Utilisation de la Programmation par contrainte appliquée à la cryptanalyse différentielle

Loïc Rouquette, Christine Solnon

Utilisation de la Programmation par contrainte appliquée à la cryptanalyse différentielle

Loïc Rouquette[1,2], Christine Solnon[2]

[1] LIRIS, UMR5201 CNRS, F-69621 Villeurbanne
[2] Univ. Lyon, INSA Lyon, CITI, INRIA, INSA Lyon, F-69621 Villeurbanne
{loic.rouquette,christine.solnon}@insa-lyon.fr

Mots-clés : *Programmation par contraintes (CP), Problème de satisfaction de contraintes (CSP), Problème d'optimisation avec contraintes (COP), Cryptanalyse différentielle.*

1 Introduction

Les algorithmes de chiffrement symétrique par bloc (tel que AES [3]) garantissent la confidentialité des communications en utilisant une clé secrète K pour chiffrer un texte initial m en un texte chiffré $f_K(m)$, de telle sorte que le texte chiffré puisse être déchiffré en utilisant la même clé. 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 m et m' pendant le chiffrement [1]. L'objectif est de calculer la probabilité p d'observer une différence en sortie δ_C sachant une différence en entrée δ_m. La différence δ_m entre les deux textes m et m' est obtenue en appliquant l'opérateur `XOR` (ou exclusif entre les bits de m et m'), noté $\delta_m = m \oplus m'$. L'ensemble des différences intermédiaires du chiffrement allant de δ_m vers δ_C s'appelle chemin différentiel. Pour réduire la taille de l'espace de recherche à explorer, Knudsen a introduit la notion de différentielle tronquée [5]. L'idée est de regrouper les bits (de δ_m, δ_K et δ_C, mais aussi de tous les états intermédiaires du processus de chiffrement) par séquences de n bits consécutifs : typiquement, $n = 8$ et chaque séquence δ_i de n 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., $\Delta_i = \texttt{faux} \iff \delta_i = 0$ et $\Delta_i = \texttt{vrai} \iff \delta_i \in [1, 2^n - 1]$. Dans un second temps, pour chaque solution abstraite on recherche la solution de probabilité optimale vérifiant les contraintes données par les variables différentielles abstraites Δ_i. Les chiffrements sont composés d'opérations linéaires et non linéaires, appelées boîtes-S. Dans le cas la cryptanalyse différentielle, seules boîtes-S participent au calcul de la probabilité étant donné que la sortie différentielle des opérations linéaires peut être calculée de façon exacte.

2 Contributions

Le chiffrement à l'origine du standard AES adopté par la NIST [1] en 2001 est Rijndael [2]. Bien que les deux chiffrements soient pratiquement identiques, le processus de normalisation a réduit les configurations possibles du chiffrement. Alors que Rijndael propose des tailles de clé de 128, 160, 192, 224 et 256 bits (idem pour les tailles de message), l'AES n'accepte que de trois tailles de clé (128, 192 et 256 bits) et la taille du message est fixée à 128 bits.

Nous implémentons et adaptons le processus de résolution, proposé par [4] pour l'AES, à Rijndael afin de calculer les caractéristiques différentielles en mode related-key. Les attaques related-key permettent, non seulement d'injecter des différences dans le message, mais permettent également d'injecter des différences dans la clé de chiffrement.

1. National Institute of Standards and Technology

Comme indiqué précédemment, le processus de résolution s'effectue en deux étapes dont l'objectif final est de trouver la caractéristique différentielle de plus grande probabilité. Les deux étapes sont modélisées en CP, mais résolues à l'aide de solveurs différents. L'étape 1 portant principalement sur des variables booléennes, est résolue à l'aide d'un solveur SAT : Picat [7]. L'étape 2 est résolue à l'aide du solveur Choco [6].

L'objectif de l'étape 1 est d'énumérer l'ensemble des solutions ayant le moins de boîtes-S possibles. Pour chaque solution de l'étape 1, le modèle de l'étape 2 est utilisé en bornant le domaine des variables à l'aide de leur équivalent booléen de l'étape 1 (relation entre Δ_X et δ_x). Le nombre de boîtes-S actives étant fixé lors de l'étape 1, il est alors possible de borner la probabilité des caractéristiques différentielles concrètes (étape 2) par $p \leq (p_{MAX})^{\#SB}$, où p_{MAX} est la probabilité maximale de passage dans une boîte-S et $\#SB$ le nombre minimal de boîtes-S actives. De plus, à chaque nouvelle solution de l'étape 2 nous pouvons mettre à jour la borne minimale de p, ce qui nous donne : $\max_{s_2 \in \text{solutions étape 2}}(p_{s_2}) \leq p \leq (p_{MAX})^{\#SB}$. L'intégration de ces deux bornes lors de la résolution permet :

1. de ne pas énumérer certaines solutions de l'étape 1 lorsque la solution optimale a été trouvée lors de l'étape 2 (i.e. $p_{sol} = (p_{MAX})^{\#SB}$) ;
2. d'éviter d'énumérer des solutions moins bonnes lors de l'étape 2.

En plus d'améliorer les performances, notre solution permet de garantir l'obtention de la solution optimale. Il est en effet possible, pour certaines configurations, que les solutions minimisant le nombre de boîtes-S actives lors de l'étape 1 ne soient pas les meilleures solutions à utiliser. Dans le cas de l'AES, une boîte-S active (i.e. dont la valeur d'entrée est différente de 0) a une probabilité de transition $p_{SB} \in \{2^{-6}, 2^{-7}\}$. Lors de l'étape 2, les transitions optimales ne peuvent pas être toujours activées, ce qui est dû autres opérations du chiffrement. De ce fait, certaines solutions avec un nombre plus important de boîtes-S, mais dont les probabilités de transition sont plus élevées, peuvent avoir une meilleure probabilité que des solutions activant moins de boîtes-S, mais dont les transitions ont une probabilité plus faible.

Nous améliorons également le modèle de l'étape 2 en décomposant l'opérateur MixColumn de l'AES à l'aide de contraintes tables. De plus, nous utilisons les informations concernant la réduction de domaine entre les variables abstraites et les variables concrètes de façon plus efficace en simplifiant certaines opérations de lors l'étape 2, notamment quand nous savons qu'un octet est fixé à 0.

3 Conclusion

Nous avons étendu et amélioré les modèles initialement proposés dans [4] pour l'AES aux 25 instances de Rijndael. Cela nous a permis de calculer les caractéristiques différentielles optimales des *related-keys* pour toutes les instances de Rijndael.

Remerciements : Le travail décrit dans ce résumé a été réalisé dans le contexte de l'ANR DeCrypt (ANR-18-CE39-0007).

Références

[1] Eli Biham et Adi Shamir. "Differential Cryptanalysis of Feal and N-Hash". en. In : *Advances in Cryptology — EUROCRYPT '91*. Sous la dir. de Donald W. Davies. T. 547. Series Title : Lecture Notes in Computer Science. Berlin, Heidelberg : Springer Berlin Heidelberg, 1991, p. 1-16. ISBN : 978-3-540-54620-7. DOI : 10.1007/3-540-46416-6_1. URL : http://link.springer.com/10.1007/3-540-46416-6_1 (visité le 13/08/2020).

[2] Joan Daemen. "The Rijndael Block Cipher". en. In : (), p. 47.

[3] Joan Daemen et Vincent Rijmen. *The design of Rijndael : AES - the advanced encryption standard*. en. OCLC : 751525895. Berlin ; London : Springer, 2011. ISBN : 978-3-642-07646-6.

[4] David Gerault et al. "Computing AES related-key differential characteristics with constraint programming". en. In : *Artificial Intelligence* 278 (jan. 2020), p. 103183. ISSN : 00043702. DOI : 10.1016/j.artint.2019.103183. URL : https://linkinghub.elsevier.com/retrieve/pii/S0004370218303631 (visité le 04/05/2020).

[5] Lars R. Knudsen. "Truncated and higher order differentials". en. In : *Fast Software Encryption*. Sous la dir. de G. Goos et al. T. 1008. Berlin, Heidelberg : Springer Berlin Heidelberg, 1995, p. 196-211. ISBN : 978-3-540-60590-4 978-3-540-47809-6. DOI : 10.1007/3-540-60590-8_16. URL : http://link.springer.com/10.1007/3-540-60590-8_16 (visité le 16/05/2019).

[6] Charles Prud'homme, Jean-Guillaume Fages et Xavier Lorca. *Choco Solver Documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S. 2016. URL : http://www.choco-solver.org.

[7] Neng-Fa Zhou, Håkan Kjellerstrand et Jonathan Fruhman. *Constraint Solving and Planning with Picat*. en. SpringerBriefs in Intelligent Systems. Cham : Springer International Publishing, 2015. ISBN : 978-3-319-25881-2 978-3-319-25883-6. DOI : 10.1007/978-3-319-25883-6. URL : http://link.springer.com/10.1007/978-3-319-25883-6 (visité le 14/01/2022).