Contrainte globale abstractXOR : résultats de complexité et algorithmes de propagation

Loïc Rouquette, Christine Solnon

Contrainte globale *abstractXOR* : résultats de complexité et algorithmes de propagation

Loïc Rouquette* **Christine Solnon**

Université de Lyon, INSA-Lyon, CNRS, LIRIS, UMR5205, F-69621, France
{loic.rouquette, christine.solnon}@insa-lyon.com

Résumé

Les algorithmes de chiffrement symétrique par bloc utilisent une clé secrète pour chiffrer un texte. Le but de l'attaque *related-key* est d'évaluer s'il est possible de retrouver la clé en étudiant la propagation des différences entre des textes et des clés donnés. Pour cela, les cryptanalystes doivent calculer des chemins différentiels maximaux. Ce problème est résolu en deux étapes : dans une première étape, chaque octet est abstrait par un booléen indiquant si l'octet contient une différence ou non et on recherche des solutions abstraites ; dans une seconde étape, on recherche une solution concrète optimale pour chaque solution abstraite. Certaines solutions abstraites peuvent ne pas admettre de solution concrète, et un enjeu important est de limiter au maximum le nombre de solutions abstraites incohérentes au niveau concret. La perte de précision au niveau abstrait vient essentiellement du fait que le chiffrement utilise massivement des XOR (ou exclusif bit à bit), et que cette opération est très mal abstraite lors de la première étape. Pour améliorer cela, nous proposons d'introduire une contrainte permettant de propager un ensemble de XOR de façon globale.

Nous étudions tout d'abord la complexité du problème consistant à décider si un ensemble de XOR admet une solution concrète. Nous montrons que si le problème est polynomial lorsque les variables ne sont pas contraintes, il devient NP-complet si on contraint certaines variables à prendre une valeur comprise entre 1 et une borne supérieure donnée. Nous introduisons ensuite la contrainte globale $abstractXOR$ qui est satisfaite s'il existe une solution abstraite pouvant être concrétisée dans le cas où les variables ne sont pas contraintes par une borne supérieure, et nous introduisons des algorithmes polynomiaux pour propager cette contrainte. Nous montrons comment utiliser cette contrainte globale pour modéliser un problème de recherche de chemin différentiel maximal pour Midori, et nous comparons les performances du modèle utilisant cette contrainte globale avec d'autres modèles.

*Papier doctorant : Loïc Rouquette est auteur principal.

Abstract

Block cipher algorithms use shared secret keys to cipher texts. The aim of the *related-key* attack is to evaluate if it is possible to find out the key by studying difference propagations. To this aim, cryptanalysts need to compute maximum differential paths. This problem is solved in two steps : in a first step, each byte is abstracted by a boolean indicating if the byte contains a difference or not and we search for abstract solutions , in a second step, we search for an optimal concrete solution for each abstract solution. Some abstract solutions may not accept concrete solutions and our goal is to limit as much as possible the number of inconsistent abstract solutions at the concrete level. These inconsistencies mainly come from the fact that ciphering massively relies on XOR operations (bitwise exclusive or), and this operation is poorly abstracted during the first step. To improve that, we introduce a constraint for propagating a set of XORs in a global way.

First we study the complexity of the problem aiming at deciding if a XOR set accepts a concrete solution. We show that if the problem is polynomial when variables are not constrained, it becomes NP-complete if we constrain some variables to take values between 1 and a given upper bound. Then, we introduce the global constraint $abstractXOR$ which is satisfied if there exists an abstract solution that can be concretized in the case where variables are not constrained by an upper bound and we introduce polynomial algorithms to propagate this constraint. We show how to use this global constraint to model a maximal differential path problem for Midori and we compare the performance of the model using this global constraint with other models.

1 Motivations

Les algorithmes de chiffrement symétrique par bloc (tels que AES [8] ou Midori [1]) garantissent la confidentialité des communications en utilisant une clé secrète K pour chiffrer un texte initial X en un texte chiffré $f_K(X)$, de telle sorte que le texte chiffré puisse être déchiffré en utilisant la même clé.

Le but de la cryptanalyse est de vérifier que la confidentialité est bien garantie. En particulier, la *cryptanalyse différentielle* évalue s'il est possible de retrouver la clé en moins de $2^{|K|}$ essais en étudiant la propagation des différences entre deux textes X et X' pendant le chiffrement [4]. La différence δX entre les deux textes X et X' est obtenue en appliquant l'opérateur XOR (ou exclusif entre les bits de X et X'), noté $\delta X = X \oplus X'$.

Les algorithmes de chiffrement ont été prouvés robustes pour ces attaques différentielles. Ainsi, l'attaque *related-key* introduite dans [3] autorise l'attaquant à injecter des différences non seulement entre deux textes X et X' mais aussi entre deux clés K et K' (même si la clé secrète K reste inconnue pour l'attaquant). Afin de concevoir cette attaque, les cryptanalystes doivent trouver des *chemins différentiels maximaux*, *i.e.*, des différences en entrée ($\delta X = X \oplus X'$ et $\delta K = K \oplus K'$) et en sortie ($\delta X_{out} = f_K(X) \oplus f_{K'}(X')$) maximisant la probabilité d'observer δX_{out} étant données δX et δK.

Résolution en deux étapes. Pour réduire la taille de l'espace de recherche à explorer, Knudsen a introduit la notion de *différentielle tronquée* [13]. L'idée est de regrouper les bits (de δX, δK et δX_{out}, mais aussi de tous les états intermédiaires du processus de chiffrement) par séquences de b bits consécutifs : typiquement, $b = 8$ et chaque séquence δ_i de b bits correspond à un octet. Chacune de ces séquences δ_i est abstraite par un booléen Δ_i indiquant si la séquence contient une différence ou non, *i.e.*,
$\Delta_i = faux \Leftrightarrow \delta_i = 0$ et $\Delta_i = vrai \Leftrightarrow \delta_i \in [1, 2^b - 1]$.

Le problème est alors résolu en deux étapes : dans une première étape, on recherche un ensemble de *solutions abstraites* où les variables booléennes satisfont une abstraction des opérations de chiffrement ; dans une seconde étape, pour chaque solution abstraite, on recherche une *solution concrète* où chaque variable concrète δ_i est affectée à 0 (resp. une valeur comprise entre 1 et $2^b - 1$) si $\Delta_i = faux$ (resp. $\Delta_i = vrai$) dans la solution abstraite.

L'ensemble des solutions abstraites calculé lors de la première étape doit permettre de trouver toutes les solutions concrètes, *i.e.*, pour chaque solution concrète il doit exister une solution abstraite telle que, pour chaque variable concrète δ_i affectée à 0 (resp. une valeur entre 1 et $2^b - 1$), la variable abstraite Δ_i soit affectée à *faux* (resp. *vrai*). En revanche, certaines solutions abstraites calculées lors de la première étape peuvent ne pas correspondre à une solution concrète. Ces solutions abstraites sont dites *b-incohérentes* car il n'existe pas de solution concrète dans le cas où le nombre de bits utilisés pour représenter les valeurs des variables concrètes δ_i est égal à b. Pour que le processus de résolution soit efficace, il est fondamental de limiter au maximum le nombre de solutions abstraites *b-incohérentes* calculées lors de la première étape.

Abstraction du XOR lors de la première étape. L'opération principale des algorithmes de chiffrement symétrique par bloc tels qu'AES ou Midori est le XOR. Lors de la première étape, le XOR concret $\oplus$ est abstrait par l'opérateur $\oplus_{\mathcal{A}}$ dont la table de vérité est :

Δ_1	Δ_2	$\Delta_1 \oplus_{\mathcal{A}} \Delta_2$
faux	*faux*	*faux*
faux	*vrai*	*vrai*
vrai	*faux*	*vrai*
vrai	*vrai*	*vrai* ou *faux*

Cet opérateur abstrait implique une perte en précision car lorsque Δ_1 et Δ_2 sont affectées à *vrai*, on ne sait pas si $\Delta_1 \oplus_{\mathcal{A}} \Delta_2$ est égal à *vrai* ou *faux*. Cela provient du fait que le résultat d'un XOR entre deux variables δ_1 et δ_2 différentes de zéro peut être soit zéro (si $\delta_1 = \delta_2$) soit différent de zéro (si $\delta_1 \neq \delta_2$). Considérons par exemple les problèmes concret (à gauche) et abstrait (à droite) :

$$\begin{array}{ll} \delta_1 \oplus \delta_2 = \delta_3 & \Delta_1 \oplus_{\mathcal{A}} \Delta_2 = \Delta_3 \\ \delta_2 \oplus \delta_4 = \delta_5 & \Delta_2 \oplus_{\mathcal{A}} \Delta_4 = \Delta_5 \\ \delta_3 \oplus \delta_5 = \delta_6 & \Delta_3 \oplus_{\mathcal{A}} \Delta_5 = \Delta_6 \end{array}$$

L'affectation $\Delta_1 = \Delta_2 = \Delta_3 = \Delta_5 = vrai$ et $\Delta_4 = \Delta_6 = faux$ est solution du problème abstrait, mais elle est b-incohérente car si $\delta_4 = \delta_6 = 0$, alors nous pouvons en déduire que $\delta_2 = \delta_3 = \delta_5$ et donc que $\delta_1 = 0$. Par conséquent, il n'existe pas de solution concrète où $\delta_4 = \delta_6 = 0$ et où chaque autre variable prend une valeur dans $[1, 2^b - 1]$, quel que soit b.

Contributions et organisation de l'article. Dans cet article, nous introduisons une contrainte globale permettant de calculer efficacement une meilleure approximation (contenant moins de solutions b-incohérentes) d'un ensemble de contraintes XOR au niveau abstrait, pendant la première étape.

Dans la section 2, nous définissons le problème XOR_∞ visant à décider s'il existe une solution à un ensemble de contraintes XOR dans le cas où les variables δ_i peuvent prendre n'importe quelle valeur entière (non bornée par $2^b - 1$). Nous introduisons un algorithme polynomial permettant de résoudre ce problème.

Dans la section 3, nous étudions la complexité de deux cas particuliers de XOR_∞ :

— cas où certaines variables sont contraintes à prendre pour valeur 0 et les autres à prendre une valeur différente de 0, dénoté $XOR_{\infty,\neq 0}$;

— cas où certaines variables sont contraintes à prendre pour valeur 0 et les autres à prendre une valeur dans $[1, 2^b - 1]$, dénoté $XOR_{b,\neq 0}$.

Nous montrons que $XOR_{\infty,\neq 0}$ est polynomial, et $XOR_{b,\neq 0}$ est NP-complet. A notre connaissance, ce résultat est nouveau. Dans [7], les auteurs s'intéressent au problème *reachability*, consistant à décider s'il est possible de déduire un terme secret s à partir d'un ensemble de message T pour un protocole de cryptographie donné, et montrent que ce problème est décidable dans le cas où T ne contient que des contraintes XOR.

Dans la section 4, nous introduisons la contrainte globale *abstractXOR* permettant de propager le fait qu'il doit exister une affectation d'un ensemble de variables booléennes qui soit ∞-cohérente pour un ensemble donné d'équations XOR, et nous introduisons des algorithmes polynomiaux pour propager cette contrainte.

Dans la section 5, nous montrons comment utiliser cette contrainte pour modéliser très simplement un problème de cryptanalyse différentielle pour Midori, et nous comparons les performances du modèle utilisant cette contrainte avec le modèle introduit dans [9].

2 Définition du problème XOR_∞

Le problème XOR_∞ vise à décider s'il existe une solution à un ensemble de contraintes XOR dans le cas où les variables peuvent prendre n'importe quelle valeur entière. On impose en plus qu'au moins une variable soit différente de 0 afin d'interdire la solution triviale où toutes les variables sont égales à 0.

Définition 1 *Une instance de XOR_∞ est définie par un couple (X_δ, C) tel que $X_\delta = \{\delta_1, \delta_2, \ldots, \delta_m\}$ est un ensemble de m variables entières, et C est un ensemble de n équations, chaque équation étant de la forme $\delta_{i_1} \oplus \delta_{i_2} \oplus \ldots \oplus \delta_{i_x} = 0$. Le problème consiste à décider s'il existe une solution non triviale,* i.e., *une affectation d'une valeur entière à chaque variable $\delta_i \in X_\delta$ telle qu'au moins une variable soit différente de 0 et que chaque équation de C soit satisfaite.*

Une instance (X_δ, C) est représentée par une matrice M comportant n lignes et m colonnes telle que $M[i, j] = 1$ si δ_j apparaît dans la $i^{\text{ème}}$ équation, et $M[i, j] = 0$ sinon. Un exemple d'une telle représentation matricielle est donné dans la figure 1. Nous considérons dans la suite que chaque ligne et chaque colonne de la matrice comporte au moins un 1 (*i.e.*, chaque variable apparaît dans au moins une contrainte, et chaque contrainte porte sur au moins une variable).

	δ_1	δ_2	δ_3	δ_4	δ_5	δ_6	δ_7
$\delta_3 \oplus \delta_6 \oplus \delta_4 = 0$	0	0	1	1	0	1	0
$\delta_5 \oplus \delta_7 \oplus \delta_6 = 0$	0	0	0	0	1	1	1
$\delta_2 \oplus \delta_5 \oplus \delta_3 = 0$	0	1	1	0	1	0	0
$\delta_2 \oplus \delta_7 \oplus \delta_1 = 0$	1	1	0	0	0	0	1

FIGURE 1 – Exemple d'instance de XOR_∞.

Algorithme 1 : normalisation(M)

```
1  r ← 1 ; j ← 1
2  tant que r ≤ n faire
3      si il existe une ligne i ∈ [r, n] tq M[i, j] = 1 alors
4          pour chaque ligne i' ∈ [1, n] tq i' ≠ i et
             M[i', j] = 1 faire
5              pour chaque colonne j' ∈ [r, m] faire
6                  M[i', j'] ← M[i', j'] ⊕ M[i, j']
7              si la ligne i' ne comporte que des 0 alors
8                  Supprimer la ligne i et décrémenter n
9          Echanger les lignes i et r et incrémenter r
10     Incrémenter j
```

Forme échelonnée réduite d'une matrice. Le premier élément non nul de chaque ligne de la matrice est appelé *pivot*, et on note c_i le numéro de colonne du pivot de la ligne i. Une matrice est sous forme *échelonnée* si pour chaque couple de lignes (i, i') tel que $i < i'$, et pour chaque colonne $j \leq c_i$, $M[i', j] = 0$. Les variables associées aux colonnes contenant des pivots sont appelées *variables de base* ; les autres variables sont appelées *variables hors-base*. Une matrice échelonnée est *réduite* si, pour chaque ligne i, la colonne c_i contient un seul 1.

L'algorithme 1 s'inspire de l'algorithme d'élimination de Gauss pour transformer une matrice quelconque en une matrice échelonnée réduite admettant le même ensemble de solutions. A chaque itération de la boucle lignes 2-9, les $r - 1$ premières lignes de la matrice M sont sous forme échelonnée réduite, et si la colonne courante j contient un élément $M[i, j]$ pouvant devenir le pivot de la ligne i, alors la ligne i est XORée avec chaque ligne $i' \neq i$ comportant un 1 en colonne j, puis les lignes i et r sont échangées et r est incrémenté. Si certaines équations du système initial sont redondantes (*i.e.*, elles peuvent être obtenues en combinant d'autres équations de telle sorte qu'elles s'annulent lorsqu'elles sont XORées avec ces équations), alors elles sont supprimées (lignes 7-8).

Cet algorithme ne change pas l'ensemble des solutions car les seules opérations effectuées sur la matrice

(en dehors des échanges de lignes) sont des XOR entre lignes (lignes 5-6), et l'équation $\delta_{i_1} \oplus \ldots \oplus \delta_{i_x} = 0$ admet les mêmes solutions que l'équation $\delta_{i_1} \oplus \ldots \oplus \delta_{i_x} \oplus \delta_{j_1} \oplus \ldots \oplus \delta_{j_y} = 0$ dès lors que $\delta_{j_1} \oplus \ldots \oplus \delta_{j_y} = 0$. L'algorithme se termine lorsque $r = n + 1$, et les n lignes de la matrice sont sous forme échelonnée réduite.

La complexité de cet algorithme est $\mathcal{O}(mn^2)$.

L'exécution de cet algorithme sur la matrice de la figure 1 permet d'obtenir la matrice suivante :

δ_1	δ_2	δ_3	δ_4	δ_5	δ_6	δ_7
1	0	0	1	0	0	0
0	1	0	1	0	0	1
0	0	1	1	0	1	0
0	0	0	0	1	1	1

Les variables de base sont δ_1, δ_2, δ_3 et δ_5, et les pivots sont coloriés en gris.

Solutions d'une forme échelonnée réduite. Si la matrice sous forme échelonnée réduite comporte autant de lignes que de colonnes, alors il n'existe pas de solution non triviale car chaque équation contraint une variable de base à prendre pour valeur 0. Sinon, il existe autant de solutions que d'affectations différentes pour les variables hors-base : pour chaque affectation, on peut construire une solution en affectant à chaque variable δ_i de la base le résultat du XOR des valeurs des variables hors base apparaissant dans la même ligne que δ_i.

Sur notre exemple, étant donnée une affectation des variables δ_4, δ_6 et δ_7, on déduit les valeurs de δ_1, δ_2, δ_3 et δ_5 en exécutant les opérations suivantes :

$$\delta_1 = \delta_4 \qquad \delta_2 = \delta_4 \oplus \delta_7$$
$$\delta_3 = \delta_4 \oplus \delta_6 \qquad \delta_5 = \delta_6 \oplus \delta_7$$

Par conséquent, pour décider si une instance du problème XOR_∞ admet une solution non triviale, il suffit de vérifier que la matrice en sortie de l'algorithme 1 a plus de colonnes que de lignes.

3 Complexité de variantes de XOR_∞

Rappelons que lors de la première étape de résolution d'un problème de cryptanalyse différentielle, on ne recherche pas une solution à une instance (X_δ, C) de XOR_∞, mais on résout un problème abstrait où chaque variable entière δ_i est abstraite par une variable booléenne Δ_i et où l'opérateur $\oplus$ entre entiers est abstrait par l'opérateur $\oplus_\mathcal{A}$ entre booléens. Dans ce contexte, pour vérifier la correction d'une solution abstraite, nous devons décider si une instance du problème XOR_∞ possède une solution dans le cas où certaines variables sont contraintes à prendre pour valeur 0 et les autres à prendre une valeur dans $[1, 2^b - 1]$. Dans la section 3.2, nous montrons que ce problème, dénoté $XOR_{b,\neq 0}$, est NP-complet. Mais auparavant nous nous intéressons au cas intermédiaire entre XOR_∞ et $XOR_{b,\neq 0}$, et nous montrons que le problème est polynomial quand certaines variables sont contraintes à prendre pour valeur 0 et les autres à prendre une valeur différente de 0 (problème dénoté $XOR_{\infty,\neq 0}$).

3.1 Complexité de $XOR_{\infty,\neq 0}$

$XOR_{\infty,\neq 0}$ est la variante de XOR_∞ où certaines variables sont contraintes à prendre des valeurs non nulles, et d'autres la valeur 0.

Définition 2 *Une instance de $XOR_{\infty,\neq 0}$ est définie par un triplet $(X_\delta, C, X_{\neq 0})$ où (X_δ, C) est une instance de XOR_∞, et $X_{\neq 0} \subseteq X_\delta$ est un sous-ensemble de variables. Le problème consiste à décider s'il existe une solution de (X_δ, C) telle que chaque variable $\delta_i \in X_{\neq 0}$ est affectée à une valeur différente de zéro et chaque variable $\delta_i \in X_\delta \setminus X_{\neq 0}$ est affectée à zéro.*

Pour décider si une instance $(X_\delta, C, X_{\neq 0})$ de $XOR_{\infty,\neq 0}$ possède une solution, nous allons utiliser la propriété suivante :

Propriété 1 *Toute matrice sous forme échelonnée réduite comportant au moins deux 1 par ligne admet au moins une solution où toutes les variables ont une valeur strictement positive.*

Preuve. L'algorithme 2 montre comment construire une telle solution : il affecte une à une les variables hors-base, en choisissant à chaque fois la première valeur positive possible. Le point clé consiste à maintenir, pour chaque variable de base, le résultat du XOR sur les valeurs des variables hors-base dont elle dépend et qui sont déjà affectées : ces valeurs sont initialisées à 0 (ligne 2) ; à chaque fois qu'une variable hors-base δ_j est affectée à une valeur v_j, pour chaque variable de base δ_{c_i} dépendant de δ_j, on met à jour v_{c_i} en le XORant avec v_j (lignes 6-7). Au moment de choisir une valeur pour une variable hors-base δ_j, on cherche l'ensemble S des valeurs interdites pour δ_j (ligne 4) : une valeur v est interdite si elle contraint une variable de base δ_{c_i} à prendre pour valeur 0, *i.e.*, $M[i, j]$ est le dernier 1 de la ligne i et le résultat du XOR sur les variables hors-base déjà affectées dont dépend δ_{c_i} (*i.e.*, v_{c_i}) est égal à v. □

Considérons par exemple la matrice échelonnée réduite de notre exemple, pour laquelle les numéros de colonne des pivots sont $P = \{1, 2, 3, 5\}$.

— Pour choisir v_4, on construit $S = \{v_1\} = \{0\}$, et on affecte v_4 à la plus petite valeur de $\mathbb{N}^+ \setminus \{0\}$, *i.e.*, $v_4 = 1$. On a $v_1 = v_2 = v_3 = 1 \oplus 0 = 1$.

Algorithme 2 : Concrétisation(M)

Input : Une matrice M sous forme échelonnée réduite telle que, pour chaque ligne $i \in [1,n]$,$\sum_{j=1}^{m} M[i,j] > 1$

Output : Une valeur $v_j > 0$ pour chaque colonne $j \in [1,m]$ telle que $\delta_1 = v_1, \dots \delta_m = v_m$ soit une solution de M

Soit $P = \{c_i : i \in [1,n]\}$ l'ensemble des numéros de colonne des pivots de M
pour *chaque colonne* $j \in P$ **faire** initialiser v_j à 0 ;
pour *chaque colonne* $j \in [1,m] \setminus P$ **faire**
  $S \leftarrow \{v_{c_i} : i \in [1,n], M[i,j]=1 \wedge \sum_{j'=j+1}^{m} M[i,j']=0\}$
  Affecter à v_j la plus petite valeur de $\mathbb{N}^+ \setminus S$
  pour *chaque ligne* $i \in [1,n]$ *tq* $M[i,j] = 1$ **faire**
    $v_{c_i} \leftarrow v_{c_i} \oplus v_j$

— Pour choisir v_6, on construit $S = \{v_3\} = \{1\}$ et on affecte v_6 à la plus petite valeur de $\mathbb{N}^+ \setminus \{1\}$, *i.e.*, $v_6 = 2$. On a $v_3 = 1 \oplus 2 = 3$ et $v_5 = 0 \oplus 2 = 2$.
— Pour choisir v_7, on construit $S = \{v_2, v_5\} = \{1,2\}$ et on affecte v_7 à la plus petite valeur de $\mathbb{N}^+ \setminus \{1,2\}$, *i.e.*, $v_7 = 3$. On a $v_2 = 1 \oplus 3 = 2$ et $v_5 = 2 \oplus 3 = 1$.

L'affectation finale est $v_1 = 1$, $v_2 = 2$, $v_3 = 3$, $v_4 = 1$, $v_5 = 1$, $v_6 = 2$, et $v_7 = 3$.

La propriété 1 nous permet de montrer la propriété suivante :

Propriété 2 *Le problème $XOR_{\infty,\neq 0}$ est polynomial.*

Preuve. Pour décider si une instance $(X_\delta, C, X_{\neq 0})$ de $XOR_{\infty,\neq 0}$ admet une solution, nous commençons par supprimer de chaque équation de C les variables n'appartenant pas à $X_{\neq 0}$, et nous supprimons les équations ne comportant que des variables n'appartenant pas à $X_{\neq 0}$. Cette transformation ne change pas l'ensemble de solutions de l'instance car $\forall X \in \mathbb{N}, X \oplus 0 = X$. Nous construisons ensuite la matrice M associée aux équations résultantes (telle que chaque colonne de M correspond à une variable de $X_{\neq 0}$ devant prendre une valeur différente de 0) et nous la mettons sous forme échelonnée réduite. Si la matrice échelonnée réduite possède une ligne comportant exactement un 1, cela implique qu'il existe une équation de la forme $\delta_i = 0$, et cette équation ne peut pas être satisfaite. Dans ce cas, l'instance n'a pas de solution. Sinon, l'instance admet au moins une solution du fait de la propriété 1.

Étant donné que la matrice peut être mise sous forme échelonnée réduite en temps polynomial en utilisant l'algorithme 1, le problème $XOR_{\infty,\neq 0}$ est polynomial. □

3.2 Complexité de $XOR_{b,\neq 0}$

$XOR_{b,\neq 0}$ est la variante de XOR_∞ où certaines variables sont contraintes à prendre des valeurs comprises entre 1 et $2^b - 1$, et d'autres la valeur 0.

Définition 3 *Une instance de $XOR_{b,\neq 0}$ est définie par un quadruplet $(X_\delta, C, b, X_{\neq 0})$ où (X_δ, C) est une instance de XOR_∞, $b \geq 1$ est un entier positif, et $X_{\neq 0} \subseteq X_\delta$ est un sous-ensemble de variables. Le problème consiste à décider s'il existe une solution de (X_δ, C) telle que chaque variable $\delta_i \in X_{\neq 0}$ a une valeur comprise entre 1 et $2^b - 1$ et chaque variable $\delta_i \in X_\delta \setminus X_{\neq 0}$ est affectée à zéro.*

La propriété 1 n'est plus valide dans le cas où les variables sont bornées, et une matrice sous forme échelonnée réduite comportant au moins deux 1 par ligne peut ne pas admettre de solution. Considérons par exemple l'ensemble d'équations suivant :

$$\delta_1 \oplus \delta_2 \oplus \delta_5 = 0 \qquad \delta_1 \oplus \delta_3 \oplus \delta_6 = 0$$
$$\delta_1 \oplus \delta_4 \oplus \delta_7 = 0 \qquad \delta_2 \oplus \delta_3 \oplus \delta_8 = 0$$
$$\delta_2 \oplus \delta_4 \oplus \delta_9 = 0 \qquad \delta_3 \oplus \delta_4 \oplus \delta_{10} = 0$$

La matrice M associée à cet ensemble d'équations est :

δ_5	δ_6	δ_7	δ_8	δ_9	δ_{10}	δ_1	δ_2	δ_3	δ_4
1	0	0	0	0	0	1	1	0	0
0	1	0	0	0	0	1	0	1	0
0	0	1	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1	1	0
0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	1	0	0	1	1

M est sous forme échelonnée réduite et respecte bien la condition d'avoir au moins deux 1 par ligne. Cependant, si $X_{\neq 0}$ contient toutes les variables, alors l'instance n'a pas de solution car chaque équation $\delta_i \oplus \delta_j \oplus \delta_k = 0$ implique que $\delta_i \neq \delta_j$ (si $\delta_i = \delta_j$ alors $\delta_k = \delta_i \oplus \delta_j = 0$). Ainsi, les variables δ_1, δ_2, δ_3 et δ_4 doivent prendre des valeurs toutes différentes. Si $b = 2$, alors il n'existe pas de solution où chaque variable est affectée à une valeur comprise entre 1 et $2^2 - 1 = 3$ car il n'y a pas suffisamment de valeurs dans $[1,3]$ pour affecter des valeurs toutes différentes à δ_1, δ_2, δ_3 et δ_4.

De fait, la propriété 3 nous montre que $XOR_{b,\neq 0}$ ne peut être résolu en temps polynomial si P≠NP.

Propriété 3 *Le problème $XOR_{b,\neq 0}$ est NP-complet.*

Preuve. Le problème appartient à la classe NP car nous pouvons vérifier en temps polynomial qu'une affectation donnée des variables de X_δ est solution. Montrons qu'il est NP-complet en réduisant le problème de coloriage d'un graphe en $XOR_{b,\neq 0}$. Étant donnés un

graphe $G = (V, E)$ et un entier x, le problème du coloriage consiste à décider s'il est possible d'affecter une valeur entière comprise entre 1 et x à chaque sommet de V de telle sorte que pour chaque arête $(i, j) \in E$, la valeur affectée à i soit différente de la valeur affectée à j. Ce problème est NP-complet [12], et il reste NP-complet même si $x = 2^b - 1$ dès lors que $b \geq 2$ (lorsque $b = 2$, on obtient le problème de coloriage avec 3 couleurs). Pour réduire une instance $(G = (V, E), x)$ d'un problème de coloriage (où $x = 2^b - 1$) en une instance $(X_\delta, C, b, X_{\neq 0})$ de $XOR_{b,\neq 0}$, on définit :

— $X_\delta = \{\delta_i : i \in V\} \cup \{\delta_{ij} : (i, j) \in E\}$
— $C = \{\delta_i \oplus \delta_j \oplus \delta_{ij} = 0 : (i, j) \in E\}$
— $X_{\neq 0} = X_\delta$

Chaque équation $\delta_i \oplus \delta_j \oplus \delta_{ij} = 0$ impose que les variables associées aux sommets i et j (*i.e.*, δ_i et δ_j) prennent des valeurs différentes, de sorte que toute solution de l'instance de $XOR_{b,\neq 0}$ est un coloriage valide de G. De même, tout coloriage valide de G correspond à une solution de l'instance de $XOR_{b,\neq 0}$: il suffit d'affecter à chaque variable δ_i associée à un sommet i la valeur affectée à i dans le coloriage et, pour chaque arête $(i, j) \in E$, d'affecter δ_{ij} à $\delta_i \oplus \delta_j$. □

4 Définition de *abstractXOR*

Pour limiter au maximum le nombre de solutions abstraites b-incohérentes lors de la première étape du calcul d'un chemin différentiel maximal, il faudrait assurer que, pour chaque solution abstraite, il existe une solution concrète satisfaisant l'ensemble des XOR et telle que, pour chaque variable booléenne Δ_i affectée à vrai dans la solution abstraite, la variable concrète δ_i ait une valeur comprise entre 1 et $2^b - 1$. Cela reviendrait à décider si une instance de $XOR_{b,\neq 0}$ admet une solution, et comme ce problème est NP-complet, nous proposons de définir une contrainte globale assurant la relaxation $XOR_{\infty,\neq 0}$ de ce problème : en relâchant la contrainte sur la borne supérieure des valeurs concrètes, on peut propager efficacement la contrainte globale.

Définition 4 *Etant donnés une instance (X_δ, C) du problème XOR_∞, et un ensemble X_Δ de variables booléennes tel que X_Δ contient une variable Δ_i pour chaque variable $\delta_i \in X_\delta$, la contrainte globale $abstractXOR_{X_\delta,C}(X_\Delta)$ est satisfaite si et seulement s'il existe une solution à l'instance de $XOR_{\infty,\neq 0}$ définie par $(X_\delta, C, X_{\neq 0})$ avec $X_{\neq 0} = \{\delta_i \in X_\delta : \Delta_i = vrai\}$.*

Notons qu'on ne cherche pas à calculer la solution concrète de $(X_\delta, C, X_{\neq 0})$, mais juste à calculer une solution abstraite (sur les booléens) pour laquelle il existe au moins une solution concrète (sur les entiers).

4.1 Vérification de la faisabilité d'*abstractXOR*

On note n et m le nombre de lignes et de colonnes de M. Etant donné un numéro de colonne $j \in [1, m]$, on note δ_j la variable entière associée à la colonne j, Δ_j la variable booléenne correspondante, et $D(\Delta_j)$ son domaine. Pour chaque ligne i, on définit :

— $var_i = \{j \in [1, m] : M[i, j] = 1\}$;
— $vrai_i = \{j \in var_i : D(\Delta_j) = \{vrai\}\}$.

On note $B \subseteq [1, m]$ l'ensemble des colonnes associées à une variable de base et $HB = [1, m] \setminus B$ l'ensemble des colonnes associées à une variable hors-base. Enfin, rappelons que nous notons c_i le numéro de colonne de la variable de base de la ligne i.

Pour vérifier la faisabilité d'*abstractXOR* pendant la recherche, nous maintenons la matrice M sous forme échelonnée réduite, nous supprimons de M les colonnes correspondant à des variables affectées à *faux*, et nous vérifions qu'il y a toujours au moins deux 1 par ligne : la propriété 1 assure qu'il existe toujours une solution dans ce cas. Nous appliquons pour cela les règles R1 à R3 à chaque fois que cela est possible, jusqu'à ce que soit un domaine devienne vide (une incohérence est détectée), soit plus aucune règle ne s'applique.

R1. Si $\exists j \in HB, D(\Delta_j) = \{faux\}$, alors supprimer la colonne j de M.

R2. Si $\exists j_1 \in B, D(\Delta_{j_1}) = \{faux\}$, alors :
— Soit i_1 le numéro de ligne tel que $M[i_1, j_1] = 1$
— Si $var_{i_1} = \{j_1\}$, alors supprimer la colonne j_1 et la ligne i_1 de M
— Sinon, il faut choisir une nouvelle base :
  — choisir $j_2 \in var_{i_1} \setminus \{j_1\}$
  — Pour chaque ligne $i_2 \neq i_1$ telle que $M[i_2, j_2] = 1$ et pour chaque colonne $j_3 \in [1, m]$, remplacer $M[i_2, j_3]$ par $M[i_2, j_3] \oplus M[i_1, j_3]$
  — Supprimer la colonne j_1

R3. Si $\exists i \in [1, n], var_i = \{j\} \wedge vrai \in D(\Delta_j)$ alors supprimer $vrai$ de $D(\Delta_j)$.

Propriété 4 *L'application de R1, R2 et R3 ne change pas l'ensemble des solutions de la contrainte globale.*

Preuve. R1 et R2 suppriment une colonne associée à une variable δ_j qui doit être affectée à 0, et nous avons vu dans la section 3.1 que cela ne change pas l'ensemble des solutions de M. R2 remplace également l'équation de la ligne i_2 (pour chaque ligne i_2 où apparaît la variable qui entre dans la base) par le résultat d'un XOR entre l'équation de la ligne i_2 et l'équation de la ligne i_1, et nous avons vu dans la section 2 que cela ne change pas l'ensemble des solutions de M. R3 supprime

vrai de $D(\Delta_j)$ dans le cas où la ligne i correspond à l'équation $\delta_j = 0$. Cette équation ne peut pas être satisfaite si $\Delta_j = vrai$, et donc Δ_j ne peut pas être affectée à *vrai* dans une solution. □

Propriété 5 *Si M est sous forme échelonnée réduite avant d'appliquer les règles alors les propriétés suivantes sont vérifiées après avoir appliqué les règles (si aucun domaine n'est vide) : (i) pour chaque colonne $j \in [1, m]$, $vrai \in D(\Delta_j)$, (ii) M est sous forme échelonnée réduite, (iii) pour toute ligne i de M, $\#var_i \geq 2$.*

Preuve. (i) est assurée par R1 et R2, qui suppriment de M les colonnes associées à des variables affectées à *faux*. (ii) est assurée par R2, qui fait entrer en base une nouvelle variable lorsqu'une colonne supprimée correspond à une variable de base. (iii) est assurée par R2 et R3 : lorsqu'une équation contient une seule variable, alors R3 enlève *vrai* du domaine de cette variable, et R2 supprime la ligne correspondant à l'équation et la colonne correspondant à la variable. □

Propriété 6 *L'application de R1, R2 et R3 jusqu'à l'obtention d'un point fixe assure la satisfiabilité d'abstractXOR$_{X_\delta, C}(X_\Delta)$ si aucun domaine n'est vide.*

Preuve. Les propriétés 5 et 1 assurent que la contrainte est satisfaite par l'affectation A telle que, pour toute variable $\Delta_j \in X_\Delta$, si $vrai \in D(\Delta_j)$ alors $A(\Delta_j) = vrai$ sinon $A(\Delta_j) = faux$. □

4.2 Maintien de la cohérence d'arc généralisée

Pour maintenir efficacement la cohérence d'arc généralisée (GAC), nous introduisons deux règles supplémentaires qui sont appliquées en plus de R1, R2 et R3 jusqu'à ce que soit un domaine devienne vide (une incohérence est détectée), soit plus aucune règle ne s'applique (la GAC est atteinte).

R4. Si $\exists i \in [1, n], var_i = \{j_1, j_2\} \wedge vrai_i = \{j_1\}$, alors supprimer *faux* de $D(\Delta_{j_2})$.

R5. Si $\exists i_1, i_2 \in [1, n], var_{i_1} \setminus \{c_{i_1}\} = var_{i_2} \setminus \{c_{i_2}\} \wedge D(\Delta_{c_{i_1}}) = \{vrai\} \wedge faux \in D(\Delta_{c_{i_2}})$ alors supprimer *faux* de $D(\Delta_{c_{i_2}})$.

Propriété 7 *L'application de R4 et R5 ne change pas l'ensemble des solutions de la contrainte globale.*

Preuve. R4 supprime *faux* de $D(\Delta_{j_2})$ dans le cas où la ligne i correspond à l'équation $\delta_{j_1} = \delta_{j_2}$ et $D(\Delta_{j_1}) = \{vrai\}$. Cette équation ne peut pas être satisfaite si $\Delta_{j_1} = vrai$ et $\Delta_{j_2} = faux$.

R5 supprime *faux* de $D(\Delta_{c_{i_2}})$ dans le cas où $D(\Delta_{c_{i_1}}) = \{vrai\}$ et les lignes i_1 et i_2 correspondent à

$$\begin{aligned} \delta_{c_{i_1}} &= \delta_{i'_1} \oplus \ldots \oplus \delta_{i'_x} \\ \delta_{c_{i_2}} &= \delta_{i'_1} \oplus \ldots \oplus \delta_{i'_x} \end{aligned}$$

Ces équations impliquent que $\delta_{c_{i_1}} = \delta_{c_{i_2}}$. Par conséquent, si $\Delta_{c_{i_1}} = vrai$, alors $\Delta_{c_{i_2}} \neq faux$. □

Propriété 8 *L'application des règles R1 à R5 jusqu'à un point fixe assure la GAC d'abstractXOR$_{X_\delta, C}(X_\Delta)$*

Preuve. Montrons que pour toute variable $\Delta_i \in X_\Delta$ et toute valeur $v_i \in D(\Delta_i)$, le couple (Δ_i, v_i) possède un support, *i.e.*, une affectation $A : X_\Delta \to \{vrai, faux\}$ qui satisfait *abstractXOR*$_{X_\delta, C}(X_\Delta)$ et telle que $A(\Delta_i) = v_i$ et $\forall \Delta_j \in X_\Delta, A(\Delta_j) \in D(\Delta_j)$.

R1, R2 et R3 assurent qu'il existe une affectation A qui est un support du couple $(\Delta_j, vrai)$ pour toute variable Δ_j telle que $vrai \in D(\Delta_j)$, et du couple $(\Delta_j, faux)$ pour toute variable Δ_j telle que $D(\Delta_j) = \{faux\}$ (cf preuve de la propriété 6).

Il reste à vérifier que pour toute variable Δ_j telle que $D(\Delta_j) = \{vrai, faux\}$, il existe un support pour le couple $(\Delta_j, faux)$. Nous distinguons pour cela trois cas, et montrons à chaque fois que si on supprime la colonne j de M et qu'on remet M sous forme échelonnée réduite, la matrice résultante aura au moins deux variables par ligne de sorte qu'il existe un support.

• *Cas 1 : δ_j n'est pas une variable de base.* Soit $S = \{i \in [1, n] : M[i, j] = 1 \wedge \#var_i = 2\}$ l'ensemble des lignes comportant δ_j et exactement une autre variable, et soit $X = \{\Delta_{c_i} : i \in S\}$ l'ensemble des variables booléennes associées aux variables de base de ces lignes. On peut affecter chaque variable $\Delta_{c_i} \in X$ à *faux* (car R4 garantit que $faux \in D(\Delta_{c_i}) \Leftrightarrow faux \in D(\Delta_j)$), et supprimer de M chaque ligne $i \in S$ et chaque colonne associée à une variable affectée à *faux*. M est alors sous forme échelonnée réduite et a au moins deux variables par ligne.

• *Cas 2 : δ_j est une variable de base apparaissant dans une ligne i contenant exactement 2 variables (δ_j et δ_k).* Dans ce cas, R4 garantit que $faux \in D(\Delta_k) \Leftrightarrow faux \in D(\Delta_j)$ et donc Δ_k peut être affectée à *faux*. Δ_k n'est pas une variable de base, et son affectation à *faux* peut impliquer d'affecter à *faux* d'autres variables (cf cas 1) mais cela sera toujours possible grâce à R4 et, une fois supprimées les colonnes des variables affectées à *faux* et les lignes ne comportant que des colonnes supprimées, la matrice résultante sera sous forme échelonnée réduite et aura au moins deux variables par ligne.

• *Cas 3 : δ_j est une variable de base apparaissant dans une ligne i contenant plus de 2 variables.* Dans ce cas, on supprime la colonne de la variable δ_j, et il faut

Maximiser $\sum_{i\in[0,r-1],j,k\in[0,3]} \log_2(p(\delta X_{i,j,k}, \delta SX_{i,j,k}))$

(C_1) $\forall i \in [0, r-2], \forall j,k \in [0,3], \delta Z_{i,j,k} \oplus \delta K_{j,k} = \delta X_{i+1,j,k}$

(C_2) $\forall i \in [0, r-2]$,

$$\begin{array}{llll} \delta Y_{i,0,0} = \delta SX_{i,0,0} & \delta Y_{i,1,0} = \delta SX_{i,2,2} & \delta Y_{i,2,0} = \delta SX_{i,1,1} & \delta Y_{i,3,0} = \delta SX_{i,3,3} \\ \delta Y_{i,0,1} = \delta SX_{i,2,3} & \delta Y_{i,1,1} = \delta SX_{i,0,1} & \delta Y_{i,2,1} = \delta SX_{i,3,2} & \delta Y_{i,3,1} = \delta SX_{i,1,0} \\ \delta Y_{i,0,2} = \delta SX_{i,1,2} & \delta Y_{i,1,2} = \delta SX_{i,3,0} & \delta Y_{i,2,2} = \delta SX_{i,0,3} & \delta Y_{i,3,2} = \delta SX_{i,2,1} \\ \delta Y_{i,0,3} = \delta SX_{i,3,1} & \delta Y_{i,1,3} = \delta SX_{i,1,3} & \delta Y_{i,2,3} = \delta SX_{i,2,0} & \delta Y_{i,3,3} = \delta SX_{i,0,2} \end{array}$$

(C_3) $\forall i \in [0, r-2], \forall j,k \in [0,3], \delta Y_{i,(j+1)\%4,k} \oplus \delta Y_{i,(j+2)\%4,k} \oplus \delta Y_{i,(j+3)\%4,k} = \delta Z_{i,j,k}$

FIGURE 2 – Problème concret du calcul de la probabilité maximale d'un chemin différentiel pour Midori128

faire entrer une variable hors-base de la ligne i dans la base (de façon similaire à ce qui est fait par R2). La matrice résultante sera sous forme échelonnée réduite, mais il reste à prouver qu'elle ne comporte pas de ligne avec une seule variable. R5 garantit cela : le seul cas où une ligne résultant du XOR de la ligne i avec une autre ligne k contenant la variable entrant dans la base peut n'avoir qu'une seule variable est le cas où les variables hors-base de la ligne i sont exactement les mêmes que les variables hors-base de la ligne k (ces variables s'annulent lors du XOR, et il ne reste plus que la variable de base δ_{c_k} de la ligne k). Dans ce cas, la règle 5 nous assure que $faux \in D(\Delta_{c_k})$, et nous pouvons donc affecter Δ_{c_k} à faux dans le support (et supprimer la ligne k et la colonne c_k).

Dans les trois cas, un support de $(\Delta_j, faux)$ est l'affectation qui affecte à *vrai* toutes les variables restant dans M après ces suppressions, et à *faux* toutes les autres variables. □

4.3 Implémentation

Les valeurs var_i et $vrai_i$, pour chaque ligne i, sont maintenues incrémentalement.

L'implémentation des règles R1 à R5 implique essentiellement de parcourir des lignes ou des colonnes de M : il faut parcourir une colonne pour la supprimer (dans R1 et R2), et parcourir une ligne pour XORer deux lignes (R2) ou décider si deux lignes ont les mêmes variables hors-base (R5). Pour une implémentation efficace de ces opérations, la matrice M (qui est très creuse) est représentée à l'aide de listes chaînées (aussi bien pour les lignes que pour les colonnes), et nous utilisons les *Dancing Links* introduits par Knuth [14] afin de restaurer efficacement l'état de la matrice lors des retour-arrière, en effectuant les opérations inverses des opérations effectuées lors de la création du point de choix. Soit l (resp. c) le nombre maximum de 1 dans une ligne (resp. colonne) de M. En utilisant les dancing links, la complexité en temps pour supprimer une colonne ou pour la restaurer est $\mathcal{O}(c)$, et la complexité en temps pour XORer deux lignes ou pour restaurer l'état de la ligne avant le XOR est $\mathcal{O}(l)$.

5 Application au calcul de chemins différentiels maximaux pour Midori

Midori [1] est un algorithme de chiffrement léger conçu pour consommer moins d'énergie qu'AES. Le chiffrement est itératif et, à chaque itération $i \in [0, r-1]$, les seules opérations qui sont effectuées sont des déplacements d'octets, des XOR entre octets, et une opération appelée *Sbox* qui substitue chaque octet par un autre octet selon une table donnée.

Description du problème concret. Le problème concret Midori128 est décrit dans la figure 2, r étant le nombre d'itérations. Les octets de la clé K et du texte X_i à chaque itération i sont regroupés dans des matrices de 4x4 octets. Les variables sont :

- $\delta K_{j,k}$ avec $j,k \in [0,3]$, qui représente la différence dans l'octet ligne j et colonne k de la clé ;
- $\delta X_{i,j,k}$ avec $i \in [0, r-1], j,k \in [0,3]$, qui représente la différence en entrée (resp. en sortie de la $i$ème itération et en sortie du chiffrement) dans l'octet ligne j et colonne k du texte quand $i = 0$ (resp. $i \in [1, r-2]$ et $i = r-1$) ;
- $\delta SX_{i,j,k}$ avec $i \in [0, r-1], j,k \in [0,3]$, qui représente la différence en sortie de la Sbox, quand la différence en entrée est $\delta X_{i,j,k}$;
- $\delta Y_{i,j,k}$, avec $i \in [0, r-2]$ et $j,k \in [0,3]$), qui représente la différence en sortie de l'opération *ShuffleCell* quand la différence en entrée est $\delta SX_{i,j,k}$;
- $\delta Z_{i,j,k}$, avec $i \in [0, r-2]$ et $j,k \in [0,3]$), qui représente la différence en sortie de *MixColumns* quand la différence en entrée est $\delta Y_{i,j,k}$.

Chacune de ces variables est une variable entière pouvant prendre une valeur comprise entre 0 et 255.

L'objectif est de trouver la solution maximisant la probabilité d'observer la différence en sortie δX_{r-1} connaissant les différences en entrée δX_0 et δK, ce qui revient à maximiser la somme des logarithmes en base 2 de la probabilité d'observer la différence en sortie de chaque Sbox $\delta SX_{i,j,k}$ connaissant la différence en entrée $\delta X_{i,j,k}$. En effet, la Sbox est la seule opération non linéaire pour laquelle la différence

(C'_0) $obj = \sum_{i\in[0,r-1],j,k\in[0,3]} \Delta X_{i,j,k}$

(C'_1) $\forall i \in [0, r-2], \forall j,k \in [0,3], \Delta Z_{i,j,k} \oplus_{\mathcal{A}} \Delta K_{j,k} \oplus_{\mathcal{A}} \Delta X_{i+1,j,k} = 0$

(C'_2) $\forall i \in [0, r-2]$,

$$\begin{array}{llll} \Delta Y_{i,0,0} = \Delta X_{i,0,0} & \Delta Y_{i,1,0} = \Delta X_{i,2,2} & \Delta Y_{i,2,0} = \Delta X_{i,1,1} & \Delta Y_{i,3,0} = \Delta X_{i,3,3} \\ \Delta Y_{i,0,1} = \Delta X_{i,2,3} & \Delta Y_{i,1,1} = \Delta X_{i,0,1} & \Delta Y_{i,2,1} = \Delta X_{i,3,2} & \Delta Y_{i,3,1} = \Delta X_{i,1,0} \\ \Delta Y_{i,0,2} = \Delta X_{i,1,2} & \Delta Y_{i,1,2} = \Delta X_{i,3,0} & \Delta Y_{i,2,2} = \Delta X_{i,0,3} & \Delta Y_{i,3,2} = \Delta X_{i,2,1} \\ \Delta Y_{i,0,3} = \Delta X_{i,3,1} & \Delta Y_{i,1,3} = \Delta X_{i,1,3} & \Delta Y_{i,2,3} = \Delta X_{i,2,0} & \Delta Y_{i,3,3} = \Delta X_{i,0,2} \end{array}$$

(C'_3) $\forall i \in [0, r-2], \forall j,k \in [0,3], \Delta Y_{i,(j+1)\%4,k} \oplus_{\mathcal{A}} \Delta Y_{i,(j+2)\%4,k} \oplus_{\mathcal{A}} \Delta Y_{i,(j+3)\%4,k} \oplus_{\mathcal{A}} \Delta Z_{i,j,k} = 0$

FIGURE 3 – Problème abstrait du calcul d'un chemin différentiel maximal pour Midori128

en sortie n'est pas connue avec certitude quand on connait l'entrée : pour la Sbox, la probabilité d'observer une différence en sortie δ_{out} étant donnée une différence en entrée δ_{in} est notée $p(\delta_{in}, \delta_{out})$ et appartient à $\{0, 2^{-2}, 2^{-3}, 2^{-4}, 2^{-5}, 2^{-6}, 1\}$. Le seul cas où $p(\delta_{in}, \delta_{out}) = 1$ est le cas où $\delta_{in} = \delta_{out} = 0$.

La contrainte (C_1) correspond à l'opération de chiffrement *AddKey* qui permet d'obtenir $\delta X_{i+1,j,k}$ en XORant $\delta Z_{i,j,k}$ et $\delta K_{j,k}$. La contrainte (C_2) correspond à l'opération de chiffrement *ShuffleCells*, et lie chaque variable $\delta Y_{i,j,k}$ à une variable différente $\delta SX_{i,j',k'}$. La contrainte (C_3) correspond à l'opération de chiffrement *MixColumns* qui permet d'obtenir $\delta Z_{i,j,k}$ en XORant les octets $\delta Y_{i,j',k}$ tels que $j' \in [0,3] \setminus \{j\}$.

Description du problème abstrait. Lors de la première étape, chaque octet $\delta X_{i,j,k}, \delta Y_{i,j,k}, \delta Z_{i,j,k}$ et $\delta K_{j,k}$ est abstrait par une variable booléenne $\Delta X_{i,j,k}, \Delta Y_{i,j,k}, \Delta Z_{i,j,k}$ et $\Delta K_{j,k}$ indiquant si l'octet est égal à 0 ou non. L'opération Sbox est ignorée lors de cette étape car $(\delta X_{i,j,k} = 0) \Leftrightarrow (\delta SX_{i,j,k} = 0)$. Pour simplifier, nous utilisons uniquement $\Delta X_{i,j,k}$ qui est associé à la fois à $\delta X_{i,j,k}$ et $\delta SX_{i,j,k}$.

Le modèle mathématique du problème abstrait est décrit dans la figure 3. La variable entière *obj* (dont le domaine est $[1, 16 * r]$) représente le nombre de différences dans des octets qui passent par une Sbox (*i.e.*, les octets $\Delta X_{i,j,k}$) : l'objectif de la première étape est d'énumérer toutes les solutions abstraites pour lesquelles *obj* est minimale (car quand $\Delta X_{i,j,k} = faux$, $p(\delta X_{i,j,k}, \delta SX_{i,j,k}) = 1$ alors que quand $\Delta X_{i,j,k} = vrai$, $p(\delta X_{i,j,k}, \delta SX_{i,j,k}) \in \{0, 2^{-2}, 2^{-3}, 2^{-4}, 2^{-5}, 2^{-6}\}$.

La contrainte (C'_0) assure que *obj* est égal au nombre de différences dans les $\Delta X_{i,j,k}$. Les contraintes (C'_1) et (C'_3) sont les abstractions des contraintes (C_1) et (C_3), où le XOR concret $\oplus$ est remplacé par le XOR abstrait $\oplus_{\mathcal{A}}$. La contrainte (C'_2) est identique à (C_2).

Modèle CP global. Un premier modèle CP, appelé *Global*, est obtenu à partir de celui de la Figure 3 en remplaçant les contraintes (C'_1) et (C'_3) par la contrainte $abstractXOR_{X_\delta, C}(X_\Delta)$ où X_Δ (resp. X_δ) contient l'ensemble des variables booléennes (resp. octets) intervenant dans les contraintes (C'_1) et (C'_3) (resp. (C_1) et (C_3)), et $C = \{C_1, C_3\}$.

Modèle CP basic. Un deuxième modèle, appelé *Basic*, est obtenu en remplaçant les contraintes (C'_1) et (C'_3) par : $\forall i \in [0, r-2], \forall j,k \in [0,3]$, $\Delta Z_{i,j,k} + \Delta K_{j,k} + \Delta X_{i+1,j,k} \neq 1$ et $\Delta Y_{i,(j+1)\%4,k} + \Delta Y_{i,(j+2)\%4,k} + \Delta Y_{i,(j+3)\%4,k} + \Delta Z_{i,j,k} \neq 1$. Ces contraintes correspondent à la relation définie par la table de vérité de l'opérateur abstrait $\oplus_{\mathcal{A}}$.

Modèle CP avancé. Le modèle Basic trouve un grand nombre de solutions abstraites qui sont incohérentes au niveau concret. Afin de diminuer ce nombre de solutions incohérentes, il est possible d'ajouter des contraintes liées à la propriété quasi-MDS (*Maximum Distance Separable*) de l'opération *MixColumns*, comme cela a été proposé dans [9]. Cette propriété est une conséquence mathématique de la définition de *MixColumns* qui a été prouvée par des cryptographes, et se traduit par un grand nombre de contraintes qui sont loin d'être triviales à formuler. Nous ne pouvons décrire ce modèle (que nous appelons *Avancé*) ici, et nous invitons le lecteur à lire [9] pour plus de détails.

Solveurs considérés. Les modèles CP Basic, Avancé, et Global ont été implémentés en Choco 4 [16]. Pour Basic et Avancé, nous avons utilisé l'heuristique *WeightedDegree* [5] qui donne de très bons résultats. Pour Global, nous utilisons une heuristique similaire : nous maintenons pour chaque variable de la contrainte globale le nombre d'incohérences où elle a été impliquée, et nous choisissons en premier les variables ayant participé au plus grand nombre d'incohérences.

Le modèle Avancé a également été implémenté en Picat-SAT [18], qui utilise le solveur SAT Lingeling [2].

Comparaison expérimentale. Nous comparons ces modèles pour différentes valeurs de r dans la table 1, pour le problème consistant à énumérer toutes les affectations des variables $\Delta X_{i,j,k}$ appartenant à une solution

	Modèle Basic en Choco			Modèle Global en Choco					Modèle Avancé de [9]			
					Vérif (R1 à R3)		GAC (R1 à R5)			Choco		SAT
r	#s	t	#pc	#s	t	#pc	t	#pc	#s	t	#pc	t
3	64	0	1 838	28	0	4 865	0	2 229	38	0	1 400	20
4	30	0	11 476	16	1	19 959	0	7 038	16	0	4 830	4
5	26	0	16 565	16	3	57 406	1	10 789	16	1	24 137	6
6	122	2	71 933	16	15	211 577	3	40 058	16	4	62 897	10
7	74	7	334 529	16	79	888 655	11	119 402	16	42	368 998	17
8	32	38	1 879 589	16	1 005	8 809 858	22	217 520	16	332	2 571 491	23
9	282	248	5 395 439	16	3 956	34 339 652	81	637 737	16	589	3 674 192	28
10	218	772	21 795 314	16	9 401	74 481 543	467	2 940 394	16	3 889	18 460 694	37
11	74	1 676	55 959 203	16	>12H		2 498	16 165 905	16	16 272	59 210 153	50
12	-	>12H		16	>12H		4 271	17 066 102	16	>12H		73

TABLE 1 – Comparaison des modèles Basic, Global et Avancé implémentés en Choco 4, ainsi que du modèle Avancé implémenté en Picat-SAT : nombre de solutions (#s), temps (t en secondes) et nombre de points de choix (#pc). Pour Global, on considère 2 propagateurs : Vérif applique les règles R1 à R3, et GAC les règles R1 à R5. La durée maximale d'exécution est de 12H.

(les valeurs des autres variables n'étant pas utiles lors de la seconde étape, nous n'énumérons pas leurs valeurs). Basic trouve beaucoup plus de solutions qu'Avancé et Global car l'abstraction qu'il considère est plus grossière. Global et Avancé trouvent le même nombre de solutions quand $r \geq 4$ mais Global trouve moins de solutions quand $r = 3$ ce qui montre que l'abstraction faite par Avancé est moins bonne. De fait, toutes les solutions trouvées par Global sont cohérentes au niveau concret (alors qu'Avancé trouve 10 solutions qui ne sont pas concrétisables quand $r = 3$).

Nous avons comparé deux propagateurs pour Global : Vérif (qui vérifie la faisabilité en appliquant les règles R1 à R3) et GAC (qui maintient la GAC en appliquant les règles R1 à R5). GAC a tendance à explorer moins de points de choix que Vérif, surtout pour les plus grosses instances. Comme la propagation de GAC est plus coûteuse, GAC n'est plus rapide que Vérif que pour les deux plus grosses instances.

Global (avec Vérif comme avec GAC) est clairement plus rapide que le modèle Avancé implémenté en Choco. En revanche, il n'est pas compétitif avec le modèle Avancé quand on utilise le solveur SAT Lingeling pour le résoudre. En effet, le modèle Avancé est essentiellement composé de variables booléennes, et ses contraintes sont de deux formes uniquement :

— $\sum_{\Delta_i \in S} \Delta_i \neq 1$
— $\sum_{\Delta_i \in S} \Delta_i = v$ où v est une variable entière

où S est un ensemble de variables Booléennes. Ces contraintes sont traduite en formules booléennes en combinant des encodages *at-most* et *at-least* qui sont très étudiés dans la communauté SAT [19]. Cela explique que les solveurs SAT soient particulièrement adaptés pour résoudre le modèle Avancé. De plus, l'utilisation du *Conflict Driven Clause Learning* [15] permet d'apprendre des *nogoods* qui accélèrent probablement la recherche.

Notons cependant que le modèle Global est très simple et est dérivé de façon immédiate de la définition de Midori, tandis que le modèle Avancé a demandé un important travail pour extraire les propriétés de *MixColumns* et les modéliser en termes de contraintes. Notre objectif est de faciliter l'utilisation de la programmation par contraintes par des cryptographes, et *abstractXOR* répond à cet objectif.

6 Conclusion

Les premiers résultats expérimentaux sur Midori128 montrent l'intéret de notre contrainte globale : elle simplifie énormément la conception du modèle (qui est dérivé de façon immédiate à partir de la définition du problème concret) tout en réduisant le nombre de solutions abstraites. Le modèle Basic, qui est également dérivé de façon immédiate à partir de la définition du problème concret, admet beaucoup plus de solutions abstraites et la majorité de ces solutions ne sont pas concrétisables ce qui rend la seconde étape du processus de résolution plus difficile. Le modèle Avancé de [9] trouve le même nombre de solutions abstraites que notre contrainte globale (sauf pour $r = 3$ où il trouve plus de solutions), mais il est bien plus difficile à concevoir car il exploite des propriétés dérivées de l'opération *MixColumns* qui sont loin d'être triviales.

En terme de passage à l'échelle, la contrainte *abstractXOR* améliore les performances de Choco, par rapport au modèle Avancé, mais elle n'est pas compétitive avec un solveur SAT. Notre objectif est donc d'améliorer les performances. Une première piste est d'améliorer les heuristiques d'ordre, qui ont un fort im-

pact sur les résultats. Une seconde piste est d'apprendre des *nogoods* (ou explications) lorsque des incohérences sont détectées afin de pouvoir retourner directement au nœud correspondant à la décision à l'origine de l'échec. Les bonnes performances de SAT nous laissent penser que cela devrait fortement améliorer les performances de Choco.

Enfin, nous souhaitons évaluer les performances d'*abstractXOR* pour rechercher des chemins différentiels maximaux pour AES, et nous comparer avec le modèle de [10, 11], mais aussi pour résoudre d'autres problèmes de cryptanalyse différentielle tels que ceux étudiés dans [6] ou [17], par exemple.

Remerciements : Le travail décrit dans cet article a été réalisé dans le contexte de l'ANR DeCrypt (ANR-18-CE39-0007). Nous remercions Charles Prud'homme pour ses réponses toujours pertinentes à nos nombreuses questions sur Choco.

Références

[1] S. Banik, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita et F. Regazzoni : Midori : A block cipher for low energy. *In ASIACRYPT*, volume 9453 de *LNCS*, pages 411–436. Springer, 2015.

[2] A. Biere : Yet another local search solver and lingeling and friends entering the sat competition 2014. pages 39–40, 01 2014.

[3] E. Biham : New types of cryptoanalytic attacks using related keys (extended abstract). *In EUROCRYPT*, volume 765 de *LNCS*, pages 398–409. Springer, 1993.

[4] E. Biham et A. Shamir : Differential cryptoanalysis of feal and n-hash. *In EUROCRYPT*, volume 547 de *LNCS*, pages 1–16. Springer, 1991.

[5] F. Boussemart, F. Hemery, C. Lecoutre et L. Sais : Boosting systematic search by weighting constraints. *In ECAI*, pages 146–150, 2004.

[6] C. Cid, T. Huang, T. Peyrin, Y. Sasaki et L. Song : Boomerang connectivity table : A new cryptanalysis tool. *In EUROCRYPT*, volume 10821 de *LNCS*, pages 683–714. Springer, 2018.

[7] H. Comon-Lundh et V. Shmatikov : Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. *In 18th Annual IEEE Symposium of Logic in Computer Science, 2003. Proceedings.*, pages 271–280, juin 2003.

[8] FIPS 197 : Advanced Encryption Standard. Federal Information Processing Standards Publication 197, 2001. U.S. Department of Commerce/N.I.S.T.

[9] D. Gérault : *Security Analysis of Contactless Communication Protocols*. Thèse de doctorat, Université Clermont Auvergne, 2018.

[10] D. Gérault, P. Lafourcade, M. Minier et C. Solnon : Revisiting AES related-key differential attacks with constraint programming. *Inf. Process. Lett.*, 139:24–29, 2018.

[11] D. Gérault, M. Minier et C. Solnon : Constraint programming models for chosen key differential cryptanalysis. *In CP*, volume 9892 de *LNCS*, pages 584–601. Springer, 2016.

[12] R. M. Karp : Reducibility among combinatorial problems. *In Symposium on the Complexity of Computer Computations*, pages 85–103, 1972.

[13] L. Knudsen : Truncated and higher order differentials. *In Fast Software Encryption*, pages 196–211. Springer, 1995.

[14] D. E Knuth : Dancing links. *Millenial Perspectives in Computer Science*, 18:4, 2000.

[15] Tero Laitinen, Tommi Junttila et Ilkka Niemelä : Conflict-Driven XOR-Clause Learning (extended version). juillet 2014.

[16] C. Prud'homme, J.-G. Fages et X. Lorca : *Choco Documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., 2016.

[17] Y. Todo, T. Isobe, Y. Hao et W. Meier : Cube attacks on non-blackbox polynomials based on division property. *In CRYPTO*, volume 10403 de *LNCS*, pages 250–279. Springer, 2017.

[18] N.-F. Zhou, H. Kjellerstrand et J. Fruhman : *Constraint Solving and Planning with Picat*. Springer, 2015.

[19] Neng-Fa Zhou et Håkan Kjellerstrand : Optimizing SAT encodings for arithmetic constraints. *In Principles and Practice of Constraint Programming - 23rd International Conference, CP 2017*, volume 10416 de *Lecture Notes in Computer Science*, pages 671–686. Springer, 2017.