Université de Versailles – Saint-Quentin-en-Yvelines
Université Paris Saclay

le 09 / 12 / 2023

Jules Baudrin (Inria Paris)

Master 1 Informatique 2023–2024
Cryptographie

Correction exercice 2 TD 4 : cryptanalyse d'AES

0. Montrer qu'un tour complet d'AES, $\mathrm{ARK}_K \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{SB}$, est aussi sécurisé que $\mathrm{MC} \circ \mathrm{SR} \circ \mathrm{ARK}_K \circ \mathrm{SB}$.

Soit $x \in \{0,1\}^{128}$, et y définie comme suit :

$$y = \mathrm{ARK}_K \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{SB}(x) = \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{SB}(x) + K. \tag{1}$$

Puisque MC et SR sont des applications bijectives, leur composée $\mathrm{MC} \circ \mathrm{SR}$ est aussi une application bijective. Ainsi, l'inverse $(\mathrm{MC} \circ \mathrm{SR})^{-1}$ existe et vaut $(\mathrm{MC} \circ \mathrm{SR})^{-1} = \mathrm{SR}^{-1} \circ \mathrm{MC}^{-1}$. Appliquant $\mathrm{SR}^{-1} \circ \mathrm{MC}^{-1}$ à gauche et à droite de l'equation (1), on obtient :

$$\mathrm{SR}^{-1} \circ \mathrm{MC}^{-1}(y) = \mathrm{SR}^{-1} \circ \mathrm{MC}^{-1} \left(\mathrm{MC} \circ \mathrm{SR} \circ \mathrm{SB}(x) + K\right).$$

Mais $\mathrm{SR}^{-1} \circ \mathrm{MC}^{-1}$ est une application linéaire[1]. On obtient donc[2] :

$$\mathrm{SR}^{-1} \circ \mathrm{MC}^{-1}(y) = \mathrm{SR}^{-1} \circ \mathrm{MC}^{-1} \left(\mathrm{MC} \circ \mathrm{SR} \circ \mathrm{SB}(x)\right) + \mathrm{SR}^{-1} \circ \mathrm{MC}^{-1} \left(K\right);$$

ce qui se simplifie en :

$$\mathrm{SR}^{-1} \circ \mathrm{MC}^{-1}(y) = \mathrm{SB}(x) + \mathrm{SR}^{-1} \circ \mathrm{MC}^{-1} \left(K\right);$$

ou encore en notant $\widetilde{K} = SR^{-1} \circ \mathrm{MC}^{-1}\left(K\right)$:

$$\mathrm{SR}^{-1} \circ \mathrm{MC}^{-1}(y) = \mathrm{SB}(x) + \widetilde{K} = \mathrm{ARK}_{\widetilde{K}} \circ \mathrm{SB}(x). \tag{2}$$

Enfin appliquant $\mathrm{MC} \circ \mathrm{SR}$ à gauche et à droite de equation (3), on obtient :

$$y = \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{ARK}_{\widetilde{K}} \circ \mathrm{SB}(x). \tag{3}$$

Finalement, y peut être calculé de deux manières équivalentes :

1. soit avec un tour complet d'AES comme dans l'equation (1) en utilisant une clé K,
2. soit en utilisant la représentation alternative de l'equation (3) dans laquelle l'ajout de la clé *modifiée* $\widetilde{K}$ a lieu entre la couche de boîte-S et les couches linéaires SR et MC.

1. En effet, l'inverse d'une application linéaire est linéaire, donc $\mathrm{SR}^{-1}, \mathrm{MC}^{-1}$ sont linéaires et la composée d'applications linéaires est linéaire.
2. Une application $\mathbb{F}_2$-linéaire f vérifie $f(x+y) = f(x) + f(y)$ pour tout x, y, autrement dit "l'image d'une somme est la somme des images".

Ce changement de représentation ne change absolument rien à la sécurité : retrouver $\widetilde{K}$ ou K est équivalent : si l'on retrouve l'une ou l'autre des clés, on passe à la deuxième en appliquant une application *publique* $\mathrm{SR}^{-1} \circ \mathrm{MC}^{-1}$ ou $\mathrm{MC} \circ \mathrm{SR}$ selon le sens.

Remarque (Échange addition/bijection linéaire). *De manière générale, si un ajout de clé **suit directement** une application **linéaire bijective** (ou si une app. lin. bij suit directement un ajout de clé), on peut toujours échanger les deux applications, quitte à adapter linéairement la clé.*

1. Expliquer pourquoi, du point de vue de la sécurité, il ne sert à rien de faire un dernier tour complet pour l'AES.

Supposons que le dernier tour de l'AES soit un tour complet.

D'après la question précédente, on peut tout à fait remplacer la représentation standard $\mathrm{ARK}_K \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{SB}$ par $\mathrm{MC} \circ \mathrm{SR} \circ \mathrm{ARK}_{\widetilde{K}} \circ \mathrm{SB}$. Notons F les premiers tours d'AES et (x, y) un couple clair/chiffré connu. On a alors $y = \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{ARK}_{\widetilde{K}} \circ \mathrm{SB} \circ F(x)$. Puisque MC et SR sont publiques, un attaquant connaissant y peut tout à fait calculer $\tilde{y} = \mathrm{SR}^{-1} \circ \mathrm{MC}^{-1}(y)$. D'après la formule, on obtient alors : $\tilde{y} = \mathrm{SR}^{-1} \circ \mathrm{MC}^{-1} \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{ARK}_{\widetilde{K}} \circ \mathrm{SB} \circ F(x)$ et en simplifiant :

$$\tilde{y} = \mathrm{ARK}_{\widetilde{K}} \circ \mathrm{SB} \circ F(x). \tag{4}$$

Avec la connaissance de y, un attaquant peut se ramener à l'equation (4), qui est exactement l'équation d'un AES dans lequel le dernier tour est simplement constitué d'une couche de boites-S suivie d'un ajout de clé. Ainsi du point de vue de la sécurité, il ne sert à rien de faire un dernier tour complet pour l'AES.

Remarque. *De manière générale, l'utilisation de n'importe quelle application publique **après le dernier ajout de clé** est inutile du point de vue de la sécurité.*

2. Expliquer pourquoi l'AES commence par un ajout de clé suivi tout de suite de SubBytes.

D'après la question précédente, les applications publiques appliquées après le dernier ARK sont inutiles. Il en va de même des applications publiques appliquées avant le premier ajout de clé.

En effet, si $y = F \circ \mathrm{ARK}_K \circ P(x)$ où P est publique et l'entrée x est connue, un attaquant peut tout à fait calculer $\tilde{x} = P(x)$ et se ramener à l'équation $y = R \circ \mathrm{ARK}_K(\tilde{x})$ qui correspond à la construction sans application publique avant le premier ARK. C'est pourquoi l'AES commence par ARK[3].

Supposons maintenant que ce premier ARK ne soit **pas** directement suivi de SB. Il serait alors suivi de SR puis MC. Mais alors, d'après la question 0, on pourrait réécrire $\mathrm{MC} \circ \mathrm{SR} \circ \mathrm{ARK}$ en $\mathrm{ARK} \circ \mathrm{MC} \circ \mathrm{SR}$ quitte à adapter la clé. On est alors précisément dans la situation décrite au paragraphe précédent et on peut simplifier $\mathrm{ARK} \circ \mathrm{MC} \circ \mathrm{SR}$ en seulement ARK car $\mathrm{MC} \circ \mathrm{SR}$ est publique et appliquée avant le premier ARK.

3. Et c'est pourquoi IP/IP^{-1} dans le DES sont inutiles.

Cette manipulation n'est possible que parce $\mathrm{MC} \circ \mathrm{SR}$ est linéaire et bijective (*c.f.* Remarque question 0). Par construction, SB est **non-linéaire** : un tel échange est donc impossible. C'est pourquoi l'AES commence par un ajout de clé suivi tout de suite de SubBytes.

Remarque (Bijections publiques et modèles d'attaque). *De manière générale, étant donné n'importe quelle* ***bijection*** *publique P, on a $P(x) = y \iff x = P^{-1}(y)$, donc connaître x c'est connaître y (car on peut calculer $P(x)$) et inversement connaître y c'est connaître x (car on peut calculer $P^{-1}(y)$). Mais le même raisonnement nous dit aussi que* ***choisir*** *x ou y est équivalent : si je veux en sortie y sachant que je contrôle x, je peux choisir $x = P^{-1}(y)$ et obtenir la sortie désirée.*
Ainsi on peut ***généraliser*** *les questions précédentes* ***à tout type de modèle d'attaque*** *: clair connu/choisi et chiffré connu/choisi.*

Remarque (Petite conclusion). *D'après les remarques "Échange addition/bijection linéaire" et "Bijections publiques et modèles d'attaque", on en déduit que quelque soit le chiffrement par blocs et le modèle d'attaque, on peut donc toujours se ramener à l'étude d'une construction qui commence par un ajout de clé suivi d'une application non-linéaire, et qui termine par une application non-linéaire suivi d'un ajout de clé.*

Hypothèse pour la suite

En vertu de la réponse à la question 1, on supposera dans toute la suite que le dernier tour d'AES est uniquement constitué de SB suivi de ARK.

3. Montrer qu'on peut casser AES réduit à 1 tour.

On considère un AES réduit à 1 tour donc $\mathrm{ARK}_{K_1} \circ \mathrm{SB} \circ \mathrm{ARK}_{K_0}$ (une addition de clé initiale et un dernier tour partiel) et on regarde une attaque à clair/chiffré connus. Avec une paire, (x, y) connue, on obtient une équation à deux inconnues de taille 128, K_0, K_1 :

$$y = \mathrm{SB}(x + K_0) + K_1.$$

Plus précisément, puisque l'addition et la couche de boites-S agissent indépendamment sur les 16 octets, l'équation peut être lue octet par octet : le i ème octet du chiffré dépend uniquement des $i$èmes octets du clair et des deux clés. Plus formellement, cela s'écrit comme ceci :

$$\forall i \in \{0, \cdots, 15\}, \qquad y^i = \mathrm{SB}(x^i + {K_0}^i) + {K_1}^i. \tag{$E_{i,x,y}$}$$

Remarque (Chiffrements en parallèle). *On est donc en train de dire, que les octets sont chiffrés indépendamment à l'aide d'une clé de 16 bits. Un AES réduit à 1 tour se comporte donc comme 16 chiffrements par blocs sur 8 bits en parallèle. D'après le cours, la recherche exhaustive coûte donc 2^{16} par chiffrement par bloc, soit 16×2^{16} au total.*

Soyons un peu plus précis et fixons nous un indice $i \in \{0, \cdots, 15\}$. La clé $({K_0}^i, {K_1}^i)$ utilisée pour chiffrer l'octet i se trouve *a priori* parmi $2^8 \times 2^8 = 2^{16}$ paires candidates. Mais on sait désormais que

$({K_0}^i, {K_1}^i)$ vérifie nécessairement l'équation $E_{i,x,y}$ et fait donc partie de l'espace $S_{i,x,y}$ des solutions de l'équation $E_{i,x,y}$.

La recherche exhaustive consiste donc à vérifier, pour toute paire (O_0, O_1) d'octets, si $(O_0, O_1) \in S_{i,x,y}$ ou si $(O_0, O_1) \notin S_{i,x,y}$ et à éliminer les paires qui ne sont pas solutions. Une équation aléatoire sur un octet (8 bits) avec 2 octets inconnues (16 bits) admet environ $2^{16}/2^8 = 2^8$ solutions. $(E_{i,x,y})$ permet donc avec 2^{16} calculs, de réduire le nombre de paires candidates pour l'octet i a environ 2^8 candidats. C'est bien, mais pas suffisant pour retrouver $({K_0}^i, {K_1}^i)$... On s'en rend encore mieux compte au global : en répétant cette étape pour les 16 octets indépendamment, on aura, avec un couple (x, y), et pour un coût de 16×2^{16} réduit l'espace de clés de 2^{256} (deux clés de 128 bits) à $(2^8)^{16} = 2^{128}$ ce qui est bien comparé à 2^{256}, mais encore beaucoup trop vaste...

Remarque (Meet-in-the-Middle). *En fait, la complexité en temps de 16×2^{16} est complètement pratique mais largement surestimé. En effet, l'équation $(E_{i,x,y})$ est équivalente à l'équation suivante.*

$$\mathrm{SB}(x^i + {K_0}^i) = y^i + {K_1}^i.$$

Ainsi, une paire (O_0, O_1) est solution de $(E_{i,x,y})$ si et seulement si le "chiffré" de x par la première moitié du chiffrement est égale au "déchiffré" de y par la deuxième moitié du chiffrement. Puisque ces deux moitiés dépendent de clés indépendantes, on est exactement dans la situation d'un "Meet-in-the-Middle". Pour retrouver toutes les paires candidates, un attaquant peut donc dresser et stocker la liste des contenant$(\mathrm{SB}(x^i + O_0), O_0)$ pour les 2^8 valeurs de O_0, puis calculer les 2^8 valeurs de $y^i + O_1$. À chaque fois qu'une de ces valeurs collisionne avec la 1ère coordonnée d'un des éléments de la liste, un couple (O_0, O_1) candidat est trouvé. Au total pour une équation sur un octet, cela coûte 2×2^8 en temps (contre le 2^{16} annoncé plus tôt), et 2^8 en mémoire.

Une première possibilité d'amélioration

En fait, il suffit de quelques autres paires clair/chiffré pour rendre cette attaque pratique. Avec plus de paires $(x_2, y_2), (x_3, y_3), \cdots$ on peut construire E_{i,x_2,y_2}, $E_{i,x_3,y_3}, \cdots$ et rendre la filtration effectuée plus haut beaucoup plus puissante ! En effet, les paires candidates (O_0, O_1) sont celles qui sont solutions de **toutes** les équations, donc les paires dans $S_{i,x,y} \cap S_{i,x_2,y_2} \cap S_{i,x_3,y_3}$ qui est beaucoup plus petit que $S_{i,x,y}$. On peut montrer que la taille *moyenne* de l'intersection de deux sous-ensembles de taille a, b tirés uniformément dans un ensemble de taille c est $\frac{ab}{c}$. Ici, on s'attend donc en moyenne à une intersection de taille $\frac{2^8 2^8}{2^{16}} = 1$ entre deux sous-ensembles de solutions de 2^8. C'est une moyenne et la taille 2^8 est une estimation, mais on se convainc aisément que quelques couples (3, 4, 5...) suffisent à éliminer toutes les clés sauf la clé utilisée par l'utilisateur.

Une deuxième possibilité avec deux paires de clairs/chiffrés.

On suppose que l'on possède deux paires (x, y) et (z, t) connues. On a donc deux équations par octet :

$$\forall i \in \{0, \cdots, 15\}, \qquad y^i = \mathrm{SB}(x^i + {K_0}^i) + {K_1}^i \qquad (E_{i,x,y})$$
$$\forall i \in \{0, \cdots, 15\}, \qquad t^i = \mathrm{SB}(z^i + {K_0}^i) + {K_1}^i. \qquad (E_{i,z,t})$$

On peut alors les combiner et obtenir en les sommant :

$$\forall i \in \{0, \cdots, 15\}, \qquad y^i + t^i = \mathrm{SB}(x^i + {K_0}^i) + {K_1}^i + \mathrm{SB}(z^i + {K_0}^i) + {K_1}^i. + {K_1}^i;$$

ce qui, une fois simplifiée donne :

$$\forall i \in \{0, \cdots, 15\}, \qquad y^i + t^i = \mathrm{SB}(x^i + {K_0}^i) + \mathrm{SB}(z^i + {K_0}^i). \qquad (F_{i,x,y,z,t})$$

Cette nouvelle équation $F_{i,x,y,z,t}$ est désormais une équation sur un octet avec **un seul** octet inconnu : ${K_0}^i$. Elle permet donc de filtrer les valeurs de ${K_0}^i$ possible et passer d'un espace de 2^8 candidats à seulement $2^8/2^8 = 1$ si elle était aléatoire. Cependant, on remarque que les solutions vont par paires : en effet un octet O_0 est solution de $F_{i,x,y,z,t}$ si et seulement si $O_0 + x^i + z^i$ est solution (Vérifiez le !). On s'attend donc à une filtration qui permet, pour un coût de 2^8, de passer de 2^8 candidats à environ 2 pour ${K_0}^i$.

Cela permet pour un coût de 16×2^8 de réduire l'espace de K_0 de 2^{128} à 2^{16} sur lequel une recherche exhaustive est instantanée !

Fait. La récupération d'une clé de tour de l'AES est suffisant pour récupérer la clé maître.

On a donc gagné.

Remarque (Filtration). *Moralité, la filtration est un outil très puissant en cryptographie. Un moyen efficace de trouver des filtres très puissants, est de chercher des équations qui dépendent du moins de bits inconnues : ici $F_{i,x,y,z,t}$ dépend seulement de 8 bits inconnus alors qu'on a au total 256 bits de clés de tour (ou 128 bits de clé maitre).*

4. Montrer qu'on peut casser AES réduit à 2 tours.

On considère un AES réduit à 2 tours donc $\mathrm{ARK}_{K_2} \circ \mathrm{SB} \circ \mathrm{ARK}_{K_1} \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{SB} \circ \mathrm{ARK}_{K_0}$ (une addition de clé initiale, un tour complet et un dernier tour partiel). Considérons une attaque à clair/chiffré connus. Avec une paire, (x, y) connue, on obtient une équation en trois inconnues de taille 128, K_0, K_1, K_2 :

$$y = \mathrm{SB} \circ \mathrm{ARK}_{K_1} \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{SB}(x + K_0) + K_2.$$

Mais on peut être plus précis en prenant comme exemple l'octet de sortie y^0 (tout est généralisable aux autres octets). Comme on le voit sur la figure 1, après un AES réduit à 2 tours, y^0 ne dépend que des octets d'entrée x^0, x^5, x^{10}, x^{15}. Concernant la clé, y^0 ne dépend que d'une diagonale de K_0, d'un octet de K_1 et d'un octet de K_2. Cela peut donc permettre de filtrer ces 6 octets indépendamment des autres.

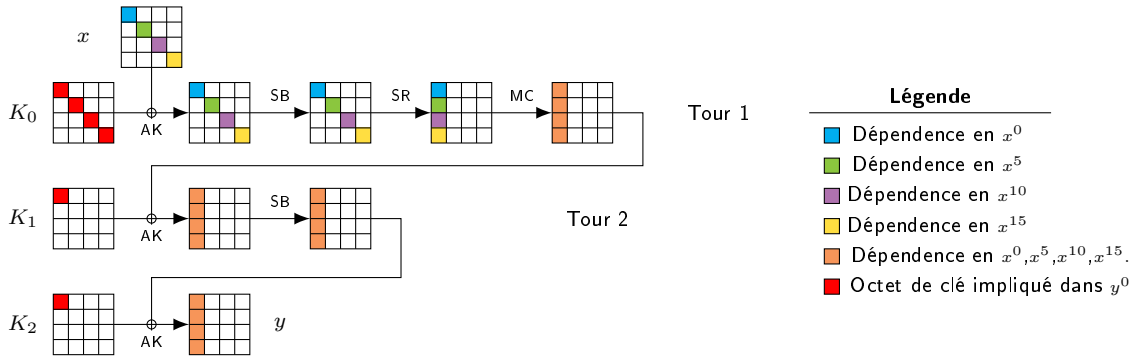

FIGURE 1 – Après 1.5 tour, un octet en sortie ne dépend que d'une diagonale en entrée.

En supposant que l'on possède deux paires (x, y) et (z, t) connues, on peut de nouveau sommer deux équations pour obtenir :

$$y + t = \mathrm{SB} \circ \mathrm{ARK}_{K_1} \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{SB}(x + K_0) + \mathrm{SB} \circ \mathrm{ARK}_{K_1} \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{SB}(t + K_0).$$

Cette équation ne dépend plus de K_2. En se restreignant à l'octet 0 en sortie, on obtient, d'après la figure 1, une équation sur un octet qui dépend de 5 octets inconnus (4 de K_0 et 1 de K_1). Le filtre permet, pour un coût de $2^{5\times 8} = 2^{40}$ de réduire les 2^{40} candidats à environ $2^{40}/2^8 = 2^{32}$ et (à beaucoup moins pour pas beaucoup plus cher si on utilise plus de paires). Cela permet par exemple de déterminer les diagonales de K_0 indépendamment les unes des autres et d'ainsi casser deux tours avec une complexité pratique.

5. Montrer qu'on peut casser AES réduit à 3 tours.

On considère un AES réduit à 3 tours :

$$\mathrm{ARK}_{K_3} \circ \mathrm{SB} \circ \mathrm{MC} \circ \mathrm{ARK}_{MC^{-1}(K_2)} \circ \mathrm{SR} \circ \mathrm{SB} \circ \mathrm{ARK}_{K_1} \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{SB} \circ \mathrm{ARK}_{K_0};$$

c'est à dire une addition de clé initiale, un tour complet, un tour en représentation alternative (où l'on a placé l'addition entre MC et SR) et un dernier tour partiel.

Considérons désormais une attaque à **clairs choisis**. On choisit de chiffrer deux messages très proches et de vérifier si les chiffrés correspondants sont eux aussi proches. Plus précisément on considère un message x quelconque, et un message proche $x + \Delta$ où $\Delta := (\delta, 0, \cdots, 0)$, $\delta \neq 0$; autrement dit x et $x + \Delta$ diffèrent sur le premier octet et sont égaux partout ailleurs.

Suivons l'évolution de la différence couche après couche dans le chiffrement .

- Pour l'**ajout de clé**, on observe, même sans connaître K, que $(x + K) + (x + \Delta + K) = \Delta$ donc une différence reste inchangée avant et après une addition de clé.
- A travers une **boîte S**, puisque S est **bijective**, les images diffèrent si et seulement si les entrées diffèrent. Ainsi avec $\delta \neq 0$, on a nécessairement $S(x^0 + \delta) + S(x^0) = \delta'$, avec $\delta' \neq 0$.
- SR ne fait que réorganiser les différences.

— Enfin, MC est linéaire donc $\mathrm{MC}(x+\Delta) = \mathrm{MC}(x) + \mathrm{MC}(\Delta)$, ce que l'on peut réécrire $\mathrm{MC}(x+\Delta) + \mathrm{MC}(x) = \mathrm{MC}(\Delta)$, autrement dit une différence Δ se transforme en $\mathrm{MC}(\Delta)$. Puisque MC est bijective, on en déduit qu'une différence $\Delta \neq 0$ et nécessairement envoyée sur une différence $\mathrm{MC}(\Delta)$ non nulle [4].

On obtient alors après 1.5 tour, la figure 2.

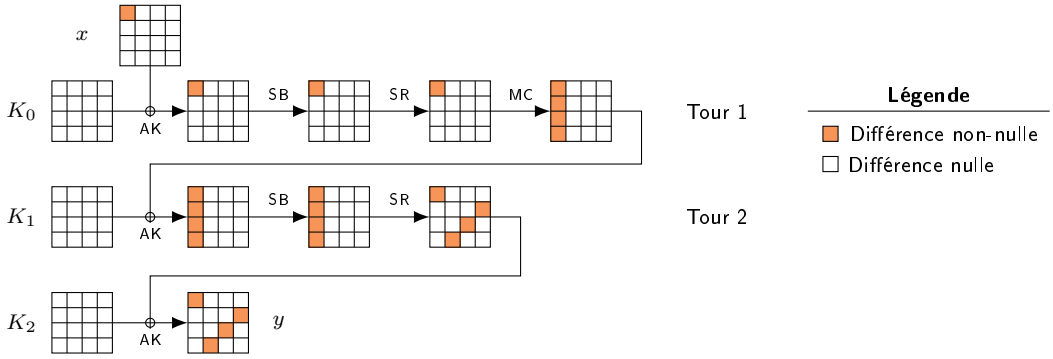

FIGURE 2 – Après 1.5 tour, une colonne de l'état ne contient qu'un octet de différence non-nulle.

Précisons la transition de la première colonne à travers MC. On a en entrée une différence de la forme $(\delta', 0, 0, 0)$, on peut alors calculer $M(\delta', 0, 0, 0) = (0\text{x}02.\delta', \delta', \delta', 0\text{x}03.\delta')$, mais puisque 0x02 et δ' sont non nuls, leur produit, effectué dans le corps à 256 éléments, l'est aussi [5]. La première coordonnée est donc non nulle. Idem pour toutes les autres, donc les 4 différences de la colonne sont non nulles après MC.

Résumons. Après 1.5 tour d'AES, on observe qu'une différence sur un octet ne se propage que sur un octet par colonne en sortie. Cette propriété distingue 1.5 tour d'AES (avec n'importe quelle clé) d'une bijection aléatoire. En effet, pour une bijection aléatoire F, on s'attend à ce que $F(x)$ soit totalement aléatoire, de même pour $F(x+\delta)$ et donc leur différence $F(x) + F(x+\delta)$ est elle aussi aléatoire. En particulier, les sorties peuvent différer sur n'importe quel octet.

Utilisons notre distingueur pour monter une attaque sur un tour de plus. On possède deux paires clair/chiffré (x, y) et $(x+\Delta, t)$. En devinant la 1ère colonne de K_2, on peut inverser partiellement le dernier tour et retrouver **la valeur** de la 1ère colonne avant le deuxième MC pour les deux chiffrements. On nomme ces valeurs $C_x, C_{x+\Delta}$. On est alors en mesure de calculer la différence $C_x + C_{x+\Delta}$. D'après le distingueur, cette différence doit être nulle sur les 3 derniers octets. Si ce n'est pas le cas, alors le guess de la colonne de K_2 est nécessairement faux. Dans le cas contraire, on a trouvé une valeur candidate pour la colonne de K_2. On filtre donc les valeurs des colonnes de K_2 indépendamment les unes des autres pour un coût de 2^{32} par colonne. On réduit ainsi l'espace des K_2 possibles à un espace suffisamment petit pour faire une recherche exhaustive.

4. 0 est l'unique image de 0 par MC.

5. En utilisant la définition d'un corps, montrer (par l'absurde par exemple) qu'un corps est *intègre*, c'est à dire qu'il vérifie la propriété : $\forall x, y, \quad xy = 0 \implies [x = 0 \text{ ou } y = 0]$.

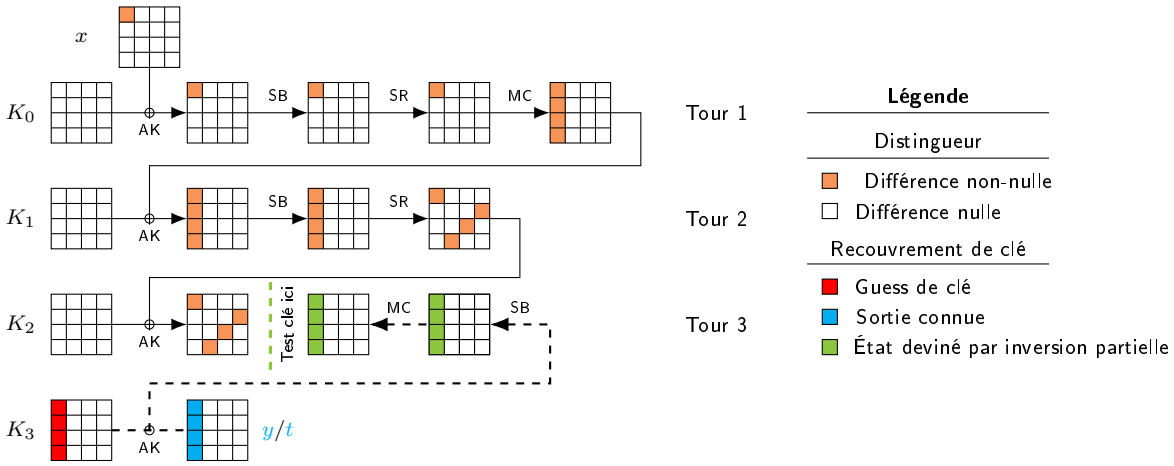

FIGURE 3 – Attaque sur 3 tours.

Remarque. *Pour connaître la puissance du filtre, il faut étudier ce qu'il se passe lorsque la clé candidate testée n'est pas la bonne et notamment quel est le nombre de faux-positifs attendus. Pour mener cette étude on* ***suppose*** *généralement que l'état par inversion partielle est totalement aléatoire lorsqu'une mauvaise clé est utilisée (c'est l'hypothèse appelée en anglais " wrong-key-randomization hypothesis"). Dans notre cas, cette hypothèse permet d'estimer à* $\frac{1}{2^{24}}$ *la proportion de faux-positifs parmi les mauvaises clés (A votre avis, pourquoi ?).*
Enfin, une fois la puissance du filtre connue, on contrebalance celle-ci en augmentant le nombre de paires clair/chiffré utilisées, comme expliqué avec la "première possibilité d'amélioration" de la question 3.

6. Montrer qu'on peut casser AES réduit à 4 tours.

Le principe est le même que pour l'attaque sur 3 tours : en utilisant des clairs choisis, on trouve une propriété qui distingue 2.5 tours d'AES d'une bijection aléatoire. Ce distingueur nous permet alors de retrouver la dernière clé de tour en filtrant certaines de ces parties indépendamment.

En revanche, on n'utilise plus 2 clairs choisis mais 256 : $\{(x, 0, \cdots, 0), x \in \{0,1\}^8\}$, c'est à dire l'ensemble des messages nuls partout sauf sur le premier octet qui prend les 256 valeurs possibles.

On étudie alors l'évolution de cet ensemble couche après couche.

- Pour l'ajout de clé, on observe sur un octet, même sans connaître K, que $x \mapsto x + K$ est une bijection donc si l'entrée prend toutes les valeurs possibles, alors la sortie aussi. Et si l'entrée prend une seule valeur alors la sortie prend une seule valeur.
- Il en va de même pour la boîte S qui est bijective.
- SR ne fait que réorganiser les octets.
- Enfin, pour MC on observe que les coordonnées ont pour forme

$$(e_1, e_2, e_3, e_4) \mapsto c_1e_1 + c_2e_2 + c_3e_3 + c_4e_4.$$

Si e_2, e_3, e_4 sont constants, on retrouve alors une fonction de la forme $e_1 \mapsto c_1 e_1 + c_5$ qui est une bijection (car c_1 est non nul donc inversible dans le corps à 256 éléments). Ainsi lorsque e_1 prend toutes les valeurs et que e_2, e_3, e_4 sont constants, chacun des octets en sortie prend une et une seule fois chaque valeur possible.

On obtient alors le distingueur de la figure 4.

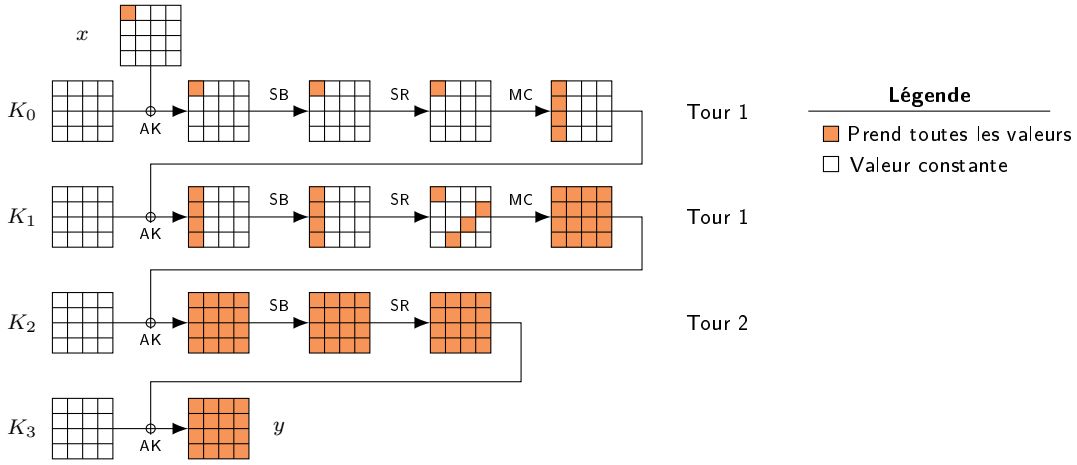

FIGURE 4 – Distingueur sur 2.5 tours.

Résumons Si l'on fait prendre toutes les valeurs au premier octet et que les autres sont constants, alors, après 2.5 tours, chacun des octets en sortie prendra une et une seule fois la même valeur.

L'attaque Comme pour 3 tours, on déchiffre partiellement, jusqu'à retrouver la valeur de la 1ère colonne pour chacun des 256 chiffrés. On vérifie si chaque octet prend une et une seule fois chaque valeur. Si c'est le cas, on a une clé candidate, sinon la clé est nécessairement fausse (*c.f.* figure 5).

Conclusion et remarques finales

- On remarquera que les attaques sur 1 et 2 tours d'AES sont en clair connu, tandis que les attaques sur 3 et 4 tours sont en clair choisi.
- De manière générale, un distingueur sur un nombre de tours donné n amène (très souvent) à une attaque sur $n+1$ ou $n+2$ tours.
- Actuellement, on peut attaquer jusqu'à 7 tours d'AES. La complexité en temps est énorme (mais inférieure à 2^{128}, c'est donc bien une attaque) et la quantité de données nécessaire (environ 2^{100} clairs choisis) est elle aussi complètement hors de portée. Cette quantité de données correspond à tellement de Go qu'il est impensable de choisir et demander le chiffrement d'autant de données, surtout si la clef secrète est changée régulièrement. Les techniques pour ces attaques sont de type "Différentielle Impossible" ou Meet in the Middle.
- Concernant les attaques pratiques sur des versions réduites d'AES, c'est à dire avec une complexité acceptable pour un ordinateur actuel, on trouve par exemple la “subspace trail

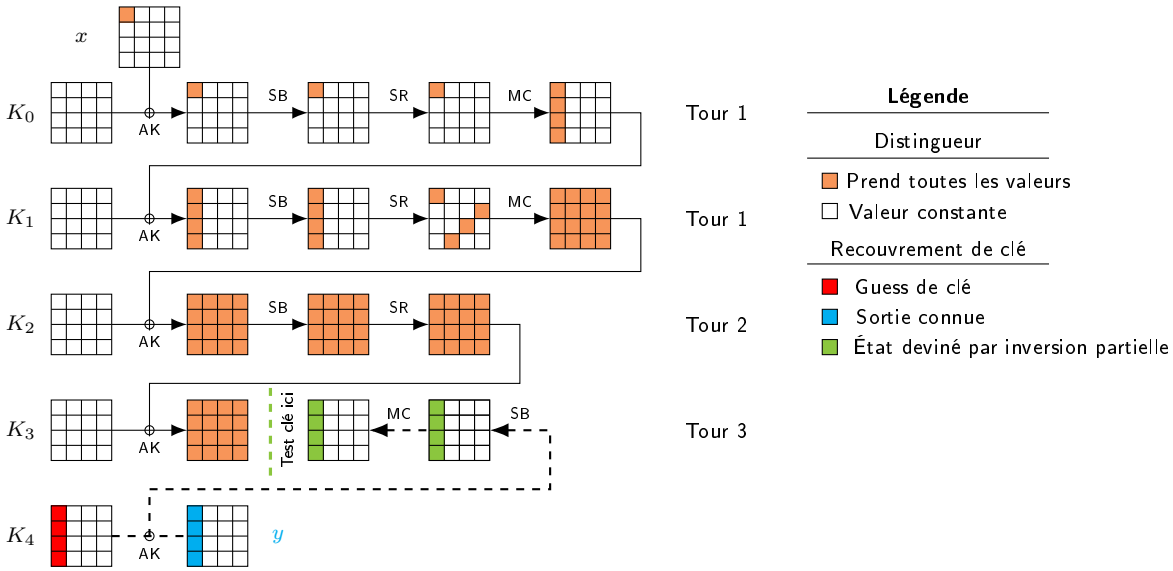

FIGURE 5 – Attaque sur 4 tours.

cryptanalysis"[6] ou encore l'"exchange attack"[7].

— Pour assurer la sécurité des chiffrements symétriques, on n'a, actuellement, pas d'autre choix que de se mettre dans la peau de l'attaquant et de regarder à quel point on peut "casser" le chiffrement ou des versions réduites à moins de tours. Le reste des tours est considéré comme la marge de sécurité.

— La sécurité d'AES est actuellement toujours assurée car la marge de sécurité est considérée comme suffisante. Ceci étant, il est nécessaire de continuer l'effort d'analyse, en cherchant de nouvelles manières d'attaquer ou de nouveaux distingueurs pour estimer plus précisément la marge de sécurité.

6. https://tosc.iacr.org/index.php/ToSC/article/view/571/513
7. https://eprint.iacr.org/2019/652