Cryptanalyse physique à textes inconnus d'algorithmes de chiffrement symétriques

Hélène Le Bouder[1]

La cryptanalyse est une discipline qui tente de briser les solutions cryptographiques. Elle est généralement basée sur une analyse logique et théorique. Dans cet article, ce sont les failles du circuit qui exécute la cryptographie, qui sont utilisées pour attaquer. On parle de cryptanalyse physique. Cet article résume l'approche utilisée dans les publications [11, 17]. Plus précisément, il présente des attaques physiques ciblant des algorithmes de chiffrement symétrique. Ces attaques utilisent la propagation de croyance basée uniquement sur l'observation de mesures, l'attaquant n'a accès ni aux données d'entrée ni à celles de sortie.

Un algorithme cryptographique est conçu pour être mathématiquement robuste. Malheureusement, une fois implémenté dans un circuit, un attaquant peut exploiter les failles de ce dernier. Les attaques physiques sont l'ensemble des techniques exploitant le fonctionnement d'un circuit. Ces attaques nécessitent un accès physique au circuit et sont divisées en deux grandes familles :

– les attaques par observation ;
– les attaques par injection de faute.

1. IMT-Atlantique, OCIF, IRISA, Rennes, France.

DOI:10.48556/SIF.1024.24.159

Cet article traite de la première famille : les attaques par observation. Celles-ci se basent sur l'observation de paramètres physiques pendant l'exécution d'un calcul.

Cet article traite plus spécifiquement d'attaques par observation dites à textes inconnus. C'est-à-dire que l'attaquant n'a accès ni aux données d'entrée ni de sortie. Les cibles choisies sont des algorithmes de chiffrement symétrique.

Dans une première partie, la section « Contexte cryptographique symétrique » aborde le contexte de la cryptographie symétrique. Ensuite, le contexte des attaques par observation est présenté dans la section « Attaques par observation ». Le déroulement des attaques par observation dites à textes inconnus, contre des algorithmes de chiffrement symétrique est décrit dans la section « Attaques sur algorithme de chiffrement symétrique ».

Contexte cryptographique symétrique

Définitions et propriétés

La cryptologie, étymologiquement la science du secret, regroupe la cryptographie et la cryptanalyse.

La cryptographie est la discipline qui assure les propriétés de confidentialité, d'intégrité, d'authentification et de non-répudiation. La cryptanalyse est la discipline qui tente de briser les solutions cryptographiques. La cryptanalyse physique est la discipline qui utilise des attaques physiques pour attaquer des algorithmes cryptographiques.

La cryptographie symétrique est un sous-domaine de la cryptographie. Elle regroupe des algorithmes qui assurent la confidentialité ou l'intégrité, en s'appuyant sur les principes de diffusion et de confusion définis par Shannon [18].

La confusion consiste à rendre la relation entre les données d'entrée et les données de sortie la plus complexe possible. La diffusion repose sur l'idée qu'un biais dans les données d'entrée ne doit pas se retrouver dans les données de sortie. Autrement dit, la connaissance de la sortie doit fournir le moins d'information possible sur les données d'entrée.

Chiffrement symétrique

Le chiffrement est l'un des sujets historiques de la cryptographie. Il s'agit de l'ensemble des méthodes permettant de rendre un message, appelé texte clair noté M, illisible à toute personne autre que le destinataire. Il ne s'agit pas de dissimuler le message, ce qui relève de la stéganographie, mais de le chiffrer à l'aide d'une clé notée K, afin de le transformer en quelque chose d'incompréhensible pour quiconque ne possède pas la clé, appelé texte chiffré, noté C. La méthode utilisée est appelée algorithme de chiffrement.

$$C = En(M, K) \tag{1}$$

L'opération inverse est appelée algorithme de déchiffrement.

$$M = Dn(C, K). \tag{2}$$

En cryptographie symétrique, la même clé est utilisée pour chiffrer et déchiffrer.

Une méthode classique pour réaliser un chiffrement symétrique est le chiffrement par blocs. Il utilise une primitive bloc prenant en entrée un bloc de taille fixe et une clé. Cette primitive retourne une sortie de même taille que le bloc d'entrée. Ensuite, une autre fonction, appelée mode de chiffrement, combine les différents blocs. La plupart des modes nécessitent de plus un vecteur d'initialisation, pour chaque opération de chiffrement.

Code d'authentification de message

Un MAC[2] est un outil de cryptographie symétrique qui assure l'intégrité. Une fonction MAC prend en entrée une clé secrète K et un message de longueur quelconque M. Elle retourne un tag de longueur fixe T.
Pour envoyer un message M protégé en intégrité, on calcule le tag :

$$T = \mathrm{MAC}(M, K) \tag{3}$$

Pour vérifier l'intégrité, on calcule $\mathrm{MAC}(M, K)$ et on teste si cela est égal à T :

– si oui, on considère que l'intégrité est respectée ;
– sinon, on considère que l'intégrité n'est pas respectée.

Un MAC peut se calculer à l'aide d'une primitive bloc et d'un mode MAC.

Chiffrement intègre

Bien souvent, lors d'un échange, la confidentialité et l'intégrité sont nécessaires. Pour garantir ces deux propriétés, plusieurs solutions symétriques sont proposées dans la littérature :

– une première solution est l'utilisation de deux outils. On combine un chiffrement et un MAC ;
– une autre solution est d'utiliser un chiffrement par blocs avec un mode garantissant les deux propriétés, tel que GCM[3] ;
– enfin, une dernière solution, proposée récemment, est d'adopter un nouvel outil : un chiffrement intègre dit tout en un.

2. *Message authentification code*.
3. *Galois/Counter mode*.

Pour développer cette solution dans un contexte de cryptographie à bas coût, le NIST[4] a lancé une compétition [15] pour établir des standards de chiffrement symétrique qui assurent également l'intégrité des données. En février 2023, il a élu Ascon [6] comme nouveau standard. Les candidats de la compétition NIST appelés en anglais AEAD[5] (attention, le mot « authenticated », qui est parfois traduit par authentifié, signifie ici intègre.), suivent le schéma illustré en Figure 1.

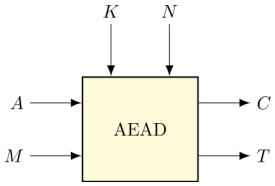


Fig. 1. Schéma générique d'un chiffrement symétrique intégré AEAD.

Ils prennent en entrée un message clair M, des données dites associées A, une clé secrète K et un vecteur d'initialisation N appelé « nonce ». Attention, N doit changer pour chaque message. Ils retournent un tag T, qui permet de vérifier l'intégrité des données, ainsi qu'un texte chiffré C.

Algorithmes cibles

Dans cet article nous présentons deux algorithmes.

L'ALGORITHME AES

L'algorithme AES[6] [14] est une primitive bloc, approuvée par le NIST en 2001, illustrée en figure 2. Il existe sous trois formes : 10, 12 ou 14 tours, avec des clés respectives de 128, 192 et 256 bits ; dans cet article, nous utilisons la version 128 bits. Et l'AES traite des blocs de 16 octets vus comme une matrice 4×4 appelée « state ». La fonction de tour de l'AES se décompose en les quatre sous-fonctions ci-dessous.

- **SubBytes** est composé de 16 s-boxes identiques. Cette fonction s-box, notée SB, est une fonction booléenne de 8 bits vers 8 qui s'applique à tous les octets du state.
- **ShiftRows**, noté SR, déplace cycliquement vers la gauche les trois dernières lignes du state de respectivement un, deux et trois éléments.
- **MixColumns**, noté MC, applique une transformation linéaire aux colonnes du state.

4. *National Institute of Standards and Technology*.
5. *Authenticated encryption with associated data*.
6. *Advanced encryption standard*.

– **AddRoundKey** effectue une opération « xor » avec la clé de tour.

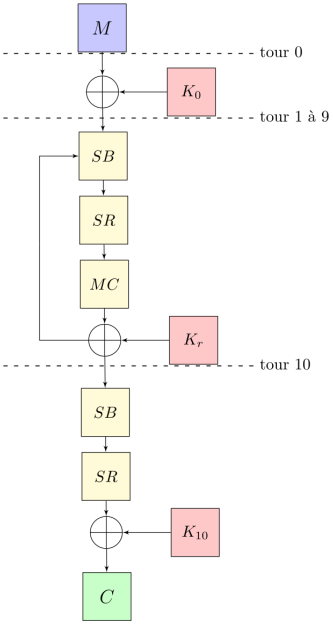


Fig. 2. Schéma de l'AES de 10 tours.

Les fonctions *SR* et *MC* servent à la diffusion ; un bit changé sur un octet affecte l'ensemble du state après deux tours. Il est important de préciser que le tour 0 de l'AES effectue uniquement un xor avec la première clé, et que le dernier tour ne contient pas de *MC*.

Une clé de tour de l'AES est représentée comme le state sous la forme d'une matrice 4×4. Ainsi, l'emplacement de chacun des octets est noté par un couple (l, c), représentant respectivement la ligne l et la colonne c dans la matrice.

Pour un AES_{128}, la clé maître K est la clé du tour 0, K_0. La clé de tour K_{r+1} est obtenue à partir de la clé du tour précédent K_r, en utilisant les équations (4) suivantes :

$$\begin{cases} K_{r+1}^{l,0} &= SB\left(K_r^{(l+1) \bmod 4,3}\right) \oplus rcon(l,r) \oplus K_r^{l,0} & \forall l \in [\![0,3]\!] \\ K_{r+1}^{l,c} &= K_r^{l,c} \oplus K_{r+1}^{l,c-1} & \forall l \in [\![0,3]\!] \text{ et } c \in [\![1,3]\!] \end{cases} \tag{4}$$

avec SB la fonction s-box, et $rcon$ désigne une matrice constante de taille 4×10.

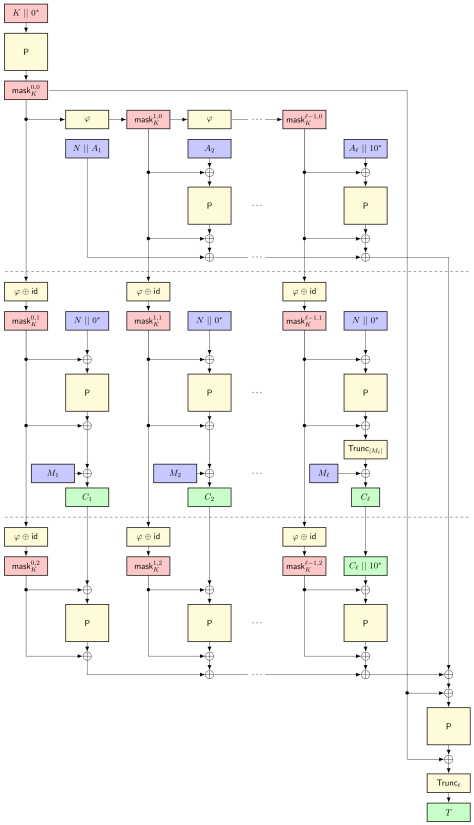


Fig. 3. L'initialisation est tout à gauche, suivie du chiffrement. La génération du tag est tout à droite.

Elephant [2] est un finaliste AEAD de la compétition de cryptographie légère du NIST. Soit P une permutation cryptographique de n bits, et φ un LFSR[7] de n bits. Soit la fonction $\text{mask} : [\![0,1]\!]^{128} \times \mathbb{N} \times \{0,1,2\} \to [\![0,1]\!]^n$ telle que :

$$\text{mask}_K^{i,j} = \text{mask}(K,i,j) = (\varphi \oplus \text{id})^j \circ \varphi^i \circ \text{P}\left(K||0^{n-128}\right), \tag{5}$$

où id est la fonction identité.

Le chiffrement En prend en entrée une clé de 128 bits K, un nonce de 96 bits N, des données associées A et un texte en clair M. Il produit un texte chiffré C

7. *Linear-feedback shift register*, registre à décalage à rétroaction linéaire.

de la même taille que M, et un tag T de ν bits. Le schéma est illustré sur la figure 3.

Le déchiffrement prend en entrée la clé de 128 bits K, le nonce de 96 bits N, les données associées A, le texte chiffré C, et le tag T de ν bits. Il retourne un texte en clair M de la même taille que C si et seulement si le tag T est correct ; sinon, il retourne un message d'erreur.

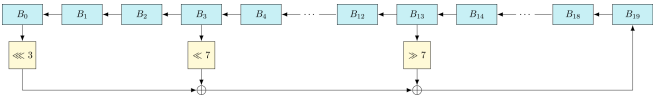


Fig. 4. 160-bit LFSR φ_{Dumbo}

Elephant se décline en trois versions qui diffèrent par la permutation cryptographique de n bits P et le LFSR φ utilisés, ainsi que par la taille du tag ν. Dans cet article, nous n'utiliserons que la version Dumbo. Dumbo utilise une permutation de 160 bits [3], le LFSR φ_{Dumbo} donné par l'équation (6), illustré sur la Figure 4, et a une taille de tag $\nu = 64$ bits.

$$\varphi_{\text{Dumbo}} : (B_0, \cdots, B_{19}) \mapsto (B_1, \cdots, B_{19}, (B_0 \lll 3) \oplus (B_3 \ll 7) \oplus (B_{13} \gg 7)) \tag{6}$$

Attaques par observation

Entre l'algorithme conçu de manière théorique et sa mise en pratique dans un circuit (appelée aussi implémentation logicielle ou matérielle), il y a différents niveaux d'abstraction. À chaque niveau d'abstraction, de nouvelles failles sont possibles, c'est pourquoi les algorithmes théoriques ne peuvent pas anticiper toutes les attaques (présentes et futures) à la fois. Voici une liste des différents niveaux.

1. **Algorithme :** la représentation abstraite.
2. **Logiciel :** le langage de programmation. Une fois conçu, l'algorithme est décrit dans un langage de programmation. Le programme est alors compilé afin d'être traduit en langage machine, sous forme d'un ensemble d'instructions.
3. **Architecture de la machine et micro-architecture :** le processeur ou CPU est le composant de l'ordinateur qui exécute les instructions machines. Un processeur construit en un seul circuit intégré est appelé microprocesseur.
4. **Circuits logiques :** une instruction de type ALU se décompose en fonctions booléennes. Elles sont mises en œuvre en électronique sous forme de portes logiques. Une porte logique réalise des opérations booléennes sur une séquence de bits.

5. **Transistors :** dispositifs semi-conducteurs qui permettent de contrôler un courant (ou une tension). Les portes logiques sont construites à partir de plusieurs transistors connectés de manière adéquate. Dans un transistor, un 0 logique correspond à une tension en dessous d'une certaine valeur, sinon cela correspond à un 1 logique.

Si un algorithme est théoriquement *sûr*, le circuit dans lequel il est implémenté ne l'est pas forcément. C'est la frontière entre la théorie et la pratique. L'algorithme est théorique, le circuit, lui, est bien concret. Ainsi, ce dernier a une consommation de courant, un temps de calcul, etc. Ce sont ces paramètres physiques qui peuvent nuire à la sécurité de notre algorithme.

Les attaques par observation, appelées aussi parfois attaques par canaux auxiliaires, notées SCA[8], s'effectuent aux niveaux circuit logique et transistors et ciblent généralement les niveaux logiciel et algorithme.

On observe depuis quelques années une augmentation de l'utilisation de petits circuits, connectés ou non, dans la vie quotidienne. Or, les systèmes embarqués et autres petits circuits sont des cibles privilégiées des attaques par observation. Le but de ces attaques est généralement :
– la recherche d'un secret, comme une clé cryptographique, ce qui est le cas dans la suite de cet article ;
– ou la rétroconception d'algorithmes.

Fuites physiques

Une attaque par observation analyse une fuite physique d'un circuit. Il s'agit de mesures physiques comme le temps, la température, la consommation de courant ou le rayonnement électromagnétique. Dans cet article, nous nous focalisons sur les deux derniers.

CONSOMMATION DE COURANT

L'idée est de se brancher directement sur le circuit pour mesurer sa consommation de courant. Un courant électrique est un déplacement de charges électriques. Chaque instruction réalisée par un circuit met en œuvre un certain nombre de transistors. À chaque instant, la mesure de l'intensité du courant électrique, appelée consommation de courant, reflète l'activité du circuit.

RAYONNEMENT ÉLECTROMAGNÉTIQUE

Les émissions d'ondes électromagnétiques d'un circuit sont causées par la circulation de courant en son sein. Cette circulation de charges dans le conducteur produit un champ électromagnétique composé d'un champ magnétique et d'un champ électrique, mathématiquement reliés par les équations de Maxwell. Ces ondes électromagnétiques se propagent dans l'espace et peuvent ainsi être

8. *Side channel analysis*.

captées par une sonde. La sonde doit généralement être positionnée aussi près que possible de la surface active du circuit ; on parle de mesures en champ proche. La figure 5 représente une plateforme permettant d'acquérir le rayonnement électromagnétique d'un circuit. Cette plateforme est la propriété de l'INRIA.

Fig. 5. EMA, plateforme de mesure de rayonnement électromagnétique du Laboratoire haute sécurité (LHS) d'Inria Rennes.

L'avantage majeur d'une analyse électromagnétique (EM) par rapport à une analyse par consommation de courant est que les mesures sont prises localement. Un autre avantage est que ces attaques sont non-invasives. Le circuit ne peut pas facilement détecter qu'il est observé, car il n'y a ni branchement ni même un contact physique. Cependant, les mesures EM sont souvent plus sensibles au bruit et dépendent fortement du positionnement de la sonde. La consommation de courant et l'EM sont étroitement liées, elles sont donc souvent modélisées et utilisées de manière très similaire.

Modèles mathématiques

Un circuit intégré est constitué d'un ensemble de portes logiques. La consommation de courant du circuit est la somme des consommations de courant de chacune de ses portes logiques. Une porte logique n'a pas une consommation de

courant constante dans le temps. Il existe deux modèles de consommation de courant pour les portes logiques :
– un modèle dynamique où une porte logique consomme de l'énergie à chaque changement d'état ;
– un modèle statique où une porte logique consomme de l'énergie même au repos.

Ces deux modèles sont couramment utilisés. On peut donc distinguer une transition de 0 vers 1 d'une transition de 1 vers 0. C'est pourquoi la consommation de courant d'une valeur présente dans le circuit, portée par un ensemble de portes logiques, est considérée comme proportionnelle :
– à son HW[9], c'est-à-dire le nombre de bits à 1 ;
– ou à sa HD[10] avec la dernière valeur calculée, c'est-à-dire le nombre de variations de bits entre deux valeurs.

Un attaquant qui récupère le HW d'un octet obtient de l'information sur celui-ci. En effet, un octet B peut prendre 256 valeurs dans $[\![0, 255]\!]$. Connaissant le HW de B, l'attaquant réduit la liste des valeurs possibles, comme le montre le tableau 1.

$\mathrm{HW}(B)$	0	1	2	3	4	5	6	7	8
$\#B$	1	8	28	56	70	56	28	8	1

Table 1. Nombre de valeurs possibles pour un octet B selon son poids de Hamming HW.

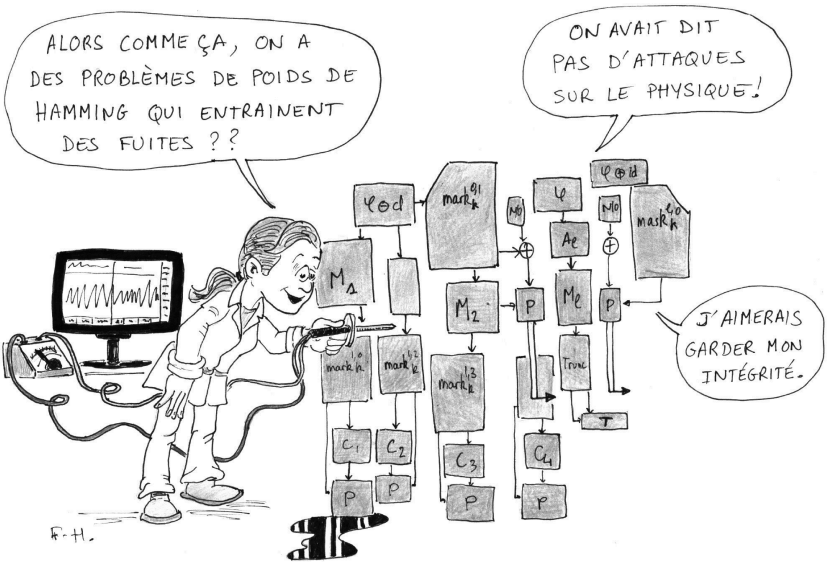
Si le poids de Hamming est souvent utilisé comme modèle représentant la consommation de courant ou le rayonnement électromagnétique, un modèle plus précis est un poids de Hamming bruité $\widetilde{\mathrm{HW}}$. C'est le modèle utilisé dans cet article. Soit B un octet, le poids de Hamming bruité $\widetilde{\mathrm{HW}}$ se calcule comme suit :

$$\widetilde{\mathrm{HW}}(B) = \mathrm{HW}(B) + \sigma_{B,t} \tag{7}$$

où $\sigma_{B,t}$ un événement de la variable aléatoire gaussienne $\mathcal{N}\left(0, \sigma^2\right)$ à un instant t . La fonction de densité de probabilité $\mathbf{F}$ associée à $\mathcal{N}\left(0, \sigma^2\right)$ est donnée par :

$$\mathbf{F}_{\sigma}(B) = \frac{1}{\sigma \cdot \sqrt{2\pi}} \cdot \exp\left(-\frac{1}{2} \cdot \left(\frac{B}{\sigma}\right)^2\right) \tag{8}$$

9. Hamming weight.
10. Hamming distance.



Attaques sur algorithme de chiffrement symétrique

Modèle de l'attaquant

Dans cet article, nous présentons des attaques à textes inconnus. Cela signifie que l'attaquant ne possède ni les données d'entrée ni de sortie de l'algorithme ciblé. Il a connaissance uniquement des traces mesurées, modélisées par des poids de Hamming bruités sur les octets. Ces attaques sont notées BSCA[11].

Les deux premières publications [11, 12] de ce type sont apparues à la même période, dans le but d'attaquer l'AES. Ces travaux ont été améliorés par Christophe Clavier *et al.* dans [4]. C'est dans cette contribution qu'apparaît pour la première fois le terme BSCA. Aujourd'hui, c'est une nouvelle famille d'attaques par observation [5].

Les attaques BSCA n'ont pas encore prouvé leur faisabilité en pratique. Elles travaillent avec des simulations de consommation de courant, généralement un poids de Hamming bruité. Néanmoins, cette nouvelle famille d'attaques est un très bon outil pour tester des primitives cryptographiques symétriques.

Algorithme de propagation de croyance

Dans la suite de cet article, nous supposons que l'attaquant a récupéré des poids de Hamming bruités. Il va alors définir un chemin d'attaque, c'est-à-dire relier

11. *Blind side channel analysis*.

ses mesures au secret attaqué (une clé) via les calculs de l'algorithme. Puis, il regroupe toutes ces informations pour obtenir le secret.

À cette fin, une technique connue sous le nom d'algorithme de propagation de croyance, BP[12] [10], est utilisée. BP est une approche d'inférence bayésienne [1]. Il a été utilisé pour la première fois par Gallager [7, 8] pour le décodage des codes correcteurs. Il a ensuite été redécouvert par Tanner [19] et formalisé par Pearl [16]. La première fois que BP a été utilisé en SCA, c'était dans l'attaque [20], puis il a été étudié dans [9]. Enfin, cet outil a été utilisé pour des attaques BSCA [11, 13].

GRAPHE DE TANNER

Un algorithme BP repose sur un graphe de Tanner. Les nœuds d'un graphe factoriel sont de deux types :

– des nœuds variables B (notés B car nous travaillons sur des octets) ;
– des nœuds d'équations $\mathcal{E}$.

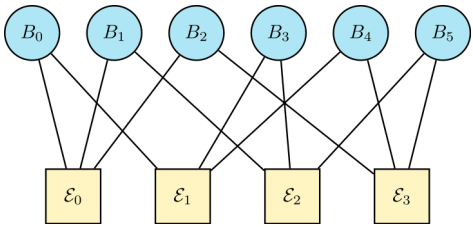


Fig. 6. Exemple d'une partie d'un graphe de Tanner. Les cercles sont des nœuds variables et les carrés sont des nœuds d'équations.

Une arête relie un nœud de variable B_i à un nœud d'équation $\mathcal{E}_j$, lorsque l'équation $\mathcal{E}_j$ représentée par le nœud d'équation implique la variable B_i. Un extrait d'un graphe de Tanner est illustré dans la Figure 6.

N désigne l'ensemble des voisins d'un nœud. Ainsi, l'ensemble des $\mathrm{N}(\mathcal{E})$ est composé de nœuds variables B, et l'ensemble des $\mathrm{N}(x)$ est composé de nœuds d'équations $\mathcal{E}$.

PROPAGATION DE LA CROYANCE

Les entrées d'un algorithme de propagation des croyances (BP) sont :

– un graphe de Tanner ;
– des probabilités a priori $\mathbb{P}_A(B_i = b)$ sur les différents nœuds de variables B.

12. *Belief propagation*.

L'algorithme BP retourne des probabilités, à partir des valeurs initiales $\mathbb{P}_A(B_i = b)$. Ces probabilités sont dites a posteriori $\mathbb{P}_P(B_i = b)$ et portent sur les différents nœuds de variables.

L'idée est que l'information se diffuse à travers le graphe de Tanner. Les nœuds voisins s'échangent de l'information et la partagent avec les autres voisins, et ainsi de suite.

En pratique, les nœuds dans le graphe échangent des messages avec leurs voisins. Plus précisément, puisque le graphe est biparti, deux types de messages sont échangés :

– des messages d'un nœud d'équation E vers un nœud de variable V, notés $\mu_{E\to V}$;
– des messages d'un nœud de variable V vers un nœud d'équation E, notés $\mu_{V\to E}$.

Les messages de variable à équation $\mu_{V\to E}(v)$ sont initialisés avec les probabilités a priori $\mathbb{P}_A(B_i = b)$.

Puis, BP fonctionne en appliquant alternativement les messages de chaque type pour mettre à jour les probabilités. À la fin de l'exécution, la valeur retournée est appelée probabilité a posteriori. Le nombre d'itérations dépend de la rapidité de convergence, qui elle-même dépend du graphe de Tanner.

Application sur différents algorithmes

ATTAQUE SUR AES

Dans cette partie, l'attaque BSCA sur AES [11] est résumée.

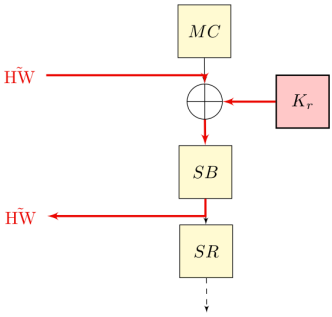


Fig. 7. Chemin d'attaque sur l'AES

Chemin d'attaque

L'acquisition de traces se déroule à chaque tour de l'AES, exceptés 0 et 10. L'objectif est de récupérer le poids de Hamming bruité des différents octets du state

avant le xor de clé et après le **SubBytes**. Il n'y a pas de **MixColumns** avant le « xor » avec K_0, et il n'y a pas de **SubBytes** après K_{10}.

Ensuite, une première étape consiste, à partir des mesures de ces poids de Hamming bruités, à attribuer une probabilité à chacun des octets des différentes clés de tours.

Graphe de Tanner

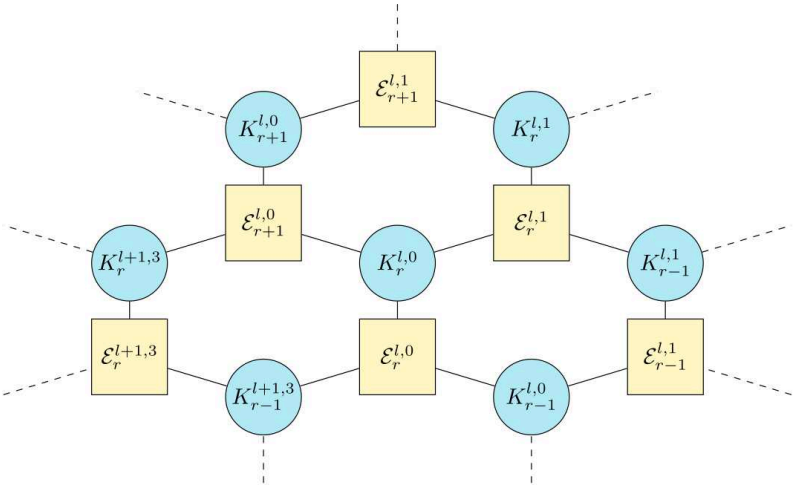


Fig. 8. Extrait du graphe de Tanner basé sur les équations liant les clés de tours de l'AES.

Dans l'AES, les différentes clés de tour sont dérivées de la clé secrète utilisée au premier tour. De plus, l'attaquant dispose de mesures permettant d'attribuer une probabilité à chacun des octets des différentes clés de tours. Ainsi, le graphe de Tanner est construit sur l'algorithme de dérivation des clés de tours. Il est illustré en Figure 8. Les équations $\mathcal{E}$ définies en #eq:KeyExpansion relient les différents octets des clés de tours.

Résultats

Les résultats illustrés dans le tableau 2 montrent que l'attaque est efficace. L'utilisation de l'algorithme BP est un excellent moyen de récupérer la clé. Même en présence d'un bruit élevé, l'attaque peut réussir. L'approche montre qu'augmenter le nombre de tours dans une primitive bloc, afin de le rendre résistant à la cryptanalyse classique, peut l'affaiblir dans notre cas.

$n\backslash\sigma$	0.1	0.2	0.3	0.5	1.0	1.5	2.0	3.0
100	✓	✓	✓	✓	×	×	×	×
1000	✓	✓	✓	✓	✓	×	×	×
10000	✓	✓	✓	✓	✓	✓	✓	×
100000	✓	✓	✓	✓	✓	✓	✓	✓

Table 2. Succès de l'attaque en fonction de l'écart-type du bruit σ et du nombre de traces n, pour 100 attaques simulées. ✓ indique que l'attaque a réussi et × indique que l'attaque a échoué.

ATTAQUE SUR ELEPHANT

Dans ce paragraphe, un autre exemple d'application de ce type d'attaque est présenté.

Chemin d'attaque

Les LFSR sont souvent utilisés en cryptographie légère, et leurs états initiaux dépendent souvent à la fois de la clé K et du nonce N. Étant donné que le nonce doit être modifié pour chaque chiffrement, les attaques sur de tels schémas sont limitées à l'algorithme de déchiffrement. Dans le cas d'Elephant, le LFSR dépend uniquement de la clé secrète. Par conséquent, notre attaque peut être appliquée dans un scénario de chiffrement. L'objectif de l'attaque est alors de récupérer l'état initial secret du LFSR d'Elephant. Récupérer l'état initial du LFSR équivaut à récupérer la clé secrète. En effet, l'état initial n'est que le résultat de la permutation connue P appliquée à la clé. Le LFSR générant un nouvel octet à chaque itération, le contenu du φ_{Dumbo} LFSR est noté comme suit :

$$(B_j, \cdots, B_{j+19}) = \mathrm{mask}_K^{j,0} \tag{9}$$

et soit B_{j+20} l'octet généré à l'itération j. De même, soit $(B_j^1, \cdots, B_{j+19}^1) = \mathrm{mask}_K^{j,1}$ et $(B_j^2, \cdots, B_{j+19}^2) = \mathrm{mask}_K^{j,2}$. Par définition du $mask$, les relations suivantes sont valables pour tout $j \geq 0$:

$$B_j^1 = B_j \oplus B_{j+1} \tag{10}$$

$$B_j^2 = B_j \oplus B_{j+2} \tag{11}$$

L'attaquant peut exploiter deux vecteurs d'attaque : d'une part, les équations (6) provenant de l'itération du LFSR, et d'autre part, les équations (10) et (11) provenant des différents masques utilisés pour la séparation des domaines. Dans un premier temps, l'attaquant obtient un HW bruité sur les différents octets des φ_{Dumbo} LFSR.

Graphe de Tanner

Dans le cas d'une attaque sur Elephant, le graphe de Tanner est construit comme suit. Les nœuds de variables sont les octets B_i, B_j^1 et B_k^2 des LFSR. Les nœuds de facteurs représentent :
– les équations de type (10) ;
– les équations de type (11) ;
– les équations de rétroaction du LFSR (6).

Le graphe de Tanner d'Elephant est illustré dans la Figure 9.

Résultats

Pour 1000 clés générées aléatoirement et 11 valeurs de bruit différentes dans $0.1 \leq \sigma \leq 1$, la moyenne, l'écart type, la médiane et les quartiles des rangs des 20 octets de clé corrects ont été calculés. Les résultats sont donnés dans 3.

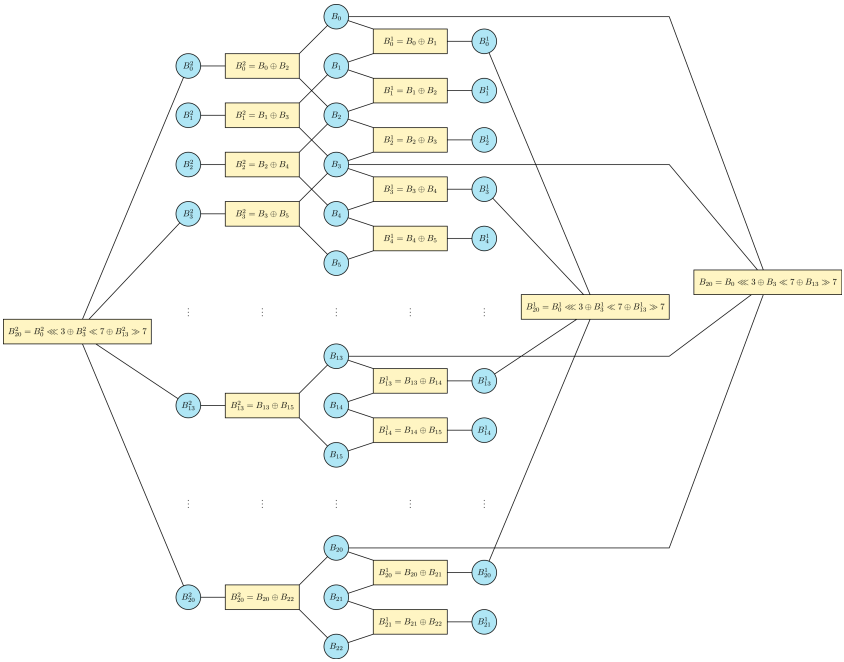


Fig. 9. Extrait du graphe de Tanner du LFSR φ_{Dumbo} d'Elephant.

σ	Moyenne	Écart type	Quartile Q1	Médiane	Quartile Q3	Max
0.1	3.54	5.97	0	3	3	27
0.2	3.67	6.11	0	3	3	31
0.3	6.00	10.76	0	3	3	97
0.4	9.59	15.72	0	3	8	97
0.5	15.97	23.03	3	8	31	153
0.6	23.65	31.41	3	8	31	157
0.7	33.73	40.11	3	27	36	213
1	70.68	61.42	31	36	92	246

Table 3. Rang des 20 octets de la clé pour 1000 clés générées aléatoirement et selon différents bruits σ.

Les résultats montrent que lorsque le niveau de bruit est faible, BP remonte les bonnes hypothèses d'octets de clé dans les premiers rangs. Un attaquant peut alors exploiter les probabilités a posteriori renvoyées par BP et utiliser des techniques d'énumération de clés pour retrouver la clé entière. Notons que, puisque le calcul du LFSR ne dépend que de la clé, il est possible de réduire le niveau de bruit en moyennant plusieurs traces.

La conception d'Elephant, où il existe des relations entre les différents masques du LFSR, constitue une vraie vulnérabilité face à cette attaque.

Conclusion

Dans cet article, le principe de la cryptanalyse physique à textes inconnus est présenté et illustré sur deux algorithmes de chiffrement symétrique. Ces travaux montrent que la cryptanalyse physique, qui utilise l'environnement physique d'un circuit pour en extraire des secrets, peut être employée avec des modèles d'attaquant ayant accès à peu d'informations. Il est ici possible de retrouver le secret uniquement à partir de la trace d'exécution du chiffrement, sans connaître ni les entrées ni les sorties. Cette approche consiste à dérouler un algorithme de propagation de croyance sur l'algorithme attaqué. Bien qu'elle en soit encore au stade théorique, elle permet d'anticiper les attaques pratiques.

Références

[1] David Barber. 2011. *Bayesian Reasoning and Machine Learning*. Cambridge University Press.

[2] Tim Beyne, Yu Long Chen, Christoph Dobraunig, and Bart Mennink. 2021. Elephant v2. *NIST lightweight competition*.

[3] Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. 2011. Spongent: a Lightweight Hash Function. In *CHES*.

[4] Christophe Clavier and Léo Reynaud. 2017. Improved blind side-channel analysis by exploitation of joint distributions of leakages. In *CHES*.

[5] Christophe Clavier, Léo Reynaud, and Antoine Wurcker. 2018. Quadrivariate improved blind side-channel analysis on boolean masked AES. In *COSADE*.

[6] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. 2014. Ascon. *Submission to the CAESAR competition*.

[7] Robert G. Gallager. 1962. Low-density parity-check codes. *IRE Trans. on Information Theory*.

[8] Robert G. Gallager. 1963. Low Density Parity check codes. MIT, Cambridge, MA.

[9] Vincent Grosso and François-Xavier Standaert. ASCA, SASCA and DPA with Enumeration: Which One Beats the Other and When? In *ASIACRYPT 2015*. Springer.

[10] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Trans. on Information Theory*.

[11] Hélène Le Bouder, Ronan Lashermes, Yanis Linge, Gaël Thomas, and Jean-Yves Zie. 2016. A Multi-round Side Channel Attack on AES Using Belief Propagation. In *FPS*.

[12] Yanis Linge, Cécile Dumas, and Sophie Lambert-Lacroix. 2014. Using the joint distributions of a cryptographic function in side channel analysis. In *COSADE*.

[13] Julien Maillard, Awaleh Houssein Meraneh, Modou Sarry, Christophe Clavier, Hélène Le Bouder, and Gaël Thomas. 2023. Blind side channel analysis on the Elephant LFSR Extended version. *SECRYPT BOOK*.

[14] NIST. 2001. Specification for the Advanced Encryption Standard. *FIPS PUB 197*.

[15] NIST. 2018. *Lightweight cryptography, standardization process*.

[16] Judea Pearl. 1982. Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach. In *AAAI*.

[17] Modou Sarry, Hélène Le Bouder, Eid Maaloouf, and Gaël Thomas. 2023. Blind side channel analysis against AEAD with a belief propagation approach. In *CARDIS*.

[18] Claude E. Shannon. 1949. Communication Theory of Secrecy Systems. *The Bell System Technical Journal* 28, 4: 656–715.

[19] Robert M. Tanner. 1981. A recursive approach to low complexity codes. *IEEE Trans. on Information Theory*.

[20] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft Analytical Side-Channel Attacks. In *ASIACRYPT 2014*.