

Circuit électronique de faible complexité protégé par masquage personnalisé

L'invention concerne un circuit électronique protégé par masquage et tirant parti d'une personnalisation propre au circuit de façon à réduire la portée des attaques par observation et réduire le coût d'implémentation de la protection. L'invention s'applique notamment au domaine de la protection des circuits qui manipulent des données confidentielles dont les circuits de cryptographie sont un exemple.

10

L'activité des circuits électroniques est observable pendant leur fonctionnement au travers des grandeurs physiques telles que la consommation de puissance, le temps de calcul ou le rayonnement électromagnétique.

15

Ces quantités physiques dépendent à la fois des architectures de calcul et des données manipulées à l'intérieur du circuit. Des informations sur les données traitées sont donc indirectement disponibles à l'extérieur du circuit par l'observation des dites quantités appelées canaux cachés ou canaux auxiliaires.

20

La dissipation de ces quantités physiques peut remettre en cause la sécurité de systèmes traitant des données secrètes protégées notamment par des méthodes de cryptographie. Ainsi, si des données secrètes sont protégées en utilisant un algorithme de cryptographie symétrique, la robustesse de la protection réside dans la capacité à garder la clé de chiffrement secrète. La dissipation des quantités physiques peut permettre à un tiers d'obtenir ladite clé en mettant en œuvre des attaques adaptées et, par conséquent, d'accéder aux données secrètes. Une attaque par observation de quantités physiques dissipées par ledit circuit est habituellement qualifiée simplement d'attaque par observation. Dans la suite de la description, un tiers utilisant des méthodes d'attaque par observation pour accéder à des données ne lui étant pas destinées est appelé attaquant, les quantités physiques dissipées, quant à elles, sont appelées fuites ou canaux cachés.

25

30

Il existe aujourd'hui des attaques par observation puissantes permettant d'accéder aux données traitées par des circuits protégés.

35

Ainsi des attaques par observation de fuites représentatives des temps de traitement des données par le circuit existent, comme celle présenté dans l'article de P. C. Kocher, J. Jaffe et B. Jun intitulé *Timing Attack on Implémentations of Diffie-Hellman, RSA, DSS and Other Systems*,
5 Proceedings of CRYPTO'96, volume 1109 LNCS, pages 104-113, Springer-Verlag, 1996.

Des attaques par observation de la consommation des circuits peuvent également être utilisées par un attaquant, en utilisant par exemple des méthodes de type DPA, ce type d'attaques étant décrit dans l'article de
10 P. C. Kocher, J. Jaffe et B. Jun intitulé *Differential Power Analysis*, Proceedings of CRYPTO'99, volume 1666 LNCS, pages 388-397, Springer-Verlag, 1999.

Ces méthodes permettent de contourner la sécurité conférée au niveau mathématique par la cryptographie.

15 S'il est relativement simple d'équilibrer un algorithme en temps de temps de traitement, il est plus difficile de protéger les circuits contre l'observation instantanée de la forme d'onde de la consommation électrique.

Il existe différentes méthodes de contre-mesure pour protéger un
20 circuit électronique contre les attaques sur les canaux cachés. Leurs caractéristiques sont notamment spécifiées par des critères communs définis au niveau international ou par des normes, comme par exemple la norme américaine FIPS 140, l'acronyme FIPS venant de l'expression anglo-saxonne « Federal Information Processing Standardization ».

25 Certaines contre-mesures ne font qu'accroître le nombre de mesures nécessaires pour qu'une attaque réussisse. C'est le cas, par exemple, des méthodes de contre-mesures utilisant un générateur de bruit non-fonctionnel mis en œuvre à côté de la logique de calcul. Par exemple, un générateur de nombre pseudo aléatoire PRNG, acronyme venant de
30 l'expression anglo-saxonne « Pseudo-Random Number Generator », initialisé aléatoirement, peut jouer ce rôle. Dans ce cas, toute mesure recueillie par un attaquant est perturbée par un bruit qui se superpose au canal caché. Les attaques deviennent plus complexes car il faut, en pratique, réaliser plus de mesures afin d'amplifier le rapport signal-sur-bruit attendu
35 pour que la technique de contre-mesure soit efficace.

D'autres techniques de contre-mesures protègent contre les attaques par observation par masquage des canaux cachés et font habituellement intervenir au cours du traitement à protéger une variable aléatoire ou pseudo-aléatoire m appelée masque. Ladite variable est utilisée
5 de manière à ce que le résultat du calcul ne dépende pas dudit masque, mais que les fuites d'information au travers des canaux cachés en dépendent.

Ainsi, les techniques de contre-mesure par masquage sont mises en œuvre en entrelaçant les données sensibles x transitant au sein du circuit
10 de cryptographie avec la variable de masque m , cet entrelacement servant à entraver l'exploitation du canal caché par un attaquant. Les données ou variables sensibles x correspondent à des variables étant à la fois prédictibles entièrement et partageants une information mutuelle non nulle avec le secret. Cette technique revient à modifier la représentation des
15 données sensibles x , vers la quantité $x \oplus m$ correspondant au chiffrement Vernam de x en appliquant la clé m à l'aide de l'opération $\oplus$ désignant une opération de type OU exclusif également désigné par le sigle XOR dans la suite de la description.

Le masque peut être conditionné par une signature propre à
20 chaque circuit auquel cas il est montré que la fuite de la clé est chiffrée par ledit masque. Cette spécificité évite des attaques dites par « catalogage » où des clones de circuits peuvent être exploités pour modéliser les fuites.

Les techniques de contre-mesures habituelles faisant appel à un masquage aléatoire résistent aux attaques directes sur la prédiction des
25 registres attaques du premier ordre, comme par exemple aux attaques de type DPA ou aux attaques de type CPA, acronyme venant de l'expression anglo-saxonne « Correlation Power Analysis ». Elles sont mises en œuvre, par exemple, en dupliquant les chemins de traitement de données dans le circuit.

30 Cette duplication implique une augmentation significative de la complexité du circuit par rapport à une implémentation non masquée.

Par ailleurs, ces contre-mesures résistent mal à des attaques d'ordre supérieur ou égal à deux. A titre d'exemple, les attaques de deuxième ordre exploitent le fait que la variance de la fuite dépend de la
35 variable sensible x . L'estimation de la variance est réalisée soit par

combinaison des fuites d'information aux deux dates où $x \oplus m$ et m d'autre part sont utilisés, soit par estimation de la distribution conjointe du couple $(x \oplus m, m)$ lorsque le masque et la donnée masquée sont utilisés simultanément. Les attaques de second ordre basées sur l'estimation de la variance sont appelées attaques « zero-offset » et sont décrites dans l'article de E. Peeters, F. Standaert, N. Donckers et J-J. Quisquater intitulé *Improved Higher Order Side-Channel Attacks with FPGA experiments*, Josyula R. Rao and Berk Sunar editors, Cryptographic Hardware and Embedded Systems - Proceedings of CHES, volume 3659 LNCS, pages 309–323. Springer-Verlag, 2005.

Un but de l'invention est notamment de pallier les inconvénients précités.

A cet effet l'invention a pour objet un circuit de cryptographie protégé par masquage, ledit circuit comportant des moyens pour chiffrer des mots binaires à l'aide d'au moins une clé k_r^c , des moyens pour appliquer des traitements linéaires et des traitements non linéaires auxdits mots, des moyens pour masquer lesdits mots. Les mots binaires sont démasqués en amont des traitements non linéaires en utilisant un masque k_r^i et masqués en aval desdits traitements en utilisant un masque k_{r+1}^i , les masques k_r^i et k_{r+1}^i faisant partie d'un ensemble de masques propres à chaque instance du circuit.

Selon un aspect de l'invention, les traitements non linéaires, le démasquage en amont des traitements non linéaires et le masquage en aval des traitements linéaires sont implémentés dans une mémoire de type ROM.

Les masques k_r^i sont, par exemple, des masques secondaires déduits de masques primaires k^i tels que $k_{r+1}^i = P(k_r^i)$ et $k_0^i = k^i$, la fonction $P(x)$ correspondant à une fonction de permutation des éléments de x , les mémoires ROM restant inchangées.

Dans un mode de réalisation, la fonction $P(x)$ est une permutation circulaire, un masque secondaire d'indice $r+1$ étant déduit d'un masque secondaire d'indice r en permutant circulairement le masque k_r^i d'un nombre de bits d choisi.

Dans un autre mode de réalisation, les masques principaux k^i sont de longueur W et composés d'un nombre entier de sous-masques de

longueur S, les masques secondaires k_r^i étant générés par permutation desdits sous-masques.

Les sous-masques des masques secondaires sont, par exemple, choisis en utilisant l'expression :

5

$$k_{r+1}^i[x] = k_r^i[\text{mod}(x - Q, W / S)]$$

dans laquelle :

r est le numéro de ronde ;

10

i est un nombre de 4 bits tirés aléatoirement ;

Q est un entier permettant de contrôler le taux de permutation entre deux masques secondaires k_r^i et k_{r+1}^i consécutifs ;

S est la longueur d'un sous-masque exprimée en bits ;

W est la longueur du masque principal exprimée en bits ;

15

mod() est une fonction définie telle que $\text{mod}(a,b) = a$ modulo b , a et b étant des nombres entiers ;

Selon un aspect de l'invention, le masque principal k^i de chiffrement est modifié régulièrement en choisissant aléatoirement un masque k^i parmi un ensemble de masques principaux mémorisés dans le circuit.

Selon un autre aspect de l'invention, l'ensemble des masques principaux mémorisés dans le circuit est différent d'un circuit à l'autre.

L'ensemble des masques principaux sont obtenus à l'aide d'un circuit de génération de masques intrinsèques au composant.

Dans un mode de mise en œuvre, la distance de Hamming entre deux masques k_r^i et k_{r+1}^i est sensiblement égale à $S/2$.

Le poids de Hamming d'un masque k^i est, par exemple, sensiblement égal à $W/2$.

Selon un aspect de l'invention, les traitements non linéaires sont mis en œuvre à l'aide de S-boxes.

Les traitements non linéaires sont appliqués, par exemple, après les traitements linéaires dans un même bloc combinatoire juste avant l'échantillonnage du résultat dans un registre.

Le circuit est, par exemple, implémenté dans un FPGA.

L'ensemble des masques principaux est obtenu, par exemple, à l'aide de la modification du fichier de configuration du circuit FPGA.

Le circuit comprend, par exemple, des moyens de reconfiguration dynamique permettant de mettre à jour l'ensemble des masques principaux et les tables implémentant les parties du circuit correspondant aux traitements non-linéaires.

Dans un mode de mise en œuvre, le circuit est implémenté dans un ASIC.

L'invention a notamment comme avantage de ne pas augmenter significativement la complexité du circuit de par la mise en œuvre de la protection par masquage, notamment concernant les parties du circuit réalisant des traitements non linéaires. L'invention a également comme avantage de permettre l'utilisation d'un jeu de masque prédéterminé de taille réduite, ledit jeu pouvant être différent d'un circuit à l'autre de manière à rendre la protection unique entre circuits issus de la même chaîne de production.

D'autres caractéristiques et avantages de l'invention apparaîtront à l'aide de la description qui suit donnée à titre illustratif et non limitatif, faite en regard des dessins annexés parmi lesquels :

- la figure 1 présente l'exemple d'une fonction de Feistel protégée par masquage ;
- la figure 2 donne un exemple de circuit de cryptographie protégé par masquage, les parties non linéaires étant démasquées ;
- la figure 3 illustre une implémentation de l'algorithme AES protégé par masquage selon l'invention.

La figure 1 présente l'exemple d'une fonction de Feistel protégée par masquage.

Le principe du masquage, connu de l'homme du métier, consiste à modifier la représentation des variables sensibles x en une représentation redondante. Cette représentation comprend au moins deux parts, une part

correspondant aux données sensibles masquées notée $x \oplus m$ et une part correspondant au masque m . La somme de ces deux parts dans le corps de Galois binaire où elles sont définies permet de retrouver la variable x en utilisant la propriété suivante :

5

$$x = (x \oplus m) \oplus m \quad (1)$$

A titre d'exemple, dans un algorithme de chiffrement par bloc tel que DES ou AES, le résultat de l'opération de cryptage d'un bloc de données
10 est issu de la répétition de plusieurs rondes. Une ronde est aussi appelée « round » en anglais et désigne un cycle de calcul dans lequel au moins deux types de transformations ont été exécutées, l'une linéaire et l'autre non-linéaire, appelée également transformation par substitution.

La transformation linéaire a pour objectif de mélanger les
15 symboles ou les groupes de symboles présentés à son entrée en suivant des règles prédéfinies et créer ainsi de la diffusion.

La transformation par substitution est réalisée habituellement à l'aide de tables de substitution dites S-boxes et contribue à casser la linéarité de la structure de chiffrement. En utilisant ce type de transformation, des
20 symboles ou des groupes de symboles sont remplacés par d'autres symboles ou groupes de symboles dans le but de créer de la confusion.

Ainsi, le couple de parts $(x \oplus m, m)$ est transformé en un couple $(\text{round}(x \oplus m'), m')$, la fonction $\text{round}()$ désignant l'opération fonctionnelle d'une ronde, tandis que m' est le nouveau masque de ronde.

25 Les parties linéaires de chaque ronde sont simplement dupliquées. La linéarité des fonctions $L()$ desdites parties implique que :

$$L(x \oplus m) = L(x) \oplus L(m) \quad (2)$$

30 La linéarité permet ainsi d'utiliser le masque $m_L = L(m)$ comme nouveau masque après transformation.

En revanche, la mise en œuvre du masquage sur les parties non-linéaires, c'est-à-dire les S-Boxes, induit un accroissement significatif en terme de coût d'implémentation. Un masque m_{NL} prenant en compte cette

transformation et permettant de retrouver $S(x)$ à partir de $S(x \oplus m)$ doit être déterminé tel que :

$$S(x) = S(x \oplus m) \oplus m_{NL} \quad (3)$$

5

Pour cela, m_{NL} peut s'exprimer à l'aide d'une fonction $S'()$ définie telle que :

$$m_{NL} = S'(x, x \oplus m) = S(x) \oplus S(x \oplus m) \quad (4)$$

10

La fonction $S'()$ possède donc deux fois plus d'entrées que la fonction $S()$. Ainsi, la mise en œuvre du masquage pour des fonctions $S()$ non linéaires revient à ajouter le carré de la complexité de S .

Afin d'illustrer la mise en œuvre du masquage dans un circuit de cryptographie, la figure 1 présente l'exemple d'une fonction de Feistel protégée par masquage. Ce type de fonction est notamment utilisé pour le chiffrement par bloc de type DES, acronyme venant de l'expression anglo-saxonne « Data Encryption Bloc ».

La mise en œuvre du masquage des données sensibles x requiert, comme explicité précédemment, un traitement en deux parties 100, 101.

La première partie 100 correspond aux traitements réalisés sur la part $x \oplus m$ et la seconde partie 101 correspond aux traitements réalisés sur la part m .

Le chiffrement des données sensibles est effectué en appliquant une clé k au bloc à chiffrer suivie d'une S-Box de fonction $S()$ et de l'application d'une fonction linéaire $L()$.

Les signaux numériques à traiter par les deux voies 100, 101 du circuit sont synchronisés en utilisant un registre 102, 103 pour chaque voie.

La première voie 100 traite la part comportant les données sensibles x masquées, c'est-à-dire $x \oplus m$. La clé k de chiffrement est appliquée en utilisant une porte XOR 106. Le signal résultant correspond à $x \oplus m \oplus k$. Une S-Box 107 permet ensuite d'obtenir le signal $S(x \oplus m \oplus k)$ 117 auquel est appliquée une fonction linéaire L 108.

La seconde voie 101 traite la part correspondant au masque m. Comme explicité précédemment, l'application d'une fonction non-linéaire S() 107 sur un signal masqué implique habituellement sa prise en compte au niveau du traitement de la seconde part. Ainsi, la fonction S'() définie par
5 l'expression (4) est mise en œuvre 113 en utilisant deux S-Boxes et 2 portes XOR 109, 112. La fonction prend en entrée d'une part le masque m 105 et d'autre part le signal $x \oplus m \oplus k$ 115 résultant de l'application de la clé de chiffrement k au niveau de la première voie 100. Sur le signal 116 résultant de l'application de S'() est appliqué une fonction linéaire L() 114 de manière
10 à prendre en compte la fonction linéaire 108 de la première voie 100. La fonction S'() peut être implémentée dans un mémoire de type ROM de manière à ce qu'elle soit protégée contre les attaques par observations. Il est en effet particulièrement difficile d'observer, par exemple, les variations de consommation électrique au sein d'une telle mémoire.

15 Même si une telle implémentation est protégée par masquage et que l'observation de l'activité liée à la fonction S'() est difficile à observer, des failles de sécurité la rende fragile, notamment contre des attaques du second ordre. Il est par exemple possible pour un attaquant de positionner deux sondes d'observation de la consommation électrique au niveau de deux
20 nœuds de circuit distincts, par exemple aux sorties 118, 119 des deux registres d'entrée de chaque voie. Une attaque basée sur l'estimation de la variance, c'est-à-dire de type « zero-offset », est dans ce cas très efficace.

La figure 2 donne un exemple de circuit de cryptographie protégé
25 par masquage sur laquelle s'appuie l'invention. Il faut noter qu'il n'existe plus de chemin de masque ce qui, avantageusement, rend impossible les attaques du second ordre comme celle décrite précédemment.

Comme explicité précédemment, lorsque les parties non masquées d'un circuit de chiffrement sont confinées dans une mémoire, il est
30 difficile d'attaquer les variables internes à ladite mémoire. En d'autres termes, une mémoire est considérée comme une boîte noire protégée contre les fuites d'informations. Seules les entrées ou les sorties sont vulnérables. Un des objectifs du circuit décrit est de tirer partie d'une l'implémentation en mémoire avec une méthode de masquage personnalisée de complexité
35 modérée.

Par ailleurs, lorsque l'utilisation de mémoires n'est pas adaptée, les éléments de calcul peuvent être ordonnancés de manière à positionner les parties non-linéaires le plus loin possible de la sortie des registres. A titre d'exemple, une attaque en corrélation est d'autant moins efficace qu'elle est
5 menée profondément dans la logique combinatoire du circuit.

L'exemple de la figure 2 présente un exemple de mise en œuvre de l'invention dans un circuit basé sur l'utilisation d'un réseau SPN, acronyme venant de l'expression anglo-saxonne « Substitution Permutation Network ». Ce type de circuit est aussi appelé réseau S-P de Shannon. Dans
10 cet exemple un cryptage en deux rondes est considéré. Des mots binaires d'indice r notés k_r^i et k_r^c sont utilisés respectivement en tant que clé de masquage et clé de chiffrement.

Le circuit présenté en exemple peut être décomposé en plusieurs étages, soit un étage d'entrée, un étage correspondant à la première ronde, un étage correspondant à la seconde ronde et un étage de sortie. Les
15 données à chiffrer sont présentées en entrée de l'étage d'entrée, par exemple sous la forme de mots de 32 bits découpés en quatre sous-mots de 8 bits. Une masquage d'entrée est appliqué en utilisant la clé k_0^i d'une longueur de 32 bits, ladite clé étant découpée en quatre sous-masques de 8 bits $k_0^i[0]$, $k_0^i[1]$, $k_0^i[2]$, $k_0^i[3]$, lesdits sous-masques étant appliqués au quatre
20 sous-mots de 8bits en utilisant quatre portes XOR 200, 201, 202, 203.

En entrée de la première ronde, un registre 204 prend en entrée les quatre sous-mots de 8 bits résultant du masquage par la clé k_0^i . Ce registre permet de synchroniser les différents flux binaires, un flux
25 correspondant à un sous-mot de 8 bits d'entrée. Une première clé de chiffrement k_0^c , découpée en quatre sous-clés de 8 bits notées $k_0^c[0]$, $k_0^c[1]$, $k_0^c[2]$, $k_0^c[3]$, est appliquée au niveau de quatre portes XOR 206, 206, 207, 208 sur les sous-mots masqués présentés en sortie dudit registre 204. La clé k_0^c est associée à la première ronde de chiffrement. Les quatre sous-mots de
30 données masqués par k_0^i et chiffrés par k_0^c sont ensuite respectivement traités par quatre blocs de traitements 209, 210, 211, 212 implémentés dans une mémoire, par exemple de type ROM. Il est à noter que la complexité d'implémentation de fonctions, notamment non linéaires, dans une mémoire ROM augmente de manière exponentielle avec le nombre d'entrées. Les
35 algorithmes cryptographiques intègrent cette contrainte et traitent les mots à

chiffrer en sous-mots de plus petite taille au niveau de la fonction non-linéaire afin de minimiser la complexité d'implémentation.

En entrée de chaque bloc de traitement, les sous-mots de 8 bits sont démasqués en appliquant les quatre sous-masques de 8 bits $k_0^i[0]$, $k_0^i[1]$, $k_0^i[2]$, $k_0^i[3]$ à l'aide de portes XOR 214. Une fonction non linéaire est ensuite appliquée, une S-box 213 pouvant être utilisée pour la mettre en œuvre.

En amont de la sortie de chaque bloc de traitement, une porte XOR 215 est utilisée afin de masquer les données en sortie, de manière à ce que les données sensibles soient masquées en sortie de l'étage de première ronde par un masque k_1^i découpé en quatre sous-masques $k_1^i[0]$, $k_1^i[1]$, $k_1^i[2]$, $k_1^i[3]$. Une transformation linéaire $L_0()$ est appliquée en sortie de ronde, celle-ci devant être prise en compte pour le masquage au sein des blocs de traitement. Pour cela, le masquage est réalisé en utilisant un masque modifié $L_0^{-1}(k_1^i)$ du masque k_1^i . Celle-ci est découpée en quatre sous-masques modifiés de 8 bits $L_0^{-1}(k_1^i[0])$, $L_0^{-1}(k_1^i[1])$, $L_0^{-1}(k_1^i[2])$, $L_0^{-1}(k_1^i[3])$, la transformation $L_0^{-1}()$ correspondant à l'inverse de la transformation linéaire $L_0()$ 216.

En sortie des blocs de traitement 209, 210, 211, 212, la transformation $L_0()$ est appliquée sur les mots binaires présentés en sortie desdits blocs. Le résultat de la première ronde de chiffrement correspond aux sous-mots binaires en sortie de la transformation linéaire 216, lesdits sous-mots correspondant à des données sensibles masquées par le masque k_1^i et donc protégés contre des attaques par observations.

L'entrée de la deuxième ronde est la sortie de la première ronde et correspond aux quatre sous-mots de 8 bits résultant de la transformation linéaire 216 de la première ronde. Ces quatre sous-mots sont présentés en entrée d'un registre 217 permettant de synchroniser les différents flux binaires.

Une seconde clé de chiffrement k_1^c propre à la deuxième ronde, découpée en quatre sous-clés de 8 bits notées $k_1^c[0]$, $k_1^c[1]$, $k_1^c[2]$, $k_1^c[3]$, est appliquée au niveau de quatre portes XOR 218, 219, 220, 221 sur les sous-mots masqués présentés en sortie du registre 217.

Les quatre sous-mots de données masqués par k_1^i et chiffrés par k_1^c sont ensuite respectivement traités par quatre blocs de traitements 222,

223, 224, 225 implémentés dans une mémoire, par exemple, de type ROM. En entrée de chaque bloc, les données sont démasquées en appliquant la clé k_1^i à l'aide de portes XOR.

Une fonction non linéaire est ensuite appliquée, une S-box
 5 pouvant être utilisée dans chaque bloc pour la mettre en œuvre. En sortie de chaque bloc de traitement, un porte XOR est utilisée pour masquer les données en sortie, de manière à ce que les données sensibles soient masquées en sortie de l'étage de seconde ronde par le masque k_2^i . Pour cela, le masquage est réalisé en utilisant un masque modifié $L_1^{-1}(k_2^i)$, la
 10 transformation $L_1^{-1}()$ correspondant à l'inverse d'une transformation linéaire $L_1()$ 226.

La transformation $L_1()$ est appliquée sur les sous-mots binaires présentés en sortie des blocs de traitement 209, 210, 211, 212. Le résultat de la seconde ronde de chiffrement correspond aux sous-mots binaires en
 15 sortie de la transformation linéaire 226, lesdits sous-mots correspondant à des données sensibles masquées et donc protégés contre des attaques par observation.

Un avantage de ce type d'implémentation est qu'il est possible, d'invertir le chiffrement par k_0^c et le masquage par k_0^i pour la ronde 1 ainsi
 20 que le chiffrement par k_1^c et le masquage par k_1^i pour la ronde 2.

De façon à réduire la complexité d'implémentation et à utiliser toujours les mêmes mémoires ROM, il est proposé de déduire les masques d'implémentation utilisés d'une étape à l'autre par une permutation d'un masque principal. Par exemple, si k^i est le masque principal, les masques
 25 secondaires k_0^i , k_1^i et k_2^i peuvent être déduits du masque principal de la manière suivante :

$$\begin{aligned} k_0^i &= k^i \\ k_1^i &= P(k_0^i) \\ 30 \quad k_2^i &= P(k_1^i) \end{aligned}$$

la fonction $P(\text{bin})$ représentant une fonction de permutation, par exemple une permutation circulaire du mot binaire bin. De plus, le masque principal k^i peut être tiré au hasard parmi un ensemble prédéfini de masques principaux. Un

exemple de permutation de masques sur la base d'un masque principal est donné dans la description à l'aide de la figure 3.

La faible taille de l'ensemble des masques principaux permet avantageusement d'utiliser des masques personnalisés propres au composant, c'est-à-dire propre à chaque instance du circuit. L'application de cette signature permet de réduire la portée des attaques de type « catalogage » car les fuites deviennent ainsi propres à un circuit et non plus à un type de circuit. Les attaques d'ordre élevé HO-DPA, acronyme venant de l'expression anglo-saxonne « higher-order differential power analysis », comme par exemple celles de type « zero-offset » sont quant à elles remises en cause du fait qu'il n'y a plus de chemin de masque spécifique. Il n'est donc plus possible de considérer le couple (variable masquée, masque).

La figure 3 illustre une implémentation de l'algorithme AES protégé par masquage.

L'algorithme de chiffrement par blocs AES, acronyme venant de l'expression anglo-saxonne « Advanced Encryption Standard », est particulièrement performant pour garder secret des messages binaires. Le message à protéger est traité par mots binaires de taille fixe, lesdits mots pouvant être de 128, 192 ou 256 bits. Les clés de chiffrement sont de longueur W, W étant égal à la longueur des mots à traiter. L'algorithme comprend trois phases de traitement, chaque phase étant composée d'une ou plusieurs rondes. La première phase R1 correspond à une ronde d'initialisation, la deuxième phase R2 correspond à N rondes utilisant la même structure de manière itérative et la troisième phase R3 correspond à une ronde finale. Le principe de ces trois phases propre à l'algorithme AES est connu de l'homme du métier.

Le circuit est par exemple un circuit FPGA ou ASIC.

Une protection par masquage selon l'invention peut être introduite de manière à protéger contre les attaques par observation des canaux cachés le circuit de cryptographie AES mettant en œuvre les trois phases R1, R2, R3.

Un générateur de nombre aléatoire 300 est utilisé afin de générer des mots binaires i , par exemple de n bits, n représentant l'entropie du

masquage. Dans l'exemple décrit dans la suite de la description, n est représenté sur 4 bits.

Un compteur CTR 301 incrémente une variable r correspondant au numéro de ronde en cours.

5 Le circuit protégé comprend une zone de mémoire 303 permettant notamment de stocker un ensemble de masques constants de longueur égale à celle des mots à chiffrer, soit 128 bits dans cet exemple.

Le nombre i généré aléatoirement 300 permet de sélectionner un masque principal k^i parmi l'ensemble des masques mémorisés 303. Par
10 conséquent, pour $n = 4$, 16 masques principaux k^i différents peuvent être sélectionnés aléatoirement.

Par ailleurs, les masques principaux mémorisés dans le composant peuvent être différents d'un composant produit à l'autre, de manière à obtenir une protection différenciée et éviter les « attaques par
15 catalogage ».

Un masque principal k^i de longueur W est composé d'un nombre entier de sous-masques de longueur S , W étant un multiple de S . Par exemple, un masque k^i de longueur $W = 128$ bits comprend par exemple 16 sous-masques de $S = 8$ bits, lesdits sous-masques étant notés $k_0^i[0]$, $k_0^i[1]$,
20 ..., $k_0^i[15]$.

A partir d'un masque principal, des masques secondaires peuvent être générés, par exemple en permutant les sous-masques composant le masque principal. Ainsi, à partir d'un unique masque principal, un masque secondaire différent peut être utilisé pour chaque ronde.

25 De façon à augmenter la robustesse face aux attaques, il existe un jeu de masques principaux k^i différent d'un composant à l'autre de manière à mettre en œuvre une protection masquage différenciée entre lesdits composants. La variable i est aléatoire et peut être générée avant chaque chiffrement.

30 Une fois le masque principal k^i choisi, un masque secondaire d'indice $r+1$ est déduit d'un masque secondaire d'indice r en permutant avec une permutation P circulairement le masque k_r^i d'un nombre de bits d choisi, le masque d'indice 0 étant initialisé tel que $k_0^i = k^i$.

d peut être choisi tel que $d = S$ bits par exemple, c'est-à-dire d'une
35 longueur correspondant à un sous masque.

Il est également possible de faire permuter le masque d'indice r d'un nombre entier de sous-masques. Ainsi, le masque k_{r+1}^i peut être généré en utilisant l'expression :

$$5 \quad k_{r+1}^i[x] = k_r^i[\text{mod}(x - Q, W / S)] \quad (5)$$

dans laquelle :

r est le numéro de ronde ;

i est un nombre de 4 bits tirés aléatoirement par le générateur 300 ;

10 Q est un entier permettant de contrôler le taux de permutation entre deux masques secondaires k_r^i et k_{r+1}^i consécutifs ;

S est la longueur d'un sous-masque exprimée en bits ;

W est la longueur du masque k^i exprimée en bits ;

15 $\text{mod}()$ est une fonction définie telle que $\text{mod}(a,b) = a$ modulo b , a et b étant des nombres entiers ;

Le masque principal peut être, par exemple, modifié au cours d'une procédure de chiffrement en tirant au hasard une nouvelle valeur de i .

20 Avantageusement, la résistance aux attaques par observation peut être optimisée en choisissant les masques principaux k^i tels que les masques secondaires soient indépendants les uns par rapport aux autres, par exemple en garantissant que la distance de Hamming entre k_r^i et k_{r+1}^i soit sensiblement égale à $S/2$.

25 Un équilibrage en moyenne des masques permet aussi de renforcer la protection, ledit équilibrage étant obtenu en garantissant que le poids de Hamming d'un masque secondaire et donc du masque principal soit sensiblement égal à $W/2$.

30 Dans l'exemple de la figure, les mots à chiffrer sont des mots de 128 bits et sont présentés en entrée du codeur dans une base de registre 302. Le mot à traiter est ensuite masqué par application de la clé $k_0^i = k^i$ non permutée à l'aide d'une porte XOR 304. Le résultat du masquage est ensuite chiffré par application d'une clé de chiffrement de longueur W notée k_0^c à l'aide d'une seconde porte XOR 305.

35 Le mot masqué par k_0^i et chiffré par k_0^c est stocké dans un registre 306, ledit registre correspondant à l'entrée de la partie du circuit réalisant la

deuxième phase de traitement R2, ladite phase correspondant à une boucle de chiffrement itératif, une itération correspondant à une ronde de traitement. Le mot mémorisé dans le registre 306 est traité par un module de contrôle 307 découpant le mot de 128 bits en 16 sous-mots de 8 bits. Le module de

5 contrôle a également pour rôle de sélectionner le masque k_r^i à utiliser pour démasquer les données en début de ronde, une ronde étant appliquée à chaque itération d'indice r . Les 16 sous-mots de 8 bits sont traités à l'aide de modules de fonction non-linéaire 308, lesdits modules étant implémentés dans une mémoire de type ROM, par exemple. Ces modules démasquent

10 309 les sous-mots présentés à leur entrée, leur applique un traitement non-linéaire 310 par exemple en utilisant des S-boxes, et masquent 311 le résultat dudit traitement. Il y a un module 308 de fonction non linéaire par sous-mot de 8 bits à traiter. Par conséquent, il y a 16 modules de fonctions non linéaires pour l'exemple de la figure 3.

15 Ces modules utilisent pour la ronde d'indice r les sous-masques $k_r^i[0], k_r^i[1], \dots, k_r^i[15]$ pour le démasquage d'entrée 309 et les sous-masques $k_{r+1}^i[0], k_{r+1}^i[1], \dots, k_{r+1}^i[15]$ pour le masquage de sortie 311. Par exemple, les 16 S-boxes peuvent être pré-calculées afin d'être masquées par les sous-masques $k_r^i[]$ puis démasquées par les sous-masques $k_{r+1}^i[]$.

20 Les 16 sous-mots en sortie des modules de traitements non linéaires sont ensuite dirigés vers un second module de contrôle 312 dont la fonction est notamment de concaténer lesdits mots en un mot de 128 bits.

Le mot de 128 bits est ensuite traité par deux modules de traitement linéaire, un premier réalisant un mélange des lignes 313 désigné

25 habituellement par l'expression anglo-saxonne « Shift Rows » et un second traitement réalisant un mélange des colonnes 314, traitement habituellement désigné par l'expression anglo-saxonne « Shift Columns ». Ces deux traitements linéaires peuvent être modélisés par une fonction $L_r()$.

Un chiffrement utilisant une clé k_r^c est appliqué sur le mot de 128

30 bits résultant desdits traitements linéaires, et ce à l'aide d'une fonction XOR 315.

Sur le mot résultant du chiffrement par k_r^c et de façon à pouvoir utiliser les mêmes S-boxes d'une ronde à l'autre, un masque k_{int}^i de 128 bits est appliqué 316. Les masques k_{int}^i sont mémorisés 303 après avoir été pré-

35 calculés en utilisant l'expression :

$$k_{\text{int}}^i = k_{r+1}^i \oplus L_r^{-1}(k_r^i) \quad (6)$$

Il existe ainsi en mémoire 16 mots pré-calculés k_{int}^i de 128 bits.

5 Les traitements de la phase R2 sont exécutés N fois de manière itérative. Lorsque les N rondes de la deuxième phase R2 ont été exécutées, la phase finale R3 est exécutée sur le mot de 128 bits extrait entre le traitement linéaire de mélange des lignes 313 et le traitement linéaire de mélange de colonnes 314.

10 Après un chiffrement final utilisant une clé k_{fin}^c appliquée par une fonction XOR 317 sur le mot extrait, un démasquage final est réalisé en appliquant un masque k_{fin}^i de 128 bits à l'aide d'une fonction XOR 318.

Les 16 masques k_{fin}^i de 128 bits sont mémorisés 303 dans le circuit protégé après avoir été pré-calculés en utilisant l'expression :

$$15 \quad k_{\text{fin}}^i = k_R^i \oplus L_r^{-1}(k_r^i) \quad (7)$$

dans laquelle la fonction $L_r^{-1}()$ représente l'inverse du traitement de mélange des lignes 313.

20 Le mot obtenu à la fin de la ronde finale, c'est-à-dire à la fin de la phase R3 de traitement correspond au résultat final du chiffrement AES. Le message chiffré obtenu est écrit dans un registre de sortie 319.

La figure 4 donne un exemple de circuit de cryptographie protégé
25 par masquage dont les traitements non-linéaires sont positionnés en fin de ronde.

Lorsque les parties du circuit correspondant aux traitements non-linéaires sont implémentés en portes logiques et non en mémoire, les fonctions de démasquage en amont et de masquage en aval desdits
30 traitements peuvent faire l'objet d'attaques.

Afin de protéger le circuit contre ces attaques, une solution est de placer les traitements non-linéaires d'une ronde en bout du cône de logique, c'est à dire juste avant l'échantillonnage par registre du résultat. Les calculs sont alors étalés du fait de leur dispersion temporelle et l'efficacité de
35 l'attaque est alors limitée.

L'exemple de la figure 4 donne un exemple de circuit de cryptographie protégé par masquage mettant en œuvre ce principe.

Cet exemple de circuit ressemble à celui présenté à l'aide de la figure 2. Le chiffrement est réalisé à l'aide d'une architecture de Feistel et est
5 réalisé grâce à la mise en œuvre d'un étage d'entrée, de deux étages de ronde appelés ronde 1 et ronde 2, puis d'un étage de sortie.

A la différence du circuit donné en exemple avec la figure 2, les traitements non-linéaires, mis en œuvre par exemple avec des S-boxes, sont placés en bout de cône de logique. Les traitements linéaires sont donc
10 placés en début de ronde. Ainsi, au niveau de l'étage correspondant à la première ronde, un traitement linéaire correspondant à une fonction de diffusion initiale $L_{-1}()$ est appliqué 401. Au niveau de l'étage correspondant à la deuxième ronde, un traitement linéaire correspondant à une fonction de diffusion $L_0()$ est appliqué 402.

15 L'inverse $L_{-1}^{-1}()$ de la fonction de diffusion initiale est appliqué 400 en sortie de l'étage d'entrée et une fonction de diffusion finale $L_1()$ est appliquée 403 à l'entrée de l'étage de sortie.

La consommation électrique en début de ronde est modélisable difficilement, alors qu'elle est rendue plus dépendante des conditions
20 environnementales en fin de ronde.

REVENDICATIONS

- 5 1- Circuit de cryptographie protégé par masquage, ledit circuit comportant des moyens pour chiffrer des mots binaires à l'aide d'au moins une clé k_r^c , des moyens pour appliquer des traitements linéaires (216) et des traitements non linéaires (226) auxdits mots, des moyens pour masquer lesdits mots, ledit circuit étant caractérisé en ce que les mots binaires sont démasqués (214) en amont des traitements non
- 10 linéaires en utilisant un masque k_r^i et masqués (215) en aval des traitements non linéaires en utilisant un masque k_{r+1}^i , les masques k_r^i et k_{r+1}^i faisant partie d'un ensemble de masques propres à chaque instance du circuit.
- 15 2- Circuit selon la revendication 1 caractérisé en ce que les traitements non linéaires, le démasquage (214) en amont des traitements non linéaires et le masquage (215) en aval des traitements linéaires sont implémentés dans des mémoires de type ROM (209).
- 20 3- Circuit selon l'une quelconque des revendications précédentes caractérisé en ce que les masques k_r^i sont des masques secondaires déduits de masques primaires k^i tels que $k_{r+1}^i = P(k_r^i)$ et $k_0^i = k^i$, la fonction $P(x)$ correspondant à une fonction de permutation des éléments de x , les mémoires ROM restant inchangées.
- 25 4- Circuit selon la revendication 3 caractérisé en ce que la fonction $P(x)$ est une permutation circulaire, un masque secondaire d'indice $r+1$ étant déduit d'un masque secondaire d'indice r en permutant circulairement le masque k_r^i d'un nombre de bits d choisi.
- 30 5- Circuit selon la revendication 3 caractérisé en ce que les masques principaux k^i sont de longueur W et composés d'un nombre entier de sous-masques de longueur S , les masques secondaires k_r^i étant générés par permutation desdits sous-masques.
- 35

- 6- Circuit selon la revendication 5 caractérisé en ce que les sous-masques des masques secondaires sont choisis en utilisant l'expression :

$$5 \quad k_{r+1}^i[x] = k_r^i[\text{mod}(x - Q, W / S)]$$

dans laquelle :

r est le numéro de ronde ;

i est un nombre de 4 bits tirés aléatoirement ;

10 Q est un entier permettant de contrôler le taux de permutation entre deux masques secondaires k_r^i et k_{r+1}^i consécutifs ;

S est la longueur d'un sous-masque exprimée en bits ;

W est la longueur du masque principal exprimée en bits ;

15 mod() est une fonction définie telle que $\text{mod}(a,b) = a$ modulo b , a et b étant des nombres entiers ;

- 7- Circuit selon l'un quelconque des revendications 3 à 6 caractérisé en ce que le masque principal k^i de chiffrement est modifié régulièrement en choisissant aléatoirement un masque k^i parmi un ensemble de masques principaux mémorisés dans le circuit.
- 20

- 8- Circuit selon la revendication 7 caractérisé en ce que l'ensemble des masques principaux mémorisés dans le circuit (303) est différent d'un circuit à l'autre.
- 25

- 9- Circuit selon la revendication 8 caractérisé en ce que l'ensemble des masques principaux est obtenu à l'aide d'un circuit interne de génération de masques.

- 30 10- Circuit selon l'un quelconque des revendications précédentes caractérisé en ce que la distance de Hamming entre deux masques k_r^i et k_{r+1}^i est égale à $S/2$.

- 11- Circuit selon l'une quelconque des revendications précédentes caractérisé en ce que le poids de Hamming d'un masque k^i est égal à $W/2$.
- 5 12- Circuit selon l'une quelconque des revendications précédentes caractérisé en ce que les traitements non linéaires sont mis en œuvre à l'aide de S-boxes (213, 310).
- 10 13- Circuit selon l'une quelconque des revendications précédentes caractérisé en ce que les traitements non linéaires sont appliqués après les traitements linéaires dans un même bloc combinatoire juste avant l'échantillonnage du résultat dans une registre.
- 15 14- Circuit selon l'une quelconque des revendications précédentes caractérisé en ce qu'il est implémenté dans un FPGA.
- 20 15- Circuit selon la revendication 13 caractérisé en ce que l'ensemble des masques principaux est obtenu à l'aide de la modification du fichier de configuration du circuit FPGA.
- 25 16- Circuit selon l'une quelconque des revendication 13 ou 14 caractérisé en ce qu'il comprend des moyens de reconfiguration dynamique permettant de mettre à jour l'ensemble des masques principaux et les tables implémentant les parties du circuit correspondant aux traitements non-linéaires
- 17- Circuit selon l'une quelconque des revendications 1 à 13 caractérisé en ce qu'il est implémenté dans un ASIC.