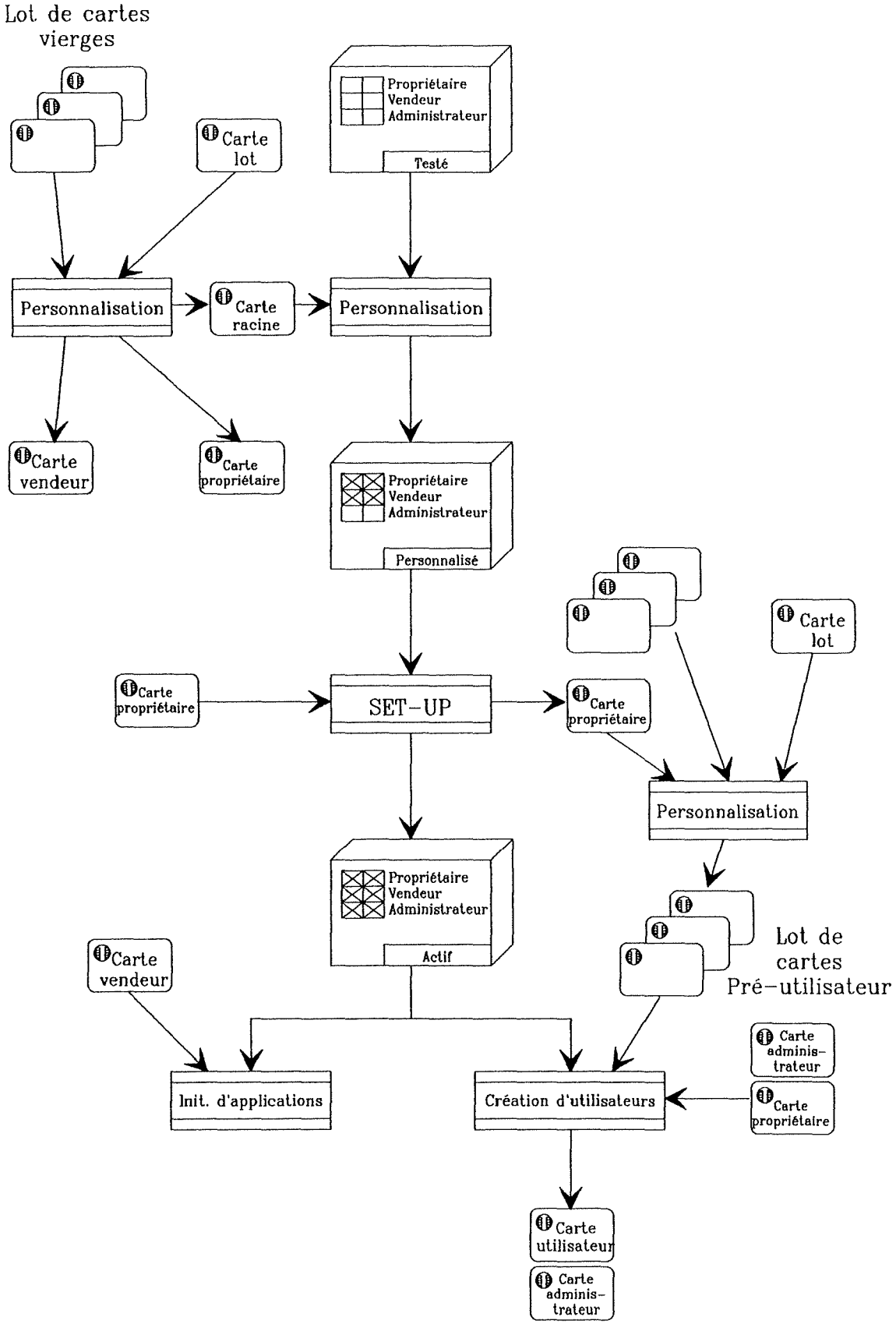


Figure 32 : Cycle de vie du processeur de sécurité (1).

b) Etat "testé"

Après avoir été assemblé en usine, le serveur de sécurité est testé et ensuite mis dans l'état "testé" avant d'être envoyé chez le vendeur agréé. A ce moment, le serveur de sécurité ne possède aucun renseignement quant à sa future attribution. Un vol à ce niveau n'entraînerait aucun préjudice si ce n'est la perte du matériel.

c) Etat "personnalisé"

Lorsqu'une personne intéressée vient acheter un serveur de sécurité, trois cartes sont créées à partir d'un lot de cartes vierges et d'une carte lot.

1. Une **carte propriétaire** qui permettra l'activation d'une procédure de création des utilisateurs futurs du système Sphinx. Cette carte étant la clé maîtresse du Sphinx, elle doit être gardée avec le maximum de sécurité dû à son rang (dans un coffre-fort, par exemple).

2. Une **carte vendeur** qui, utilisée conjointement avec la carte propriétaire, permettra la maintenance du système durant sa vie active. Cette maintenance consistera principalement en l'ajout de nouveaux logiciels dans le système.

3. Une **carte racine** utilisée, quant à elle, pour personnaliser le processeur de sécurité. Ce dernier mémorise ainsi les caractéristiques essentielles concernant le vendeur et le propriétaire.

C'est également lors de cette phase de personnalisation que différents logiciels sont fournis à l'acheteur et autorisés dans le processeur de sécurité. Ces logiciels sont :

1. L'ensemble logiciel permettant l'exécution des protocoles de sécurité de base.

2. Le logiciel de "Set-Up" qui sera nécessaire à l'étape suivante.

3. Des logiciels que l'acheteur veut d'ores et déjà utiliser sur son système Sphinx.

L'acheteur n'a donc plus qu'à placer le processeur de sécurité chez lui pour disposer du système Sphinx (par 2.) auquel il aura déjà appris quelques protocoles (implémentés dans 3.).

Dorénavant, le processeur de sécurité ne répondra aux sollicitations du monde extérieur que sur présentation de la carte propriétaire ou des cartes propriétaire et vendeur simultanément.

d) Etat "actif"

Une fois personnalisé, le processeur de sécurité doit être installé sur le site d'emploi. Il faut ensuite que le propriétaire du système crée les différentes cartes qui seront employées par les utilisateurs afin d'être reconnus en tant qu'utilisateur accrédité et d'accéder ainsi aux ressources fournies par le système Sphinx. Pour cela, le propriétaire exécute le programme de "Set-Up" fourni dans l'étape précédente. La présence de la carte propriétaire est nécessaire pour empêcher qu'un fraudeur puisse s'emparer du système. Lors de ce processus de "Set-Up", une carte mère est créée. Cette carte pourra servir ensuite lors d'un processus de personnalisation des futures cartes utilisateurs. Les cartes qui sortent de cette personnalisation sont "personnalisées" au sens où on l'entend pour la carte à puce mais elles doivent encore être personnalisées pour l'utilisateur. Dans ce but, il faut encore inscrire les caractéristiques de son futur propriétaire et "embosser" cette carte.

Au cours de la phase de "Set-Up", le propriétaire crée une carte pour un utilisateur particulier qui sera, en fait, le (ou le premier) **administrateur** du système Sphinx. La notion de type d'utilisateur sera détaillée dans le prochain chapitre.

Enfin, après avoir stocké les caractéristiques de cet utilisateur dans le processeur de sécurité, la phase de "Set-Up" fera passer ce dernier dans l'état "actif".

Une fois dans l'état "actif", le processeur de sécurité détient les informations nécessaires pour reconnaître le vendeur, le propriétaire et un utilisateur spécial. Il est alors à même d'authentifier ces derniers grâce à la carte que chacun d'eux possède. De plus, il dispose du logiciel nécessaire pour charger de nouveaux logiciels et pour créer de nouveaux utilisateurs. Les règles de sécurité régissant l'exécution de ces logiciels seront explicitées au chapitre suivant.

5.5.2 Cycle de vie lié au hardware

Au cours de sa vie active, différents événements peuvent survenir. Il peut s'agir d'événements "normaux", prévus et acceptés par Sphinx. Ils rentreront donc dans le cadre du fonctionnement normal du système que nous allons examiner dans une première section. La réaction du système en cas d'attaque physique sera analysée dans une seconde section. La réaction à une attaque logique se traduit, quant à elle, par une réponse polie mais ferme envoyée au demandeur de service non habilité.

a) Fonctionnement normal de Sphinx

En phase active, il faut distinguer deux types d'utilisation du système Sphinx. Le premier type consiste à gérer le système. Cette gestion se fait uniquement à partir du PC hôte et sur présentation d'au moins une des cartes suivantes : carte propriétaire, carte vendeur et carte administrateur. Le deuxième type consiste à demander au Sphinx l'exécution d'un logiciel particulier. A ce moment, le Sphinx authentifie le demandeur et, en fonction des droits de ce dernier, charge et exécute le logiciel ou refuse l'une ou l'autre de ces actions.

b) Réaction aux attaques physiques

Différents cas doivent, ici, être envisagés selon le type de protection physique utilisé.

1) Processeur de sécurité dans un coffre-fort

Le processeur de sécurité se trouve dans une pièce dont l'accès est strictement contrôlé. Dans ce cas, aucune attaque physique contre le processeur n'est envisagée et les seules attaques possibles sont de type logique et donc gérées par le Sphinx. On peut ainsi envisager de conserver les clés indispensables au bon fonctionnement du système en mémoire E^2PROM.

2) Processeur de sécurité coffre-fort

Si le processeur de sécurité n'est pas mis dans un lieu protégé, on peut envisager de le placer dans une "coquille protectrice". On utilisera alors un blindage capable d'empêcher physiquement les tentatives élémentaires d'analyse des différents circuits constituant le processeur de sécurité (à l'aide, par exemple, d'un analyseur logique). De plus, ce blindage sera capable de détecter les intrusions physiques plus élaborées. Nous

pensons principalement au micro-forage, au "pelage" et aux attaques à l'aide de différents acides ou solvants (voir section suivante). Une fois ainsi détectée, la fraude peut être immédiatement signalée au processeur de sécurité qui peut alors prendre les mesures nécessaires. La réaction envisagée est alors d'effacer une partie de la mémoire RAM sauvegardée à l'aide du dispositif prévu. Les différentes clés indispensables qui seront, cette fois, placées en mémoire RAM, seront alors perdues, rendant inutile l'intrusion et mettant, de facto, le processeur de sécurité dans un état supplémentaire : l'état "bloqué".

Mais après une telle réaction, le Sphinx, dépossédé de ses clés, ne sera plus en état de travailler, même pas pour son propriétaire. Il faut donc envisager une procédure de déblocage dans le cycle de vie du processeur de sécurité. Celle-ci s'effectuerait sous le contrôle de la carte propriétaire et grâce à un "back-up" des informations vitales. On peut envisager de faire ce back-up sur une carte de secours. Cette carte, en cours de fonctionnement normal, pourrait être conservée dans un coffre-fort, avec la carte propriétaire et la carte utilisée lors de la "pré-personnalisation" des cartes utilisateur.

Le cycle de vie complet est alors schématisé à la figure 33.

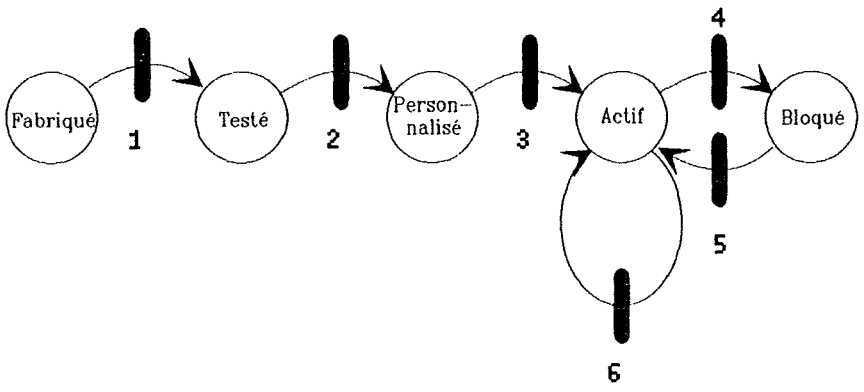


Figure 33 : Cycle de vie du processeur de sécurité (2).

3) Blindage - Détection de fraude physique

Pour détecter des attaques physiques, on peut emballer le dispositif à protéger d'un (ou plusieurs) grillage(s) aux mailles très fines et éventuellement "couler" le tout dans un matériau résistant (résine).

Tout d'abord, ce dispositif agit comme une cage de Faraday et empêche ainsi tout espionnage externe du circuit par analyse des variations du champ magnétique [SAURA,89].

Ensuite, on peut appliquer un courant dans le grillage et détecter tout changement dans ce courant. Si le grillage est attaqué par une micro-foreuse, attaqué par "pelage" ou par acide, les dégâts inévitablement causés au grillage se traduira immédiatement par une variation de ce courant circulant et le dispositif pourra réagir.

5.6 Résumé du chapitre

Dans ce chapitre, nous nous sommes efforcé de présenter globalement le projet auquel nous avons participé lors de notre stage de fin d'étude. Ce projet consistait à spécifier et à développer un serveur de sécurité capable de fournir les fondements nécessaires à l'implémentation de protocoles de sécurité de haut niveau. Puis nous avons précisé ce que nous entendions par ce terme et fixé les idées au moyen d'un exemple concret : une gestion de droit d'accès à des serveurs situés dans un réseau ouvert.

Nous avons ensuite exposé notre environnement de travail, pour pouvoir présenter de manière plus approfondie l'architecture, tant matérielle que logicielle, du serveur de sécurité. Ceci nous a permis de définir le système SPHINX, un ensemble formé d'un serveur, d'un processeur spécialisé et de protocoles de sécurisation de base.

La dernière partie du chapitre a enfin été consacrée à la présentation du cycle de vie typique du système Sphinx lors d'une utilisation en environnement réel.

6. Protocoles de sécurité de base du serveur de sécurité

6.1 Principes de base et entités manipulées

Avant d'expliciter les protocoles de sécurité de base du système Sphinx, nous allons définir les différentes entités qui seront manipulées. Une fois ces notions bien assimilées, nous pourrons aller plus en avant dans notre examen du "coeur logiciel" du Sphinx.

6.1.1 Entités relatives à la gestion des applications

A la demande du client, un programmeur du système Sphinx peut développer une **application** spécifique. Ainsi, le client pourra acheter, par exemple, une application de distribution de tickets d'accès aux ressources d'un réseau local basée sur le protocole Kerberos que nous avons examiné dans le chapitre précédent.

Cette application sera implémentée sous la forme d'un ou plusieurs **modules**. Un module est un ensemble de **fonctions** nécessaires au bon déroulement de l'application. Dans notre exemple désormais connu, l'"application Kerberos" pourrait être implémentée sous forme d'un "module Kerberos" constitué des fonctions suivantes :

- Demander un ticket initial ;
- Demander un ticket pour un serveur ;
- Initialiser le dialogue avec un serveur.

Nous appellerons **module exécutable** l'ensemble du code exécutable réalisant les fonctions d'un module et donc de l'application correspondante. Dans cet ensemble sont reprises les informations de service nécessaires au module "loader/starter" (tailles, table de relocation ...) pour placer les différents codes exécutables dans la mémoire du processeur de sécurité. C'est sous cette forme que l'on stocke toutes les applications sur le PC hôte en attendant leur téléchargement.

Module, application et code exécutable (voir figure 34) sont les trois entités manipulables par la gestion des applications.

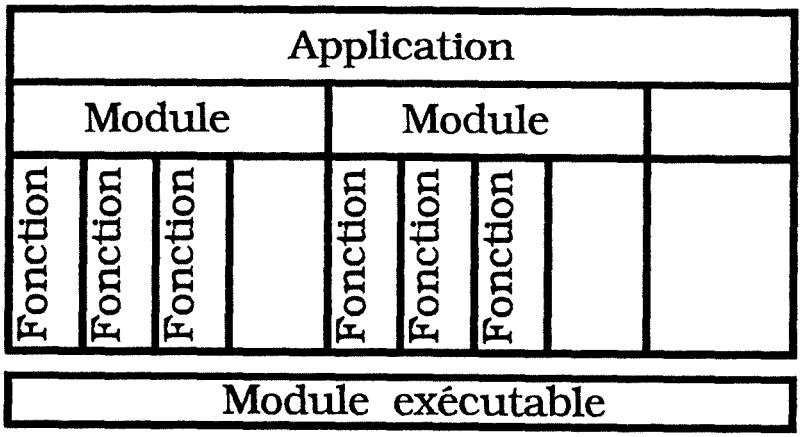


Figure 34 : Entités du niveau de gestion des applications.

6.1.2 Entités relatives à la gestion des utilisateurs

Comme nous l'avons déjà esquissé, tous les utilisateurs du système ne possèdent pas le même statut. Certains ont un rôle particulier et donc des besoins et des droits différents. Pour refléter cet état de fait et pour homogénéiser la gestion des utilisateurs, le Sphinx manipule six types d'utilisateurs, **six groupes d'utilisateurs** (voir figure 35. Chaque groupe est ainsi caractérisé d'une part par des privilèges et, en contre-partie, par des contraintes de sécurité à respecter.

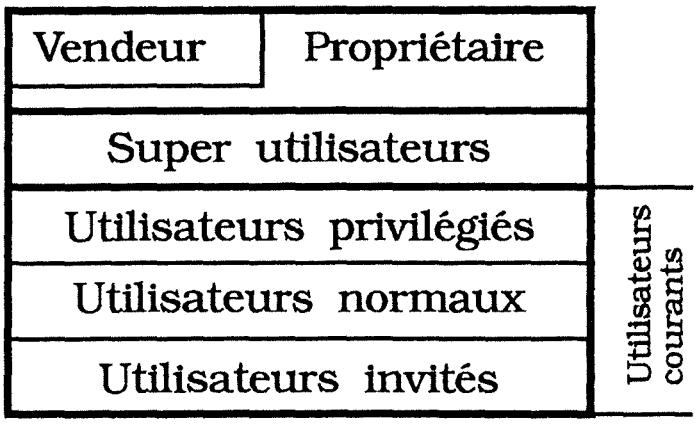


Figure 35 : Entités du niveau de gestion des utilisateurs.

a) Groupe des utilisateurs courants

Sous le terme générique d'**utilisateur courant**, on reprend trois groupes d'utilisateurs :

- le groupe des **utilisateurs invités** ;
- le groupe des **utilisateurs normaux** ;
- le groupe des **utilisateurs privilégiés**.

Il n'y a pas de différence fondamentale entre ces différents types d'utilisateurs. Mais cette distinction permet d'affiner la gestion des privilèges des différents utilisateurs, ces privilèges étant définis au niveau du groupe.

On peut ainsi imaginer de ne pas soumettre les "invités" à un protocole d'identification mais, en contre-partie, réduire fortement leurs privilèges. Les deux autres groupes seraient identifiés grâce à un protocole solide et ils auraient des privilèges plus ou moins élevés.

b) Groupe des super-utilisateurs

Un **super utilisateur** est un utilisateur qui, en outre, dispose des privilèges nécessaires pour créer d'autres utilisateurs courants et mettre à jour leur privilèges. C'est donc un administrateur du système Sphinx.

c) Propriétaire du Sphinx

Le **propriétaire** du Sphinx est "le seul maître à bord". Lui seul peut autoriser l'introduction de nouveaux logiciels et créer tous les autres utilisateurs. Il est le seul à pouvoir créer des super utilisateurs qui pourront eux-mêmes créer d'autres utilisateurs courants.

d) Vendeur du Sphinx

Le **vendeur** du Sphinx est la seule personne autorisée à personnaliser le Sphinx. C'est également lui qui peut développer une application (un module) et autoriser son exécution sur le Sphinx. Ainsi, le propriétaire est assuré de l'authenticité des logiciels qu'il utilise sur son système et le vendeur est sûr que l'on ne pourra pirater ses logiciels.

6.1.3 Entités relatives aux dialogues

Le dialogue entre les trois correspondants (station de travail - PC hôte - Processeur de sécurité) s'effectue via des **requêtes**. Une requête est une demande d'exécution de fonction que l'on envoie dans un message accompagnée des paramètres nécessaires. Lorsqu'un poste reçoit une requête qui lui est destinée, il se réserve le droit d'exécuter ou non la fonction demandée et de renvoyer les résultats. Les résultats, envoyés sous la forme d'un message également, forment ainsi une **réponse** à la requête. Certaines requêtes ne nécessitent pas de réponse.

Les messages peuvent transiter en clair sur le réseau mais, si cela est nécessaire, on peut sécuriser ces échanges de diverses façons. On peut par exemple envisager de chiffrer ou de signer les messages qui transitent sur la ligne. Le protocole qui gère la sécurisation des messages, appelé **protocole d'échange**, peut donc varier selon les besoins, ajoutant ainsi de la souplesse au système.

6.2 Gestion des utilisateurs

6.2.1 Liste des utilisateurs accrédités

a) Introduction

Un des premiers problèmes qui s'est présenté lors de la réalisation de la gestion des utilisateurs a été de concevoir la liste des utilisateurs autorisés à accéder au Sphinx. Il fallait que cette base de données possède les quatre caractéristiques suivantes :

1. **Dynamique** : il faut pouvoir, à tout moment, modifier des renseignements, en ajouter, en supprimer.

2. **Extensible** : le nombre d'utilisateurs du système peut être très élevé et il n'était pas question de le limiter à une taille arbitraire.

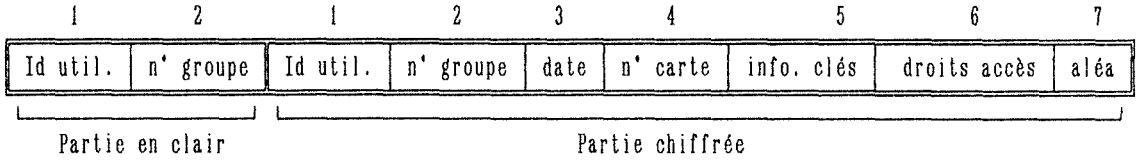
3. **Durable** : on ne peut se permettre de perdre la liste des utilisateurs car, si cela arrivait, il faudrait, dans le meilleur des cas, recréer tous les utilisateurs. Cette situation n'est pas acceptable d'un point de vue financier et encore moins d'un point de vue organisationnel.

4. **Sûre** : la liste des utilisateurs du système est assurément le point névralgique du système. Si la manipulation de cette liste par un fraudeur est possible, il est bien évident que ce dernier peut mettre la main sur le système sans aucun problème.

Les trois premières caractéristiques excluaient le stockage de cette liste dans un des trois types de mémoire du processeur de sécurité. La dernière solution immédiatement envisageable était donc le stockage sur la mémoire secondaire du PC hôte (typiquement, sur son disque dur). Mais l'exportation de ces données stratégiques en dehors de l'enceinte sécurisée du processeur de sécurité pose d'énormes problèmes pour atteindre la quatrième caractéristique : la sécurité. Dans la section suivante, nous allons examiner une solution satisfaisante à ce problème.

b) Structure de la liste externe

Les renseignements nécessaires pour contrôler l'accès au Sphinx ont été stockées sous la forme d'un fichier d'enregistrements concernant les utilisateurs autorisés à utiliser le système. Chacun de ces enregistrements est divisé en deux parties - la première en clair, la seconde chiffrée - et il reprend, pour un utilisateur donné, les renseignements suivants :



1. **Identificateur** de l'utilisateur ;
2. **Numéro du groupe** (groupe 1 à 6) auquel l'utilisateur appartient ;
3. La **date de validité** de la carte à microprocesseur de l'utilisateur ;
4. Le **numéro de série de la carte** à microprocesseur de l'utilisateur ;
5. Des informations concernant les **clés** publiques et privées associées à l'utilisateur ;
6. Des informations concernant les **droits d'accès** de l'utilisateur à différentes ressources du réseau ;
7. Une information permettant de vérifier la **validité** de l'enregistrement (aléa).

La deuxième partie de l'enregistrement est chiffrée à l'aide d'une clé différente par groupe (il y a donc six clés différentes). L'aléa est identique pour tous les utilisateurs d'un groupe donné.

Le fichier se présente donc sous la forme (logique) suivante :

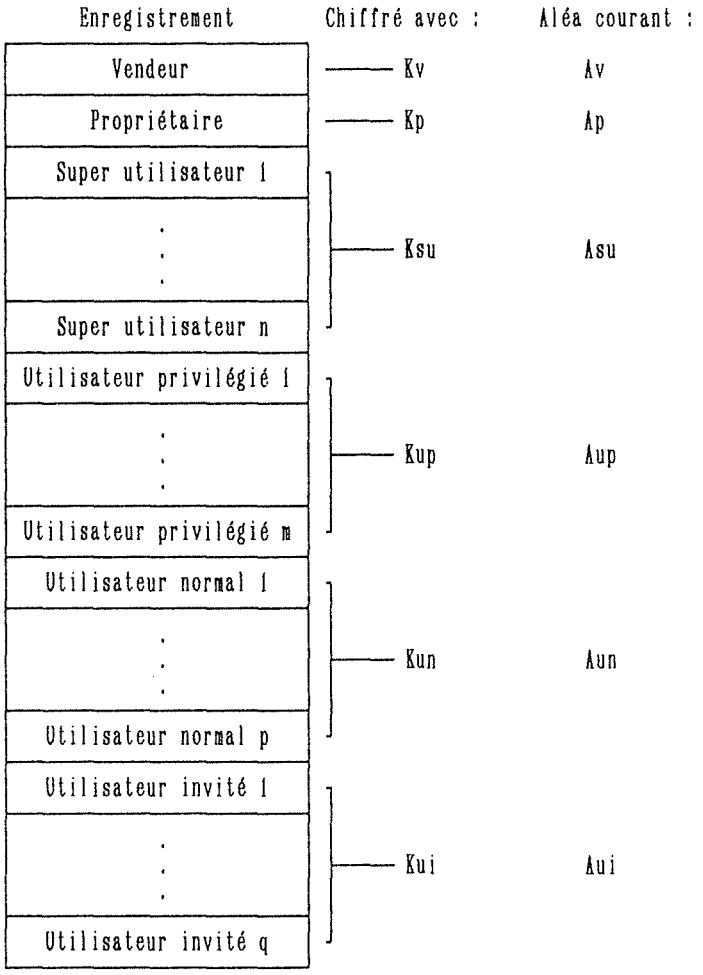


Figure 36 : Structure du fichier externe des utilisateurs.

c) Justification des choix

1) Partie en clair - partie chiffrée

L'emploi d'une partie en clair était nécessaire pour permettre un accès aisé aux enregistrements au sein de la liste. Les informations en clair sont cependant répétées dans la partie chiffrée pour empêcher qu'un fraudeur ne puisse dissocier la partie claire de la partie chiffrée. Ceci lui permettrait sinon de se construire un enregistrement contenant les informations secrètes d'un autre utilisateur associé à ses propres informations. Il aurait donc été possible de se faire passer pour cet autre

utilisateur. La seconde partie de l'enregistrement est chiffrée pour empêcher quiconque de lire et de modifier de façon cohérente les informations vitales pour la sécurité du Sphinx.

2) Date de validité - numéro de carte

La date de validité permet de contrôler et de réduire dynamiquement l'utilisation du système Sphinx dans le temps.

Le numéro de la carte permet d'associer une carte à un utilisateur. Un utilisateur est ainsi identifié par son identificateur et sa carte. Cette précaution permet d'invalider une carte perdue, volée ou périmée tout en gardant, pour l'utilisateur toujours autorisé (muni d'une nouvelle carte), le même identificateur.

3) Clés publiques chiffrées

Il peut paraître paradoxal de chiffrer des clés publiques mais, comme nous l'avons montré précédemment, une clé publique doit tout de même être protégée contre les modifications illicites. Ainsi, en mettant ces clés dans la partie chiffrée de l'enregistrement, le système Sphinx notarise les clés publiques des utilisateurs.

4) Problème du rejeu - Version "up-to-date"

Un problème courant en cryptographie, qui risquait de se présenter ici et qu'il fallait empêcher à tout prix, était le rejeu. En effet, supposons qu'un utilisateur se voit interdit d'accès au Sphinx ou voit ses privilèges diminués. Il pourrait avoir mémorisé l'état du fichier externe ou de son enregistrement avant cette dernière modification et rien ne l'empêcherait, sur le PC hôte, de rétablir l'ancienne version de la liste de validité. Il récupérerait ainsi de façon frauduleuse ses anciens droits. Pour éviter pareille situation, un double mécanisme de contrôle de validité de l'enregistrement est mis en oeuvre grâce à la clé de chiffrement et à l'aléa de l'enregistrement. A chaque suppression ou mise à jour d'un enregistrement dans un groupe, ces deux valeurs de contrôle sont remplacées par deux nouvelles valeurs choisies aléatoirement. Il faut donc **regénérer** la partie du fichier concernant le groupe intéressé, c'est-à-dire reprendre chaque enregistrement du groupe, le déchiffrer, mettre à jour l'aléa et rechiffrer avec la nouvelle clé. On peut ainsi vérifier qu'un enregistrement appartient à la dernière version de la liste externe de validité en comparant l'aléa (déchiffré avec la dernière version de la clé,

appelée **clé courante**) avec l'**aléa courant** (la dernière version de l'aléa). Ces deux valeurs certifiant la fraîcheur des enregistrements sont évidemment sauvegardées en toute sécurité sur le processeur de sécurité.

Remarquons que ce travail fastidieux ne doit pas obligatoirement être effectué lors de l'ajout d'un nouvel enregistrement. On peut également n'effectuer cette manipulation coûteuse qu'en fin de journée, par exemple, ou après une quelconque période de temps jugée optimale.

6.2.2 Structure interne des utilisateurs

Dans le processeur de sécurité, les renseignements concernant les utilisateurs sont stockés au niveau du groupe. Sphinx dispose ainsi d'une table à six entrées (une par groupe) qui contiennent chacune :

1. Une **clé maître** employée pour diversifier, à partir du numéro de série de la carte, les clés qui seront utilisées pour réaliser le protocole d'authentification des utilisateurs ;

2. Un ensemble de **clés maîtres** (clés de diversification également) pour la gestion des clés des cartes à puce (MK, AK, CK, EK : se référer à la section consacrée à la carte DES-D2) ;

3. L'**aléa courant** pour le groupe (se référer à la structure de la liste de validité externe) ;

4. La **clé courante** de chiffrement des enregistrements concernant le groupe (même remarque) ;

5. Un **lien** permettant de retrouver le protocole destiné à authentifier les utilisateurs faisant partie du groupe ;

6. Un **lien** vers le protocole utilisé pour les échanges de messages avec l'utilisateur (notion de protocoles d'échange).

Groupe	Clé authentif.	Clés cartes				Aléa	Clé	Protocole authentif.	Protocole échange
Vendeur						Av	Kv	(1)	
Propriétaire						Ap	Kp	(1)	
Super utili.						Asu	Ksu	(3)	
Utili. privil.						Aup	Kup	(3)	
Utili. normal						Aun	Kun	(3)	
Utili. invité						Aui	Kui	(2)	

Comme nous le verrons plus tard, dans le paragraphe consacré à l'interpréteur de commandes :
(1) Il s'agit d'un protocole d'authentification de base utilisant la carte à microprocesseur de ces deux types d'utilisateurs.
(2) Aucun protocole d'authentification n'est prévu pour les invités, dans une première phase.
(3) On emploie ici un protocole d'authentification mutuelle réalisé à l'aide de la carte à puce de l'utilisateur.

Figure 37 : Structure interne des utilisateurs.

6.2.3 Opérations permises

Les fonctions que l'on peut exécuter pour gérer les utilisateurs sont uniquement accessibles lorsque le processeur de sécurité est dans l'état actif. Seul le propriétaire et les super utilisateurs disposent des privilèges nécessaires pour les activer. De plus, elles ne sont accessibles qu'à partir du PC hôte. Ainsi, ces fonctions ne s'exécutent qu'avec la carte propriétaire ou super utilisateur dans le lecteur du PC hôte.

Les fonctions reprises dans le module de gestion des utilisateurs sont :

1. Création d'un nouvel utilisateur ;
2. Suppression d'un utilisateur existant ;
3. Contrôle d'accès d'un utilisateur ;
4. Consultation des renseignements sur un utilisateur ;
5. Mise à jour des renseignements sur un utilisateur ;
6. Mise à jour des attributs d'un groupe ;
7. Changement de groupe d'un utilisateur ;
8. Déblocage d'une carte ;
9. Renouvellement d'une carte.

a) Création d'un nouvel utilisateur

Rappelons que le propriétaire du système Sphinx peut créer autant de super utilisateurs et d'utilisateurs courants qu'il veut, alors qu'un super utilisateur ne peut créer que des utilisateurs courants.

Cette fonction demande les renseignements nécessaires à la création d'un nouvel utilisateur. Elle vérifie ensuite si l'utilisateur que l'on veut créer n'existe pas déjà dans la liste externe de validité. Après quoi, elle crée et stocke un nouvel enregistrement et crée la carte à microprocesseur correspondante. Durant l'opération, la carte du propriétaire ou du super utilisateur se trouve dans le lecteur du PC hôte et la carte pré-utilisateur, qui deviendra la carte utilisateur, se trouve dans le lecteur du processeur de sécurité. Le PIN pour la carte est généré aléatoirement et communiqué à l'utilisateur.

b) Suppression d'un utilisateur existant

Le propriétaire ou le super utilisateur peut choisir un utilisateur dans la liste externe de validité et demander qu'il soit retiré du cercle des utilisateurs autorisés du Sphinx. La carte de l'utilisateur que l'on veut exclure n'est évidemment pas nécessaire. A la fin de cette fonction, l'enregistrement correspondant aura disparu de la liste de validité, la partie de cette liste concernant le groupe auquel appartenait l'utilisateur aura été regénérée et les nouvelles valeurs de la clé courante et de l'aléa courant auront été mises à jour dans la table des groupes.

Remarquons que la régénération de la liste externe ne fonctionne qu'après avoir testé si le fichier est valide avant l'opération. S'il n'en était pas ainsi, il suffirait de modifier la liste et de faire en sorte que le Sphinx la régénère immédiatement après pour mystifier le système de contrôle des enregistrements.

c) Contrôle d'accès

Le contrôle d'accès est effectué au moins une fois (à l'initialisation de la communication) par le système qui doit absolument s'assurer de l'identité de l'utilisateur pour lequel il travaille. Le contrôle d'accès est personnalisé pour chaque groupe. Sphinx peut connaître le protocole qu'il doit utiliser pour tel ou tel groupe grâce au lien présent dans la table des groupes. Quelques protocoles d'authentification envisagés seront décrits

plus tard dans l'exposé, lorsque nous traiterons de l'interpréteur de commandes.

d) Consultation des renseignements utilisateur

Cette fonction est la seule du module de gestion des utilisateurs qui est accessible de n'importe quelle station de travail. Cela est dû au fait que son exécution ne requiert pas obligatoirement la carte de l'utilisateur dans le lecteur relié au PC hôte. Le but de cette fonction est de fournir l'ensemble des informations publiques dont dispose Sphinx à propos de l'un de ses utilisateurs. Les **informations publiques** sont les informations contenues dans les enregistrements de la liste externe de validité et qui peuvent être divulguées à tout le monde. Il s'agit, dès lors, des clés publiques RSA et de la date de validité de l'autorisation d'accès.

e) Mise à jour des renseignements utilisateur

Cette fonction permet au propriétaire et aux super utilisateurs de modifier certaines informations concernant les utilisateurs. Ces modifications ne peuvent, à l'heure actuelle, porter que sur la date de validité des autorisations. Les autres informations sont gérées exclusivement par Sphinx qui peut ainsi en assurer la cohérence et la sécurité. Par exemple, on ne peut modifier "manuellement" une clé publique RSA car cette opération est réalisée dans de meilleures conditions de sécurité par une fonction de génération de clé publique du module "Driver RSA". Remarquons qu'ici, comme dans la suppression d'un utilisateur, il faut régénérer le fichier externe.

f) Mise à jour des attributs d'un groupe

Les paramètres que l'on peut modifier dans la table des groupes sont les protocoles d'authentification et d'échange. Ceci permet de moduler le niveau de sécurité à appliquer lors des dialogues avec les utilisateurs appartenant à un groupe déterminé. La sécurité requise pour un groupe résulte d'un compromis entre la sécurité désirée et le prix, en temps d'attente principalement, que l'on est prêt à payer.

g) Changement de groupe d'un utilisateur

Grâce à cette fonction, les administrateurs du système peuvent modifier les privilèges d'un utilisateur en le changeant de groupe. Il faut

donc, après activation de cette fonction, régénérer la liste externe de validité.

h) Déblocage d'une carte

Comme nous l'avons vu, une carte à microprocesseur peut s'auto-bloquer après trois présentations de PIN erroné. Le déblocage d'une carte, sous le contrôle du Sphinx et d'un administrateur, est donc prévu. Pour que la hiérarchie entre les différents types d'utilisateurs soit respectée, un super utilisateur peut débloquer les cartes des utilisateurs courants mais seul le propriétaire peut débloquer les cartes des utilisateurs.

i) Renouvellement d'une carte

Cette opération est activée lorsqu'il faut remplacer la carte d'un utilisateur. Le vol ou la perte d'une carte, ou plus simplement la "mort" d'une carte, justifie ce remplacement. Ce remplacement est possible sans changer l'identificateur de l'utilisateur grâce à l'existence indépendante du numéro de carte dans la liste externe de validité. Comme l'exécution de cette fonction induit des modifications au niveau de la liste externe, celle-ci doit ensuite être régénérée.

6.3 Gestion des applications

6.3.1 Structure interne des applications

a) Introduction

Rappelons que le système Sphinx a été conçu pour exécuter des applications pour les utilisateurs. Un application se présente sous la forme d'un module composé de différentes fonctions. Le module est la plus petite entité que l'on peut télécharger, c'est-à-dire transférer de la mémoire secondaire du PC hôte vers la mémoire centrale du processeur de sécurité. Cette notion de téléchargement sera formalisée au cours des sections qui suivent. Dans l'immédiat, nous allons examiner la structure de données qui permet de mémoriser et de manipuler ces différents concepts.

b) Structure de données

Pour sa gestion des applications, le système Sphinx manipule deux tables. La première, appelée table des modules reprend les renseignements

relatifs aux modules. La seconde, la table des fonctions regroupe bien entendu les informations concernant les différentes fonctions. Comme ces deux tables sont placées en mémoire EEPROM, elles sont signées grâce à un MAC DES et la signature est sauvegardée en mémoire RAM volatile. Elle peut ainsi être effacée en cas d'attaque, ce qui rendrait inutile toute modification de la table.

Chaque ligne de la table des modules contient les champs suivants :

1. Identificateur du module ;
2. Identificateur sécurisé du module ;
3. Type du module (indique s'il s'agit d'une procédure, d'une tâche utilisateur ou d'une tâche système) ;
4. Signature du module ;
5. Droits d'accès des groupes ;
6. Lien vers la première fonction du module.

Chaque ligne de la table des fonctions reprend les champs suivants :

1. Identificateur de la fonction ;
2. Identificateur sécurisé de la fonction ;
3. Droits d'accès des différents groupes au module ;
4. Lien vers la fonction suivante du module.

Un petit dessin valant mieux qu'un long discours, on peut voir sur la figure 36, une représentation graphique de cette structure.

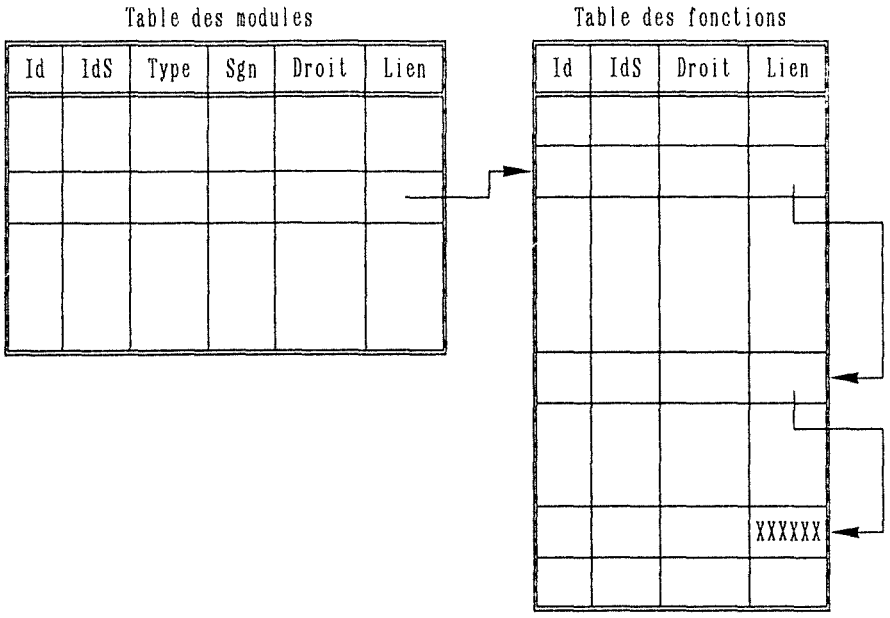


Figure 38 : Structure interne des applications.

c) Justification des choix

1) Identification

Un identificateur de module est le numéro de série qui est utilisé dans le catalogue des applications déjà développées sur le système Sphinx. L'identificateur de la fonction est un numéro identifiant une fonction au sein d'un module. Typiquement, les fonctions seront numérotées à partir de 0. L'identifiant d'une fonction est donc constitué du couple (identificateur de module, identificateur de fonction). Comme un numéro de série n'est pas très explicite, une description en langage naturel accompagne ce numéro. Le lien entre numéro de série et descripteur entraînera l'emploi des "identificateurs de sécurité". Nous reviendrons sur cette notion lors de la description de la structure externe des applications.

2) Signature

La signature du module est obtenue en calculant un MAC à base de DES sur le module exécutable. La gestion de cette signature apparaîtra tout au long du développement des différentes fonctions réalisant la gestion des applications.

3) Droits d'accès

Les droits que l'on peut avoir sur une application sont de trois types :

1. **Droit d'exécuter** une fonction ;

2. **Droit de télécharger** un module ;

3. **Droit de supprimer** un module de la mémoire centrale du processeur de sécurité.

Nous insistons sur le fait que le droit d'exécution est défini sur les fonctions alors que les droits de téléchargement et de suppression sont définis au niveau du module. Ceci explique la présence d'un champ intitulé "droit" dans les deux tables.

Les droits sont définis au niveau des groupes d'utilisateurs. Ainsi, chacun des trois droits que nous venons de voir peut être représenté par un vecteur de six booléens (flags). Chacun de ces six booléens exprime alors si oui ou non les six groupes (respectivement) possèdent le droit en question.

Exemple :

Considérons la fonction de création d'un utilisateur courant. Ses droits d'exécution sont :

1	2	3	4	5	6
0	1	1	0	0	0

Cela signifie que seul le propriétaire (groupe 2) ou un super utilisateur (groupe 3) peut exécuter la fonction de création d'un utilisateur courant.

6.3.2 Fichier externe des applications

a) Introduction

Comme nous l'avons déjà souligné, l'emploi de numéro de série pour identifier un module ne rend pas très compréhensible la manipulation des modules. Par contre, cela permet de gagner de la place mémoire dans le processeur de sécurité. On garde donc les numéros de série mais l'on maintient, sur le PC hôte, un fichier des applications faisant le lien entre le numéro de série du produit et le nom habituellement employé pour le désigner. Ceci permettra de rendre plus conviviale l'interface de l'utilisateur qui, lors d'une session de gestion des applications, ne manipulera plus les produits que par leur nom externe.

b) Structure du fichier

Le **fichier externe des applications** contient un enregistrement par module autorisé sur le système Sphinx. Chaque enregistrement contient :

1. L'identificateur du module (numéro de série) ;
2. Le nom explicite du module ;
3. Les droits des groupes sur le module ;
4. L'identificateur des fonctions ;
5. Les noms explicites des fonctions ;
6. Les droits des groupes sur les fonctions ;
7. Un lien vers l'entrée correspondant à la fonction dans la structure interne des applications.

c) Explication des choix

1) Droits d'accès

Les droits d'accès se trouvent déjà, en sécurité, dans la structure interne des applications. La redondance est introduite pour faciliter l'accès à ces informations. Ceci ne pose aucun problème de sécurité car, pour le Sphinx, les seuls droits qu'il prend en compte sont ceux qu'il mémorise dans sa propre structure interne.

2) Problème de sécurité

Si la liste externe était gérée sans précautions, elle laisserait la porte ouverte à une attaque très dangereuse pour le système. En effet, supposons qu'un administrateur veuille modifier, à la hausse, les droits d'un certain groupe vis-à-vis d'un module. D'un point de vue externe, la seule chose que voit cet administrateur, ce sont les noms explicites des modules et les droits qui lui sont associés. Si un fraudeur modifiait le lien entre nom explicite et numéro de série, l'administrateur croirait augmenter les droits sur un module alors qu'il les augmente, en interne, pour le module spécifié par le numéro de série introduit par le fraudeur. On peut aisément imaginer les dégâts que pourraient occasionner un tel cheval de Troie.

Pour contrecarrer ce type d'attaque, on calcule un MAC DES sur l'ensemble (nom explicite - identificateur), tant au niveau du module que de la fonction. Le résultat du calcul cryptographique est appelé **identificateur**

sécurisé et il est stocké dans la structure interne des applications. Il suffit alors au Sphinx de recalculer cette valeur sur le couple (identificateur, nom externe) qui lui est proposé et de comparer avec la valeur qu'il a mémorisée pour s'assurer que le couple n'a pas été dissocié.

6.3.3 Opérations permises

Les fonctions que l'on peut exécuter pour gérer les applications sont uniquement accessibles lorsque le processeur de sécurité est dans l'état actif. De plus, elles ne sont accessibles qu'à partir du PC hôte et uniquement lorsque personne d'autre n'utilise le système.

Les fonctions reprises dans le module de gestion des applications sont :

1. Initialisation d'une application dans le système ;
2. Mise à jour des droits des groupes sur un module ;
3. Mise à jour des droits des groupes sur une fonction ;
4. Recouvrement des applications ;
5. Suppression d'une application ;
6. Téléchargement sécurisé d'une application.

a) Initialisation d'une application dans le système

Lorsque l'on veut utiliser une nouvelle application sur le système Sphinx, trois temps sont nécessaires :

1. Achat du module correspondant auprès de son vendeur ;
2. Initialisation du module ;
3. Téléchargement sécurisé du PC hôte vers le processeur de sécurité.

L'initialisation d'un module se fait donc par le propriétaire du système et son vendeur simultanément. Il faut copier le module exécutable sur la mémoire secondaire du PC hôte et, ensuite, fournir la signature du module au Sphinx afin qu'il l'enregistre dans sa structure interne des applications. La signature est calculée par le Sphinx sur le texte original du module exécutable que lui fournit le vendeur.

La présence conjointe du propriétaire et du vendeur assure deux qualités :

1. Le propriétaire est assuré de l'intégrité du logiciel qu'il achète (pas de virus ni de cheval de Troie) ;

2. Le vendeur est assuré que personne ne pourra lui "pirater" son logiciel (copie illicite).

b) Mise à jour des droits des groupes sur un module

Cette fonction reçoit l'enregistrement de la liste externe des application correspondant au module, accompagné des nouveaux droits que l'on veut appliquer au module. Le Sphinx peut recalculer un MAC sur le couple (identificateur, nom explicite) et le comparer à l'identificateur sécurisé qu'il a mémorisé. S'il n'y a pas égalité, le système conclut à une attaque et refuse de poursuivre. Le Sphinx peut également vérifier l'égalité entre les anciens droits qui lui ont été fournis aux anciens droits qu'il a lui-même mémorisés. De même, si les droits ne correspondent pas, le Sphinx refuse de continuer. Si toutes les conditions précédentes sont vérifiées, le Sphinx modifie les droits dans sa structure interne des applications et recalcule la signature sur cette structure. Le PC hôte, quant à lui, met à jour le fichier externe des applications.

c) Mise à jour des droits des groupes sur une fonction

Cette fonction est semblable à la fonction précédente. Elle se différencie de cette dernière par une double vérification. La première s'effectue au niveau des modules, comme cela a été expliqué ci-dessus. La seconde est faite au niveau de la fonction où l'on compare également l'identificateur sécurisé et les anciens droits des groupes sur les modules (il y a ici deux types de droits).

d) Recouvrement des applications

Cette fonction est exécutée en cas de perte du fichier externe des applications. Elle examine la structure interne des applications pour fournir à l'extérieur les renseignements qui permettront de reconstruire ce fichier. Elle fournit de plus les renseignements "publics" qui se trouvent dans la structure interne des applications ainsi que des renseignements relatifs à l'état des modules en mémoire (téléchargé ou non).

e) Suppression d'une application

Cette fonction élimine un module de la liste interne des applications. Ceci signifie que le module ne pourra plus être utilisé sur le processeur de sécurité, à moins qu'il ne soit à nouveau initialisé. L'exécution de cette fonction se fait uniquement sur demande du propriétaire du système et après des contrôles de même type que ceux effectués lors d'une mise à jour des droits d'un module. En fin d'exécution, la signature sur la structure interne des applications est recalculée et le fichier externe mis à jour.

f) Téléchargement sécurisé d'une application

Une application initialisée sur le système ne se trouve pas perpétuellement dans la mémoire centrale du processeur de sécurité. Elle se trouve le plus souvent sur la mémoire externe du PC hôte, sous forme de module exécutable. Lorsque l'on veut utiliser un application qui ne figure pas en mémoire centrale, il faut tout d'abord l'y placer. Ceci se fait par téléchargement. Le module exécutable est envoyé au processeur de sécurité qui calcule sur celui-ci un MAC DES. Le système compare ensuite le résultat de son calcul avec la signature du module qui avait été effectuée lors de l'initialisation du module. C'est après vérification de cette signature que le système fait enfin appel au "loader/starter" qui effectuera les opérations nécessaires pour reloger les codes des différentes fonctions en mémoire centrale.

6.4 Interpréteur de commandes

6.4.1 Introduction

L'**interpréteur de commandes** est l'ensemble logiciel destiné à recevoir les messages et à analyser leur contenu. Rappelons ici que la première étape d'analyse se fait grâce au protocole d'échange que nous avons vu précédemment. Ainsi, lorsque le message est chiffré ou signé, l'interpréteur de commandes déchiffre ou (respectivement) vérifie la signature, en accord avec le protocole employé par l'utilisateur et défini dans la table des groupes. Si le message contient une requête, l'interpréteur de commandes se réserve le droit de leur donner suite ou non selon les droits de l'utilisateur qui a fait cette requête. Pour cela, il

doit avoir préalablement reconnu cet utilisateur. Nous allons donc examiner un dialogue complet et typique entre un utilisateur et le système Sphinx. Un dialogue est constitué des quatre phases suivantes :

1. ouverture de session ;
2. authentification ;
3. vie normale : cycle requête - réponse ;
4. fermeture de session.

6.4.2 Ouverture de session

Tous les dialogues entre les utilisateurs et Sphinx se font dans le cadre d'une **session**. L'utilisateur envoie donc, avec son identité et son groupe d'appartenance, une demande d'ouverture de session. A la suite de cette demande, le système Sphinx crée une occurrence d'un interpréteur de commandes et le rattache à la session. Le dialogue peut alors réellement commencer.

6.4.3 Authentification

La première action effectuée par l'interpréteur de commandes est de lancer l'exécution d'un protocole d'authentification qui correspond au groupe duquel se réclame l'utilisateur. A la suite de cette authentification, l'interpréteur de commandes sera personnalisé pour l'utilisateur actuel.

On peut définir six protocoles d'authentification différents, un par groupe d'utilisateur. Dès à présent, trois ont été définis.

a) Authentification de base du vendeur et du propriétaire

Le vendeur ou le propriétaire est authentifié grâce à sa carte placée dans le lecteur du Sphinx via le protocole de base suivant.

1. Sphinx possède dans la table des groupes une clé de diversification K_d pour l'authentification d'un utilisateur de type vendeur ou propriétaire. Cette clé a été stockée lors de la phase de personnalisation. Sphinx dispose également, grâce à la liste externe de validité, du numéro N de la carte accréditée de l'utilisateur. Il peut ainsi calculer la clé d'authentification enregistrée dans cette carte, soit $K_{auth} = \{ N \} K_d$.

2. Sphinx génère un aléa et l'envoie à la carte afin qu'elle le renvoie après l'avoir chiffré au moyen de cette clé commune.

3. Sphinx peut également effectuer ce calcul en interne et peut donc comparer le résultat qu'il reçoit au résultat qu'il vient de calculer. En cas d'égalité, le vendeur (le propriétaire) est authentifié.

De façon plus synthétique, grâce à nos conventions de notation, on obtient :

1. Sphinx : calcule K_{auth} = { N } K_d ; génère aléa R
2. Sphinx --> Carte : R
3. Carte : possède K_c (qui doit être égale à K_{auth})
4. Carte --> Sphinx : { R } K_c
5. Sphinx : Si { R } K_c = { R } K_{auth}
   Alors Authentification correcte

b) Authentification d'un invité

Un invité n'étant pas destiné à posséder des privilèges importants, le protocole d'authentification peut être très léger, peut-être même inexistant. Un minimum de sécurité peut être obtenu grâce à un protocole d'identification basé sur mot de passe. Un invité n'est donc pas obligé de posséder une carte utilisateur pour accéder au Sphinx.

c) Authentification d'un utilisateur

Ce protocole, résumé à la figure 39, est mis en oeuvre pour authentifier un utilisateur se réclamant du groupe super utilisateur, utilisateur privilégié ou utilisateur normal. Il est effectué grâce à la carte que doit posséder ce type d'utilisateur et qui doit être placée dans le lecteur attaché à une station de travail.

1. Sphinx possède dans la table des groupes une clé de diversification K_d pour l'authentification d'un utilisateur. Cette clé a été stockée lors de la phase de création de cet utilisateur. Sphinx dispose également, grâce à la liste externe de validité, du numéro N de la carte accréditée de l'utilisateur. Il peut ainsi calculer la clé d'authentification enregistrée dans cette carte, soit K_{auth} = { N } K_d.

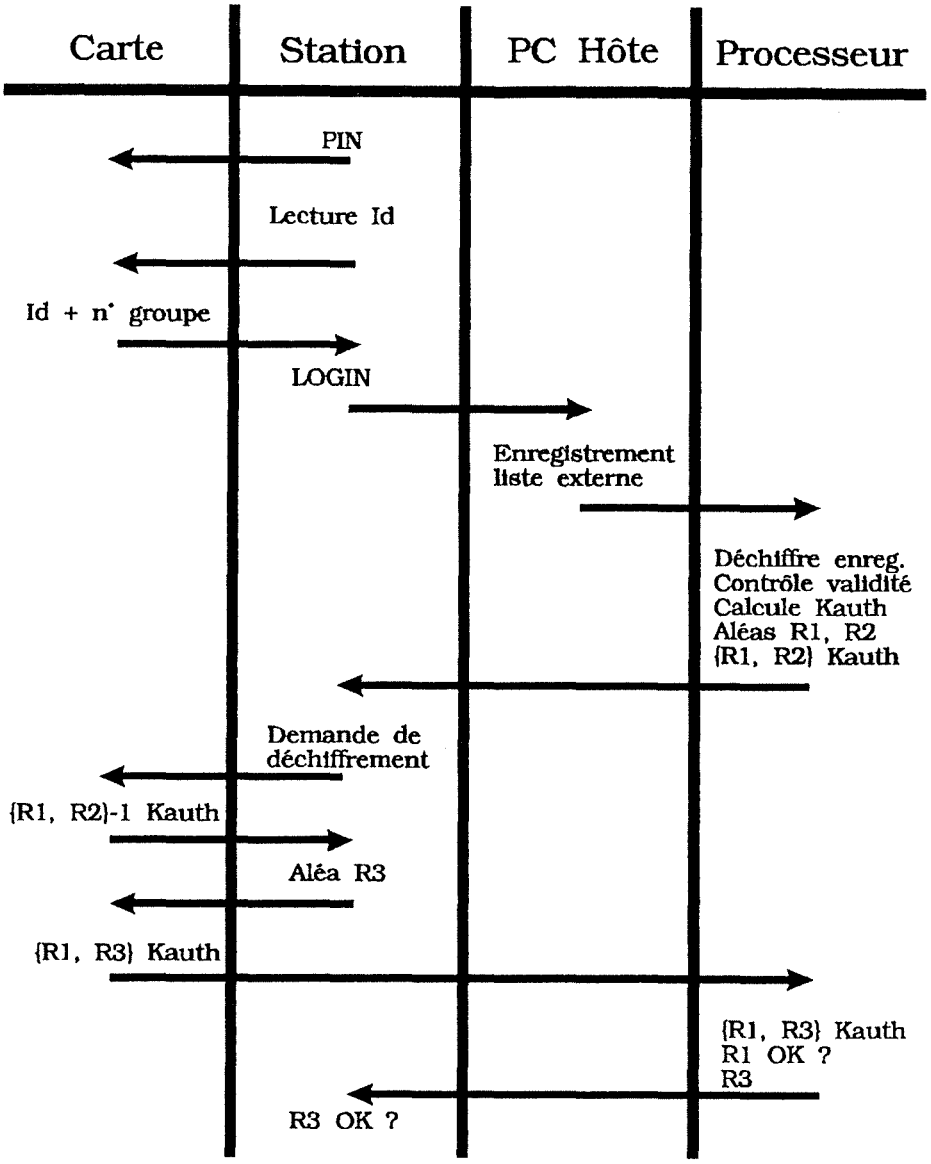


Figure 39 : Authentification d'un utilisateur.

2. Sphinx génère ensuite deux aléas R1 et R2 qu'il envoie concaténés et chiffrés avec la clé d'authentification.

3. La carte, qui est la seule à posséder également cette clé, peut déchiffrer le message pour obtenir à nouveau R1 et R2 en clair. Elle génère ensuite un aléa R3 et envoie au Sphinx R1 et R3 concaténés et chiffrés avec la clé d'authentification.

4. Sphinx, après avoir déchiffré le message, peut comparer la valeur de R1 qu'il reçoit à celle qu'il avait envoyée. S'il y a égalité, le

Sphinx est assuré de l'authenticité de la carte. Il renvoie alors l'aléa R3 seul, chiffré avec la clé d'authentification.

5. La carte peut enfin déchiffrer ce dernier message et comparer le R3 qu'elle reçoit au R3 qu'elle avait envoyé. L'authentification est alors mutuelle.

6.4.4 Cycle normal de fonctionnement

Dans le cadre d'une session et après authentification, l'interpréteur de commande attend les requêtes. Si la requête qu'il reçoit lui ordonne de fermer la session, il s'exécute. Sinon, il traite la requête et attend la suivante. Lorsque la requête n'est pas une demande de fermeture de session, l'interpréteur regarde la fonction que l'utilisateur lui demande d'effectuer. Si le module exécutable n'existe pas ou s'il n'est pas téléchargé, l'utilisateur est averti que sa requête est annulée. Si ce module exécutable est actuellement dans sa mémoire centrale, Sphinx peut envisager, à condition que l'utilisateur en possède le droit, de lancer l'exécution de la fonction.

Il est à remarquer que les modules de gestion des applications et des utilisateurs sont des modules standard et sont donc invoqués de la manière que nous venons de décrire. Il est bien évident que les droits d'exécution sur ces modules sont réservés aux super utilisateurs et au propriétaire.

6.5 Résumé du chapitre

Dans ce chapitre, nous avons donné les principaux résultats obtenus au cours de l'analyse fonctionnelle des trois modules essentiels au fonctionnement sécurisé du système Sphinx. Ces trois modules sont : la gestion des utilisateurs accrédités du système, la gestion des applications autorisées à s'exécuter sur le système et l'interpréteur des commandes du système.

Après avoir clairement défini les notions que nous allions utiliser dans la suite, nous nous sommes attardés sur les structures de données nécessaires pour les deux premiers modules. Ceci nous a permis de montrer

les problèmes de sécurité qu'elles posaient et d'expliquer ensuite une solution à chacun de ces problèmes. L'intérêt de l'emploi de techniques cryptographiques de base a été mis en évidence dans les développements de ces solutions.

Ensuite, nous avons dégagé les fonctions nécessaires à la gestion du système. Nous avons également expliqué la façon de gérer nos solutions, à base de cryptographie, à l'intérieur des fonctions qui ressortaient de l'analyse des besoins.

Dans la dernière partie, l'examen de l'interpréteur de commande nous a permis de montrer plus en détail le fonctionnement du système ainsi que le fonctionnement des protocoles d'authentification complexes utilisés pour identifier de façon stricte les utilisateurs.

Conclusions

Tout au long de notre travail, dont l'ossature est schématisée par la figure 40, nous avons voulu montrer l'escalade dans les moyens techniques développés pour combattre la fraude informatique. Dans ce but, nous avons montré, de façon presque historique, comment chaque solution apportée à un problème entraînait elle-même de nouveaux problèmes à résoudre.

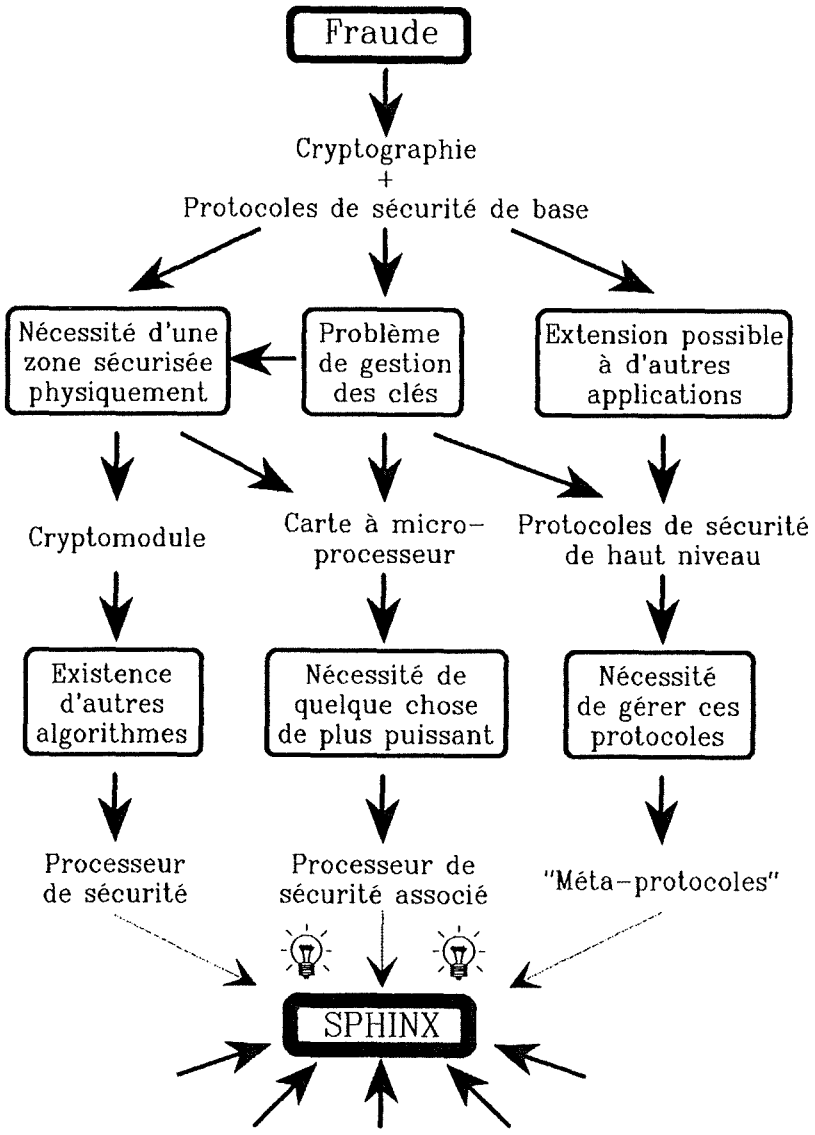


Figure 40 : Fil conducteur de notre travail.

Les premières solutions techniques solides développées pour contrer la fraude étaient basées sur la cryptographie et, plus particulièrement, sur les protocoles de sécurité de base que nous avons examinés dans un premier temps. Très vite, on s'est aperçu, à l'usage, que l'utilisation de la cryptographie posait un énorme problème de gestion des clés, ainsi que le problème, intimement lié, de la nécessité d'une zone physiquement protégée. Ces problèmes ont alors été résolus par l'apparition de cryptomodules et, plus récemment, par l'utilisation de la carte à microprocesseur. De plus, de nouveaux protocoles de sécurité, plus complexes, sont apparus pour gérer les clés ainsi que pour répondre à de nouveaux besoins. Mais, le problème de la sécurité se posant de façon de plus en plus critique, les solutions précédentes se sont avérées insuffisantes. En particulier, l'apparition de nombreux protocoles a entraîné la nécessité d'une gestion de cet ensemble de protocoles.

Ainsi, en constatant dans la pratique tous ces besoins, est venue l'idée de développer le système SPHINX, un ensemble matériel - pour la sécurité physique - et logiciel - pour la sécurité logique et la gestion des ressources. Dans la deuxième partie de notre exposé, nous avons voulu montrer comment le système que nous avions développé semblait résoudre la plupart des problèmes causés par les différentes solutions existantes.

L'historique et l'exposé des solutions adoptées pour développer le système SPHINX nous ont également permis de mettre en évidence et d'appliquer quatre règles, que nous trouvons fondamentales, lors de la conception d'une solution technique permettant de combattre la fraude informatique.

1. La sécurité est un domaine en perpétuelle évolution, très fortement lié à l'état de l'art. Avant tout choix de solution, il faut donc se renseigner de façon sérieuse sur l'existant. Pour cela, il est nécessaire d'effectuer une étude bibliographique poussée et de faire appel aux spécialistes, par l'intermédiaire des normes par exemple. Cette considération donne une raison d'être supplémentaire aux premiers chapitres de ce travail.

2. La sécurité informatique ne peut se limiter à sa composante logique. L'emploi d'une zone physiquement protégée, sous la forme d'un cryptomodule, d'une carte à microcircuit ou d'un processeur de sécurité nous paraît indispensable.

3. La sécurité doit être pensée dès le début du développement d'un système, dès les couches les plus basses. En effet, l'"ajout" de sécurité à un ensemble existant est, non seulement complexe, mais il n'apporte bien souvent qu'un sentiment de sécurité illusoire et dangereux à l'utilisateur. C'est dans cet esprit que le SPHINX a été développé dès les couches matérielles, ce qui a permis une utilisation, ou plutôt une intégration, homogène de toutes les techniques éprouvées de la sécurité. Cette intégration de la cryptographie, des protocoles de base et de haut niveau, d'un processeur de sécurité et de cartes à puce (symbolisée par les flèches montantes de la figure 40) donne au système SPHINX la cohérence nécessaire à une sécurité et une performance de haut niveau.

4. Les fraudeurs sont dynamiques et la technique évolue de jour en jour. Il ne faut donc surtout pas oublier que tout système de sécurité, comme un cryptosystème par exemple, possède une durée de vie limitée. Ainsi, si le système est trop rigide, il sera très vite dépassé et totalement inutile ou, pire, dangereux. Encore une fois, nous avons essayé de développer le SPHINX pour tenir compte de ce problème. Nous avons, tout au long de ce travail, mis l'accent sur l'ouverture de notre système : gestion dynamique des applications et des utilisateurs ; souplesse au niveau des protocoles d'authentification et d'échange ; modularisation de l'ensemble du logiciel ; possibilité d'adapter l'architecture matérielle.

Nous avons montré l'utilité de notre système dans le cadre actuel de la sécurité ainsi que les qualités principales que nous avons voulu lui conférer. Une étude pratique plus poussée serait évidemment nécessaire pour valider ce système, mais d'ores et déjà, la réalisation de la notion de serveur de sécurité apparaît comme un moyen prometteur pour combattre la fraude informatique au sein d'un réseau.

Bibliographie

[ABRAM,87] *
Marshall D. ABRAMS & Albert B. JENG
Network Security : Protocol Reference Model and the Trusted Computer System Evaluation Criteria
in : IEEE Network Magazine, vol. 1, n° 2, April 1987, pp. 24-33.

[ACHEM,86] *
Mohammed ACHEMLAL & Michel MOURRIER
Dynamic Signature Verification
in : *Information Security : The Challenge*, IFIP Security on Information Systems Security, Monte-Carlo, December 2-4 1986, pp.433-441.

[AKL,83] *
Selim G. AKL
On the Security of Compressed Encoding
in : *Advances in Cryptology*, Proceedings of Crypto 83 (University of California, Santa Barbara), Plenum Press, New-York, 1984, pp. 209-230.

[ANSI X3.92,81]
ANSI
American National Standard Data Encryption Algorithm
American National Standards Institute, 1430 Broadway, New-York, New-York 10018, 1981, 15 p.

[ANSI X3.106,83]
ANSI
American National Standard for Informations Systems - Data Encryption Algorithm - Modes of Operation
American National Standards Institute, 1430 Broadway, New-York, New-York 10018, 1983, 18 p.

[ANSI X9.9,82]
ANSI
Financial Institution Message Authentication X9.9
American National Standard ANSI X9.9, 1982, 16 p.

[AVL,89]
Axel van LAMSWEERDE
Méthodologie de développement de logiciels
Cours dispensé aux F.U.N.D.P. Namur, 1989.

[AXIS,89]
AXIS
Les cartes à mémoire : les clés indispensables pour les systèmes de communication
Ed. Milan-Midia, collection Ecomedia, 1989, 127 p.

[BALME,sd]
René-Pierre BALME
Peut-on rendre un ordinateur inviolable ?
Article extrait du n° 921 de 01 INFORMATIQUE.
in : *Mediacom*, Edition 50, Paris, n° 3, pp. 10-12.

[BERNA,89]
Alain BERNARD
Le chiffrement appliqué à la sécurité des réseaux
in : [VINCE,89], pp. 16-18.

[BORDA,89]
Pascal BORDAT
Les applications de la carte à micro-circuit dans le domaine de la sécurité du système d'information
in : [VINCE,89], pp. .

[BRANS,87] *
Dennis K. BRANSTAD
Considerations for Security in the OSI Architecture
in : *IEEE Network Magazine*, vol. 1, n° 2, April 1987, pp. 34-40.

[BRICK,88]
Ernest F. BRICKELL and Andrew M. ODLYZKO
Cryptanalysis : A Survey of Recent Results
in : *Proceedings of the IEEE*, Vol. 76, n° 5, May 1988, pp. 578-593.

[CAMPA,88]
Mireille CAMPANA & Marc GIRAULT
Comment utiliser les fonctions de condensation dans la protection des données
in : *Sécuricom 88*, 1988, pp. 91-110.

[CARRO,86]
John M. CARROL & Stephen MARTIN
Cryptographic Requirements for Secure Data Communications
in : *Information Security : The Challenge*, IFIP Security on Information Systems Security, Monte-Carlo, December 2-4 1986, pp.90-98.

[CHAMO,89]
Jean-Pierre CHAMOUX
Aspects juridiques de la sécurité informatique
in : [VINCE,89], pp. 34-37.

[CHORL,86] *
B. J. CHORLEY & W. L. PRICE
An Intelligent Token for Secure Transactions
in : *Information Security : The Challenge*, IFIP Security on Information Systems Security, Monte-Carlo, December 2-4 1986, pp.442-450.

[COHEN,85] *
Fred COHEN
A Secure Computer Network Design
in : *Computer & Security*, Elsevier Science Publishers, n° 4, 1985, pp. 189-205.

[COHEN,86]
Gérard COHEN
Un aperçu sur les codes de correction d'erreur
Publication Ministère des PTT - Sup Télécom, Aout 1986, 18 p.

[COLLI,88] *
Simon COLLIN
Zero Knowledge Proofs
in : *Computer Fraud & Security Bulletin*, Elsevier Science Publishers Ltd., England, vol. 10, n° 10, pp. 5-7, 1988.

[DAVIE,84]
D.W. DAVIES & W.L. PRICE
Security for Computer Networks - An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer
Ed. J. Wiley & Sons, 1984, ??? p.

[DAVIO,83]
Marc DAVIO et al.
Analytical Characteristics of the DES
in : *Advances in Cryptology*, Proceedings of Crypto 83 (University of California, Santa Barbara), Plenum Press, New-York, 1984, pp. 171-202.

[DEMIL,83] *
Richard DEMILLO & Michael MERRIT
Protocols for Data Security : Can Two Mutually Suspicious Participants play Poker over the Telephone ? Certainly, if They are Clever enough to Institute a Secure Protocol
in : *IEEE*, February 1983, pp. 39-50.

[DENNI,81]
Dorothy E. DENNING & Giovanni Maria SACCO
Timestamps in Key Distribution Protocols
in : *Communication of the ACM*, vol. 24, n°8, Augustus 1981, pp. 533-536.

[DIFFI,76] *
Whitfield DIFFIE and Martin E. HELLMAN
New Directions in Cryptography
in : *IEEE Transactions on Information Theory*, Vol. IT-22, n° 6, November 1976, pp. 644-654.

[DIFFI,79]
Whitfield DIFFIE & Martin E. HELLMAN
Privacy and Authentication : An Introduction to Cryptography
in : *Proceedings of the IEEE*, vol. 67, n° 3, March 1979, pp. 397-427.

[DIFFI,88] *
Whitfield DIFFIE
The Firts Ten Years of Public-Key Cryptography
in : *Proceedings of the IEEE*, Vol. 76, n° 5, May 1988, pp. 560-577.

[DROR,89]
Asael DROR
Secret Codes
in : [RASH,89], pp. 267-274.

[ELGAM,85] *
Taher ELGAMAL
A Public Key Cryptosystem and a Signature Sheme Based on Discrete Logarithms
in : *IEEE Transactions on Information Theory*, Vol. IT-31, n° 4, July 1985, pp. 469-472.

[ESTRI,89] *
Deborah ESTRIN, Jeffrey C. MOGUL & Gene TSUDIK
Visa Protocols for Controlling Interorganizational Datagram Flow
in : *IEEE Journal on Selected Areas in Communications*, vol. 7, n° 7, May 1989, pp. 486-498.

[FAIRF,85] *
R.C. FAIRFIELD, A. MATUSEVICH & J. PLANY
An LSI Digital Encryption Processor (DEP)
in : *IEEE Communications Magazine*, vol. 23, n° 7, July 1985, pp. 30-41.

[FAK,86]
Viiveke FAK
How to choose Good Cryptographic Protection
in : *Information Security : The Challenge*, IFIP Security on Information Systems Security, Monte-Carlo, December 2-4 1986, pp.210-213.

[FERRE,86]
Ronald C. FERREIRA
The Smart Card ?
in : *Information Security : The Challenge*, IFIP Security on Information Systems Security, Monte-Carlo, December 2-4 1986, pp.487-503.

[FERRE,89]
R. FERREIRA
Note on the Security of Hashing Functions in Message Authentication
Notes internes, Février 1989, 12 p.

[FERRE,89a]
Ronald C. FERREIRA & Jean-Jacques QUISQUATER
Integration of Public Key Cryptosystems in a Smart Card, Some Scenarios
ISO TC 68/SC6/WG7 Input Document, May 1989, 7 p.

[FERRE,89b]
R.C. FERREIRA
The Smart Card : A High Security Tool in EDP
in : *Philips TDS Review*, vol. 47, n° 3, September 1989, pp. 1-19.

[FIPSP,80]
FIPS
DES Modes of Operation
Federal Information Processing Standards Publication, 1980 December 2, 26 p.

[FRITZ,88]
C. FRITZER, K. PRESTTUN, J.T. RICHARDSEN & G. SOBERG
Système d'information expérimental réparti et sécurisé
in : *Revue des Télécommunications*, vol. 62, n° 3/4, 1988, pp. 310-317.

[GAILL,88]
Y. GAILLY
Définition d'une architecture de sécurité pour les grands réseaux avec création d'une boite noire
in : *Sécuricom 88*, 1988, pp. 339-357.

[GARDN,sd] *
Martin GARDNER
MATHEMATICAL GAMES : A New Kind of Cipher that would take millions of Years to Break
in : ???, pp. 120-124.

[GASPA,89]
David GASPARD
La carte à puce au secours de la sécurité informatique
in : *Electronique hebdo*, n° 135, 7 Décembre 1989, p. 12.

[GAUBE,89a]
Christian GAUBERT
Sept leçons de sécurité des réseaux
in : *TELECOM*, Revue de l'association amicale des ingénieurs de l'ENST, n° 80, été 89, pp. 19-24.

[GAUBE,89b]
M. Christian GAUBERT
Dix leçons de sécurité
in : *Sécurité Informatique*, n°41-42-43, Septembre-Octobre-Novembre 1989, pp. 4-6, pp. 4-6, pp. 4-5.

[GERAR,89]
Philippe GERARD
Pour plus de ... PC sécurisés
in : *TELECOM*, Revue de l'association amicale des ingénieurs de l'ENST, n° 80, été 89, pp. 40-42.

[GIRAU,86]
M. GIRAULT, M. CAMPANA & A. BAUVAL
Cryptographie : introduction et application à la carte à mémoire
Extrait de l'*Echo des Recherches*, n° 124, 1986
in : *Mediacom*, Edition 50, Paris, n° 3, pp. 31-42.

[GIRAU,88a]
Marc GIRAULT
Hash Functions for Digital Signatures
Working Draft ISO/IEC JTC1/SC 20/WG 2 N 118, 1988-04-08, 6 p.

[GIRAU,88b]
Marc GIRAULT
Hash Functions for Digital Signatures
Project ISO/CEI JTC 1/SC 20/WG 2 N 124 Part 1 & 2, to be registered as a DP, 1988-05-11, 8 p.

[GOUDE,89]
Jean-Luc GOUDET
Peut-on pirater la carte à puce ?
in : *Science & Vie Micro*, n°58, Février 1989, pp. 87-91.

[GREEN,89]
Ross M. GREENBERG
Know The Viral Ennemy
in : [RASH,89], pp. 275-284.

[GRISS,89]
André GRISSONNANCHE
La gestion de la sécurité du système d'information dans l'entreprise
in : [VINCE,89], pp. 11-15.

[GROVE,89]
Derrick GROVER et al.
The Protection of Computer Software - its Technology and Applications
Cambridge University Press, British Informatics Society Ltd, 1989, 263 p.

[GUEZ,88]
F. GUEZ, C. ROBERT & A. LAURET
Les cartes à microcircuit
Ed. Masson, Paris, 1988, 188 p.

[GUILL,sd]
Louis GUILLOU
Une nouvelle science : la garantique
Article extrait du n° 44 hors série de la revue *Sciences & Avenir*
in : *Mediacom*, Edition 50, Paris, n° 3, pp. 5-8.

[GUILL,87] *
Louis C. GUILLOU & Jean-Jacques QUISQUATER
Procédé d'authentification d'accréditations ou de messages à apport nul de connaissance et de signature de messages
Demande de brevet - France n°87 12366 du 7 septembre 1987, 33 p.

[GUILL,88.2] *
Louis Claude GUILLOU & Jean-Jacques QUISQUATER
Authentication Schemes Based upon Publication of Problems Constructed from Secret Solution
in : ??? Pour la science, 19-05-88, 8 p.

[GUILL,88.1] *
Louis C. GUILLOU, Marc DAVIO & Jean-Jacques QUISQUATER
L'état de l'art en matière de techniques à clé publique
in : *Ann. Télécommun.*, 43, n° 9-10, 1988, pp. 489-505.

[GUILL,89]
Louis-Claude GUILLOU
La normalisation internationale des cartes à micro-circuit à contacts
in : [VINCE,89], pp. 23-29.

[HELLM,87] *
Martin E. HELLMAN
Commercial Encryption
in : *IEEE Network Magazine*, vol. 1, n° 2, April 1987, pp. 6-10.

[HEURK,86] *
Ph. van HEURK
A Smart Card for the Belgian Banks
in : *Information Security : The Challenge*, IFIP Security on Information Systems Security, Monte-Carlo, December 2-4 1986, pp.267-275.

[HUET,sd]
Jérome HUET
Droit de l'informatique : panorama sur les cartes de paiement
in : *Mediacom*, n° 3, Edition 50, Paris, pp 49-51.

[INDJO,89]
M.D. INDJOUDJIAN
La cryptographie à clefs publiques
in : *La jaune et la rouge*, Novembre 1989, pp. 17-20.

[ISO 7498,84]
Systèmes de traitement de l'information - Interconnexion de systèmes ouverts - Modèle de référence de base
Norme Internationale ISO 7498, 15-10-1984, 51 p.

[ISO 7498-2,89]
Information Processing Systems - Open Systems Interconnection - Basic Reference Model - Part 2 : Security Architecture
International Standard ISO 7498-2, 15-02-1989, 32 p.

[ISO/DP 8227,82]
Information Processing - Data Encipherment - Specification of Algorithm DEA 1
Draft Proposal ISO/DP 8227, 1982-12-16, 24 p.

[ISO 8372,87]
Traitement de l'information - Modes opératoires d'un algorithme de chiffrement par blocs de 64 bits
Norme internationale ISO 8372, 15-08-1987, 6 p.

[ISO 8372,87.2]
Information processing - Modes of operation for a 64-bit block cipher algorithm
International Standard ISO, 1987-08-15, 7 p.

[ISO 8731-1,87]
Banques - Algorithmes approuvés pour l'authentification des messages - Partie 1 : DEA
Norme internationale ISO, 1987-06-01, 2 p.

[ISO 8732,88]
Banques : Gestion de clé
Norme internationale ISO 8732, 1988-11-15, 82 p.

[ISPHO,89]
Willem ISPHORGING
La sécurité et les cartes à mémoire
in : *TELECOM*, Revue de l'association amicale des ingénieurs de l'ENST, n° 80, été 89, pp. 38-39.

[JOLY,89] *
Louis-Noël JOLY & Mouadh LOUZIR
La sécurité informatique dans la banque
in : [VINCE,89], pp. 30-33.

[JUENE,87] *
Robert R. JUENEMAN
Electronic Document Authentication
in : *IEEE Network Magazine*, vol. 1, n° 2, April 1987, pp. 17-23.

[JUNG,87] *
Achim JUNG
Implementing the RSA Cryptosystem
in : *Computer and Security 6*, Ed. Elsevier Science Publishers B.V., 1987, pp. 342-350.

[JURGE,83] *
H. JÜRGENSEN & D.E. MATTHEWS
Some Results on the Information Theoretic Analysis of Cryptosystems
in : *Advances in Cryptology*, Proceedings of Crypto 83 (University of California, Santa Barbara), Plenum Press, New-York, 1984, pp. 303-356.

[KOCHA,89]
Martin KOCHANSKI
How Safe is It ?
in : [RASH,89], pp. 257-266.

[KOBLI,87]
Neal KOBLITZ
A Course in Number Theory and Cryptography
Graduate Texts in Mathematics, 114, Springer-Verlag New York Inc., 1987, 208 p.

[KRAYE,87]
Rola KRAYEM
La sécurité des systèmes informatiques ouverts et la protection d'accès avec la carte à microprocesseur
Thèse de doctorat d'ingénieur de l'Ecole Nationale Supérieure des Télécommunications, 1987, 143 p.

[LAMER,88]
J.-M. LAMERE, Y. LEROUX & J. TOURLY
La sécurité des réseaux : méthodes et techniques
Dunod Informatique, Bordas, Paris, 1987, 374 p.

[LAMER,89a]
J.-M. LAMERE
Le piratage informatique, mythes et réalités
in : [VINCE,89], pp. 7-10.

[LAMER,89b]
Jean-Marc LAMERE
Bonjour les dégats !
in : *TELECOM*, Revue de l'association amicale des ingénieurs de l'ENST, n° 80, été 89, pp. 14-17.

[LANGL,89]
Renaud de LANGLADE
La sécurité de votre informatique passe par la sécurité de vos cablages d'immeubles et de la couche physique
in : *TELECOM*, Revue de l'association amicale des ingénieurs de l'ENST, n° 80, été 89, pp. 47-48.

[LAROU,68]
Larousse Sélection
Dictionnaire Encyclopédique
Librairie Larousse, vol. 1, 1968, 992 p.

[LAW,89] *
Carl Edgard LAW
X.400 and OSI : Electronic Messaging into the 1990's
IBC Technical Services Ltd, 1989, 186 p.

[LEGLU,sd]
Dominique LEGLU
Les codes secrets de l'informatique
in : *Mediacom*, Edition 50, Paris, n° 3, pp 13-19.

[LEMIR,86] *
James R. LEMIRE
A New Key Management Approach for Open Communication Environments
in : *Information Security : The Challenge*, IFIP Security on Information Systems Security, Monte-Carlo, December 2-4 1986, pp.199-209.

[LEPRI,89] *
Bruno LEPRINCE & Pascal LAUTHIER
Sécurité dans les réseaux locaux
in : *TELECOM*, Revue de l'association amicale des ingénieurs de l'ENST, n° 80, été 89, pp. 59-61.

[LETWI,88]
Gordon LETWIN
Inside OS/2
Microsoft Press, 1988, 289 p.

[LOUIS,88]
Christophe LOUIS
La protection des logiciels : une solution pour ordinateurs personnels utilisant la carte à micro-calculateur
Thèse de Doctorat de l'Université Pierre et Marie Curie, Mai 1988, 166 p.

[LOUIS,89]
Christophe LOUIS
Programmation d'un D.E.S. rapide
Notes internes, Février 1989.

[MASSE,88] *
James L. MASSEY
An Introduction to Contemporary Cryptology
in : *Proceedings of the IEEE*, Vol. 76, n° 5, May 1988, pp. 533-549.

[MEUNI,89]
Jean-Louis MEUNIER
L'assurance du risque informatique
in : [VINCE,89], pp. 38-39.

[MEYER,82]
Carl H. MEYER & Stephen M. MATYAS
Cryptography : A New Dimension in Computer Data Security - A Guide for the Design and Implementation of Secure Systems
Ed. John Wiley & Sons, 1982, 755 p.

[MEYER,88]
Carl H. MEYER & Michael SCHILLING
Chargement sécurisé d'un programme avec code de détection de manipulation
in : *Sécuricom 88*, 1988, pp. 111-130.

[MITCH,89] *
Christopher MITCHELL, Michael WALKER & David RUSH
CCITT/ISO Standards for Secure Message Handling
in : *IEEE Journal on Selected Areas in Communications*, vol. 7, n° 7, May 1989, pp. 517-524.

[MOLLI,83]
Jean MOLLIER
La signature numérique des messages chiffrés
Extrait du numéro de Mars 1983 de la revue *POUR LA SCIENCE*
in : *Mediacom*, Edition 50, Paris, n° 3, pp 20-27.

[MYERS,89]
Ware MYERS
Micro View : The Road to the Supersmart Card
in : *IEEE Micro, Chips, Systems, Software and Applications*, IEEE Computer Society, California, vol. 9, n° 4, August 1989, pp. 96-95.

[NAKAO,89] *
Kouji NAKAO & Kemji SUZUKI
Proposal on a Secure Communications Service Element (SCSE) in the OSI Application Layer
in : *IEEE Journal on Selected Areas in Communications*, vol. 7, n° 7, May 1989, pp. 505-516.

[NEANT,89]
François NEANT
Sécurité informatique : l'organisation, encore de l'organisation, toujours de l'organisation
in : *TELECOM*, Revue de l'association amicale des ingénieurs de l'ENST, n° 80, été 89, pp. 12-13.

[NEEDH,78]
Roger M. NEEDHAM & Michael D. SCHROEDER.
Using Encryption for Authentication in Large Networks of Computers
in : *Communications of the ACM*, vol. 21, Num. 12, December 1978, pp. 993-999.

[NEWMA,87]
David B. NEWMAN Jr., Jim K. OMURA & Raymond L. PICKHOLTZ
Public Key Management for Network Security
in : *IEEE Network Magazine*, vol. 1, n° 2, April 1987, pp. 11-16.

[NORA,sd]
Christine NORA
Les réseaux informatiques et la sécurité
in : *Mediacom*, Edition 50, Paris, n° 3, pp 28-30.

[OMURA,88]
Jim OMURA, Byron MAYO & David HAMMOND
Gestion des clés publiques par le système SEEK
in : *Sécuricom 88*, 1988, pp. 131-146.

[PAILL,86] *
Jean-Claude PAILLE & Marc GIRAULT
The Security Processor C.R.I.P.T.
in : *Information Security : The Challenge*, IFIP Security on Information Systems Security, Monte-Carlo, December 2-4 1986, pp.127-139.

[PARK,88]
Stephen K. PARK & Keith W. MILLER
Random Number Generator : Good Ones are Hard to Find
in : *Communications of the ACM*, vol. 31, n° 10, October 1988, pp. 1192-1201.

[PINKA,89]
Denis PINKAS
Un modele de controle d'accès pour un système distribué à base d'autorité de confiance
in : *Sécuricom 89*, 1989, pp. 257-270.

[PIONN,89]
Jean PIONNIER
Télécom et sécurité
in : *TELECOM*, Revue de l'association amicale des ingénieurs de l'ENST, n° 80, été 89, pp. 9-11.

[PHILI,89]
PHILIPS Telecomunication and Data Systems
D2 High Security Smart Card : Programmer's Guide
Philips Telecomunication and Data Systems, Apeldoorn, 1989, 146 p.

[POMER,83] *
Carl POMERANCE, J.W. SMITH & S.S. WAGSTAFF
New Ideas for Factoring Large Integers
in : *Advances in Cryptology*, Proceedings of Crypto 83 (University of California, Santa Barbara), Plenum Press, New-York, 1984, pp. 81-85

[PRICE,88]
Dr Wyn L. PRICE
Security Standards for Data Networks
in : *Computer Fraud & Security Bulletin*, Elsevier Science Publishers Ltd., England, vol. 10, n° 10, pp. 7-11, 1988.

[QUISQ,89] *
Jean-Jacques QUISQUATER & Louis Claude GUILLOU
Des procédés d'authentification basés sur une publication de problèmes complexes et personnalisés dont les solutions maintenues secrètes constituent autant d'accréditations
in : *Sécuricom 89*, 1989, pp. 149-158.

[RANEA,87]
Pierre-Guy RANEA, Jean-Michel FRAY & Yves DESWARTE
SATURNE : un système tolérant les intrusions par la fragmentation-dissémination
in : *Bulletin de liaison de la Recherche en Informatique et en Automatique* (INRIA), n° 114, 1987, pp. 20-24.

[RASH,89]
Wayne RASH Jr. et al.
In depth : Security
in : *Byte*, June 1989, pp.254-291.

[RIVES,78]
R.L. RIVEST, A. SHAMIR & L. ADLEMAN
A Method for Obtaining Digital Signatures and Public-key Cryptosystems
in : *Communications of the ACM*, vol. 21, n° 2, February 1978, pp. 120-126.

[RIVES,88]
Ronald L. RIVEST
Cryptologie
Draft of January 21, 1988. Prepared for the Handbook of Theoritical Computer Science, chap. 13, 33 p.

[RONAL,89]
Edmund RONALD
Le DES en échec ? Adi Shamir, le briseur de codes
in : *01 Informatique*, 24 Avril 1989, p. 41.

[SAURA,89]
Jean-Claude SAURAT
Sécurité des systèmes : un petit détail souvent oublié, les émissions parasites compromettantes
in : *TELECOM*, Revue de l'association amicale des ingénieurs de l'ENST, n° 80, été 89, pp. 44-45.

[SECINF,89]
Pour la première fois la carte à puce intègre l'algorithme RSA
in : *Sécurité informatique*, Publi-News 3, Nanterre, vol. 4, n° 42, Octobre 1989, pp. 1 et 7.

[SEELE,89]
Donn SEELEY
Password Cracking : A game of Wits
in : Communications of the ACM, vol. 32, n° 6, June 1989.

[SELEZ,87]
A.J. SELEZNEFF
Smart Card
in : *Philips TDS Review*, vol. 45, n° 4, December 1987, pp. 32-46.

[SIMMO,88] *
Gustavus J. SIMMONS
A Survey of Information Authentication
in : *Proceedings of the IEEE*, Vol. 76, n° 5, May 1988, pp. 603-620.

[SMID,88]
Miles E. SMID and Dennis K. BRANDSTAD
The Data Encryption Standard : Past and Future
in : *Proceedings of the IEEE*, Vol. 76, n° 5, May 1988, pp. 550-559.

[STEIN,88]
Jennifer G. STEINER, Clifford NEUMAN & Jeffrey I. SCHILLER
KERBEROS : An Authentication Service for Open Network Systems
To be presented at Winter USENIX 1988.

[STEPH,89]
Peter STEPHENSON
Personal and Private
in : [RASH,89], pp. 285-289.

[TARDO,85] *
Joseph J. TARDO
Standardizing Cryptographic Services at OSI Higher Layers
in : *IEEE Communications Magazine*, vol. 23, n° 7, July 1985, pp. 25-29.

[TSUDI,89] *
Gene TSUDIK
Datagram Authentication in Internet Gateways : Implementations of Fragmentation and Dynamic Routing
in : *IEEE Journal on Selected Areas in Communications*, vol. 7, n° 7, May 1989, pp. 499-504.

[UGON,85]
M. UGON
La carte à microprocesseur et la protection des communications
Extrait de *De nouvelles architectures pour les communications*, Ed. Eyrolles, Paris, 22/25, Octobre 1985
in : *Mediacom*, Edition 50, Paris, n° 3, pp 43-46.

[UGON,89]
Michel UGON
La carte à microcalculateur : un antivirus informatique - Propos sur l'intégrité des logiciels
in : *TELECOM*, Revue de l'association amicale des ingénieurs de l'ENST, n° 80, été 89, pp. 31-35.

[VINCE,89]
Jacques VINCENT-CARREFOUR et al.
La sécurité informatique
in : *Arts & Manufactures*, la revue des centraliens, n°409, juin-juillet-aout 1989, pp. 1-39.

[VOYDO,85] *
Victor L. VOYDOCK & Stephen T. KENT
Security in High-Level Network Protocols
in : *IEEE Communications Magazine*, vol. 23, n° 7, July 1985, pp. 12-24.

[WILLI,83]
H.C. WILLIAMS
An Overview of Factoring
in : *Advances in Cryptology*, Proceedings of Crypto 83 (University of California, Santa Barbara), Plenum Press, New-York, 1984, pp. 71-80.

[WINTE,83]
Robert S. WINTERNITZ
Producing a One-Way Function from DES
in : *Advances in Cryptology*, Proceedings of Crypto 83 (University of California, Santa Barbara), Plenum Press, New-York, 1984, pp. 203-207.

[WONG,88]
Dr Ken WONG
Computer Fraud and retail Banking - What are the risks ?
in : *Computer Fraud & Security Bulletin*, Elsevier Science Publishers Ltd., England, vol. 11, n° 1, pp. 8-14, 1988.

Index