Université de Lorraine
54506 VANDOEUVRE-LES-NANCY
Campus Aiguillettes

Rapport pour l'UE Initiation à la Recherche

Cryptanalyse différentielle : Les attaques Boomerang

MASTER INFORMATIQUE 2024-2025

Julien VOLSFELTS
Neil HADIBI
Aéna ARIA

Encadrante : **Marine MINIER** (LORIA, Université de Lorraine)

Janvier-Mai 2025

Décharge de responsabilités

L'Université de Lorraine n'entend donner ni approbation ni improbation aux opinions émises dans ce rapport, ces opinions devant être considérées comme propres à leur auteur.

Remerciements

Nous tenons particulièrement à remercier Madame Marine MINIER, notre encadrante, pour sa bienveillance face à nos fréquentes interrogations et l'aide qu'elle nous a apporté au cours de cette UE. Il nous paraît évident que son encadrement nous a permis de comprendre un sujet d'une complexité technique plutôt conséquente pour nous. Nous la remercions également d'avoir accepté d'encadrer un second groupe suite à notre demande, et d'avoir adapté le sujet original pour nous.

Nous remercions également Sarah ATTATFA, Nathan REGAUDIE et Esteban DOUILLET avec qui nous avons pu collaborer pour la première moitié du sujet. Leurs remarques et questions pertinentes nous ont permis de mieux comprendre les points complexes de ce sujet.

Table des matières

1 Introduction

Dans un monde où les objets connectés et les systèmes embarqués sont de plus en plus répandus, la sécurité des données qu'ils manipulent est un facteur très important comme nous l'explique l'article *Improving the reliability of SKINNY-Hash for IoT applications with less power overhead* [BM25]. Ces écosystèmes présentent des contraintes de calcul et de mémoire et sont ainsi très limités. L'algorithme Skinny, en particulier la version Skinny 128-256 est particulièrement adapté pour répondre à ces exigences.

Cet algorithme a été pensé pour résister aux attaques dites différentielles, cependant un autre type d'attaque similaire pourrait le compromettre : les attaques Boomerang. Nous nous sommes donc intéressés dans notre travail à la résistance de l'algorithme Skinny 128-256 face à l'attaque Boomerang. Pour ce faire, nous nous sommes appuyés sur des outils proposés par Hossein Hadipour, Patrick Derbez et Loïc Rouquette.

Dans ce rapport, nous allons dans un premier temps expliquer le principe de la cryptographie symétrique et du chiffrement par bloc (section 2). Ensuite, nous étudierons la cryptanalyse différentielle (section 3) et le principe des attaques boomerang (section 4). Nous pourrons alors introduire l'algorithme étudié, Skinny 128-256 (section 5). Nous détaillerons alors le fonctionnement d'un outil automatique de cryptanalyse Boomerang (section 6), les deux outils de Hadipour et Derbez avec un comparatif de résultats (section 7) et nous finirons par l'adaptation de l'outil de Rouquette pour le transformer de Skinnyee à Skinny (section 8) avant de faire une conclusion (section 9).

2 Cryptographie Symétrique et Chiffrement par Bloc

2.1 Définition

La cryptographie symétrique, aussi appelée cryptographie à clé secrète, est la plus ancienne forme de chiffrement. Elle permet de chiffrer et déchiffrer des messages à l'aide d'une seule et même clé, la clé secrète. Ainsi, l'expéditeur et le destinataire ont besoin de se partager cette même clé par un autre canal sécurisé. Si la sécurité lors de l'échange de clé n'est pas garantie, toute communication devient vulnérable.

Le chiffrement par bloc est une des deux grandes catégories de chiffrement symétrique (bloc et flot). Chaque message est divisé en blocs de taille fixe entre 64 et 512 bits, et chaque bloc est chiffré indépendamment à l'aide d'une clé secrète, d'un algorithme de chiffrement, et d'un mode de chiffrement. Ces modes définissent la manière dont les bloc sont traités pour former le chiffré final. Par ailleurs, certains modes comme ECB peuvent apporter des failles de sécurité.

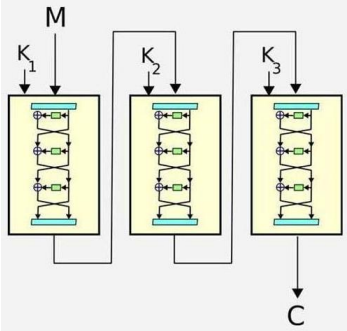
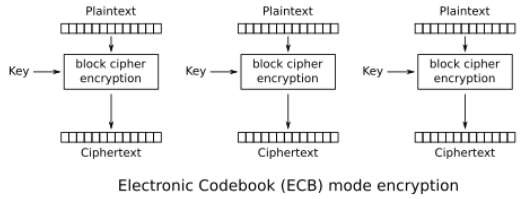
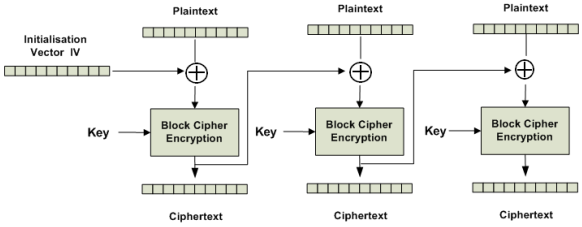


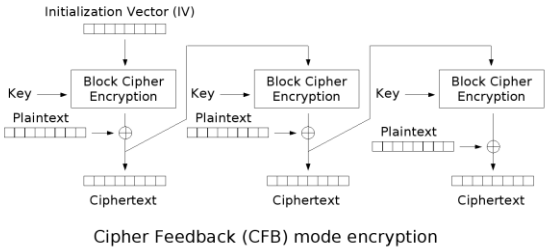
FIGURE 1 – Schéma de chiffrement par bloc [Wik]



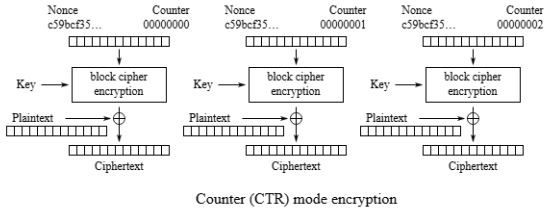
(a) ECB (Electronic Code Book)



(b) CBC (Cipher Block Chaining)



(c) CFB (CipherFeedback)



(d) CTR (Counter Mode)

FIGURE 2 – Méthodes de chiffrement par bloc

Le mode ECB chiffre chaque bloc indépendamment. Le mode CBC applique une opération XOR bit à bit avec le bloc chiffré précédent. Le mode CFB transforme le chiffrement par bloc en chiffrement de flux, et le mode CTR chiffre chaque bloc en combinant le texte clair avec un compteur chiffré.

2.2 Principe de fonctionnement

Tout d'abord, l'expéditeur applique un algorithme de chiffrement avec une clé secrète pour transformer le bloc de message clair en un texte chiffré. Le destinataire, quant à lui, reçoit le message chiffré et utilise cette même clé pour récupérer le bloc de message original. On obtient alors la relation suivante :

$$C = E_K(M)$$
$$M = E_K^{-1}(C)$$

Soient C le message chiffré, M le message en clair, E l'algorithme de chiffrement et K la clé secrète.

Selon le mode de chiffrements utilisé, les blocs vont être chiffrés et déchiffrés différemment. Par exemple, pour ECB :

$$C_i = E_K(M_i)$$
$$M_i = E_K^{-1}(C_i)$$

avec M_i et C_i le bloc à l'indice i du message clair (resp. chiffré).

CBC utilise le chiffrement du bloc précédent pour chiffrer le bloc courant. On obtient la relation suivante :

$$C_i = E_K(M_i \bigoplus C_i - 1)$$

$$M_i = E_K^{-1}(C_i) \bigoplus C_{i-1}$$

avec $C_0 = IV$ (un vecteur d'initialisation aléatoire). CFB et CTR sont plus ou moins similaires dans la méthode de chiffrement, avec pour CTR et CFB :

$$C_i = M_i \bigoplus E_K(CTR_i) \text{ et } M_i = C_i \bigoplus E_K(CTR_i)$$
$$C_i = M_i \bigoplus E_K(C_{i-1}) \text{ et } M_i = C_i \bigoplus E_K(C_{i-1})$$

Le schéma ci-dessous montre un réseau de Feistel, une structure largement utilisée dans les algorithme de chiffrement par bloc. Un réseau de Feistel est subdivisé en plusieurs tours et décompose un bloc M en deux moitiés L et R : $M = L_0|R_0$ (par exemple, un bloc de 64 bits peut être divisé en 2 partie de 32bits). À chaque tour i, on applique la fonction suivante sur ces 2 moitiés :

$$L_{i+1} = R_i$$

$$R_{i+1} = L_i \bigoplus F_K(R_i)$$

La fonction F_K est une fonction de ronde paramétrée par une clé intermédiaire, cette clé est tirée de la clé originale par une méthode appelée "key schedule", ce qui permet de renforcer la robustesse de l'algorithme. Ainsi la moitié des données sont encodée avec la clef, puis ce résultat est mélangé avec l'autre moitié du bloc grâce à l'opération XOR. Cette structure a l'avantage majeur que le déchiffrement suit exactement les mêmes étapes que le chiffrement, en inversant l'ordre des clés de ronde.

Cela garantit l'inversibilité du chiffrement même si F_K n'est pas inversible, ce qui est une des grandes forces du réseau de Feistel et sert donc de base structurelle pour construire des algorithmes de chiffrement robuste.

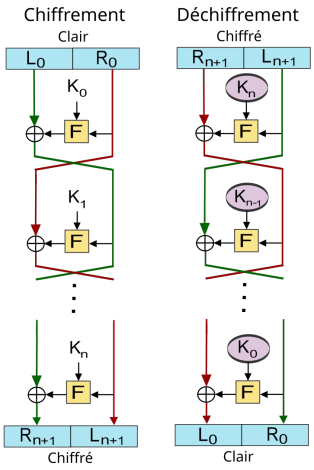


FIGURE 3 – Réseau de Feistel à n tours utilisant l'opérateur XOR [Wik24]

3 Cryptanalyse Différentielle

La cryptanalyse différentielle est une technique d'attaque introduite au début des années 1990 par Eli Biham et Adi Shamir dans leur publication *Differential Cryptanalysis of the Data Encryption Standard* (Springer 1993). [BS90]

La cryptanalyse différentielle consiste à étudier une paire de texte en clair avec une différence donnée et à observer la manière dont cette différence se propage à travers le chiffrement, et de déterminer quelle est la clé secrète la plus probable à l'aide des informations obtenues.

L'opérateur de différence la plus couramment utilisée pour cela est le XOR, en raison de sa simplicité et de ses propriétées algébriques. Or, ce n'est pas la seule opération utilisée, et on peut aussi utiliser une soustraction modulaire ou une différence arithmétique. Ce choix se fait surtout en fonction du type de données à traiter et de la structure de l'algorithme à analyser.

Une attaque différentielle repose sur l'étude des différences en sortie du chiffrement avec les différences introduites en entrée dans les messages en clairs appelées "différentielles". En analysant statistiquement un grand nombre de paires de texte, on peut identifier des motifs pouvant indiquer un biais dans la propagation des différences, ce qui permet à l'attaquant d'identifier certaines informations non-négligeables sur certaines sous-parties de la clé secrète.

Les propriétés statistiques des différentielles dépendent fortement de la nature des boîtes de substitution S (boîte-S ou S-Box) qui compose l'algorithme de chiffrement. Pour chacune de ces boîtes-S, on peut mesurer l'effet d'une différence Δ_X placée en entrée sur la différence de sortie Δ_Y. Pour cela, on associe la relation suivante pour calculer une paire de différentielles (Δ_X, Δ_Y) :

$$\Delta_Y = S(X) \oplus S(X \oplus \Delta_X)$$

Où Δ_Y correspond à la différence en sortie, Δ_X, la différence appliquée au texte en entrée et S la fonction de substitution. Cette relation permet de construire des tables de différences pour chaque S-Box, qui seront ensuite utilisées pour identifier les différentielles les plus probables. Cette table permet de mesurer pour chaque différence en entrée Δ_X, la fréquence d'apparition des différences en sortie donc Δ_Y. Cette table est définit comme suit :

$$DDT[\Delta_Y][\Delta_X] = |\{X | S(X) \bigoplus S(X \bigoplus \Delta_X) = \Delta_Y\}|$$

Elle regroupe toutes les sorties les plus probables de valeurs ΔY pour une entrée ΔX. Une valeur de DDT élevée induit nécessairement une probabilité de différentiel élevée. Cette probabilité est estimé par :

$$Pr[S(X) \bigoplus S(X \bigoplus \Delta_X) = \Delta_Y] = \frac{DDT[\Delta_Y][\Delta_X]}{2^n}$$

Les différentielles sont alors classés par ordre d'importance, ce qui joue un rôle important dans la construction de caractéristique différentielle. Ainsi, le choix des différences appliquées en entrée Δ_X est donc très important pour le reste de l'attaque. Ce choix détermine l'efficacité globale de l'attaque à travers la manière dont les différences vont se propager dans l'algorithme. Un bon choix de différence maximise alors la probabilité d'obtenir un différentiel spécifique, et donc de réussir l'attaque.

4 Attaque Boomerang

L'attaque boomerang est une attaque proposée par David Wagner en 1999 dans sa publication *The Boomerang Attack*[Wag99].

Cette attaque peut être vu comme une extension de la cryptanalyse différentielle. Celle-ci est utile tout particulièrement pour les algorithmes de chiffrement résistant à la cryptanalyse différentielle classique dans lequels aucune caractéristique différentielle ne présente une probabilité suffisamment élevée sur l'ensemble des tours.

Le principe de cette attaque repose tout d'abord sur l'introduction d'une coupure virtuelle dans l'algorithme de chiffrement E. Pour cela on considère que l'algorithme de chiffrement est symétrique. Cette coupure permet d'observer plus facilement comment les différences se propagent dans l'algorithme de chiffrement. Il est donc décomposer en 2 parties : une première moitié avant E_0 et une seconde après la coupure E_1. Ensuite, considérons une paire de textes en clair $(M1, M2)$ avec :

$$M2 = M1 \bigoplus \Delta_{M1}$$

On récupère ensuite les chiffrés C1 et C2, respectivement $E_1(E_0(M1))$ et $E_1(E_0(M2))$ qu'on décale de γ. On obtient alors 2 nouveaux chiffrés $(C3, C4)$ que nous allons déchiffrer pour obtenir les blocs de message en clair $(M3, M4)$ avec :

$$M4 = M3 \bigoplus \Delta_{M3}$$

On se retrouve ainsi avec un total de quatre blocs à la fin de l'attaque, organisés en deux paires de messages, chacune étant associée à une différence spécifique, respectivement : Δ_{M1} et Δ_{M3}. On associe à chacune de ces différences, entre E_0 et E_1, une caractéristique différentielle. Soit :

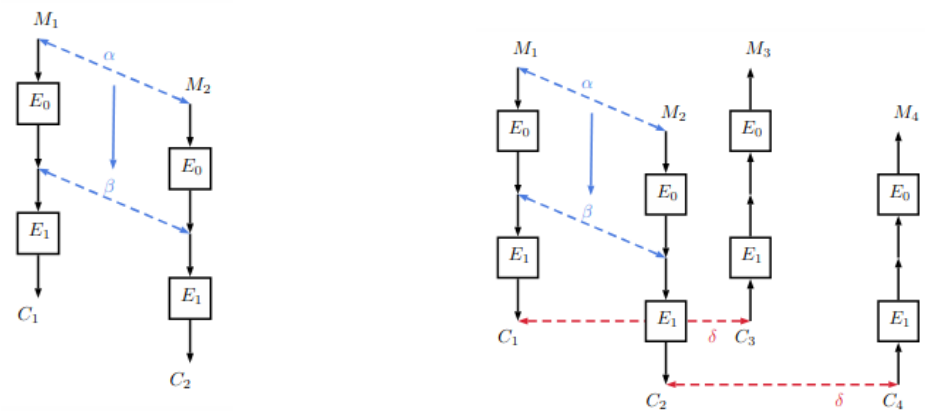
$$\alpha \to \beta$$

$$\delta \to \gamma$$

On suppose ici que la paire $(M1, M2)$ est choisie de manière a être compatible avec la caractéristique différentielle E_0. Les 2 caractéristique différentielles correspond au résultat de la propagation de la différence dans l'algorithme. Les 2 blocs chiffrés en sortie de la première partie de l'algorithme $E_0(M1)$ et $E_0(M2)$ sont ensuite traités dans la seconde partie de l'algorithme, produisant les chiffrés suivant :

$$C1 = E_1(E_0(M1))$$

$$C2 = E_1(E_0(M2))$$



(a) Première partie du carré différentiel

(b) Première et seconde partie du carré différentiel

FIGURE 4 – Carrés Différentiels

Ces chiffrés sont ensuite décalé via une différence notée δ. Cette différence nous permet d'obtenir deux nouveaux chiffrés respectivement $(C3, C4)$ avec

$$C3 = C1 \oplus \delta$$

$$C4 = C2 \oplus \delta$$

Ces 2 blocs sont ensuite déchiffrés par l'inverse de E_1, nous ramenant à un état intermédiaire. Nous obtenons à ce stade, une différence de γ entre les paires $(E_0(M1), E_1^{-1}(C3))$ et $(E_0(M2), E_1^{-1}(C4))$ à condition que la caractéristique différentielle pour E_1^{-1} soit respectée. Il ne reste alors qu'à appliquer E_0^{-1} pour obtenir les messages $M3$ et $M4$.

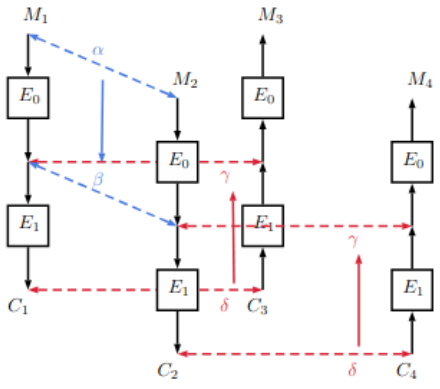


FIGURE 5 – Carré Différentiel partiel

Si toutes les conditions sont remplies :

$$\alpha \to \beta$$

$$\delta \to \gamma$$

$$M3 \bigoplus M4 = M1 \bigoplus M2 = \Delta_{M1} = \Delta_{M3} = \alpha$$

c'est-à-dire que la paire $(M1, M2)$ respecte la caractéristique différentielle de la première moitié E_0, que les paires $(M1, M3)$ et $(M2, M4)$ sont compatibles avec la caractéristique différentielle inverse de E_1 et que la paire $(M3, M4)$ obtenue après les étapes de déchiffrement présente exactement la même différence que la paire d'entrée $(M1, M2)$. Alors les 2 textes devraient différer exactement de la même manière que les 2 blocs d'origine. D'où le nom "Attaque Boomerang" : la différence revient à son point de départ :

$$M3 \bigoplus M4 = M1 \bigoplus M2 = \Delta_{M1} = \Delta_{M3} = \alpha$$

Les deux caractéristiques différentielles permettent de construire un carré appelé "carré différentiel". Ce carré différentiel a pour but de valider ou de négliger des hypothèses sur certaines portions de la clé secrète que nous cherchons, à partir de probabilité non négligeable. C'est en répétant le même processus sur un grand nombre de paires de texte, que l'attaquant peut affiner ses hypothèses sur la clé secrète.

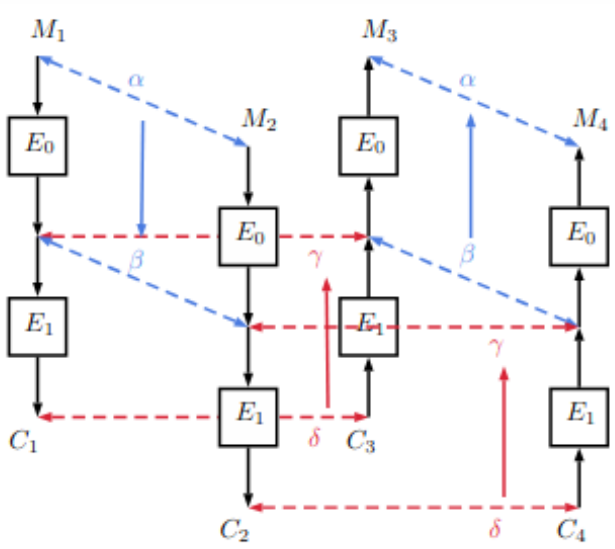


FIGURE 6 – Carré Différentiel complet

Les deux caractéristiques différentielles $\alpha \to \beta$ et $\delta \to \gamma$ possèdent respectivement une probabilité p et une probabilité q de réussite. Comme le chemin se fait deux fois en parallèle (un pour $M1 \to M3$ et un pour $M2 \to M4$) la probabilité total de réussite dans un carré différentiel est p^2q^2 puisque

chaque chemin connaît une probabilité de réussite égal à $p \times q$. Plus p ou q est élevé et plus la probabilité que le carré "revienne" (l'effet boomerang) est élevée.

Dans un contexte plus général où l'algorithme de chiffrement est divisé en 3 parties : $E = E_0 \cdot E_m \cdot E_1$. la probabilité total est égale à $(p \times p) \cdot (q \times q) \cdot r$ où r est la probabilité que la perturbation/différence traverse correctement la partie intermédiaire E_m. Cette probabilité peut être noté :

$$r = \Pr\left[E_m^{-1}\left(E_m(x_1) \oplus \gamma\right) \oplus E_m^{-1}\left(E_m(x_1 \oplus \beta) \oplus \gamma\right) = \beta\right]$$

La différence β est préservée lors de la traversée de E_m dans les deux chemins ($M1 \rightarrow M3$ et $M2 \rightarrow M4$).
Ainsi, la réussite de l'attaque boomerang généralisée dépend d'une part de la qualité des caractéristiques différentielles dans les parties E_0 et E_1 mais aussi de la capacité à conserver la différence dans la partie intermédiaire E_m.

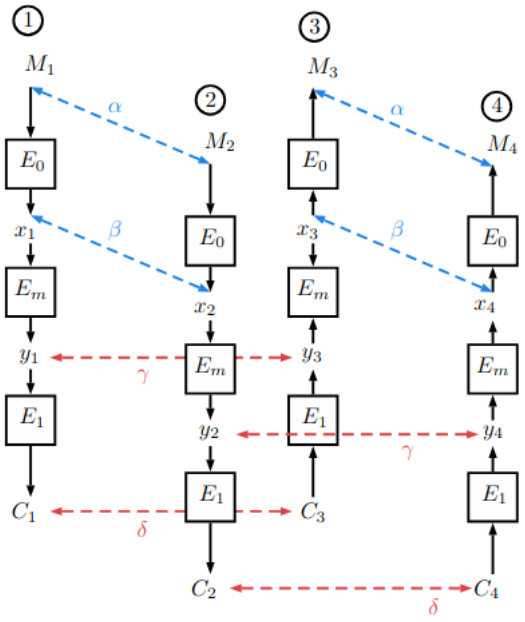


FIGURE 7 – Carré Différentiel dans une attaque Boomerang généralisée

5 SKINNY 128-256

Skinny est une famille de chiffremment dit par bloc légers, conçue pour offrir un bon compromis entre sécurité et efficacité en particulier lorsqu'il s'agit d'environnement technologique comme les systèmes embarqué ou encore l'IoT (*Internet of Things*) qui impliquent certaines contraintes strictes de performances et de consommation.

Développé par Beierle, Jean, Peyrin, Sudre et Wang, ils expliquent dans leur article *The SKINNY Family of Block Ciphers and its Low-Latency Variant MANTIS* présentée à la conférence CRYPTO 2016[Wik] ce qu'est l'algorithme Skinny

Skinny offre un niveau de sécurité comparable à certains algorithme de chiffrement avancé comme *AES* avec un niveau de complexité d'implémentation matérielle et logicielle moins élevé. Nous nous concentrons dans notre projet sur une version spécifique de l'algorithme Skinny : *Skinny 128-256*. Cette version fonctionne sur des blocs de 128 bits et utilise une clé de 256 bits appelé *Tweakey* qui combine la clé de chiffrement et une valeur publique, le tweak qui correspond à un paramètre secondaire qui peut être modifié sans toucher à la clé principale.

Le chiffrement s'effectue en un nombre de tour (*Rounds*) avec un maximum de 56 tours. Chaque tour effectue sur une série d'opérations qui suit un **réseau de substitution-permutation**. Cette structure *SPN (Substitution permutation Network)* effectue une série de quatre opérations principales à chaque tour :

1. *SubCells* : Chaque mot de 8 bits appelé *byte* du bloc connaît une transformation non linéaire par la S-Box de taille 8×8. Si x est un byte de 8 bits alors la S-Box applique :

   $$x \to S(x)$$

   Cette transformation est appliqué sur tous les bytes qui composent le bloc de 128 bits, pour un total de 16 bytes. La S-Box est un élément primordial dans la structure de l'algorithme Skinny, et celle-ci est choisie pour ses propriétés de résistance aux attaques différentielles et linéaires.

2. *AddConstants* : Cette étape ajoute des constantes spécifiques à chaque tour du chiffrement par l'opération **XOR**. Ces constantes sont utilisées pour renforcer la diffusion à travers le chiffrement. Ce principe de diffusion, appelé le principe de confusion-diffusion de Shannon, permet de complexifier la relation entre la texte chiffré et la clé utilisé lors du chiffrement (Confusion) mais aussi de répandre l'effet qu'à 1 seul bit du texte en clair sur de nombreux bits du texte chiffré (Diffusion).

3. *AddRoundTweakey* : Cette étape génère une sous-clé spécifique à chaque tour à partir du Tweakey global et d'un algorithme de cadencement de clé (*Key Scheduler*). Cette clé fait l'objet d'un bloc de données après avoir été injecté via l'opération **XOR**.
   L'équation est :

   $\text{State} \leftarrow \text{State} \oplus TK_i$ (l'opération $\oplus$ ne s'applique qu'aux deux premières lignes de l'état)

où TK_i est la clé du tour i issue du tweakey

4. *ShiftRows* : Cette étape permute les positions de chaque bytes sur le bloc. Ce bloc peut être vu comme une matrice de 4x4 de mots de 8 bits où chaque ligne est fait l'objet d'un décalage circulaire. Le décalage dépend de l'indice de la ligne : La première ligne n'est pas décalée, La deuxième ligne est décalée d'une position vers la gauche, la troisième de 2 positions vers la gauche et la quatrième de 3 positions vers la gauche.

5. *MixColumns* : Cette étape applique une transformation linéaire appliquée colonne par colonne sur le bloc de données, dans le but de propager une informations verticalement pour assurer la diffusion. L'opération associée à cette transformation est généralement représentée par une multiplication matricielle sur des bytes dans un corps fini : $\mathbb{F}_2^8$. Ce corps représente simplement l'ensemble de tous les vecteur de 8 bits et peut être représenté par

$$\mathbb{F}_2^8 = \{(b_0, b_1, b_2, b_3) | b_i \in \{0, 1\}\}$$

où l'ensemble $\{0, 1\}$ représente le corps fini $\mathbb{F}_2$ composé de l'addition (XOR) et la multiplication. L'opération MixColumns est la suivante :

$$ColonneSortante = M * ColonneEntrante$$

où M est la matrice de 4×4, $ColonneEntrante$ représente la colonne d'entrée, et plus précisément un vecteur colonne de 4 bytes.

Ainsi les caractéristiques de l'algorithme Skinny rendent l'algorithme particulièrement intéressant pour des analyses différentielles, en particulier l'attaque boomerang. On peut à partir des S-Box construire la table différentielle.

De plus, la présence de transformations non-linéaires (S-Box) et linéaires (MixColumns et ShiftRows) permet aux attaquants de définir des paires de textes en clair qui vont propager des différences sur plusieurs tours, puisque les caractéristiques différentielles peuvent être représentées précisément, ce qui est primordial dans une attaque de type boomerang.

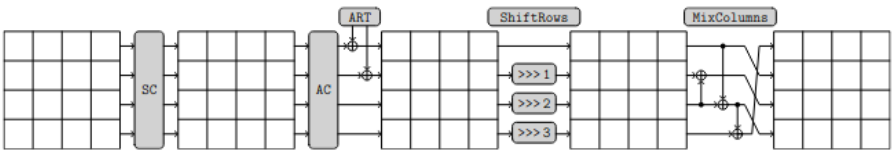


FIGURE 8 – Fonction de tour de Skinny 128-256 [B+16]

La figure 8 représente la fonction de tour de l'algorithme Skinny composée des transformations *SubCells (SC)*, *AddConstants (AC)*, *AddRoundTweakey (ART)*, *ShiftRows (SR)* et *MixColumns (MC)*.

6 Outils Automatiques

Il existe des outils automatiques pour estimer les probabilités d'attaque boomerang pour un algorithme donné. Ces outils sont très complexe, et nous n'étudions pas leur caractéristiques précises d'implémentation. Cependant, il est important d'expliquer leur fonctionnement général pour les utiliser et les comparer.

6.1 Step 1

Les outils que nous allons manipuler sont tous constitués de deux étapes distinctes : la *Step 1* et la *Step 2*. Dans la *Step 1*, l'outil effectue une approximation de la propagation des différences pour trouver les chemins d'activation des S-Box. Dans l'outil proposé par Hossein Hadipour que nous verrons dans la section suivante, la sortie de cette étape affiche les différences ainsi que les valeurs de clé actives utilisées pour construire la caractéristique différentielle. Voici ci-dessous le code python définissant les données d'entrée de cette étape et un exemple de résultat associé.

```
1  def main():
2      r0, w0 = 3, 4
3      r1, w1 = 2, 4
4      rm, wm = 1, 2
5      time_limit = -1
6      skinny = Skinnyrk(r0, r1, rm, w0, w1, wm, 1, time_limit)
7      skinny.make_model()
8      skinny.solve_model()
```

Dans ce script l'algorithme de chiffrement est découpé en trois parties : r0, rm et r1 avec r0 la partie supérieure (E_0), rm la partie centrale (E_m) et r1 la partie inférieure (E_1) . À chacune de ces parties est associée un poids "w". Ensuite le code construit un problème d'optimisation associée à l'algorithme de chiffrement. Ce problème est ensuite résolu par *solve.model()*. Ci-dessous, voici un exemple de sortie.

```
Upper Trail:

 x0:    0000000000000000            y0: 0000000000000000
 x1:    0000000000000000            y1: 0000000000000000
 x2:    0000000000000000            y2: 0000000000000000
 x3:    0000000000000000            y3: 0000100000000000
 x4:    0000000001000000            y4: 0000000001000000
 x5:    0001000000010001            y5: 0001001000010001
 x6:    0111000101010101            y6: 0111000101010101
 x7:    1010011111010010            y7: 1010011111010010
 x8:    1111101011111111
 tku1:    00000000000000000100000000000000
 tku2:    00000000000000008000000000000000


 Lower Trail:

 x0:    0000010100100100            y0: 0000000100100100
 x1:    0000000000001000            y1: 0000000000001000
 x2:    0001000000000000            y2: 0000000000000000
 x3:    0000000000000000            y3: 0000000000000000
 x4:    0000000000000000            y4: 0000000000000000
 x5:    0000000000000000            y5: 0000000000000000
 x6:    0000000000000000            y6: 0100000000000000
 x7:    0100010000000100
 tkl1:    00000000000600000000000000000000
 tkl2:    00000000008100000000000000000000

 Middle Rounds:

 xu5:    0001000000010001
 xl3:    0000000000000000
 tku1:    00000000000000000100000000000000
 tku2:    00000000000000008000000000000000
 tkl1:    00000000000600000000000000000000
 tkl2:    00000000008100000000000000000000
 Time used = 2.4447128772735596
```

Le résultat obtenu est strucuré en trois parties *Upper Trail, Lower Trail et Middle Rounds*. On y observe les valeurs des états *(x et y)* ainsi que les sous-clés actives *(tku1,tku2,tkl1,tkl2)*. Ces différentes données représentent les chemins d'activations des S-Boxes et la façon dont les différences ce propagent dans l'algorithme de chiffrement.

6.2 Step 2

Dans la *Step 2*, l'outil évalue les poids réels et effectue un calcul précis de probabilité associée à chaque caractéristique à partir des valeurs de l'état $\mathbf{x}$ et des valeurs des sous-clés actives *(tku1,tku2,tkl1,tkl2)*.

On a le fichier .yaml suivant qui contient nos données :

```yaml
blocksize: 128
rounds: 7
mode: 2
sweight: 0
endweight: 128
timelimit: -1
fixedVariables:
    tk1_0 : "00020000000000000000000000000000"
    tk2_0 : "00800000000000000000000000000000"
    x_0 : "00200000010000000000000100000100"
    x_7 : "00000000000000000006000000000000"
```

Dans ce fichier on fixe un certains nombres de conditions pour rechercher des caractéristiques différentielles sur l'algorithme Skinny 128-256. On défini tout d'abord une taille de bloc de 128 bits, un nombre de tour égale à 7 et un intervalle de pondération allant de 0 à 128. Cette pondération correspond à un coût différentiel, c'est-à-dire la somme des poids des différentes opérations (S-Box actives) dans le chiffrement.

Chaque opération est associé à une probabilité très faible, exprimée sous forme logarithmique négative en base 2 sous la forme 2^{-x} avec $x \in \mathbb{R}^+$. Certaines variables sont fixées : les variables "tk1_0" et "tk2_0" représentes les tweakeys et "x_0" et "x_7" les états d'entrée et de sortie du chiffrement. Une fois l'outil exécuté, on obtient :

```
Current weight: 13.0
Number of trails: 2
        Current Probability: 2^(-12.0)
Time used = 0.6695 seconds

Current weight: 16.415
Number of trails: 2
        Current Probability: 2^(-11.87071376879294)
Time used = 1.4280 seconds

Current weight: 18.0
Number of trails: 4
        Current Probability: 2^(-11.79054359385267)
Time used = 2.1713 seconds

Current weight: 21.0
Number of trails: 4
        Current Probability: 2^(-11.780828459425601)
Time used = 2.9364 seconds

Current weight: 22.0
```

```
Number of trails: 2
        Current Probability: 2^(-11.778409863681768)
Time used = 3.6714 seconds

Current weight: 23.0
Number of trails: 2
        Current Probability: 2^(-11.777202084601516)
Time used = 4.3841 seconds
```

Lors de l'exécution, l'outil génère progressivement des caractéristiques valides de plus en plus complexe où chaque résultat affiché représente une caractéristique différentielle trouvée. Chacune de ces caractéristiques possède un coût (un certain poids). À chacun de ses coûts on associe une probabilité, par exemple une caractéristique de poids égale à 13.0 correspond à une probabilité approximative égale à 2^{-12}. Le nombre de trails indique combien de chemins compatibles avec les contraintes fixées conduisent à ce résultat. Nous pouvons observé une évolution des poids, passant initialement d'un poids égal à 13 à un poids égal à 23. De plus les probabilité sont croissantes, passant de 2^{-12} à $2^{-11.77}$.

7 Outils Hadipour/Derbez

Hossein Hadipour a développé des outils conçus pour la cryptanalyse boomerang des algorithmes de chiffremment par blocs légers comme expliqué dans son article *Revisiting Differential-Linear Attacks via a Boomerang Perspective With Application to AES, Ascon, CLEFIA, SKINNY, PRESENT, KNOT, TWINE, WARP, LBlock, Simeck, and SERPENT* [HDE24].

Un autre outil réalisé par Patrick Derbez et Pierre-Alain Fouque permet aussi d'effectuer des attaques boomerang sur Skinny 128-256 [Der21].

Les outils mis en place par Hadipour et Derbez repose sur l'utilisation d'outils MILP pour formuler et résoudre des contraintes différentielles et/ou boomerang. Ces outils sont des programmes qui suivent les caractéristiques suivantes : une fonction objectif $f^T x$ où f est un vecteur colonne des coefficients constant et x vecteur colonne des inconnues à optimiser. Les problèmes MILP sont aussi constitué de contraintes linéaires et des restistrictions sur certaines composantes de x pour avoir des valeurs entières. L'équation est la suivante :

$$min_x f^t x \text{ subject to } \begin{cases} x(intcon) \\ A \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub \end{cases}$$

où $f^T x$ est la fonction objectif à minimiser, représentant par exemple dans le cas de la cryptanalyse le nombre total de bits actifs pour mesurer la diffusion ou des différences propagées.

Chaque variable du vecteur x est compris dans l'intervalle

$$[lb, ub]$$

où certaines d'entre elles (spécifié par intcon) connaissent une restriction vers une valeur entière. A, A_eq, b et b_eq représentent respectivement les matrices et vecteurs associées aux contraintes linéaires.

Les outils de Hadipour et Derbez simulent la boîte S, ainsi que les permutations et injections de clé. Chaque S-Box est modélisée par un ensemble de contraintes logique est traduitent en contraintes MILP, les permutations de colonnes ou de bits dans l'algorithme Skinny sont exprimées en réarrangeant les indices dans le vecteurs x ou en utilisant des matrices de permutations.

Ces outils joue un rôle essentiel dans la découverte de nouvelles attaques Boomerang contre des algorithmes de chiffrements à bloc léger comme Skinny et ses variantes, et ont permis de standardiser et automatiser l'analyse de nombreux algorithmes de chiffrement légers. Ils facilitent la génération automatique de caractéristiques différentielles optimales. Autrement dit, plus besoin de tester manuellement des milliers de cas possibles.

La flexibilité est aussi mise en avant par le fait que les mêmes modèles et solveurs utilisés pour une variante d'algorithme de chiffrement peuvent être utilisés sur une autre variante en ne changeant que certains paramètres nécessaires comme les S-Box ou les perturbations. Ainsi, les attaques ne sont pas que simulées mais sont aussi optimisées.

Skinny-128 est bien adapté à la modélisation MILP et rentre donc bien dans l'ensemble des algorithmes de chiffrement pouvant être traités par ces outils. La S-Box sera exploitée pour générer des contraintes MILP et les permutations comme ShifRows ou encore MixColumns modélisées par des matrices linéaires, ou encore pour générer des permutations d'indices qui seront directement traitables par contraintes MILP. Les outils de Hadipour et Derbez peuvent alors découvrir les caractéristiques différentielles optimisées associées.

7.1 Résultats pour Skinny 128-256

NB TOUR	PR	Temps exécucution Hadipour (s)	Temps exécution Derbez (s)
5	2^(-22.00)	1.7641	1.322
6	2^(-15.56)	1.9350	1.614
7	2^(-11.78)	2.3847	2.045
8	2^(-13.45)	11.554	10.013
9	2^(-12.78)	24.0362	22.237

Ce tableau présente les résultats d'attaques boomerang sur le chiffrement Skinny 128-256 avec différents nombres de tours. Chaque probabilité de différentiation PR est exprimée en puissance de 2 négative, comme $2^{-11.78}$.

Les deux dernières colonnes du tableaux indiquent les temps d'exécutions correspondants à chaque outil en fonction du nombre de tours. On remarque sur ce tableau que le temps d'exécution augmente de manière exponentielle en fonction du nombre de tour. Ces outils sont très gourmands en temps de calcul.

8 SKINNYee

Dans cette partie nous nous concentrons sur un outil proposé par Loïc Rouquette initialement conçu pour fonctionner sur Skinnyee, une variante de Skinny avec des caractéristiques différentes pour faciliter une attaque boomerang. L'objectif ici d'adapter cet outil afin qu'il puisse être utilisé pour analyser et fonctionner avec Skinny dans sa forme classique.

Cet outil nous a posé plusieurs difficultés liées à la structure de l'outil, qui utilise un nombre conséquent de technologies différentes qui communiquent par appels successifs entre eux, ce qui rendait le débuggage de nos modifications particulièrement compliqué. Parmis les technologies utilisées dans cet outil, on y trouve :

- Python, un langage de script commun, utilisé pour traîter les sorties des outils
- Snakemake, un langage d'automatisation de compilation similaire à Make, utilisé pour lancer les scripts Python de la Step 1 et 2 pour une intervalle de nombre de rounds donnée
- Rust, un langage bas niveau connu pour sa sécurité mémoire, utilisé pour traîter des données liées aux distingueurs
- Kotlin, un langage hérité de Java, utilisé pour exécuter les sources Minizinc
- Gradle pour compiler ces sources Kotlin en un fichier .jar exécutable par les autres parties de l'outil
- Minizinc, un langage de modélisation qui permet de décrire des problèmes d'optimisation et de satisfaction de contraintes que l'on appelle des CSP (Constraint Satisfaction problems). Ce language est utilisé dans divers domaines où nous devons résoudre des problèmes complexes avec beaucoup de contraintes par exemple l'intelligence artificielle ou encore la recherche opérationnelle. Il est utilisé ici pour modéliser Skinnyee.
- OrTools, une bibliothèque d'optimisation développé par Google qui est utilisé pour résoudre divers types de problèmes d'optimisation combinatoire. Elle est utilisé comme solveur des sources Minizinc.

Cet outil étant donc très complexe à utiliser dans sa forme originale, nous avons alors décidé de ne garder que les sources Minizinc et d'écrire notre propre script Python pour gérer l'exécution du solveur, le traîtement des données entre la Step 1 et la Step 2, et l'affichage final des résultats.

Nous nous sommes tout particulièrement concentré sur le fait de pouvoir exécuter automatiquement le fichier *boomerang-truncated-differential-characteristic.mzn* correspondant à la Step 1 et le fichier *differential-characteristic.mzn* correspondant à la Step 2, en veillant à récupérer certaines informations renvoyées par la Step 1 servant à définir les tableaux **DX** et **DRTK** utilisés dans la Step 2.

Le fichier *boomerang-truncated-differential-characteristic.mzn* se charge de la recherche de distingueurs boomerang dans le chiffrement Skinnyee. Le schéma de chiffrement est divisé en 3 partie : un chemin différentiel supérieur E_0, un chemin intermédiaire E_m et un chemin inférieur E_1. Le paramètre RM définit le nombre de tour total au sein de E_m, RE_0Em et RE_mE1 déterminent respectivement les longueur des trails supérieur et inférieur.

Le tweakey dans ce programme est représenté par deux tableaux TK_0 pour E_0 et TK_1 pour E_1. Ces derniers sont propager par les permutations définies dans Pt, Orb et Orb_inv. L'état X subit alors les opérations Subcells, ShiftRows, MixColumns et AddroundTweakey pour chaque tour. La fonction objectif vise à minimiser le coût total du chemin boomerang.

Le fichier *differential-characteristic.mzn* a pour but de minimiser une pondération, c'est-à-dire la somme des probabilités logarithmiques (toujours en ajoutant un signe "-" et un facteur 10) des différentes opérations non linéaires rencontrées au cours des tours dans l'algorithme de chiffrement. Concrètement, ce fichier recherche le chemin différentiel le plus probable dans le chiffrement. Le tweakey est divisé en 4 sous-parties. Dans notre travail, 2 des 4 sous-parties ont été enlevées pour correspondre au modèle de Skinny (les deux sous-parties supplémentaires étant une modification de Skinnyee). Chaque sous-partie représente une composante des clés ajoutés à chaque tour par l'opération **AddRoundKey**. Ces sous-parties de clés évoluent à chaque tour selon certaines règles de permutations définies dans le tableau **Pt** ou selon des transformations via des LFSR (Linear Feedback Shift Register).

En parallèle, l'état **X** subit des différences via des transitions à travers les S-Box (**SubCells**), des permutations de lignes (**ShiftRows**) et une diffusion par colonnes (**MixColumns**). Un poids est associé à chaque opération S-Box, cette pondération est stocké dans un tableau nommé **P**. Ce tableau est issue de la table des différence. L'objectif à minimiser est défini par la somme totale de ces pondérations.

L'intérêt principal était de pouvoir passer les résultats obtenues grâce à la sortie du fichier *boomerang-truncated-differential-characteristic.mzn* comme entrée du fichier *differential-characteristic.mzn* pour les tableaux *DX* et *DRTK*, où *DX* est un tableau à 3 dimensions où chaque entrée correspond à une ligne, une colonne et à un tour spécifique. Ce tableau permet de suivre la propagation d'une différence entre les tours et donc de déterminer comment les différences se propagent. Le tableau *DRTK* est un tableau à deux dimensions où chaque entrée correspond à une différence introduite par les Tweakeys (les clé de tour) pendant le chiffrement. On peut ainsi vérifier les conséquences qu'ont les clés de tours au fur et à mesure de l'algorithme.

Le fichier *boomerang-truncated-differential-characteristic.mzn* sert concrètement à générer des caractéristiques différentielles tronquées dans le cadre de l'attaque boomerang. Les caractéristiques différentielles tronquées permettent d'analyser les différences partielles entre des textes clairs et leurs chiffrés correspondants. *differential-characteristic.mzn* quant à lui, affine l'analyse faite par le fichier *boomerang-truncated-differential-characteristic.mzn*, en prenant en compte la propagation des différences à travers les différents tours contenus dans le chiffrement et optimise les caractéristiques différentielles obtenues. Les tableaux **DX** et **DRTK** sont définis comme suit en minizinc :

```
array[int] of bool: DX_step2in; % recuperation de la sortie de la "Step 1"
array[ROUNDS, ROWS,COLS] of bool: DX = array3d(ROUNDS,ROWS,COLS,DX_step2in);
```

```
array[int] of bool: DRTK_step2in; % recuperation de la sortie de la "Step 1"
array[ROUNDS_BUT_LAST,ROWS_AND_COLS] of bool:
    DRTK = array2d(ROUNDS_BUT_LAST, ROWS_AND_COLS,DRTK_step2in);
```

Les tableaux **DX_step2in** et **DRTK_step2in** représentent respectivement la propagation des différences et les différences introduites par les tweakeys de la Step 1. La sortie de la step 1 est encodée sous forme unidimensionnelle, ils sont alors reconvertis en tableaux multidimensionnels (en 3D pour **DX** et en 2D pour **DRTK**) avant de la réinsérer dans la step 2.

Les tableaux **DK** et **DRTK** contiennent respectivement $ROUNDS \times ROWS \times COLS$ éléments et $ROUNDS\_BUT\_LAST \times ROWS\_AND\_COLS$. L'interconnexion automatique entre les deux étapes, est rendue possible par le fichier main.py ci-dessous :

```
print("<============================ "Step 1" ============================")
res = os.system("minizinc --solver cp boomerang-truncated-differential-characteristic.mzn > step1.t
print("minizinc --solver cp boomerang-truncated-differential-characteristic.mzn > step1.txt\n")

with open("step1.txt", "r") as file:
    result_content = file.read()
    print(result_content)
    split_res = result_content.split(";")
    extract = lambda x : x[x.find("[")+1:x.find("]")]
    TK0 = extract(split_res[1].strip()).split(", ")
    TK1 = extract(split_res[2].strip()).split(", ")
    X0 = extract(split_res[5].strip()).split(", ")
    X1 = extract(split_res[6].strip()).split(", ")

    TK0Bis_str = " ".join(TK0)
    TK1Bis_str = " ".join(TK1)

```

```
    TK0B = re.findall(r"\d+: (false|true)", TK0Bis_str)
    cpt = 0
    for i in TK0B:
        if cpt > 0:
            TK0.append(i)
        cpt = 1

    TK1B = re.findall(r"\d+: (false|true)", TK1Bis_str)
    cpt = 0
    for i in TK1B:
        if cpt > 0:
            TK1.append(i)
        cpt = 1

    DX = [X0[i] != X1[i] for i in range(len(X0))]
    DRTK = [TK0[i] != TK1[i] for i in range(len(TK0))]

print("Nombre d'elements dans DX : ", len(DX))
print("Nombre d'elements dans DRTK : ", len(DRTK))

with open("step2in.dzn", "w") as resfile:
    resfile.write(f"DX_step2in = {str(DX).lower()};\n")
    resfile.write(f"DRTK_step2in = {str(DRTK).lower()};\n")

print("<============================ "Step 2" ============================>")
print("minizinc --solver cp differential-characteristic.mzn step2in.dzn\n")
os.system("minizinc --solver cp differential-characteristic.mzn step2in.dzn")
```

Ce script python récupère les résultat de la "Step 1" envoyé dans le fichier main.py et extrait automatiquement les valeurs des états intermédiaires X0 et X1 ainsi que celles des clés de rondes (TK0 et TK1). Une comparaison est ensuite faite entre états et entre clés pour construire les 2 tableaux DX et DRTK utile pour la "Step 2". Cette automatisation permet un gain de temps non négligeable, tout en réduisant les erreurs humaines et en facilitant l'exploration de caractéristiques différentielles pertinentes utile pour l'attaque.

Nous avons pu réaliser une grande partie de la conversion de l'outil pour le faire correspondre à Skinny. Cependant, nous n'avons pas pu vérifier le bon fonctionnement de l'outil converti par manque de temps pour finaliser la conversion. En particulier, le nombre de tours dans les fichiers *boomerang-truncated-differential-characteristic.mzn* et *differential-characteristic.mzn* ne sont pas définis par la même valeur. De plus ,le nombre d'éléments requis dans les tabelaux **DX** et **DRTK** ne semble pas correspondre entre les deux fichiers pour un même nombre de round.

9 Conclusion

Ce projet avait pour but d'étudier les attaques Boomerang sur l'algorithme de chiffrage symétrique par bloc Skinny dans sa variante 128-256 à l'aide d'outils automatiques.

Nous avons dû tout d'abord étudier la cryptanalyse différentielle classique, ce qui nous a permis de comprendre le principe des attaques Boomerang dans sa forme générale.

Ensuite, nous avons expliqué le fonctionnement de Skinny et pourquoi il était intéressant de l'utiliser dans le cadre des attaques Boomerang, en particulier par sa modélisation facilitant la recherche d'attaque, et sa S-Box qui se veut résistante aux attaques différentielles mais qui peut être vulnérable aux attaques boomerang.

Nous avons alors étudié les caractéristiques des outils automatiques d'attaque Boomerang, en particulier la distinction importante entre la Step 1, estimant un chemin d'attaque dans le schéma de chiffrement, et la Step 2, permettant de calculer les probabilités liées à ce chemin.

Par la suite, nous avons essayé les outils de Hossein Hadipour et de Patrick Derbez afin de simuler des attaques Boomerang, puis nous avons comparé les résultats et les temps d'exécutions obtenus. Nous avons alors remarqué que le temp de calcul des deux outils augmentait de façon exponentielle avec le nombre de tour de Skinny.

Enfin, nous avons essayé d'adapter un autre outil créé par Loïc Rouquette originalement fait pour Skinnyee, une variante de Skinny, afin de la rendre compatible avec Skinny dans sa version classique dans le but d'obtenir un troisième outil utilisable. Nous avons rencontrés beaucoup de difficultés lors de l'adaptation, en particulier à cause de la structure complexe de l'outil et de son caractère technique. Nous avons du alors extraire les sources critiques de l'outil pour refaire un outil structurellement plus simple.

Les différentes difficultés rencontrées nous ont empêché de finir complètement cette conversion, cependant une grande partie du travail d'adaptation a pu être réalisé. Ce travail a été ensuite transmis aux stagiaires de notre encadrante Marine Minier afin de finir la conversion de l'outil.

Pour conclure, ce projet nous a permis d'en apprendre plus sur la cryptanalyse différentielle, les attaques Boomerang et le fonctionnement des algorithmes de chiffrement, ce qui était un bon complément à l'UE Cryptographie que nous avions tous les trois suivi ce semestre. Nous avons pu aussi nous familiariser avec des langages de description de contraintes comme Minizinc et des solveurs comme Or-tools.

Références

[B+16] Boris B. et al. *SKINNY: Lightweight Block Cipher for Secure Embedded Systems*. Rapp. tech. 2016/660. IACR, 2016. URL : https://eprint.iacr.org/2016/660.pdf.

[BM25] Sonal BARGE et Gerardine Immaculate MARY. *Improving the reliability of SKINNY-Hash for IoT applications with less power overhead*. Technical Report (journal publication). Multidisciplinary Science Journal (Malque Publishing), 2025. URL : https://malque.pub/ojs/index.php/msj/article/view/6988.

[BS90] Eli BIHAM et Adi SHAMIR. *Differential Cryptanalysis of the Data Encryption Standard*. Rapp. tech. Technion - Israel Institute of Technology, 1990. URL : https://biham.cs.technion.ac.il/Reports/differential-cryptanalysis-of-the-data-encryption-standard-biham-shamir-authors-latex-version.pdf.

[Der21] Patrick DERBEZ. *BoomerangSkinny: MILP-based tools for boomerang cryptanalysis of SKINNY*. Technical Report. Accessed: 2025-05-10. Inria GitLab, 2021. URL : https://gitlab.inria.fr/pderbez/boomerangskinny.

[Gre93] George D. GREENWADE. "The Comprehensive Tex Archive Network (CTAN)". In : *TUGBoat* 14.3 (1993), p. 342-351.

[HDE24] Hossein HADIPOUR, Patrick DERBEZ et Martin EICHLSEDER. *Revisiting Differential-Linear Attacks via a Boomerang Perspective With Application to AES, Ascon, CLEFIA, SKINNY, PRESENT, KNOT, TWINE, WARP, LBlock, Simeck, and SERPENT*. Rapp. tech. 2024/255. IACR Cryptology ePrint Archive, 2024. URL : https://eprint.iacr.org/2024/255.pdf.

[Wag99] David WAGNER. *The Boomerang Attack*. Tech report. Springer, 1999. URL : https://link.springer.com/chapter/10.1007/3-540-48519-8_12.

[Wik] WIKIPEDIA CONTRIBUTORS. *Schéma de chiffrement par bloc*. https://fr.wikipedia.org/wiki/Chiffrement_par_bloc.

[Wik24] WIKIPÉDIA, L'ENCYCLOPÉDIE LIBRE. *Réseau de Feistel*. 2024. URL : https://fr.wikipedia.org/wiki/R%C3%A9seau_de_Feistel.

Annexes

UNIVERSITÉ DE LORRAINE

Déclaration sur l'honneur contre le plagiat

(à joindre obligatoirement à tout travail de recherche ou dossier remis à un enseignant)

Je soussigné(e),

Nom, Prénom,
Aria, Aéna

Régulièrement inscrit à l'Université de Lorraine

N° de carte d'étudiant : 32106658

Année universitaire : 2024-2025

Niveau d'études : L ou M

Parcours : Master Informatique

N° UE : 8JU31N01

Certifie qu'il s'agit d'un travail original et que toutes les sources utilisées ont été indiquées dans leur totalité. Je certifie, de surcroît, que je n'ai ni recopié ni utilisé des idées ou des formulations tirées d'un ouvrage, article ou mémoire, en version imprimée ou électronique, sans mentionner précisément leur origine et que les citations intégrales sont signalées entre guillemets.

Conformément à la loi, le non-respect de ces dispositions me rend passible de poursuites devant la commission disciplinaire et les tribunaux de la République Française.

Fait à Nancy, le 11/05/2025

Signature :

7

UNIVERSITÉ DE LORRAINE

Déclaration sur l'honneur contre le plagiat

(à joindre obligatoirement à tout travail de recherche ou dossier remis à un enseignant)

Je soussigné(e),

Nom, Prénom,
Volsfelts Julien

Régulièrement inscrit à l'Université de Lorraine

N° de carte d'étudiant : 32111979

Année universitaire : 2024-2025

Niveau d'études : L ou M

Parcours : Informatique

N° UE : 8JU31N01

Certifie qu'il s'agit d'un travail original et que toutes les sources utilisées ont été indiquées dans leur totalité. Je certifie, de surcroît, que je n'ai ni recopié ni utilisé des idées ou des formulations tirées d'un ouvrage, article ou mémoire, en version imprimée ou électronique, sans mentionner précisément leur origine et que les citations intégrales sont signalées entre guillemets.

Conformément à la loi, le non-respect de ces dispositions me rend passible de poursuites devant la commission disciplinaire et les tribunaux de la République Française.

Fait à , le 11/05/2025

Signature :

7

Annexes

UNIVERSITÉ DE LORRAINE

Déclaration sur l'honneur contre le plagiat

(à joindre obligatoirement à tout travail de recherche ou dossier remis à un enseignant)

Je soussigné(e),

Nom, Prénom,
HADIBI, Neil

Régulièrement inscrit à l'Université de Lorraine

N° de carte d'étudiant : 32109335

Année universitaire : 2024-2025

Niveau d'études : L ou (M)

Parcours : Master Informatique

N° UE : 8JU31N01

Certifie qu'il s'agit d'un travail original et que toutes les sources utilisées ont été indiquées dans leur totalité. Je certifie, de surcroît, que je n'ai ni recopié ni utilisé des idées ou des formulations tirées d'un ouvrage, article ou mémoire, en version imprimée ou électronique, sans mentionner précisément leur origine et que les citations intégrales sont signalées entre guillemets.

Conformément à la loi, le non-respect de ces dispositions me rend passible de poursuites devant la commission disciplinaire et les tribunaux de la République Française.

Fait à Nancy, le 11/05/2025

Signature :

7