ROYAUME DU MAROC

OFFICE MAROCAIN DE LA PROPRIETE INDUSTRIELLE ET COMMERCIALE
---

(19)

المملكة المغربية

المكتب المغربي للملكية الصناعية و التجارية
---

(12) FASCICULE DE BREVET

(11) N° de publication :
MA 27811 A1

(51) Cl. internationale :
H04L 9/06

(43) Date de publication :
01.03.2006

(21) N° Dépôt :
28621

(22) Date de Dépôt :
22.11.2005

(30) Données de Priorité :
23.05.2003 EP 03011696.6

(86) Données relatives à l'entrée en phase nationale selon le PCT :
PCT/EP2004/050854 19.05.2004

(71) Demandeur(s) :
NAGRAVISION SA, 22 ROUTE DE GENEVE CH-1033 CHESEAUX-SUR-LAUSANNE (CH)

(72) Inventeur(s) :
JUNOD, PASCAL ; VAUDENAY, SERGE

(74) Mandataire :
M. MEHDI SALMOUNI-ZERHOUNI

(54) Titre : **DISPOSITIF ET PROCEDE DE CHIFFREMENT ET DE DECHIFFREMENT D'UN BLOC DE DONNEES**

(57) Abrégé : Le but de cette invention est de proposer une nouvelle méthode d'encryption qui offre un haut niveau de sécurité combiné avec une grande vitesse d'exécution. Cet objectif est obtenu par une méthode permettant de crypter ou de décrypter des blocs de données X à Y, en base à une clé principale R, cette méthode utilisant plusieurs modules connectés en série, chaque module utilisant une sous-clé RA dérivée de la clé principale R et comprenant les étapes de: - introduction d'au moins deux valeurs initiales XOL et XOR, et mélange de ces valeurs afin de former une valeur mélangée X1, - obtention d'une valeur X2 en mélangeant une première partie RAH de sous-clé RA avec la valeur X1, - obtention d'une valeur X3 en appliquant la valeur X2 à une couche de substitution, la couche de substitution comprenant au moins une boîte de substitution (sbox), chaque boîte de substitution contenant au moins une table de constantes pour laquelle l'introduction sert de pointeur et la constante visée sert de sortie, - obtention d'une valeur X4 en utilisant une boîte de diffusion de type multipermutation en base à la valeur X3, - obtention d'une valeur X5 en mélangeant une seconde partie RAL de sous-clé RA avec la valeur X4, - obtention de la

valeur X6 en appliquant à la valeur X5 une couche de substitution, - obtention d'une valeur X7 en mélangeant une première partie RAH de sous-clé Ra avec la valeur X6, - mélange de la valeur X7 avec au moins les deux valeurs XOL et XOR initiales afin d'obtenir au moins les deux valeurs X8L et X8R, X8L et X8R représentant la valeur de sortie X8 du module, cette méthode utilisant au moins deux modules, où pour chaque module une nouvelle sous-clé RA est générée à partir de la clé principale R, les valeurs initiales XO du premier module étant une division des données d'entrée X, les valeurs de sortie X8L et X8H du dernier module formant les données de sortie Y, et cette méthode comprenant en outre l'étape d'application à au moins une des valeurs X8L ou X8R d'une fonction d'orthomorphisme avant l'application de ces valeurs à l'entrée XOR et XOL du module suivant.

Résumé

Le but de cette invention est de proposer une nouvelle méthode d'encryption qui offre un haut niveau de sécurité combiné avec une grande vitesse d'exécution. Cet objectif est obtenu par une méthode permettant de crypter ou de décrypter des blocs de données X à Y, en base à une clé principale R, cette méthode utilisant plusieurs modules connectés en série, chaque module utilisant une sous-clé RA dérivée de la clé principale R et comprenant les étapes de:

- introduction d'au moins deux valeurs initiales X0L et X0R, et mélange de ces valeurs afin de former une valeur mélangée X1,
- obtention d'une valeur X2 en mélangeant une première partie RAH de sous-clé RA avec la valeur X1,
- obtention d'une valeur X3 en appliquant la valeur X2 à une couche de substitution, la couche de substitution comprenant au moins une boîte de substitution (sbox), chaque boîte de substitution contenant au moins une table de constantes pour laquelle l'introduction sert de pointeur et la constante visée sert de sortie,
- obtention d'une valeur X4 en utilisant une boîte de diffusion de type multipermutation en base à la valeur X3,
- obtention d'une valeur X5 en mélangeant une seconde partie RAL de sous-clé RA avec la valeur X4,
- obtention de la valeur X6 en appliquant à la valeur X5 une couche de substitution,
- obtention d'une valeur X7 en mélangeant une première partie RAH de sous-clé Ra avec la valeur X6,
- mélange de la valeur X7 avec au moins les deux valeurs X0L et X0R initiales afin d'obtenir au moins les deux valeurs X8L et X8R, X8L et X8R représentant la valeur de sortie X8 du module,

cette méthode utilisant au moins deux modules, où pour chaque module une nouvelle sous-clé RA est générée à partir de la clé principale R, les valeurs initiales X0 du premier module étant une division des données d'entrée X, les valeurs de sortie X8L et X8H du dernier module formant les données de sortie Y, et cette méthode comprenant en outre l'étape d'application à au moins une des valeurs X8L ou X8R d'une fonction d'orthomorphisme avant l'application de ces valeurs à l'entrée X0R et X0L du module suivant.

DISPOSITIF ET PROCEDE DE CHIFFREMENT ET DE DECHIFFREMENT D'UN BLOC DE DONNEES

La présente invention fait référence à un dispositif et à une méthode d'encryption et de décryption d'un bloc de données connu sous forme de chiffrement par blocs, la taille du bloc d'entrée et du bloc de sortie étant le même.

Cette opération est contrôlée en utilisant une clé qui pourrait être soit de la même taille que le bloc soit de taille différente, généralement de taille plus grande.

Cette invention fait référence à une méthode d'encrytion / décryption symétrique à l'opposé de la méthode asymétrique. La méthode symétrique est caractérisée par l'utilisation de la même clé servant à crypter et à décrypter les données tandis que la méthode asymétrique utilise une première clé pour crypter et une deuxième clé pour décrypter les données.

Des méthodes bien connues sont celles de DES (clé 56-bits), CAST (clé 128-bits), Blowfish (clé 448-bits), Twofish (clé 256-bits), et Rijndael (également connu comme AES, clé 256-bits). En fonction des applications requises, elles possèdent leur propre avantage et inconvénient.

Plusieurs brevets décrivant ces méthodes ont été publiés. Le brevet USA 5.214.703 décrit la méthode connue comme IDEA™ qui est basée sur un procédé d'encryption d'opérations de 8.5 tours pour une longueur de bloc de 64 bits, chacun des tours utilisant 6 sous-clés dérivées de la clé principale. Le noyau est constitué par une méthode Lai-Massey utilisant un modulo d'addition 2^{16}, modulo de multiplication $2^{16}+1$ et une fonction exclusive OU de manipulation de bits .

Les deux exigences principales pour une méthode d'encryption sont la robustesse contre n'importe quelle forme de cryptanalyse et la vitesse de calcul. On obtient un facteur clef de robustesse par l'effet de diffusion, c'est-à-dire que lorsqu'un bit change dans les données d'entrée, tous les bits de sortie sont influencés d'une manière imprévisible.

La vitesse de calcul est déterminée principalement par le type d'opérations mathématiques et logiques nécessaires. Un nombre d'opérations plus complexes (division, multiplication) peut prolonger le temps d'exécution du procédé d'encryption.

Le but de cette invention est de proposer une nouvelle méthode d'encryption qui offre un niveau de sécurité élevé combiné avec une vitesse d'exécution élevée.

Cet objectif est obtenu par une méthode pour crypter ou décrypter des blocs de données X à Y, basé sur une clé principale R, cette méthode utilisant plusieurs modules connectés en série,

chaque module utilisant une sous-clé Ra dérivée de la clé principale R et comprenant les étapes de:

- introduction d'au moins deux valeurs initiales X0L et X0R,
- mélange d'au moins deux valeurs X0L et X0R afin de former une valeur mixte X1,
- obtention d'une valeur X2 en mélangeant une première partie RAH de sous-clé Ra avec la valeur X1,
- obtention d'une valeur X3 en appliquant la valeur X2 à une couche de substitution, la couche de substitution comprenant au moins une boîte de substitution (sbox), chaque boîte de substitution contenant au moins une table de constantes pour laquelle l'entrée sert aussi de pointeur et la constante pointée sert de sortie,
- obtention d'une valeur X4 utilisant une boîte de diffusion de type multipermutation en base à la valeur X3,
- obtention d'une valeur X5 en mélangeant une seconde partie RAL de sous-clé RA avec la valeur X4,
- obtention de la valeur X6 en appliquant à la valeur X5 une couche de substitution,
- obtention d'une valeur X7 en mélangeant une première partie RAH de sous-clé RA avec la valeur X6,
- mélange de la valeur X7 avec au moins les deux valeurs initiales X0L et X0R afin d'obtenir au moins les deux valeurs X8L et X8R, X8L et X8R représentant la valeur de sortie X8 du module,

cette méthode utilisant au moins deux modules, où pour chaque module une nouvelle sous-clé Ra est générée à partir de la clé principale R, les valeurs initiales X0 du premier module étant une division des données d'entrée X, les valeurs de sortie X8L et X8H du dernier module formant les données de sortie Y, et cette méthode comprenant en outre l'étape d'application à au moins une des valeurs X8L ou X8R d'une fonction d'orthomorphisme avant l'application de ces valeurs à l'entrée X0R et X0L du module suivant.

Les deux parties principales de la méthode sont la couche de substitution et la matrice de multipermutation.

Le but de la couche de substitution est de transformer la valeur d'entrée en une valeur de sortie sans relation algébrique simple. C'est pourquoi la manière la plus rapide consiste à utiliser une table de verrouillage contenant des constantes avec lesquelles on peut obtenir le résultat de confusion désiré.

Étant donné que dans cette forme de réalisation, les données d'entrée ont une longueur de 32 bits, le nombre de constantes sera de 2^{32} valeurs chacune de d'une longueur de 32 bits.

Conformément à une forme de réalisation préférée, les données d'entrée sont fractionnées en groupes d'une longueur de 8-bits réduisant ainsi le nombre de constantes à 256 octets.

Les données d'entrée de 32 bits ou 64 bits sont ensuite divisées en octets de 8 bits et appliquées à la boîte de substitution de manière à obtenir une sortie de 8 bits. Les données d'entrée sont utilisées sous forme de pointeur d'adresse et la constante signalée est la sortie.

En fonction de la méthode d'implantation, les tables de constante sont les mêmes pour tous les groupes de données d'entrée (32 bits ou 64 bits). Dans une autre forme de réalisation, les tables de constante sont différentes pour chaque groupe de données d'entrée.

Les constantes stockées dans cette table sont une permutation fixe de numéros qui sont tous différents, codifiées par un nombre de bits égal à la largeur de la table.

La deuxième partie principale de la méthode est la matrice de multipermutation. La matrice de multipermutation est une matrice carrée qui garantit que chaque sous-matrice carrée possible possède un déterminant différent de zéro; les éléments de la matrice sont les éléments d'un champ fini. L'opération de mélange consiste à multiplier un vecteur d'éléments d'entrée par la matrice, donnant comme résultat un vecteur qui est défini comme la sortie.

Brève description des dessins

- la figure 1 montre le schéma fonctionnel du module principal dans la version de 64 bits,
- la figure 2 montre le procédé principal incluant un exemple avec deux modules
- la figure 3 montre la partie interne du module principal, dans la version de 64 bits
- la figure 4 montre le schéma fonctionnel du module principal dans la version de 128 bits,
- la figure 5 montre le schéma fonctionnel de la fonction d'orthomorphisme,
- la figure 6 montre le sous-système de génération de la boîte de substitution,
- la figure 7 montre la partie interne du module principal, dans la version de 128 bits.
- la figure 8 montre le procédé principal incluant un exemple avec deux modules dans la version 128 bits, et
- la figure 9 montre une version alternative de la boîte de substitution.

Description détaillée de l'invention

La Figure 1 montre le squelette du procédé d'encryption (ou de décryption) qui représente le module MOD. Les données d'entrée X0 de 64 bits, qui sont représentées en deux parties X0L et X0R de 32 bits chacune, sont d'abord mélangées dans l'élément mélangeur MX afin d'obtenir la valeur X1. Cet élément mélangeur est prévu pour fournir une image de 32 bits de deux fois 32 bits de données. Ceci peut être obtenu de différentes manières, comme en utilisant une fonction XOR, par l'addition d'un modulo, ou en utilisant n'importe quel droit de groupe

L'étape suivante est illustrée avec le bloc f32 qui a une entrée de 32 bits X1 et une sortie de 32 bits X7 ainsi qu'avec l'utilisation d'une sous-clé RA. La description détaillée de ce bloc est donnée en référence à la figure 3 (voir ci-dessous).

La sortie X7 du bloc f32 est appliquée aux deux blocs de mélange MX qui sont connectés avec les deux entrées X0L et X0H.

Les données X8L et X8R obtenues représentent les deux sorties X8 de 64 bits du module MOD.

La figure 2 montre le procédé complet utilisant au moins deux modules MOD. Les données d'entrée X sont d'abord appliquées à un module de division SP qui convertit l'entrée X de 64 bits en deux valeurs de sortie X0L1 et X0R1, chacune d'une longueur de 32-bits,

La fonction de ce module de division SP pourrait être obtenue de différentes façons comme en sélectionnant les bits les plus bas pour X0L1 et les bits les plus élevés pour X0R1, ou chaque bit impair pour X0L1 et même les bits pour X0R1. D'autres méthodes de division des données d'entrée X pourraient être utilisées aussi longtemps que tous les bits de X sont inclus dans X0L1 et X0R1.

Les sorties X0L1 et X0R1 sont ensuite utilisées sous forme d'entrées dans le premier module MOD1. Ce premier module traite les données en utilisant une première sous-clé RA1. Le traitement pour X0L1 et X0R1 est le même que celui décrit conformément à La Figure 1. Les sorties de ce premier module MOD1 sont deux sorties X8L1 et X8R1. Une fonction d'orthomorphisme est appliquée à l'une de ces sorties, par exemple à X8L1 comme illustré dans La Figure 2. La sortie obtenue à partir de cette fonction d'orthomorphisme porte la référence X0L2. L'autre valeur X8R1 obtenue à partir du traitement par le premier module MOD1 est utilisée sous forme d'entrée, ainsi que la sortie X0L2 obtenue à partir de la fonction d'orthomorphisme, dans un deuxième module de traitement MOD2. Ce deuxième module MOD2 va traiter leurs données d'entrée basées sur une deuxième sous-clé RA2. Les sorties de ce deuxième module portent comme référence X8L2 et X8R2 dans La Figure 2. Ces sorties sont assemblées de manière à former les données cryptées Y dans le module assembleur AS. Ce module AS a la même fonction que le module de division SP mais fonctionne de façon inverse. Il est à noter que la forme de reconstruction de la sortie Y pourrait être différente au module de division SP mais l'objectif reste le même. Tous les bits de X8L2 et X8R2 devraient être présents à la sortie Y.

La figure 3 montre de façon détaillée, les fonctions du bloc f32 de la Figure 1. Dans ce bloc, des données X1 d'une longueur de 32-bits représentent l'entrée. Ces données sont séparées en blocs d'une longueur de 8-bits (X1a, X1b, X1c, X1d) à travers un bloc de division SPMU,

également appelé X1' dans la figure 3. Ce bloc a la même fonction que celle décrite en relation avec le bloc SP de la figure 2. Chacun de ces blocs de 8-bits est mélangé avec une première partie RAH de la sous-clé RA de manière à obtenir une valeur X2a, X2b, X2c, X2d (formant la valeur X2). Cette opération de mélange est la même que celle décrite en relation avec le bloc MX de la figure 1.

La génération des deux sous-clés RAH et RAL se fait à travers le module de division SP. Ce module a la même fonction que celle décrite dans la figure 1.

Chacune de ces valeurs X2a à X2d est appliquée à une couche de substitution, comprenant au moins une boîte de substitution (sbox), chaque boîte de substitution contenant une table de constantes pour laquelle l'entrée sert de pointeur et la constante signalée sert de sortie. La donnée de sortie porte comme référence X3a, X3b, X3c, X3d (formant la valeur X3) dans La Figure 3.

Une méthode pour générer cette table de constante consiste à utiliser un générateur pseudo-aléatoire. On devrait éliminer toutes les valeurs en duplicata de manière à ce que chaque constante de cette table soit unique.

Cette donnée est introduite dans une boîte de diffusion Mu4 de type multipermutation (4.4). La donnée de sortie de cette boîte de diffusion porte comme référence X4a, X4b, X4c, X4d respectivement (formant la valeur X4). La boîte de diffusion consiste en une multiplication du vecteur d'entrée (X4a, X4b, X4c, X4d) par une matrice carré 4x4 Mu4, dont les éléments appartiennent au champ fini de 256 éléments; ces éléments sont indiqués par Mu(i, J), où i fait référence à l'index en ligne et j à l'index en colonne. Le résultat de la multiplication du vecteur (X4a, X4b, X4c, X4d) par la matrice Mu4 est un vecteur (Y4a, Y4b, Y4c, Y4d) où ces valeurs sont obtenues telles qu'indiquées à continuation:

$$Y4a = Mu4(1, 1)*X4a + Mu4(1, 2)*X4b + Mu4(1, 3)*X4c + Mu4(1, 4)*X4d$$

$$Y4b = Mu4(2, 1)*X4a + Mu4(2, 2)*X4b + Mu4(2, 3)*X4c + Mu4(2, 4)*X4d$$

$$Y4c = Mu4(3, 1)*X4a + Mu4(3, 2)*X4b + Mu4(3, 3)*X4c + Mu4(3, 4)*X4d$$

$$Y4d = Mu4(4, 1)*X4a + Mu4(4, 2)*X4b + Mu4(4, 3)*X4c + Mu4(4, 4)*X4d$$

Ici „+" indique l'addition dans le champ fini et „*" sa multiplication. Les éléments de Mu4 sont choisis de manière à ce que la quantité de calculs nécessaires pour évaluer les quatre expressions antérieures est minimale. Le nombre de multiplications par la constante "1" (indiquées à continuation par "identités") a par conséquent été choisi de manière à être aussi grand que possible.

Les données sont ensuite mélangées avec une seconde partie RAL de sous-clé Ra afin d'obtenir une valeur X5a, X5b, X5c, X5d (formant la valeur X5).

Chacune de ces valeurs X5a à X5d est ensuite appliquée à un bloc de substitution (sbox) de manière à obtenir une valeur X6a, X6b, X6c, X6d (formant la valeur X6). Ces valeurs sont mélangées avec une première partie RAH de sous-clé RA afin d'obtenir des nouvelles valeurs X7a, X7b, X7c, X7d (formant la valeur X7).

Ces valeurs X7a, X7b, X7c, X7d sont ensuite assemblées de manière à former les données de sortie X7 dans le module d'assemblage tel que décrit en relation avec la figure 2. Ces données correspondent aux données de sortie X7 du bloc f32 dans La Figure 1

Lors du procédé d'encryption, la clé principale R est divisée en plusieurs sous-clés, une par module MOD. Dans l'exemple de la figure 3, la première sous-clé RA1 est utilisée en combinaison avec le module MOD1 et la deuxième sous-clé RA2 est utilisée en combinaison avec le module MOD2.

Pour obtenir les données X basées sur les données Y et la clé R, le même procédé que celui qui a été décrit en référence à la figure 3 est appliqué à la seule différence que les sous-clés sont générées dans l'ordre inverse. La sous-clé RA2 est ensuite appliquée au premier module MOD1 et la sous-clé RA1 est appliquée au deuxième module MOD2.

Conformément au principe général de cette invention, le nombre de modules MOD connectés en série n'est pas limité à deux modules. Afin d'obtenir une bonne robustesse, une expérience a montré que 9 tours sont optimaux pour obtenir un résultat que l'on pourrait qualifier de procédé d'encryption. Ce nombre pourrait s'étendre à 12 ou plus de manière à obtenir plus de robustesse.

La figure 4 décrit une forme de réalisation du module MOD64 conçu pour le traitement des données d'une longueur de 128-bits. Les entrées X0LL et X0LR sont mélangées dans l'élément mélangeur MX de manière à former la valeur de sortie X1L et de la même manière, les valeurs X0RL et X0RR sont mélangées de manière à former la valeur X1R.
L'étape suivante est illustrée avec la couche f64 qui possède deux entrées de 32 bits X1L et X1R et deux sorties de 32 bits X7L et X7R ainsi qu'en utilisant une sous-clé RA. La description détaillée de ce bloc est donnée en référence à la figure 7 (voir ci-dessous).

Chacune de ces sorties est mélangée avec deux données d'entrée du module MOD64 à l'intérieur du même élément de mélange MX. Dans notre exemple, la valeur de sortie X7L est mélangée avec l'entrée X0LL et X0LR respectivement et la valeur de sortie X7R est mélangée avec l'entrée X0RL et X0RR respectivement. Une autre combinaison de mélange est également

possible, telle qu'un mélange de la valeur de sortie X7L avec X0LL et X0RR dans une configuration transversale.

La figure 5 est une illustration d'une forme de réalisation de la fonction d'orthomorphisme. Les données d'entrée sont indiquées par ZI et les données de sortie par ZO. La longueur des données n'est pas un résultat de cette fonction. Les données d'entrée ZI sont d'abord divisées en deux valeurs ZL et ZR de la même taille avec le module de division SP. Les deux valeurs sont ensuite mélangées avec l'élément de mélange également appelé MX et la sortie de l'élément est appliquée à l'unité d'assemblage AS. L'autre valeur fractionnée ZR est directement appliquée au module d'assemblage sans modification. Ce module comprend deux entrées et combine ces données de manière à former la valeur de sortie ZO. Ce module travaille à l'inverse du module de division SP. La particularité de cette forme de réalisation est que les entrées du module d'assemblage sont croisées par rapport aux sorties du module de division SP. La sortie ZR située à droite du module de division SP est appliquée à l'entrée située à gauche du module d'assemblage AS et la sortie située à gauche ZL du module de division SP, après avoir été mélangée avec l'autre sortie du module de division SP, est appliquée à l'entrée située à droite du module d'assemblage AS.

En ce qui concerne la boîte de substitution, il existe différentes possibilités de réaliser cette fonction. On a décrit antérieurement une méthode basée uniquement sur une table de constante. La première étape pour réduire la taille de la table est de fractionner l'entrée et d'appliquer cette partie à une table beaucoup plus petite.

L'exemple de la figure 3 montre une boîte de substitution travaillant avec une longueur de données de 8-bits englobant ainsi une table de 256 constantes.

Dans certains cas, en particulier là où la taille de la mémoire pose un problème, d'autres alternatives sont requises. Une telle alternative est décrite en référence aux figures 6 et 9.

La figure 3 montre un sous-système Cbox de cette boîte de substitution, sous-système comprenant une entrée C divisée en deux entrées CL et CR et deux sorties CL'et CR'.

Le cœur de ce sous-système est le module TA comprenant une table de constante de $2^{(n/2)}$ éléments, chacun de n/2 bits, où n représente la longueur de la valeur d'entrée C.

Pour une entrée ayant une longueur de 8 bits, la table de constante comprend 16 (2^4) éléments, chacun d'une longueur de 4-bits. Ces éléments sont générés de façon aléatoire, en tenant en compte que chaque élément possède une valeur unique.

La figure 9 décrit comment utiliser le module Cbox pour construire une boîte de substitution. La valeur d'entrée CI est d'abord fractionnée en deux parties CL1 et CR1 et appliquée au premier module Cbox1 comme décrit en référence à la figure 3. La sortie dudit module Cbox1 est envoyée vers le module suivant Cbox2. Une des sorties du premier module, dans ce cas CL1', avant l'application au deuxième module CVBox2, est fournie à une fonction d'orthomorphisme OR.

L'exécution de la boîte de substitution utilise généralement au moins deux sous-systèmes Cbox, chacun ayant une table de constante TA différente. Dans l'exemple illustré, la boîte de substitution est formée en utilisant trois sous-systèmes Cbox et les sorties du dernier sous-système ne possèdent pas de fonction d'orthomorphisme OR conformément à la forme de réalisation.

La figure 7 est une alternative de la forme de réalisation décrite dans la figure 3, conçue pour des données d'une longueur de 64-bits. La structure conçue pour 32 bits est quasi totalement doublée afin de traiter des données de 64-bits. Les données d'entrée X1 sont divisées dans un vecteur avec des éléments d'une longueur de 8-bits (X1a à X1h) et traitées de la même manière comme décrit en relation avec la figure 3. La différence principale réside dans la boîte de diffusion Mu8 qui est une matrice carré de 8x8 éléments du champ fini de 256 éléments. Les éléments de la matrice sont indiqués par Mu8(i, j), où i fait référence à l'index en ligne et j à l'index en colonne. Pour un vecteur d'entrée (X3a, ..., X3h), la multiplication par la matrice Mu8 fournit le vecteur de sortie (Y3a, ..., Y3h) de la façon suivante ("+" est l'addition et "*" la multiplication dans le champ fini):

$$Y3a = Mu8(1,1)*X3a + Mu8(1,2)*X3b + Mu8(1,3)*X3c + Mu8(1,4)*X3d + Mu8(1,5)*X3e + Mu8(1,6)*X3f + Mu8(1,7)*X3g + Mu8(1,8)*X3h;$$

$$Y3b = Mu8(2,1)*X3a + Mu8(2,2)*X3b + Mu8(2,3)*X3c + Mu8(2,4)*X3d + Mu8(2,5)*X3e + Mu8(2,6)*X3f + Mu8(2,7)*X3g + Mu8(2,8)*X3h;$$

$$Y3c = Mu8(3,1)*X3a + Mu8(3,2)*X3b + Mu8(3,3)*X3c + Mu8(3,4)*X3d + Mu8(3,5)*X3e + Mu8(3,6)*X3f + Mu8(3,7)*X3g + Mu8(3,8)*X3h;$$

$$Y3d = Mu8(4,1)*X3a + Mu8(4,2)*X3b + Mu8(4,3)*X3c + Mu8(4,4)*X3d + Mu8(4,5)*X3e + Mu8(4,6)*X3f + Mu8(4,7)*X3g + Mu8(4,8)*X3h;$$

$$Y3e = Mu8(5,1)*X3a + Mu8(5,2)*X3b + Mu8(5,3)*X3c + Mu8(5,4)*X3d + Mu8(5,5)*X3e + Mu8(5,6)*X3f + Mu8(5,7)*X3g + Mu8(5,8)*X3h;$$

Y3f = Mu8(6,1)*X3a + Mu8(6,2)*X3b + Mu8(6,3)*X3c + Mu8(6,4)*X3d + Mu8(6,5)*X3e + Mu8(6,6)*X3f + Mu8(6,7)*X3g + Mu8(6,8)*X3h;

Y3g = Mu8(7,1)*X3a + Mu8(7,2)*X3b + Mu8(7,3)*X3c + Mu8(7,4)*X3d + Mu8(7,5)*X3e + Mu8(7,6)*X3f + Mu8(7,7)*X3g + Mu8(7,8)*X3h;

Y3h = Mu8(8,1)*X3a + Mu8(8,2)*X3b + Mu8(8,3)*X3c + Mu8(8,4)*X3d + Mu8(8,5)*X3e + Mu8(8,6)*X3f + Mu8(8,7)*X3g + Mu8(8,8)*X3h;

La figure 8 décrit le procédé complet utilisant deux tours d'exécution du module MOD64. Le module de division SP divise les données d'entrée d'une longueur de 128-bits X en quatre parties, à savoir X0LL1, X0LR1, X0RL1 et X0RR1 (formant la valeur X0). Deux parties du résultat du module MOD64-1 sont ensuite appliquées à une fonction d'orthomorphisme OR, avant d'être utilisées comme entrée du module suivant MOD64-2.

La position de la fonction d'orthomorphisme OR par rapport aux sorties du module MOD64 n'est pas décisive. On peut sélectionner les deux sorties de gauche ou les deux sorties de droite en fonction de la mise en place de cette méthode.

La sortie Y est obtenue directement du dernier module MOD64, sans avoir de fonction d'orthomorphisme OR à l'une de ces sorties.

Dans le cas où on utilise plus de deux modules MOD64, la fonction d'orthomorphisme OR est placée entre chaque module MOD64. Même si dans la forme de réalisation préférée la position de la fonction d'orthomorphisme OR est la même indépendamment du nombre de modules, dans une autre forme de réalisation, la position de cette fonction d'orthomorphisme OR peut être changée afin d'être connectée à une sortie différente du module MOD64.

Revendications

1. Méthode pour crypter ou décrypter des blocs de données X à Y, en base à une clé principale R, cette méthode utilisant au moins deux modules principaux connectés en série (MOD), chaque module principal (MOD) utilisant une sous-clé (RA) dérivée de la clé principale (R), comprenant les étapes de:

- introduction d'au moins deux valeurs initiales X0L et X0R,
- mélange d'au moins deux valeurs X0L et X0R de manière à former une valeur mélangée X1,
- obtention d'une valeur X2 en mélangeant une première partie RAH de sous-clé RA avec la valeur X1,
- obtention d'une valeur X3 en appliquant la valeur X2 à une couche de substitution, la couche de substitution comprenant au moins une boîte de substitution (sbox), chaque boîte de substitution contenant une table de constantes pour laquelle l'entrée sert de pointeur et la constante visée sert de sortie,
- obtention d'une valeur X4 en utilisant une boîte de diffusion de type multipermutation basée sur la valeur X3,
- obtention d'une valeur X5 en mélangeant une seconde partie RAL de sous-clé RA avec la valeur X4,
- obtention de la valeur X6 en appliquant un bloc de substitution à la valeur X5,
- obtention d'une valeur X7 en mélangeant une première partie RAH de sous-clé RA avec la valeur X6,
- mélange de la valeur X7 avec au moins deux valeurs X0L et X0R initiales afin d'obtenir au moins les deux valeurs X8L et X8R, X8L et X8R représentant la valeur de sortie X8 du module,

où pour chaque module principal (MOD) une nouvelle sous-clé (RA) est générée à partir de la clé principale (R), les valeurs initiales X0L et X0R du premier module étant un sous-ensemble des données d'entrée X, les valeurs de sortie X8L et X8H du dernier module formant les données de sortie Y, et cette méthode comprend en outre l'étape d'application à au moins une des valeurs X8L ou X8R d'une fonction d'orthomorphisme avant l'application de ces valeurs à l'entrée X0R et X0L du module suivant principal.

2. Méthode d'encryption ou de décryption selon la revendication 1, où les données d'entrée ont une longueur de 64 bits et les données d'entrée X sont divisées en deux valeurs initiales X0L et X0H d'une longueur de 32 bits, et les deux valeurs de sortie X8L et X8H forment les données de sortie Y.

3. Méthode d'encryption ou de décryption selon la revendication 1, où les données d'entrée ont une longueur de 128 bits et les données d'entrée X sont divisées en quatre valeurs initiales X0LL, X0LR, X0RL et X0RR d'une longueur de 32 bits, et les quatre valeurs de sortie X8LL, X8LR, X8RL et X8RR forment les données de sortie de 128 bits, une première partie X1L

de la valeur X1 est obtenue en mélangeant la valeur X0LL avec X0LR et la seconde partie X1R de la valeur X1 est obtenue en mélangeant la valeur X0RL avec X0RR, une première partie X7L de la valeur X7 est mélangée avec deux des quatre valeurs initiales X0LL, X0LR, X0RL et X0RR et la seconde partie X7R de la valeur X7 est mélangée avec les deux autres parties des valeurs initiales X0LL, X0LR, X0RL et X0RR.

4. Méthode d'encryption ou de décryption selon la revendication 1, où la couche de substitution comprend plusieurs boîtes de substitution (sbox), chaque boîte ayant une entrée de 8-bits et une sortie de 8-bits, l'entrée de la couche de substitution étant divisée en parties d'une longueur de 8-bits.

5. Méthode d'encryption ou de décryption selon la revendication 4, où la table de constantes (TA) de la boîte de substitution (sbox) contient pour une entrée donnée une sortie unique.

6. Méthode d'encryption ou de décryption selon la revendication 4, où la table de constantes est la même pour chaque boîte de substitution (sbox).

7. Méthode d'encryption ou de décryption selon la revendication 4, où la table de constantes est différente pour chaque boîte de substitution (sbox).

8. Méthode d'encryption ou de décryption selon la revendication 4, où la table de constantes de la boîte de substitution (sbox) change pour chaque exécution de module principal.

9. Méthode d'encryption ou de décryption selon la revendication 1, où la longueur des données est de 64 bits et la boîte de diffusion est une fonction matrice Y3 = M * X4, l'argument M définissant des additions 4*4, des multiplications par une constante, ou des identités, dont au moins une ligne et une colonne comprend trois identités.

10. Méthode d'encryption ou de décryption selon la revendication 9, où les lignes et les colonnes restantes de l'argument M comprennent deux identités.

11. Méthode d'encryption ou de décryption selon la revendication 1, où la longueur des données est de 128 bits et la boîte de diffusion est une fonction matrice Y3 = N * X3, l'argument N définissant 8*8 additions, multiplications par une constante, ou identités, dont au moins une ligne et une colonne comprennent sept identités.

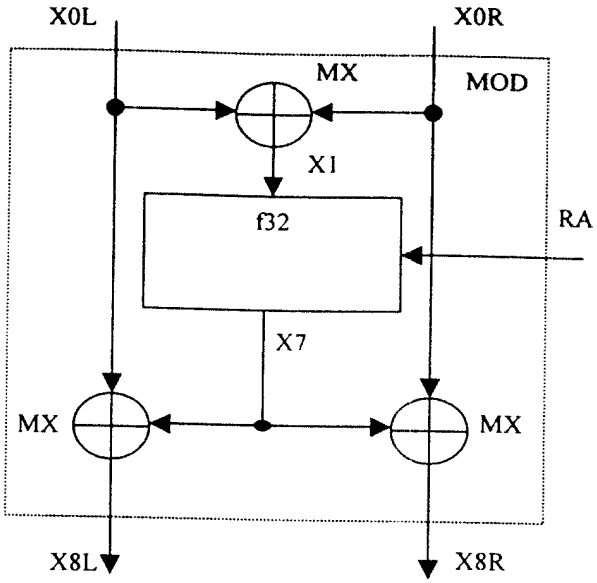

X0L
X0R
MX
MOD
X1
f32
RA
X7
MX
MX
X8L
X8R

FIG. 1

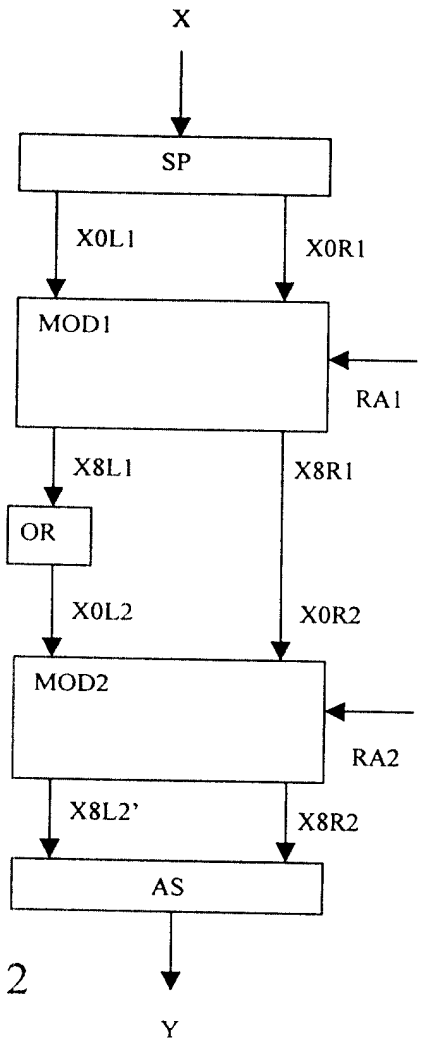

X
SP
X0L1
X0R1
MOD1
RA1
X8L1
X8R1
OR
X0L2
X0R2
MOD2
RA2
X8L2'
X8R2
AS
Y

FIG. 2

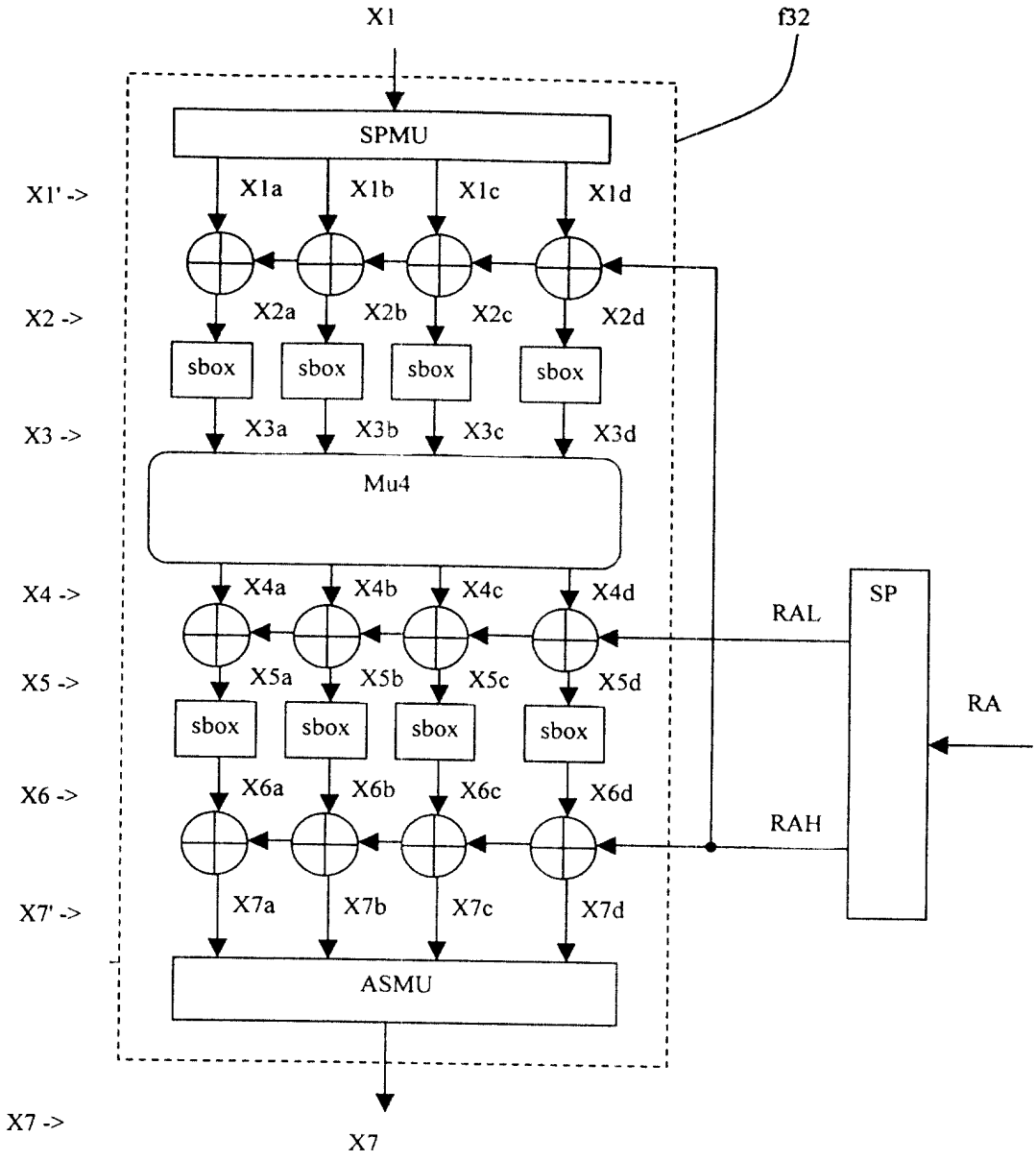

X1
f32
SPMU
X1' ->
X1a
X1b
X1c
X1d
X2 ->
X2a
X2b
X2c
X2d
sbox
sbox
sbox
sbox
X3 ->
X3a
X3b
X3c
X3d
Mu4
X4 ->
X4a
X4b
X4c
X4d
RAL
SP
X5 ->
X5a
X5b
X5c
X5d
RA
sbox
sbox
sbox
sbox
X6 ->
X6a
X6b
X6c
X6d
RAH
X7' ->
X7a
X7b
X7c
X7d
ASMU
X7 ->
X7

FIG. 3

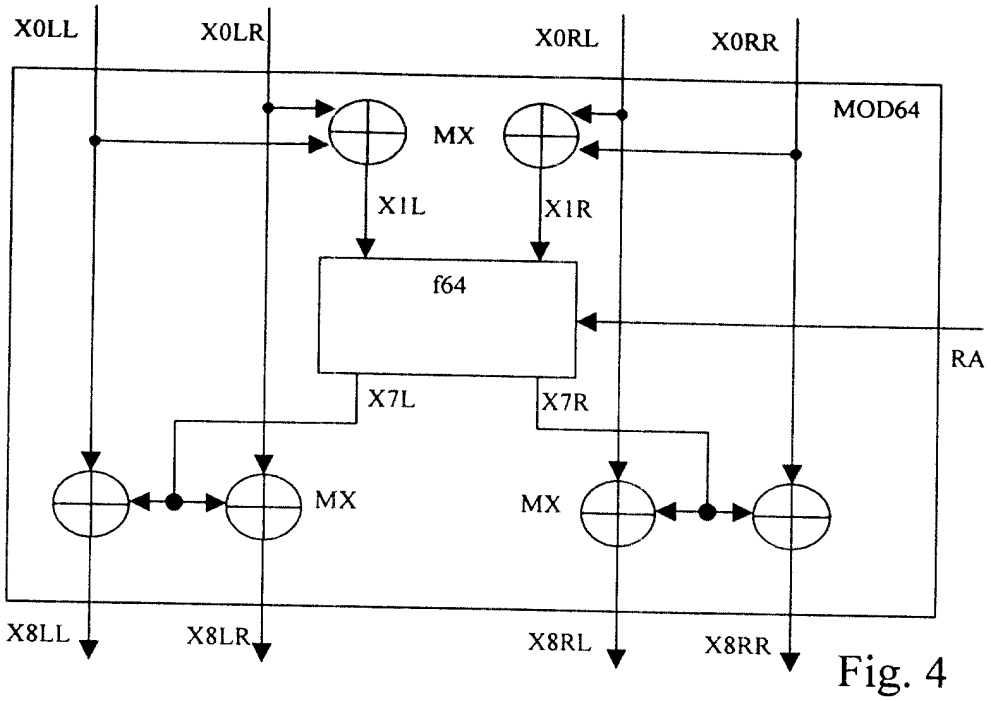


Fig. 4

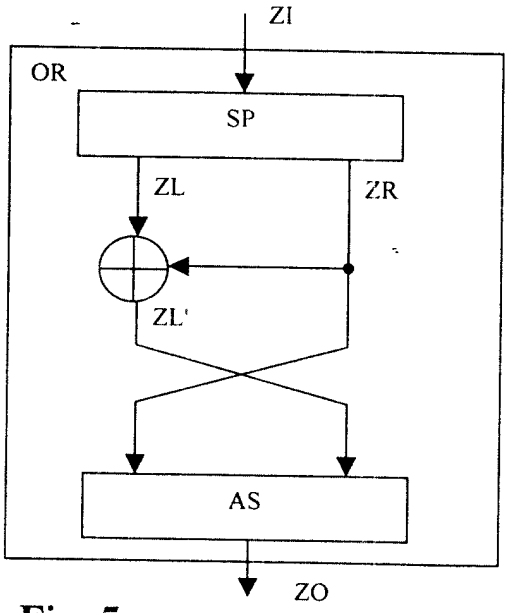


Fig. 5

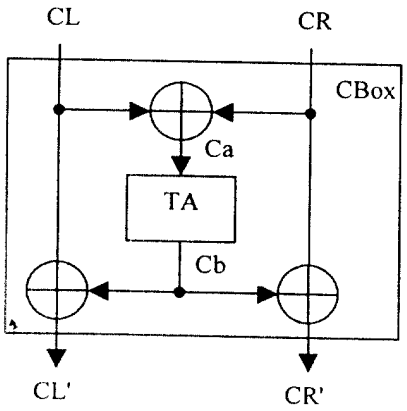
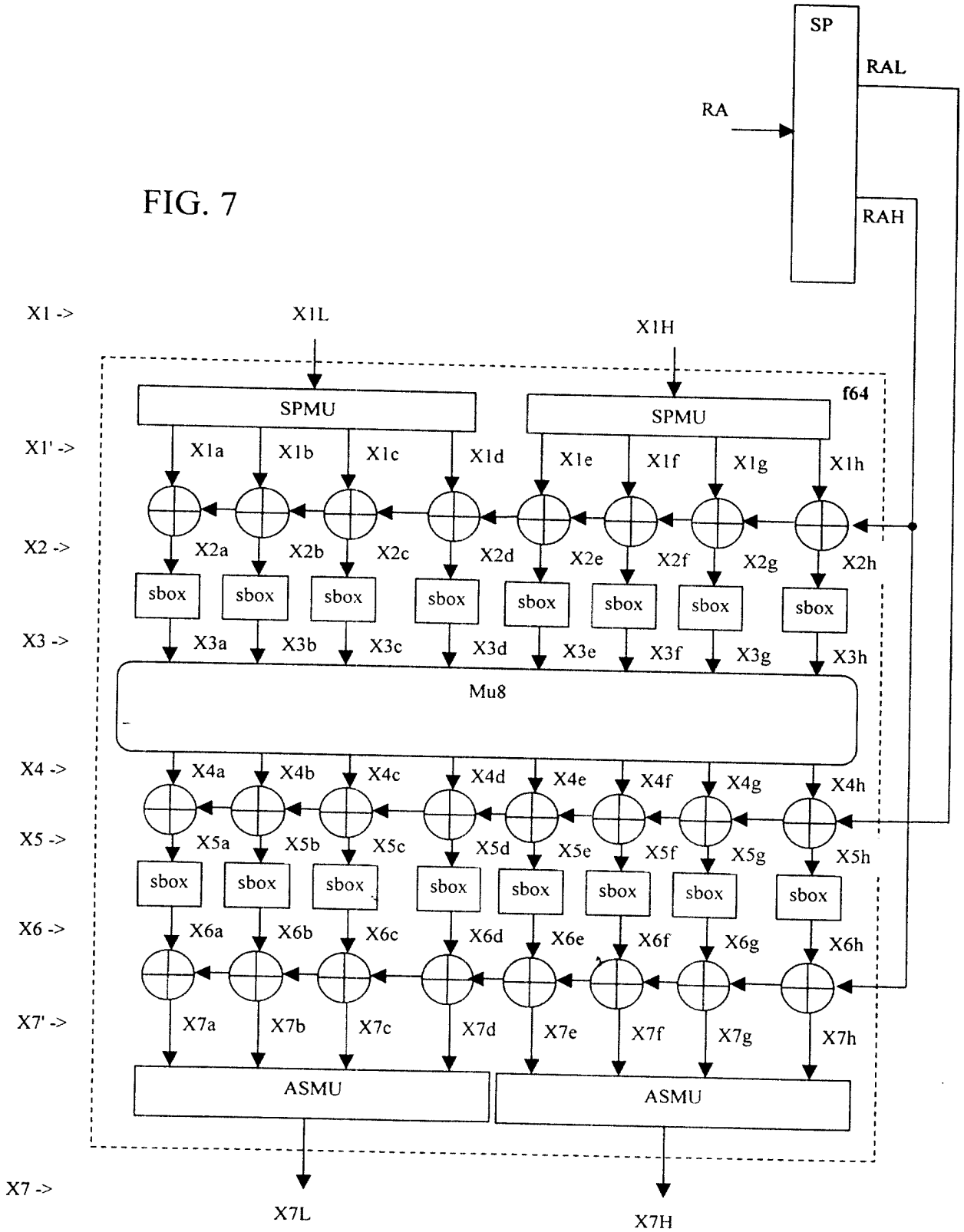


Fig. 6

FIG. 7


SP
RAL
RA
RAH
X1 ->
X1L
X1H
f64
SPMU
SPMU
X1' ->
X1a
X1b
X1c
X1d
X1e
X1f
X1g
X1h
X2 ->
X2a
X2b
X2c
X2d
X2e
X2f
X2g
X2h
sbox
X3 ->
X3a
X3b
X3c
X3d
X3e
X3f
X3g
X3h
Mu8
X4 ->
X4a
X4b
X4c
X4d
X4e
X4f
X4g
X4h
X5 ->
X5a
X5b
X5c
X5d
X5e
X5f
X5g
X5h
X6 ->
X6a
X6b
X6c
X6d
X6e
X6f
X6g
X6h
X7' ->
X7a
X7b
X7c
X7d
X7e
X7f
X7g
X7h
ASMU
ASMU
X7 ->
X7L
X7H

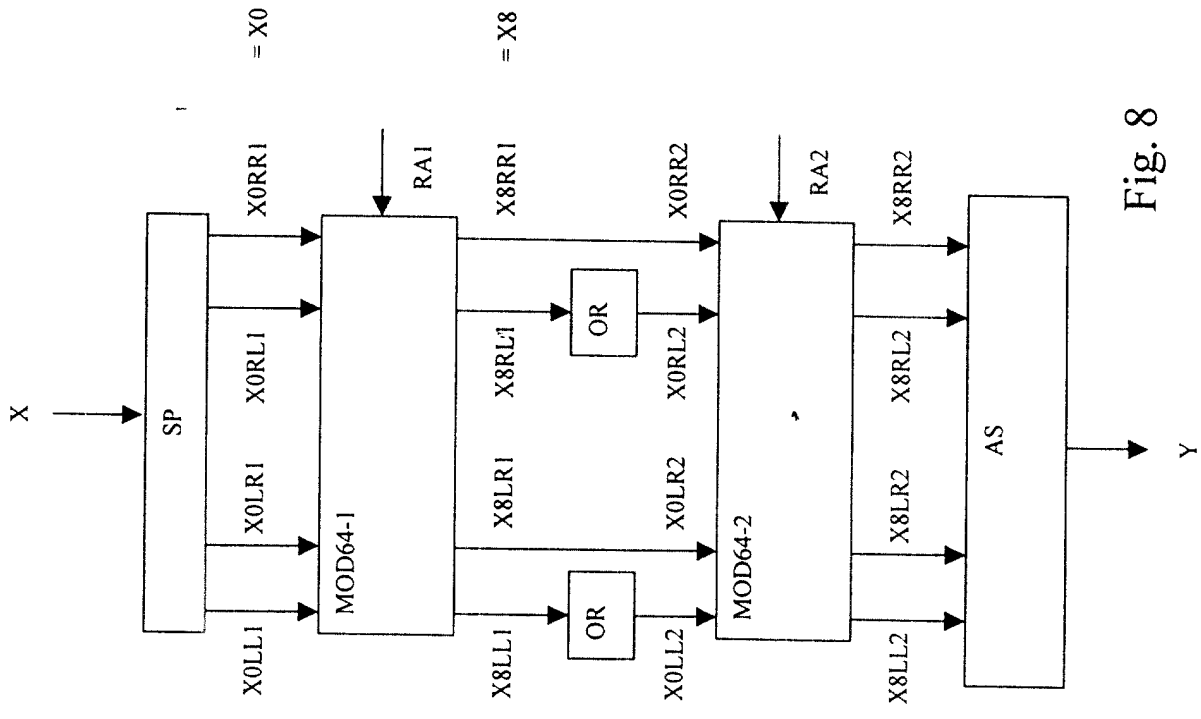

X
SP
X0RR1
X0RL1
X0LR1
X0LL1
MOD64-1
RA1
= X0
X8RR1
X8RL1
X8LR1
X8LL1
= X8
OR
OR
X0RR2
X0RL2
X0LR2
X0LL2
MOD64-2
RA2
X8RR2
X8RL2
X8LR2
X8LL2
AS
Y

Fig. 8

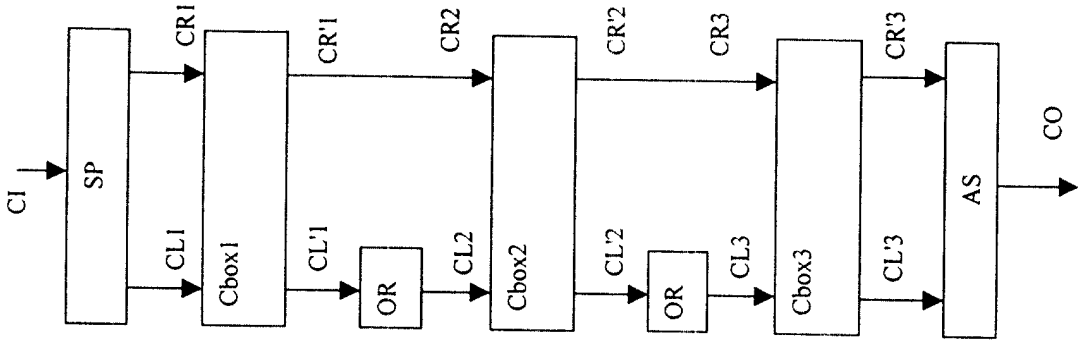

CI
SP
CL1
CR1
Cbox1
CL'1
CR'1
OR
CL2
CR2
Cbox2
CL'2
CR'2
OR
CL3
CR3
Cbox3
CL'3
CR'3
AS
CO

Fig. 9