Cryptanalyse de l'AES à 5 tours

Louis Coumau, Axel Durbet et Dhekra Mahmoud

Université de Bordeaux

26 février 2021

université de BORDEAUX

Table des matières

université de BORDEAUX

Quelques informations sur l'AES

Quelques informations sur l'AES

- Type de chiffrement : Symétrique
- Auteurs : Joan Daemen et Vincent Rijmen
- Année de sortie : 1997
- Gagnant du concours du NIST en 2000

Propriétés intéressantes de l'AES

Si le chiffrement de Rijndael est si important, c'est qu'il possède de nombreuses de propriétés intéressantes :

- Résistance à toutes les attaques connues à l'époque.
- Il est possible d'implémenter l'AES aussi bien sous forme logicielle que matérielle (câblé).
- Rapidité du code sur la plus grande variété de plates-formes (logicielles et matérielles) possible.
- Simplicité dans la conception.
- Besoins en ressources et mémoire très faibles.

Les fonctions qui composent l'AES

Les fonctions majeures

L'AES est un chiffrement par bloc composé de quatre fonctions majeures (dans $\mathbb{F}_{256}$) :

- SubBytes (S-box ou SB)
- ShiftRows (SR)
- MixColumns (MC)
- AddRoundKey (AK)

SubBytes(S-box ou SB)

`SubBytes` peut s'écrire comme une table de substitution.

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Figure 1 – $SB(\texttt{0xed}) = \texttt{0x42}$

université de BORDEAUX

ISubBytes(IS-box ou ISB)

ISubBytes est l'inverse de SubBytes .

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Figure 2 – $SB(0x42) = 0xed$

université de BORDEAUX

ShiftRows (SR)

Le `ShiftRows` est une permutation circulaire vers la gauche aux lignes du tableau, respectivement de 0, 1, 2, 3 cases comme illustré ci-dessous :

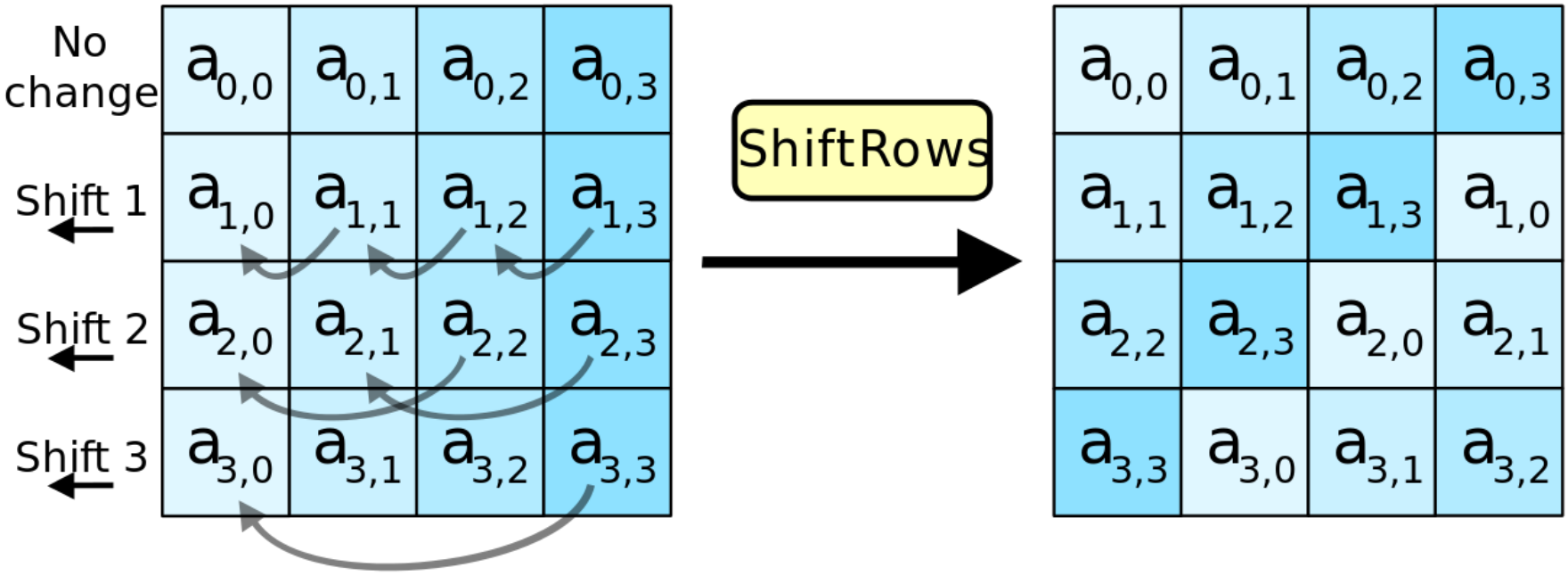


Figure 3 – Fonctionnement du `ShiftRows`

IShiftRows (ISR)

L'`IShiftRows` est une permutation circulaire vers la droite aux lignes du tableau, respectivement de 0, 1, 2, 3 cases. C'est l'inverse du `ShiftRows` .

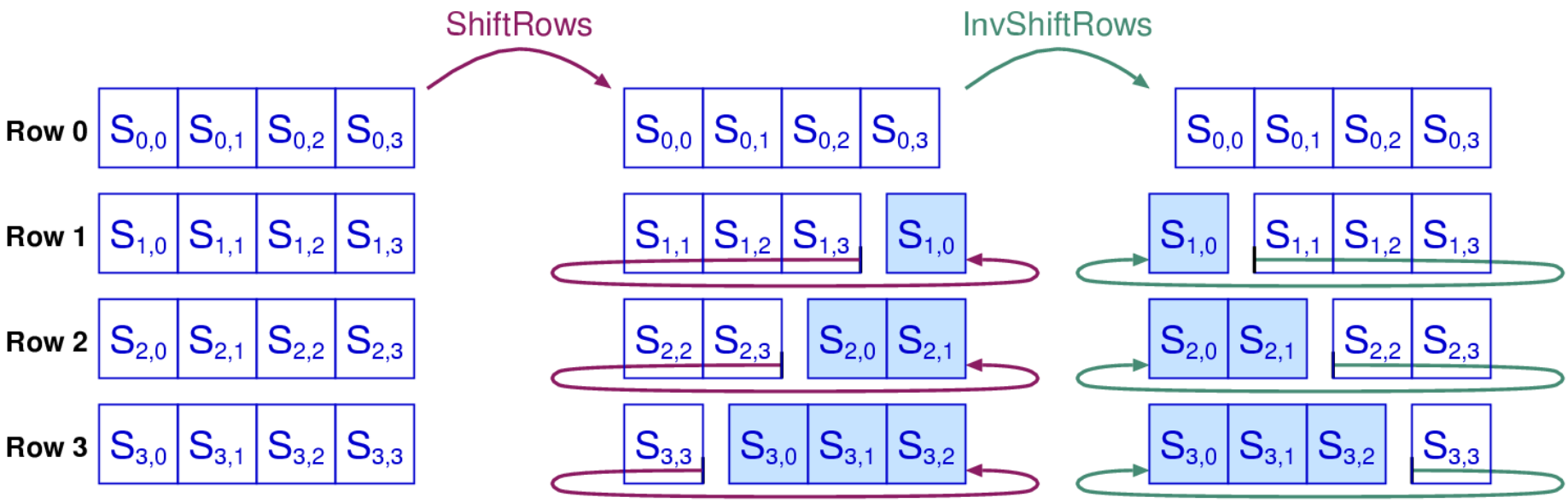


Figure 4 – Fonctionnement de `IShiftRows`

MixColumns (MC)

Le `MixColumns` est un produit matriciel à coefficient dans $\mathbb{F}_{256}$. Soit la matrice A l'entrée du `MixColumns` et la matrice B le résultat de la fonction. Alors $M \times A = B$.

Avec $M = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$

IMixColumns (IMC)

Avec les même notations que précédemment, l'`IMixColumns` se calcule avec le produit matriciel $M^{-1} \times B = A$.

Avec : $M^{-1} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix}$

AddRoundKey (AK)

`AddRoundKey` est une addition bit à bit ($\oplus$ ou XOR) de la clé de tour K_i, case par case.

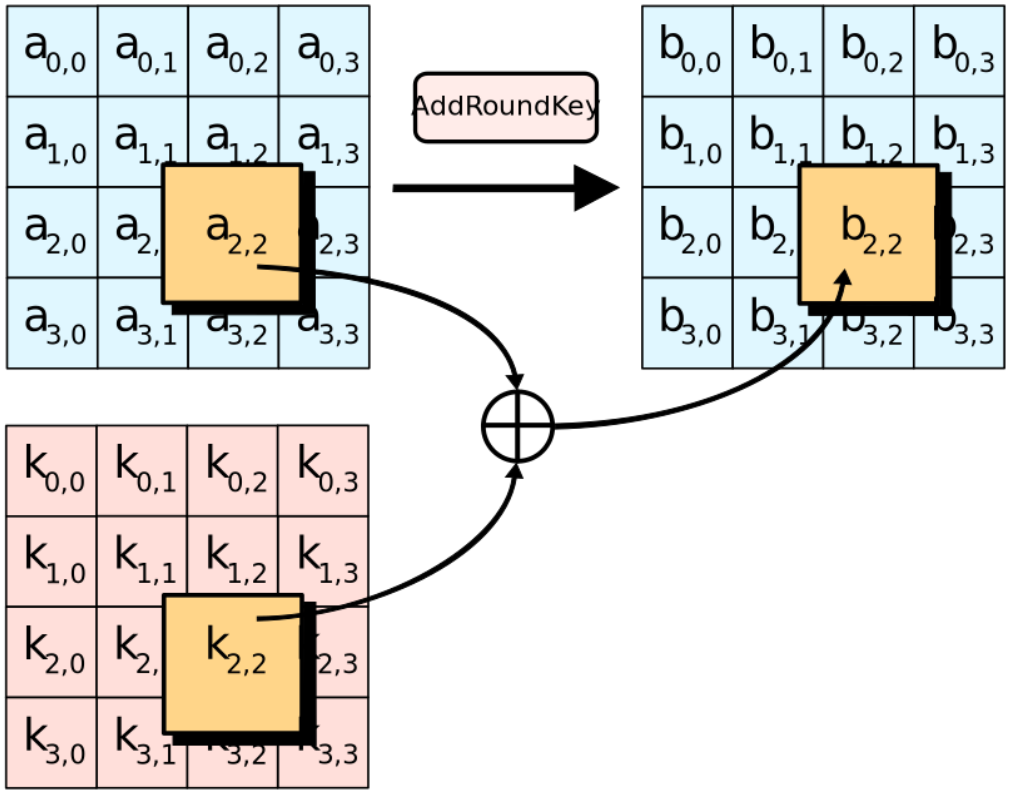


Figure 5 – `AddRoundKey`

Exemple d'un tour d'AES

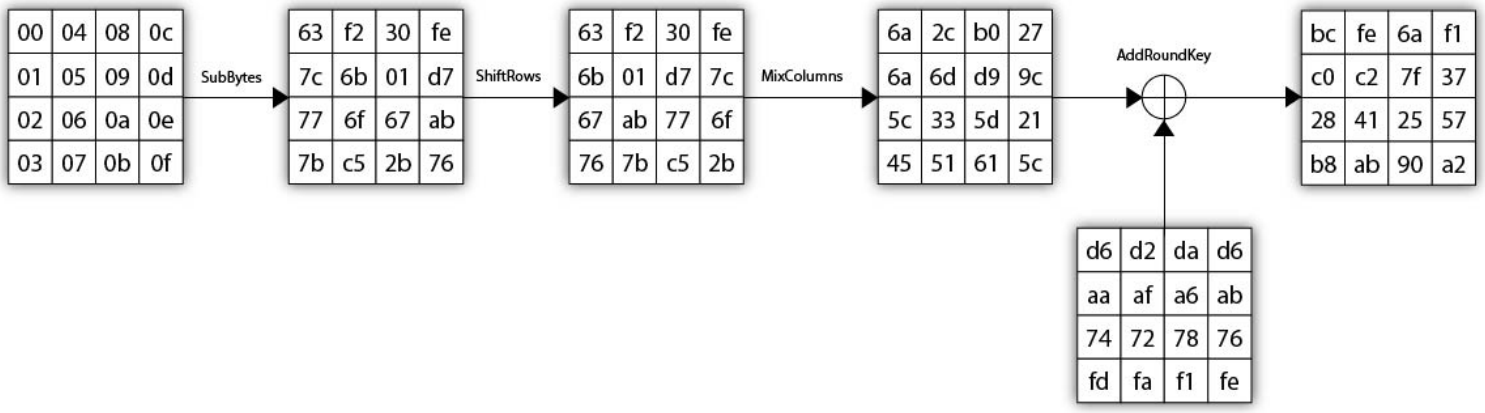
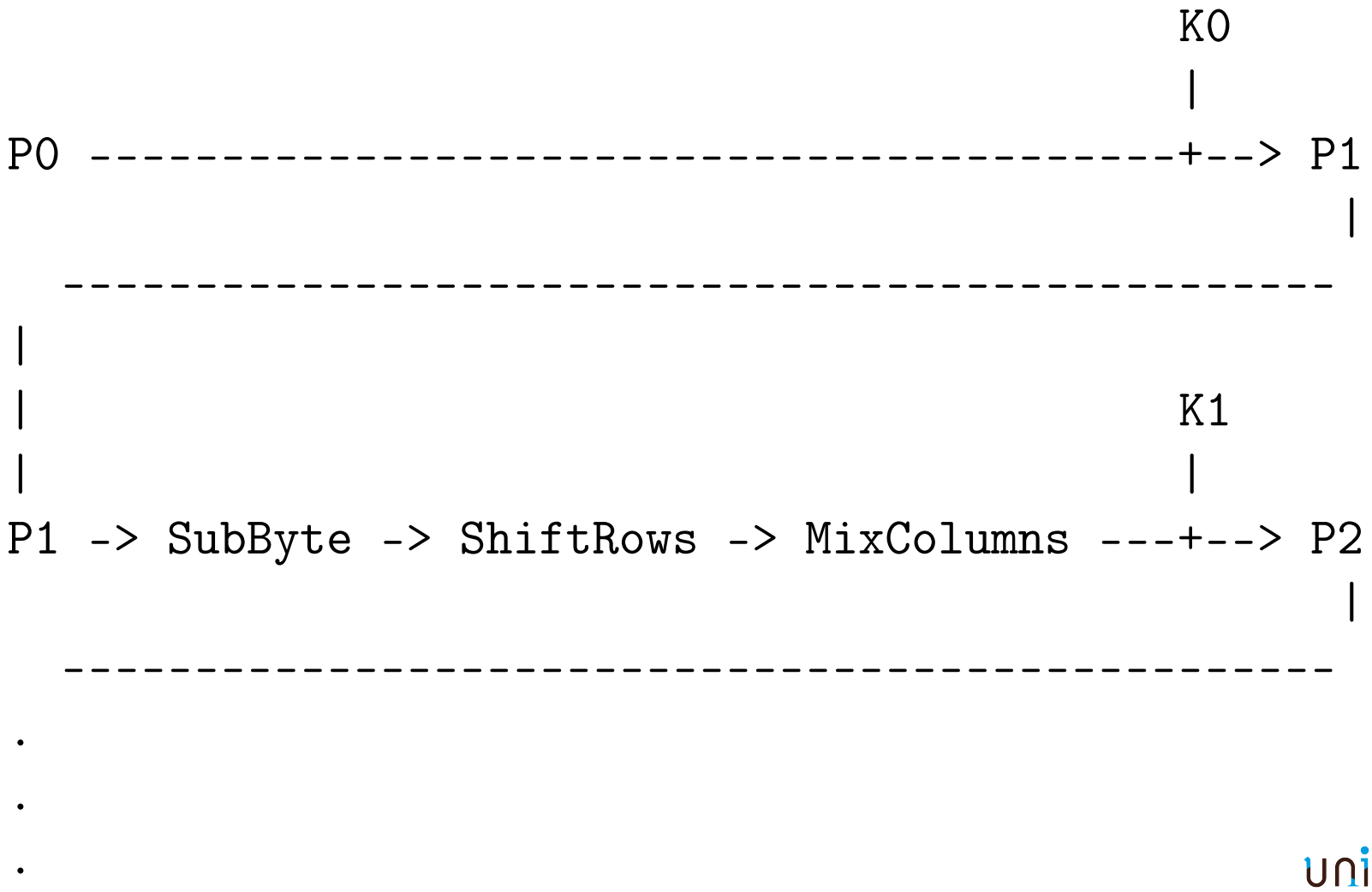
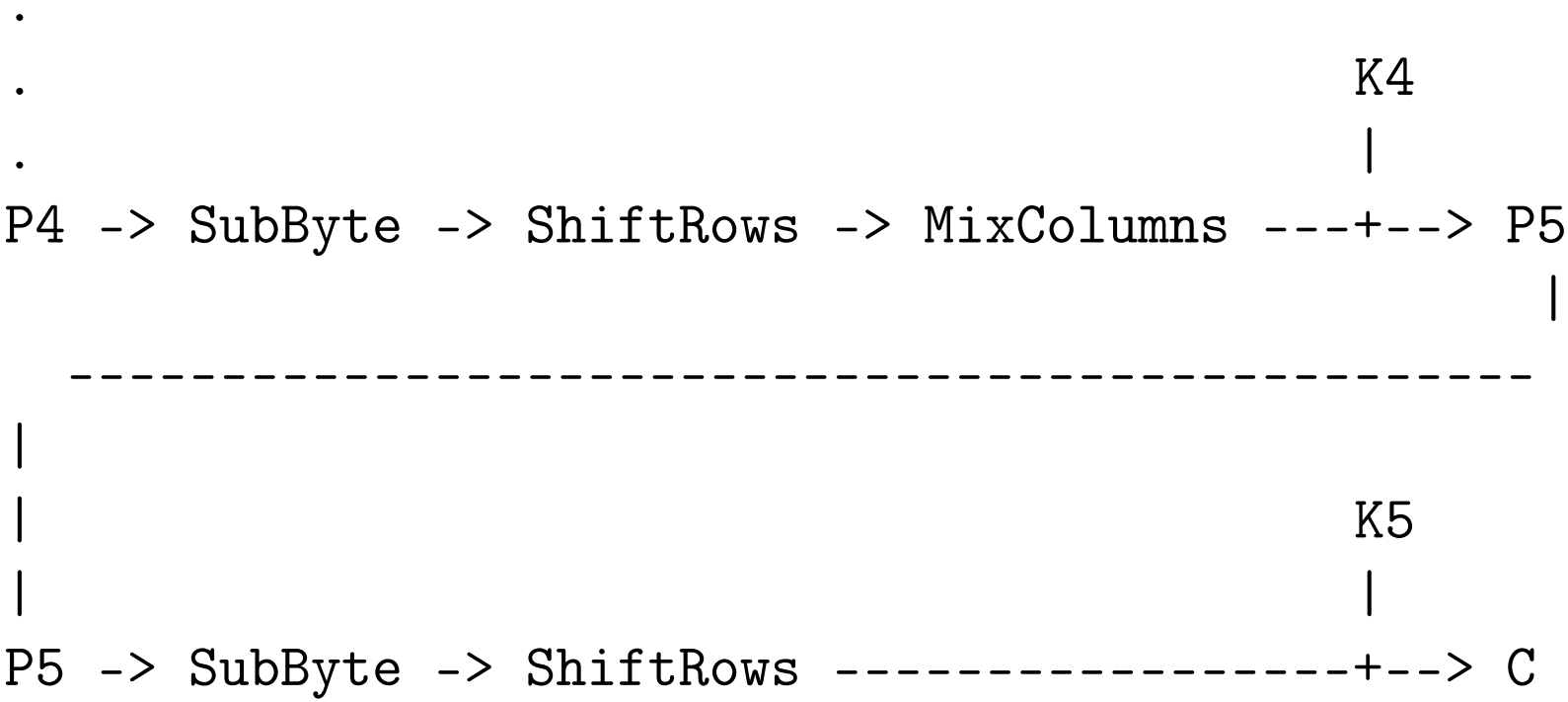


Figure 6 – Un tour d'AES

Exemple d'un tour d'AES



Exemple Sur 5 tours



Remarques sur le chiffrement

Remarques sur le chiffrement à 5 tours :

- On a $n + 1$ clés pour n tours.
- La suite de sous-clé est générée en utilisant une fonction inversible.
- Une seule sous-clé permet de retrouver les autres.

L’attaque carré

Le chiffrement carré

Le chiffrement carré

- Square est une construction itérative de chiffrement de 16 blocs de 128 bits.
- La fonction de tour correspondante est constituée de quatre transformations élémentaires.

Les fonctions élémentaires

- `Linear Transformation` :
- `Nonlinear Transformation`
- `Byte Permutation`
- `Bitwise RoundKey Addition`

Remarques

- L'attaque carré fonctionne uniquement sur les chiffrements qui possèdent une structure dite carrée :
  - A priori, les structures que l'on manipulent se présente sous la forme d'une matrice carrée.
  - Les opérations élémentaires de l'algorithme de chiffrement sont inspirées de celles du chiffrement carré.
- Cette attaque reste donc valable pour l'AES car il hérite de nombreuses propriétés du chiffrement Square.

Les Λ-sets

Définition

Un Λ-set est défini comme suit :

$$\forall x, y \in \Lambda : \begin{cases} x_{i,j} \neq y_{i,j} & \text{si } (i,j) \in \lambda \\ x_{i,j} = y_{i,j} & \text{sinon} \end{cases}$$

Voyons un exemple simple d'un Λ-set avec un seul octet actif (l'octet d'indice $(0,0)$) :

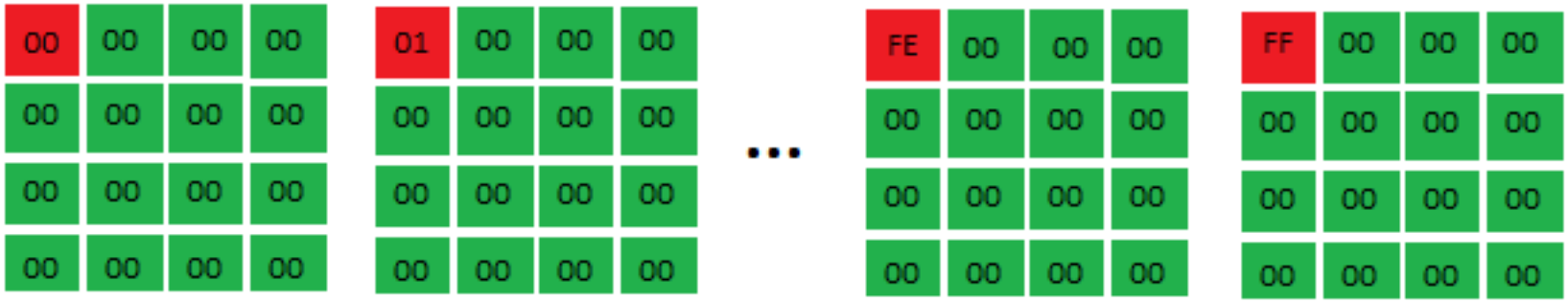


Figure 7 – Exemple d'un Λ-set

Propriétés

Les propriétés des Λ-sets :

- L'application de `SubBytes` ou `AddRoundKey` sur un Λ-set donne généralement un autre Λ-set avec les indices des octets actifs inchangés.
- L'application de `MixColumns` n'aboutit pas nécessairement à un Λ-set. Mais, une colonne d'entrée avec un seul octet actif donne une colonne de sortie avec les quatre octets actifs.
- L'application de `ShiftRows` donne un autre Λ-set avec les indices des octets actifs changés.
- Les valeurs des octets actifs sont équilibrées (balanced).

Le principe de l'attaque

Le principe de l'attaque

- Il s'agit de travailler avec un ensemble de clairs qui constituent un Λ-set pour en exploiter les propriétés.
- La structure de notre ensemble (Λ-set) est conservée jusqu'à l'entrée de la *MC* du troisième tour.
- A la sortie du troisième tour, tous les octets sont équilibrés. Cette propriété va être détruite en passant dans la S-box du quatrième tour.

Le principe de l'attaque

- En faisant une supposition sur la dernière clef de tour, on doit remonter jusqu'à l'entrée du quatrième tour pour la vérifier en s'assurant de la propriété d'équilibre des octets.

L'attaque Square sur l'AES 4 tours

Le principe de l'attaque

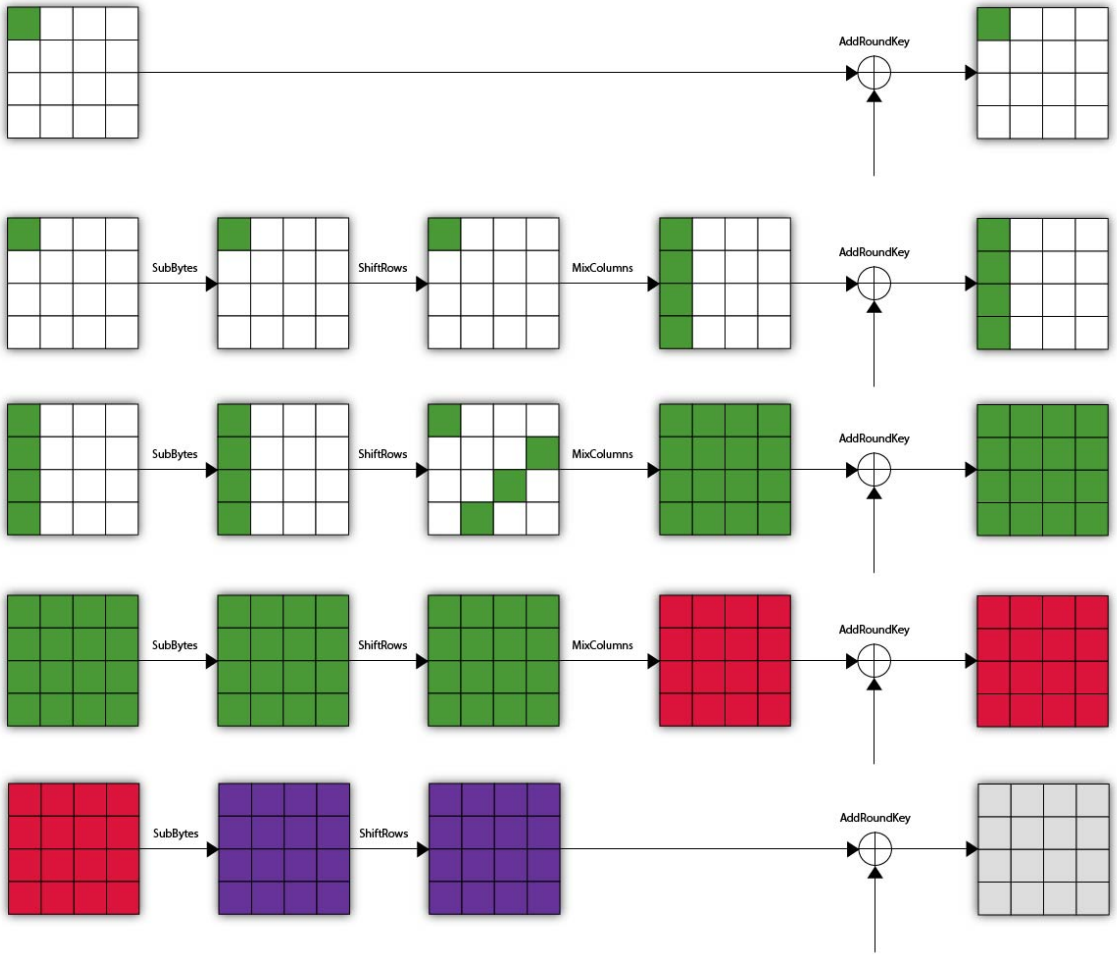


Figure 8 – Illustration de l'attaque Square sur 4 tours.

université de BORDEAUX

Pseudo-code de l'attaque

Algorithme 3.1.2.1. [Attaque carrée sur l'AES 4 tours]

Entrée : 2 Λ-sets (3.1.2.1) différents chiffrés et G^{-1} (3.1.2)

Sortie : La clé

1. *On applique* `IShiftRows` *à tous nos chiffrés.*
2. *Pour chaque octet de la dernière clé, faire :*
   (a) *Assigner une valeur à l'octet de la clé.*
   (b) *Pour chaque chaque chiffré du Λ-set 1 faire la somme (de XOR) suivante : $\sum_{i=1}^{256} ISbox(a_{i,j} \oplus K_j)$ [2] avec $a_{i,j}$ le j-ème octet[3] du i-ème chiffré du Λ-set et K_j le j-ème octet de la clé.*
   (c) *On réitère l'opération sur chaque chiffré du Λ-set 2.*
   (d) *Vérifier si les deux sommes sont nulles.*
   — *Si oui : Passez a l'octet de clé suivant.*
   — *Si non : Changer la valeur de l'octet de la clé.*
3. *Remonter avec la sous-clé vers la clé initiale grâce à G^{-1}.*
4. *Retourner la clé.*

Extension d'un tour à la fin

Extension d'un tour à la fin

- On va d'abord faire une hypothèse sur 4 octets de la sous-clé 5 (la dernière).
- Remonter à la fin du tour 4 (l'avant dernier) et faire une hypothèse sur l'octet de la clé correspondant par les opérations inverses.
- On peut revenir au début du tour 4 et vérifier la propriété d'équilibre car on retrouve un Λ-set.
- Si elle est vérifié (la somme est nulle) alors nos 4 octets de la sous-clé 5 sont bons.

Pseudo-code

Entrée : 5 Λ-*sets (3.1.2.1) différents chiffrés et* G^{-1} *(3.1.2)*

Sortie : La clé K_0

1. *(Initialisation) On définit les ensembles des positions suivants :*
   (a) $I_0 = [0, 7, 10, 13]$, $I_1 = [1, 4, 11, 14]$, $I_2 = [2, 5, 8, 15]$, $I_3 = [3, 6, 9, 12]$ [4]
   (b) $J = [0, 2, 1, 3]$
2. *(Initialisation) On crée les clés* K_5 *et* K_4 *avec tous les octets à* 0
3. *Pour* $n \in [0, 1, 2, 3]$
4. *Pour toutes les valeurs possibles de* K_5 *aux indices* I_n *:*
5. *Pour toutes les valeurs possibles de* K_4 *à l'indices* J_n *:*
6. $b \longleftarrow 0$
7. *Pour tous les* Λ*-sets :*
8. *Pour tous les chiffrés* C_i *:*
9. $C'_i \longleftarrow$ *On remonte le tour 5 de* C_i *avec la clé* K_5 [5]
10. $C'_i \longleftarrow$ *On remonte le tour 4 de* C'_i *avec la clé* K_4
11. $b \longleftarrow b \oplus C'_i[J_n]$
12. *Si* $b = 0$ *:*
13. *Alors on a trouvé les bonnes valeurs de* K_5
14. *aux indices* I_n.
15. *Arrêter ou passer à la valeur de n suivante.*
16. **À cette étape, on a tous les octets de* K_5 ***
17. $K_0 \longleftarrow$ *On remonte la clé* K_5 *avec* G^{-1} *5 fois*
18. *On retourne* K_0

Extension d'un tour au début

Extension d'un tour au début

- On va d'abord modifier nos clairs de telle sorte à avoir un Λ-set à l'entrée du MC du quatrième tour.
- On élimine, donc, un tour en appliquant un tour inverse à nos clairs.
- Nos clairs posséderaient 4 octets actifs sur la diagonale.
- On fait une hypothèse sur la diagonale de la clef K_0 et on effectue la même attaque de l'AES à 4 tours.

Extension d'un tour au début

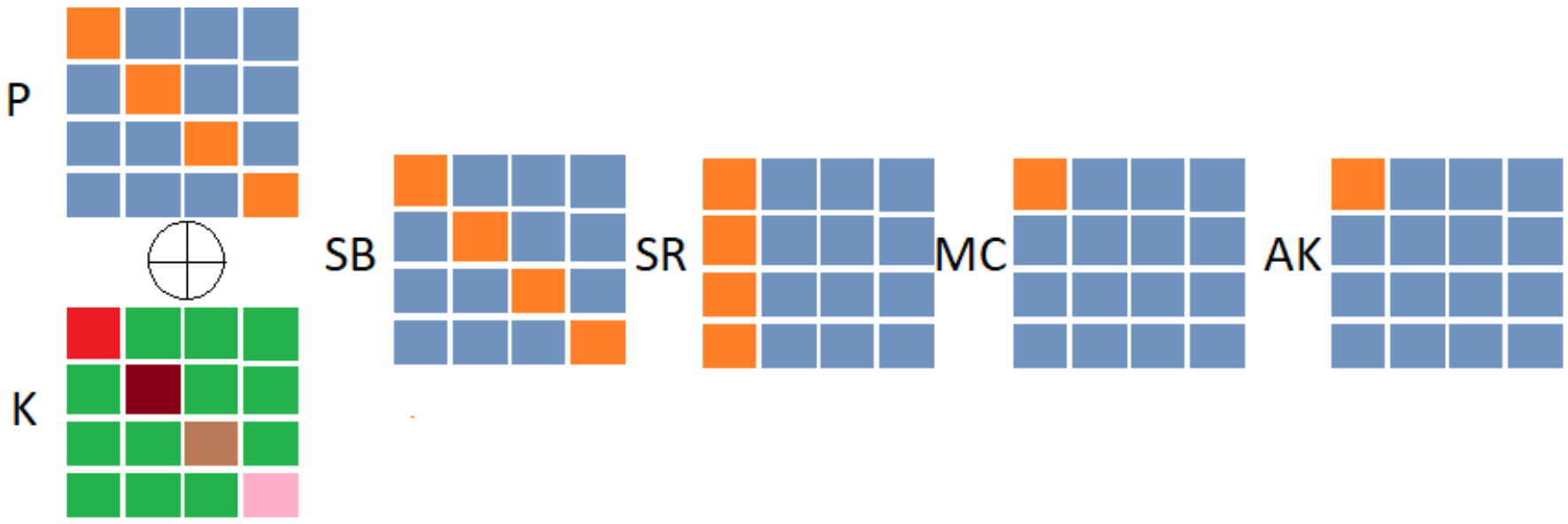


Figure 9 – Évolution de nos clairs au cours du premier tour.

Pseudo-code

Entrée : 4 Λ-*sets (3.1.2.1) différents possédant un seul octet actif sur la première colonne et* G^{-1} *(3.1.2)*

Sortie : La clé secrète

1. *On applique respectivement l'inverse des fonctions* **`MixColumns`** , **`ShiftRows`** *et* **`SubBytes`** *sur chaque élément des* Λ*-sets.*
2. *On initialise les octets de nos clefs* K_0 *et* K_5 *à 0.*
3. *Pour chaque valeur possible des octets* 0, 5, 10 *et* 15 *de* K_0 *faire :*
   (a) *On additionne bit à bit les éléments des* Λ*-sets avec* K_0*. Puis, on les chiffre.*
   (b) *Pour chaque valeur possible* k *de l'octet* 0 *de la clef* K_5 *faire :*
      i. *Pour chaque chiffré* $c_{i,j}$ *du* Λ*-set* i*, on calcule la somme suivante :* $b_i = \sum_{j=1}^{256} ISbox(c_{i,j,0} \oplus k)$ *telle que* $c_{i,j,0}$ *correspond à l'octet* 0 *du* j*-ème chiffré du* i*-ème* Λ*-set.*
      ii. *Si les quatre* b_i *sont nuls alors, on obtient les bonnes valeurs pour les quatre octets de* K_0 *ainsi que la bonne valeur de l'octet* 0 *de* K_5 *et on sort de la grande boucle.*
      iii. *Sinon, on passe à l'octet suivant.*
4. *On récupère les* 15 *autres octets de la clef* K_5 *en suivant la même démarche de l'attaque réalisée sur l'***`AES`** *à* 4 *tours (3.1.2.3).*
5. *On retourne la clé.*

L’attaque yoyo

Définitions

Définition (Zero difference pattern)

Le vecteur différence à 0 (zero difference pattern) :
Soit $\alpha \in \mathbb{F}_q^n$, *le vecteur différence à 0* $\nu(\alpha) = (z_0, z_1, z_2, z_3, ..., z_{n-1})$ *retourne* $z_i = 1$ *si la i-ème coordonnée de* α *est nulle et 0 sinon.*
Une autre façon de l'écrire :

$$z_i \mapsto \begin{cases} 1 & \text{si } \alpha_i = 0 \\ 0 & \text{sinon} \end{cases}$$

Par exemple : $\nu((17, 34, 98, 0, 23, 11, 0)) = (0, 0, 0, 1, 0, 0, 1)$.

Définitions

Définition (S)

$S = SB \circ MC \circ SB$. S est une super S-box qui agit sur les colonnes. Elle possède toutes les propriété d'une boite S standard.

Définition (Q')

$Q' = SR \circ MC \circ SB$.

Définition (L)

$L = SR \circ MC \circ SR$ est une application linéaire.

Définition (La fonction ρ)

Pour un vecteur $\omega \in \mathbb{F}_2^n$ et deux états α et $\beta \in \mathbb{F}_q^n$, on définit un nouvel état, $\rho^\omega(\alpha, \beta)$ tel que la i-ème coordonnée de ce nouvel état est :

$$\rho^\omega(\alpha, \beta)_i = (\alpha_i \times \omega_i) \oplus (\beta_i \times (\omega_i \oplus 1))$$

Une autre façon de le voir :

$$\rho^\omega(\alpha, \beta)_i \mapsto \begin{cases} \alpha_i & \text{si } \nu_i = 1 \\ \beta_i & \text{sinon} \end{cases}$$

Par exemple, si : $\omega = (0, 1, 1)$, $\alpha = (11, 22, 33)$ et $\beta = (88, 77, 66)$ alors : $\rho^\omega(\alpha, \beta) = (88, 22, 33)$

Comment voir l'AES

On fait une analyse différentielle donc on ne prend pas en compte les AK. Si on pose un tour d'AES comme étant : $MC \circ SR \circ SB$ et le premier tour étant $MC \circ SB \circ SR$.
Alors on a :

$$R^5 = SR \circ S \circ L \circ S \circ Q' \circ SR$$

Et :

$$R^{5'} = S \circ L \circ S \circ Q'$$

Théorème fondamental

Théorème

Soit S une S-box et L une application linéaire qui agit sur des vecteurs de tailles n.
Soit α et $\beta \in \mathbb{F}_q^n$, on pose : $A = \rho^\omega(\alpha, \beta)$ et $B = \rho^\omega(\beta, \alpha)$.

$$\nu(S \circ L \circ S(\alpha) \oplus S \circ L \circ S(\beta)) = \nu(S \circ L \circ S(A) \oplus S \circ L \circ S(B))$$

SimpleSwap

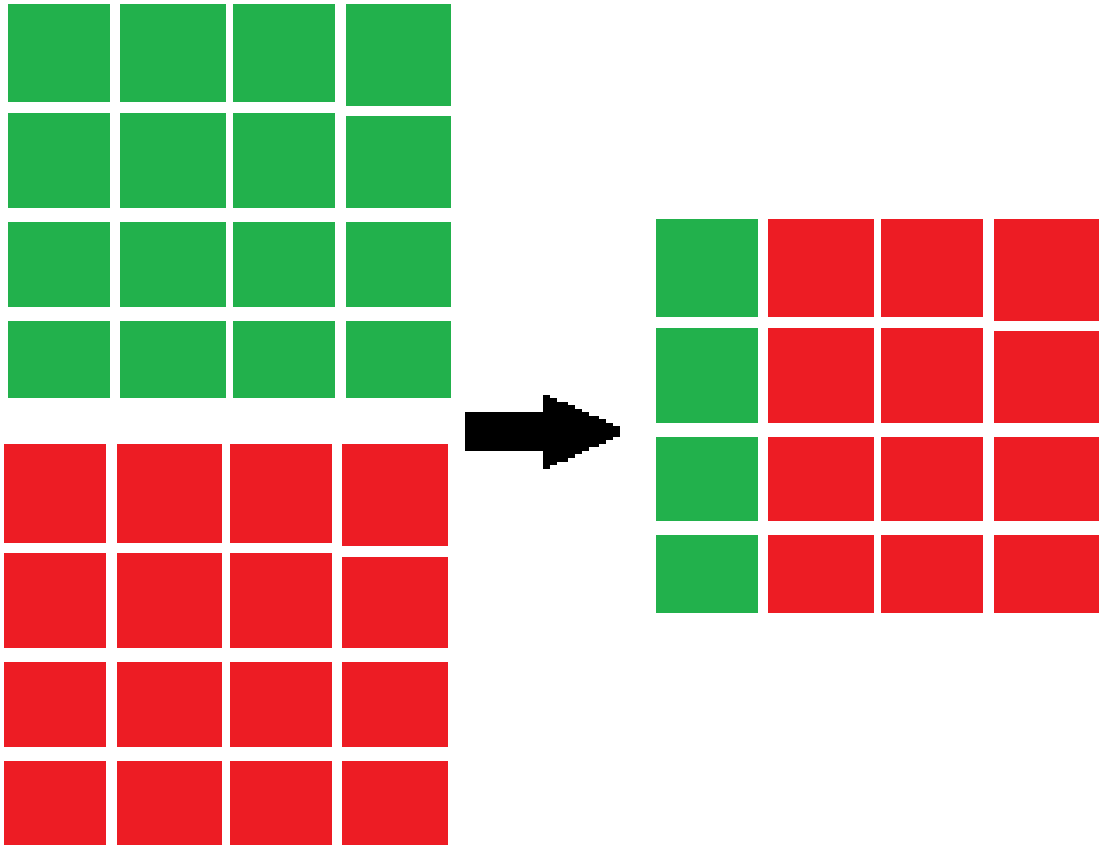


Figure 10 – Fonctionnement du `SimpleSwap`

SimpleSwap

On définit le `SimpleSwap` (cas particulier de ρ) une fonction qui échange le premier mot de différent de chaque texte et retourne le texte échangé :

Entrée : Un couple de textes x_0 *et* x_1
Sortie : x'_0

1. $x'_0 = x_1$
2. *Pour* i *allant de* 0 *à* 3 *faire :*
3. $\quad$ *Si* $(x_0)_i \neq (x_1)_i$ *:*
4. $\quad\quad (x'_0)_i = (x_0)_i$
5. $\quad\quad$ *retourner* x'_0
6. *retourner* x'_0

Le principe de l'attaque

Le principe de l'attaque

- Soient P_0 et P_1 deux clairs choisis tels que :
  - Ils ont les mêmes colonnes 2, 3 et 4.
  - Ils ont comme première colonne $(0, i, 0, 0)$ et $(1, i \oplus 1, 0, 0)$ telle que i varie entre 0 et 255.
- Soit (k_0, k_1, k_2, k_3) la première colonne de K_0 "supposé". Tous les autres octets sont réduits à 0.

Le principe de l'attaque

- $R^5 = S \circ L \circ S \circ SR \circ MC \circ SB$
- Le "zero pattern" se préserve à travers $S \circ L \circ S$
- Il suffit alors de caractériser cette différence à 0 de la différentielle de P_0 et P_1 avant $S \circ L \circ S$.
- SR est une simple permutation circulaire fixée :
  - Caractériser cette différence à 0 avant SR ou après SR revient au même.
- On suivra donc la différence à 0 de la différentielle indiquée juste après $MC \circ SB$.

Le principe de l'attaque

- Après $MC \circ SB$, la différentielle à la troisième ligne de la première colonne, y, s'écrit sous la forme suivante :
  - $y = s(k_0) \oplus s(1 \oplus k_0) \oplus s(k_1 \oplus 1 \oplus i) \oplus s(i \oplus k_1)$
- y est nul si $i = k_0 \oplus k_1$ ou $i = k_0 \oplus k_1 \oplus 1$
- i parcourt toutes les valeurs possibles de 0 à 255 => On tombera nécessairement sur la bonne valeur de $k_0 \oplus k_1$.
- A priori, il suffit de faire une recherche exhaustive sur trois octets de la clef d'une même colonne et de déduire le quatrième grâce à $i = k_0 \oplus k_1$

Le principe de l'attaque

- Le vecteur différence à 0 de la première colonne à ce stade-là et dans le cas où $i = k_0 \oplus k_1$ est égal à $(0, 0, 1, 0)$
- En appliquant $S \circ L \circ S \circ SR$ et en créant une nouvelle paire de chiffrés A et B à partir des chiffrés α et β (en utilisant `SimpleSwap`), le vecteur différence à 0 serait conservé.
- Il suffit de vérifier, que pour toutes les paires de clairs créées en déchiffrant les chiffrées construits de cette manière-là, le vecteur différence à 0 de leur différentielle après $MC \circ SB$ est bien conservé.

Pseudo-code

Entré : $\mathcal{P}_0$, $\mathcal{P}_1$ [13]
Sortie : $D(K_0)$, *la diagonal de la clé* K_0

1. *Pour* i *de* 0 *à* 255 *:*
2. *On définit* S *une liste de couple*
3. *Pour* j *de* 0 *à* 4 *:*
4. $p_0 \longleftarrow \mathcal{P}_0[i], p_1 \longleftarrow \mathcal{P}_1[i]$
5. $c_1 \longleftarrow Encryption(p_0), c_0 \longleftarrow Encryption(p_1)$
6. $c_1{}' \longleftarrow SimpleSwap(c_1, c_0), c_1{}' \longleftarrow SimpleSwap(c_1, c_1)$
7. $p_0{}' \longleftarrow Decryption(c_1{}'), p_1{}' \longleftarrow Decryption(c_1{}')$
8. $p_0 \longleftarrow SimpleSwap(p_0{}', p_1{}'), p_1 \longleftarrow SimpleSwap(p_1{}', p_0{}')$
9. *On ajoute le couple* (p_0, p_1) *à* S
10. *On définit* K *une clé*
11. *Pour* $k0$ *de* 0 *à* 255
12. *Pour* $k2$ *de* 0 *à* 255
13. *Pour* $k3$ *de* 0 *à* 255
14. $K[0] = k0,\ K[5] = (k0 \oplus i),\ K[10] = k2,\ K[15] = k3$
15. *Pour tous les couples* (p_0, p_1) *de* S
16. $K' \longleftarrow \texttt{ShiftRows}(K)$
17. $p_0{}' \longleftarrow \texttt{AddRoundKey}(p_0, K'), p_1{}' \longleftarrow \texttt{AddRoundKey}(p_1, K')$
18. $p_0{}' \longleftarrow \texttt{SubBytes}(p_0{}'), p_1{}' \longleftarrow \texttt{SubBytes}(p_1{}')$
19. $p_0{}' \longleftarrow \texttt{MixColumns}(p_0{}'), p_1{}' \longleftarrow \texttt{MixColumns}(p_0{}')$
20. *Si* $p_0{}' \oplus p_1{}' = 0$
21. **On a trouvé la diagonal de* $K0$***
22. *On retourne* K

université de BORDEAUX

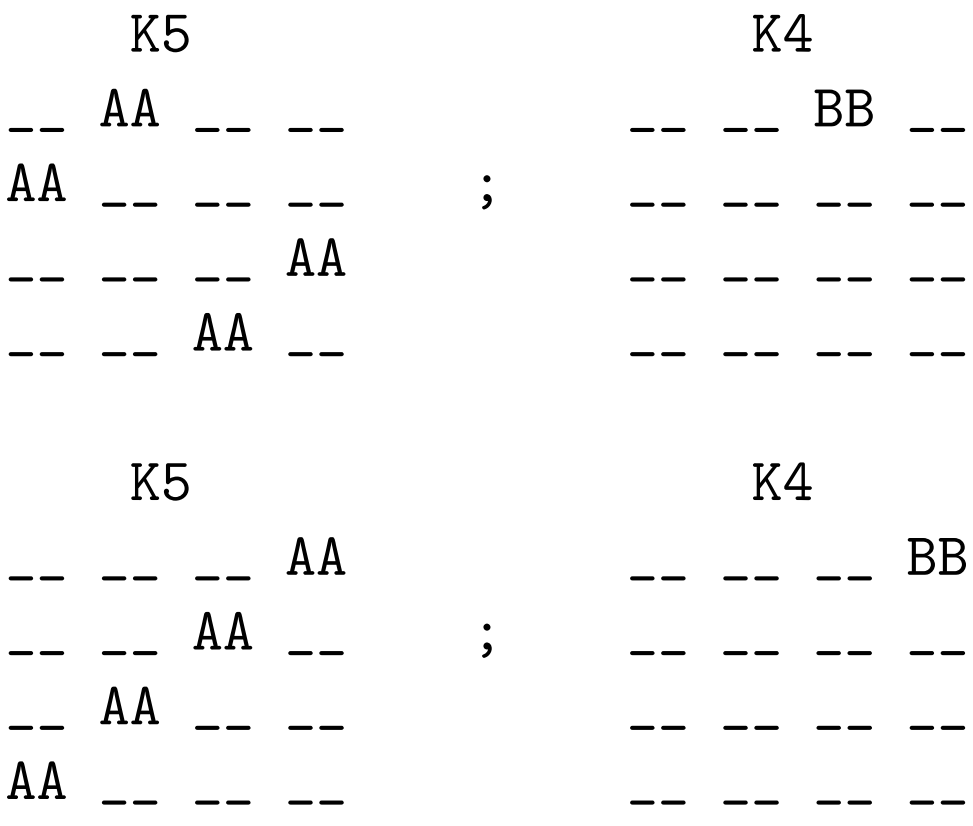
Complexité

Complexité de l'attaque carré type 1

```
    K5                    K4
AA __ __ __           BB __ __ __
__ __ __ AA     ;     __ __ __ __
__ __ AA __           __ __ __ __
__ AA __ __           __ __ __ __
```

```
    K5                    K4
__ __ AA __           __ BB __ __
__ AA __ __     ;     __ __ __ __
AA __ __ AA           __ __ __ __
__ __ AA __           __ __ __ __
```

Complexité de l'attaque carré type 1



Complexité de l'attaque carré type 1

Pour déterminer une seule diagonale de K_5 :

- 4 octets de la clé K_5
- 1 octet de la clé K_4

$$(2^8)^4 \times 2^8 = 2^{40}$$

Complexité :

$$4 \times 2^{40} = 2^{42}$$

De ce fait, pour ce nombre de tours, elle est plus efficace qu'une recherche exhaustive qui vaudrait, elle :

$$(2^8)^{16} = 2^{128}$$

Complexité de l'attaque carré type 2

```
    K0                          K5
AA __ __ __                 BB __ __ __
__ AA __ __      ;          __ __ __ __
__ __ AA __                 __ __ __ __
__ __ __ AA                 __ __ __ __
```

Pour déterminer la diagonale de K_0 :

- 4 octets de la clé K_0
- 1 octet de la clé K_5

Complexité :

$$(2^8)^4 \times 2^8 = 2^{40}$$

Complexité de l'attaque yoyo

Pour i de 0 à 255

```
     K0
k0 __    __ __
__ k0+i  __ __
__ __    k2 __
__ __    __ k3
```

On génère k_0, k_2 et k_3 i fois. On en déduit donc ce produit :

$$(2^8)^4 = 2^{32}$$

Résultat de l'attaque carré type 1

OTF	N-L	S/E	T
2×4	2	S	5,67
2×4	3	S	8,56
2×4	4	S	11,39
2×4	5	S	14,26
3×4	5	S	3431,75
4×4	5	S	10 jours* [1]
5×4	5	S	7 ans*

1. Les temps suivit d'un * sont estimés.

Résultat de l'attaque carré type 2

OTF	N-L	S/E	T
2	4	S	0,06
3	4	S	13,18
4	4	S	3434 $\approx$ 1h
5	4	S	256h $\approx$ 10j*

Résultat de l'attaque Yoyo

OTF	S/E	T
1	S	0,00s
2	S	0,08s
3	S	24,16s

Conclusion

Question

Merci de votre attention !