TP1 : AES et Cryptanalyse Différentielle

Christina Boura
christina.boura@irif.fr

10 décembre 2024

1 Boîte-S de l'AES

La boîte-S de l'AES est une permutation non-linéaire sur $\mathbb{F}_2^8$. Sa représentation sous forme de table est donnée dans la table 1. Par exemple, l'image de `0x1f` par la boîte-S est `0xc0`. Cette valeur se trouve à l'intersection de la ligne `0x10` et de la colonne `0x0f`.

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

TABLE 1 – Représentation en table de la boîte-S de l'AES

1. Implémentez le calcul de la table de distribution des différences (DDT) pour une boîte-S de taille arbitraire et appliquez-le à la boîte-S de l'AES. Quelle est l'uniformité différentielle de S ? Quelles observations peut-on faire sur la répartition des valeurs au sein de cette table ?
2. Mathématiquement, une boîte-S peut être vue comme une fonction booléenne vectorielle $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$, qui se décompose en m coordonnées $s_0, s_1, \ldots, s_{m-1}$, où chaque coordonnée est une fonction booléenne des n variables d'entrée. Cela s'exprime sous la forme :

$$S(x) = (s_0(x), s_1(x), \ldots, s_{m-1}(x)), \quad x \in \mathbb{F}_2^n.$$

Le tableau ci-dessous illustre une boîte-S $S : \mathbb{F}_2^3 \to \mathbb{F}_2^3$, générée aléatoirement.

x	0	1	2	3	4	5	6	7
$s_0(x)$	1	1	0	1	1	0	0	0
$s_1(x)$	1	0	1	1	0	0	1	0
$s_2(x)$	0	0	1	1	1	0	0	1
$S(x)$	3	1	6	7	0	5	2	4

Pour analyser les propriétés mathématiques d'une boîte-S donnée, la représentation sous forme de table n'est pas toujours appropriée, car elle ne fournit aucune information sur la structure algébrique de la fonction. Pour cette raison, il est parfois préférable de représenter une boîte-S en utilisant la *forme normale algébrique* (ANF) de ses fonctions coordonnées.

Définition 1 (Forme algébrique normale)**.** *Soit f une fonction booléenne en n variables. La forme algébrique normale (ANF) de f est l'unique polynôme multivarié suivant :*

$$f(x) = \bigoplus_{u \in \mathbb{F}_2^n} c_f(u) x^u,$$

où $x^u = x_0^{u_0} x_1^{u_1} \cdots x_{n-1}^{u_{n-1}}$ et $c_f(u)$ peut être calculé à l'aide de la transformée de Möbius comme suit :

$$c_f(u) = \bigoplus_{v \in \mathbb{F}_2^n,\, v \preceq u} f(v) \in \mathbb{F}_2,$$

avec $v \preceq u$ si et seulement si $(u_i = 0 \implies v_i = 0)$ pour tout $i = 0, \ldots, n-1$.

Exemple La forme normale algébrique (ANF) des trois coordonnées s_0, s_1, s_2 de la boîte-S $[3, 1, 6, 7, 0, 5, 2, 4]$ est donnée ci-dessous. La variable x est un vecteur de 3 bits (x_0, x_1, x_2) :

$$S(x) = \begin{cases} s_0(x) = x_0x_1 + x_0x_2 + x_1x_2 + x_1 + x_2 + 1, \\ s_1(x) = x_0x_1 + x_0x_2 + x_0 + x_1x_2 + x_2 + 1, \\ s_2(x) = x_0x_2 + x_1x_2 + x_1. \end{cases}$$

Le module `sage.crypto.sbox.SBox` de `SageMath` est très pratique pour calculer diverses propriétés cryptographiques des boîtes-S. Toutes les fonctionnalités et des exemples d'utilisation peuvent être trouvés ici :

https://doc.sagemath.org/html/en/reference/cryptography/sage/crypto/sbox.html

Utiliser ce module pour calculer la forme algébrique normale (ANF) de chacune des 8 coordonnées de la boîte-S de l'AES. Puisque chacune des coordonnées est une fonction booléenne, le module suivant permettant de calculer diverses propriétés des fonctions booléennes vous sera utile :

https://doc.sagemath.org/html/en/reference/cryptography/sage/crypto/boolean_function.html

2 Résistance de l'AES à la cryptanalyse différentielle

Le but de cette partie est de démontrer, par deux approches distinctes, que l'AES est résistant à la cryptanalyse différentielle. Plus précisément, nous prouverons que toute caractéristique différentielle de l'AES sur 4 tours comporte au moins 25 boîtes-S actives. En combinant cette observation avec le fait que la probabilité d'une transition différentielle à travers une boîte-S de l'AES est au plus 2^{-6}, nous montrerons que la probabilité de n'importe quelle caractéristique différentielle sur 4 tours de l'AES est au plus $2^{-25 \cdot 6} = 2^{-150}$.

Pour atteindre cet objectif, nous adopterons deux approches :

1. Dans la section 2.1, nous utiliserons des arguments mathématiques basés sur des observations simples concernant les opérations effectuées dans un tour de l'AES.
2. Dans la section 2.2, nous appliquerons la méthode de programmation linéaire mixte par contraintes, connue sous le nom de *Mixed Integer Linear Programming (MILP)*, pour démontrer le même résultat.

2.1 Démonstration "à la main"

Dans ce qui suit, on note M la matrice 4×4 utilisée pour définir `MixColumns`. La permutation `ShiftRows (SR)` envoie les 4 octets d'une colonne d'un état x dans 4 colonnes différentes de $\texttt{SR}(x)$. Il en est de même pour la permutation $\texttt{SR}^{-1}$.

1. Montrer que pour tout état de l'AES $x \neq 0$, la somme du nombre de colonnes non nulles des états x et $\texttt{SR} \circ \texttt{MC} \circ \texttt{SR}(x)$ est supérieure ou égale à 5, où 5 est le branch number de la matrice M.
2. Utiliser le résultat ci-dessus pour montrer qu'un chemin de 4 tours de l'AES possède au minimum 5^2 boîtes-S actives.

2.2 Avec une approche programmation par contraintes

2.2.1 Introduction à la programmation linéaire mixte par contraintes (MILP)

La *programmation linéaire (Linear Programming, LP)* est une méthode permettant de résoudre des problèmes d'optimisation sous des contraintes linéaires. Le problème peut être défini de manière générique comme suit :

On cherche à optimiser une fonction en des variables $x \in \mathbb{R}^d$, où x représente un vecteur de d variables réelles positives ($x \geq 0$). La *fonction objectif* est une combinaison linéaire des variables x, et peut être soit maximisée soit minimisée. Elle est généralement de la forme :

$$f(x) = c^T x = \sum_{i=1}^{d} c_i x_i$$

où $c = (c_1, c_2, \ldots, c_d)$ est un vecteur de coefficients donné.

Les variables x doivent respecter un ensemble de J contraintes linéaires. Les contraintes sont définies par une matrice $A \in \mathbb{R}^{J \times d}$ et un vecteur $b \in \mathbb{R}^J$, sous la forme :

$$Ax \leq b \quad \Longleftrightarrow \quad \sum_{i=1}^{d} a_{ji} x_i \leq b_j,$$

où a_{ji} est l'élément de la matrice A correspondant à la j-ème contrainte et à la i-ème variable.

Ainsi, un problème de programmation linéaire générique se formule comme suit :

$$\max_{x \in \mathbb{R}^d} \{c^T x \mid Ax \leq b,\ x \geq 0\}$$

ou, pour un problème de minimisation :

$$\min_{x \in \mathbb{R}^d} \{c^T x \mid Ax \leq b,\ x \geq 0\}.$$

Dans le cadre de la programmation linéaire mixte (MILP), certaines variables peuvent être restreintes à des valeurs entières ou binaires, permettant ainsi de modéliser des problèmes combinatoires. Par exemple, si x_1 et x_2 doivent représenter des nombres entiers, on ajoute la contrainte $x_1, x_2 \in \mathbb{Z}$.

2.2.2 Application à l'AES

Mouha et al. [1] ont été les premiers à proposer l'utilisation de l'approche MILP pour établir une borne inférieure sur le nombre de boîtes-S actives pour r tours de l'AES. C'est cette approche que nous allons implémenter.

1. Nous allons utiliser des variables binaires pour représenter l'activité des octets des états successifs de l'AES : $x_i = 1$ signifiera simplement que l'octet x_i est actif. Pour modéliser une caractéristique différentielle sur r tours, combien de variables sont nécessaires ?
2. Quelle fonction objective devons-nous maximiser ou minimiser ?
3. Comment modéliser la propagation d'une différence à travers les couches `SubBytes`, `ShiftRows` et `AddRoundKey` ?
4. Même question pour l'opération `MixColumns`.
5. Écrire le modèle MILP de manière formelle.

2.2.3 Implémentation avec `SageMath`

Le logiciel `SageMath` propose un solveur MILP, qui permet, à partir d'un modèle MILP correctement défini, de fournir une solution au problème. Dans cette partie, vous devez implémenter le modèle défini dans la section précédente en utilisant `SageMath` et son solveur dédié pour le résoudre.

La page suivante :

https://doc.sagemath.org/html/en/reference/numerical/sage/numerical/mip.html

explique comment résoudre des problèmes MILP avec `SageMath`. Lisez attentivement cette documentation avant de rédiger votre propre code.

Références

[1] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In Chuankun Wu, Moti Yung, and Dongdai Lin, editors, *Inscrypt 2011*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2011.