Construction de Luby-Rackoff et réseaux de Feistel

Franck Jeannot

Montréal, Canada, Juillet 2018, S548, v1.0

Abstract

A review of the Luby-Rackoff construction, seen as a major demonstration in the usage of block ciphers and on the analysis of Feistel Networks.

Keywords: CSB, CSPRB, CSPRNG, DES, Feistel, generator, Luby, permutations, PFE, PRF, PRP, pseudo-random, Rackoff

1. Introduction

Michael **Luby** [1] et Charles **Rackoff** [2], en 1985 [3], ont montré [1] qu'une permutation pseudo-aléatoire (**PRP** *pseudo random permutation*) « **forte** », peut être construite sur la base d'une construction spécifique basée sur une **fonction pseudo-aléatoire** (**PRF** *pseudorandom function*) elle-même basée sur un réseau de **Feistel** [2]. Le résultat de cette construction reste théorique car les blocs de chiffrement utilisés en pratique ont des design ad hoc, cependant, le résultat peut être vu comme une justification de l'usage de blocs de chiffrement de type réseaux de Feistel [4]. Cette construction se base aussi sur des idées issues du **DES** (Data Encryption System). Luby et Rackoff ont conclu notamment [1] sur l'existence, de ce qu'ils nomment « *générateur de permutation super pseudo-aléatoire inversible* » (*super pseudorandom invertible permutation generator*) [5] sur la base de l'existence d'un générateur de fonction pseudo aléatoire [3].

1. https://en.wikipedia.org/wiki/Michael_Luby
2. https://fr.wikipedia.org/wiki/Charles_Rackoff
3. CRYPTO '85 Proceedings, publié en 1986
4. Source : Trad. libre et adaptée de "A Graduate Course in Applied Cryptography", p. 139, Dan Boneh and Victor Shoup, v 0.4, Sept 2017
5. Notion renforçant le caractère pseudo-aléatoire, voir p. 6 de https://omereingold.files.wordpress.com/2014/10/lr.pdf

2. Historique

Manuel **Blum** et Silvio **Micali** ont introduit en 1984 la notion de générateur pseudo-aléatoire [6] [7] « cryptographiquement fort » [4] [5] (**CSB, CSPRB** *cryptographically strong pseudo random generator*) [8]. Leonid A. **Levin** a ensuite prouvé [6], [7] en 1985, que de tels générateurs existaient si et seulement si des fonctions à sens unique existent. **Goldreich, Goldwasser et Micali** ont ensuite montré [8], qu'il était possible de construire des fonctions pseudo-aléatoires en utilisant des **CSB**. Sur ces bases la **construction de Luby-Rackoff** a été proposée.

3. Fonction pseudo-aléatoire

Une fonction pseudo aléatoire (ou *pseudorandom function* **PRF**) est une « *fonction dont l'ensemble des sorties possibles n'est pas efficacement distinguable des sorties d'une fonction aléatoire* » [9] [10]. On parle aussi de **PFE** (**P**seudo-random **F**unction **E**nsembles) [11]. Voir ***Goldreich, Goldwasser et Micali*** [8].

Une **PRF**, $F_K\{0,1\}^* \mapsto \{0,1\}^*$ est indistinguable par le calcul (*computationally indistinguishable*) [12] [13] d'une fonction aléatoire $f_{\text{rand}}\{0,1\}^* \mapsto \{0,1\}^*$

Par équivalence on a $\forall \epsilon > 0$:

$$Pr\left[\; 1 \leftarrow A \rightleftharpoons \boxed{F_K(\cdot)} \;\right] - Pr\left[\; 1 \leftarrow A \rightleftharpoons \boxed{f_{\text{rand}}(\cdot)} \;\right] < \epsilon \qquad (1)$$

FIGURE *(1): Indistinguabilité entre PRF et fonction aléatoire (Adapté de "Crypto Notes - Lecture 7" de Stuart Walker, University of Bristol)*

6. https://en.wikipedia.org/wiki/Blum%E2%80%93Micali_algorithm
7. https://en.wikipedia.org/wiki/Cryptographically_secure_pseudorandom_number_generator
8. On utilisera indistinctement **CSB**, **CSPRB**, "*cryptographically secure pseudo-random number generator*", (**CSPRNG**) ou *cryptographic pseudo-random number generator* (**CPRNG**)
9. Source : https://fr.wikipedia.org/wiki/Fonction_pseudo-al%C3%A9atoire
10. https://en.wikipedia.org/wiki/Pseudorandom_function_family
11. https ://omereingold.files.wordpress.com/2014/10/lr.pdf
12. https://en.wikipedia.org/wiki/Computational_indistinguishability
13. https://en.wikipedia.org/wiki/Ciphertext_indistinguishability

4. Permutation pseudo aléatoire

Une permutation pseudo aléatoire (ou *pseudorandom permutation* (**PRP**) est une fonction qui ne peut pas être distinguée d'une permutation aléatoire sur la base d'un effort « *pratiquement réalisable* » [14].

5. Ronde de Feistel

On appelle **rondes**, les *étapes de permutation et de substitution répétées* d'un algorithme de chiffrement [2].

6. Chiffrement par bloc

Les algorithmes chiffrent le texte en clair en blocs (chiffrés) et decryptent le texte chiffré en blocs **à l'aide d'une clé secrète**.

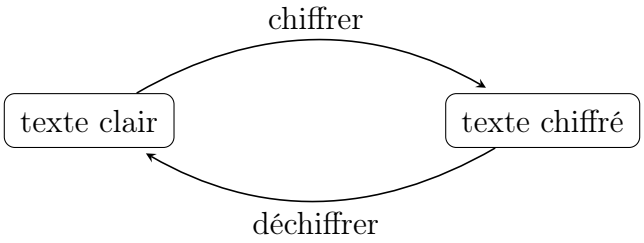

FIGURE *(2): Cycle chiffrement-déchiffrement*

7. Réseaux de Feistel

De nombreux algorithmes de chiffrement prennent la forme de réseaux de **Feistel** [2] [9] [10].
Dans un réseau de **Feistel**, l'entrée est divisée de façon égale en blocs gauche (**L**eft) et droit (**R**ight), et la sortie du i'ème tour du chiffre est calculée comme suit pour $i = 0, 1, \ldots, n$, considérant f comme la fonction de la **ronde** et K_i les sous clés de chaque ronde.

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_i) \end{aligned}$$

14. https://en.wikipedia.org/wiki/Pseudorandom_permutation

On peut écrire pareillement [15] :

$$\begin{aligned} L_{i+1} &= R_i \\ R_{i+1} &= L_i \oplus f(R_i, K_i) \end{aligned}$$

Considérant n rondes, on a le **texte chiffré** donné par (R_{n+1}, L_{n+1}) et le texte en clair (ou déchiffré) (R_0, L_0). Une des propriétés des réseaux de Feistel est que la fonction f peut être aussi complexe que souhaitée, tout en gardant toujours la capacité de récupération de l'entrée :

$$\begin{aligned} R_{i-1} &= L_i \\ L_{i-1} &= R_i \oplus f(L_i, K_i) \end{aligned}$$

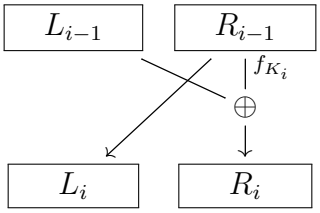

FIGURE *(3): Représentation de réseau de Feistel simplifié (Adapté de David R. Kohel - 2010)*

Ainsi, même si f n'est pas inversible, l'algorithme lui-même est inversible, permettant le déchiffrage (si on connait la clé K_i). Les réseaux de Feistel font partie intégrante des algorithmes tels que DES, Lucipher, FEAL, Khufu, LOKI, GOST, Blowfish,... et autres algorithmes à chiffrement par bloc.

On peut éliminer L_i en définissant $R_{-1} = L_0$ de telle manière que l'entrée soit $R_{-1}R_0$ et que la ronde soit de la forme $R_i = R_{i-2} \oplus f_{K_i}(R_{i-1})$, ce qui dans ce cas donne un diagramme de type suivant [16]

15. c'est alors la formulation utilisée dans la section "Construction details" de https://en.wikipedia.org/wiki/Feistel_cipher

16. Inspiré de David R. Kohel en XY-pic https://bitbucket.org/mvngu/crypto-book/raw/

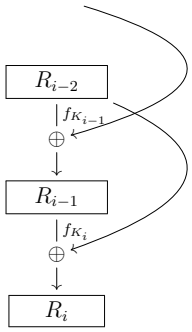

FIGURE *(4): Réseau de Feistel simplifié sans L_i (Adapté de David R. Kohel - 2010)*

L'extrant ou sortie finale de l'algorithme de chiffrement de Feistel est la paire inversée $R_nL_n = R_nR_{n-1}$ qui permet d'inverser l'algorithme en utilisant une séquence inversée des clés.

On prouve ensuite qu'en inversant la séquence de la clé interne on obtient le chiffrement inverse, en comparant les séquences de chiffrement et de déchiffrement R_i et R'_j.

Chiffrement. Un message $M = L_0R_0 = R_{-1}R_0$, est chiffré via l'itération en suivant la séquence de clé $K_1, K_2, \ldots, K_n$:

$$R_{i+1} = R_{i-1} \oplus f_{K_{i+1}}(R_i), \tag{2}$$

Déchiffrement. Supposons que l'on commence avec $C = R_nR_{n-1} = R'_{-1}R'_0$, et une séquence de clé inversée $K'_1, K'_2 \ldots, K'_n = K_n, K_{n-1} \ldots, K_1$. Le déchiffrement suit le même algorithme que pour le chiffrement en suivant la séquence de clé suivante :

$$R'_{j+1} = R'_{j-1} \oplus f_{K'_{j+1}}(R'_j). \tag{3}$$

En posant $j = r - i - 1$, on a $K'_{j+1} = K'_{r-i} = K_{i+1}$. On veut donc voir les relations telles que :

$$R'_{-1} = R_r,\ R'_0 = R_{r-1}, \ldots, R'_{r-1} = R_0,\ R'_r = R_{-1}.$$

On veut montrer que $R'_j = R_i$ pour autant que $i + j = n - 1$. Clairement cette relation existe pour $(i, j) = (n, -1)$ and $(i, j) = (n - 1, 0)$. Si on assume que la relation existe pour $j - 1$ et j on prouve que la relation existe pour $j + 1$. La séquence de déchiffrement (3) peut être remplacée par :

$$R'_{j+1} = R'_{j-1} \oplus f_{K'_{j+1}}(R'_j) = R'_{r-i-2} \oplus f_{K'_{r-i}}(R'_{r-i-1}) = R_{i+1} \oplus f_{K_{i+1}}(R_i)$$

L'expression $R_{i+1} = R_{i-1} \oplus f_{K_{i+1}}(R_i)$ de (3) peut être réarrangée en ajoutant (soustrayant) $f_{K_{i+1}}(R_i)$ aux deux côtés pour avoir : $R_{i+1} \oplus f_{K_{i+1}}(R_i) = R_{i-1}$. On conclut que $R'_{j+1} = R_{i-1}$, et donc l'égalité existe par induction.

8. Construction de Luby-Rackoff

Michael Luby et Charles Rackoff ont prouvé [1] que 3 rondes d'une fonction **CSB** étaient suffisantes [17] pour permettre à un algorithme de chiffrement par bloc de **réaliser une permutation pseudo-aléatoire** alors que 4 rondes sont suffisantes pour en faire une permutation pseudo-aléatoire « **forte** » (ce qui signifie qu'il reste pseudo-aléatoire même pour un adversaire qui détient un "oracle access" à sa permutation inverse) [11]. Par la suite de ces travaux, de nombreuses revues [12], optimisations et simplifications ont été faites, par exemple Maurer en 1993 [13] (simplifications) et Lucks [14] (généralisations). L'algorithme proposé par Luby et Rackoff opère sur une chaîne $L \cdot R$ de $2n - bit$ où L et R sont d'une taille de n bits chacun et peut être simplifié de la manière suivante (f_i sont des fonctions pseudo-aléatoires indépendantes ; $V \cdot W$ est la sortie) :

$$\begin{aligned} S &= L \oplus f_1(R) \\ T &= R \oplus f_2(S) \\ V &= S \oplus f_3(T) \\ W &= T \oplus f_4(T) \end{aligned}$$

8.1. Précisions sur les preuves mathématiques de la sécurité des réseaux de Feistel post Luby-Rackoff

Il a été précisé et démontré **en 2003** par Jacques Patarin [15] [16] les considérations de sécurité suivant le nombre de rondes de Feistel utilisées :

- 3 rondes : « fort » contre toutes attaques adaptatives a texte clair choisi quand le nombre de requêtes est $m \ll 2^{n/2}$ (*all adaptative chosen plaintext attacks ...*)
- 4 rondes : « fort » contre toutes attaques adaptatives a texte clair et chiffré choisi quand le nombre de requêtes est $m \ll 2^{n/2}$ (*all adaptative chosen plaintext and chosen ciphertext attacks ...*)

De plus pour $\epsilon > 0$ avec $m \ll 2^{n(1-\epsilon)}$:

- 4 rondes et plus : « fort » contre toutes attaques a texte clair connues (*KPA Known Plaintext attacks*)
- 7 rondes et plus : « fort » contre toutes attaques adaptatives a texte clair (*CPA Choosen Plaintext Attacks*)
- 10 rondes et plus : « fort » contre toutes attaques adaptatives a texte clair et chiffré (*CPCA Choosen Plaintext and Ciphertext Attacks*)

17. Voir section "Theoretical work" de https://en.wikipedia.org/wiki/Feistel_cipher

Références

[1] Luby, Michael and Rackoff, Charles, How to construct pseudo-random permutations from pseudo-random functions, Advances in Cryptology, CRYPTO '85 Proceedings (1986) 447–447.

[2] Jeannot, Franck, Les réseaux de Feistel classiques et généralisés.
URL https://www.franckjeannot.com/wp-content/uploads/reseaux_feistel.pdf

[3] Naor, Moni and Reingold, Omer, On the construction of pseudorandom permutations: Luby-Rackoff revisited, Journal of Cryptology 12 (1) (1999) 29–66. doi:10.1007/PL00003817.
URL https://omereingold.files.wordpress.com/2014/10/lr.pdf

[4] Blum, Manuel and Micali, Silvio, How to generate cryptographically strong sequences of pseudo-random bits, SIAM J. Comput. 13 (4) (1984) 850–864. doi:10.1137/0213053.
URL https://people.csail.mit.edu/silvio/Selected%20Scientific%20Papers/Pseudo%20Randomness/How_To_Generate_Cryptographically_Strong_Sequences_Of_Pseudo-Random_Bits.pdf

[5] Pieprzyk, Josef, How to construct pseudorandom permutations from single pseudorandom functions, Advances in Cryptology, EUROCRYPT '90 (1991) 140–150.
URL https://pdfs.semanticscholar.org/7d58/9564998287a942149c4fee931852b22d8997.pdf

[6] Levin, Leonid A., One-way functions and pseudorandom generators, Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing (1985) 363–365.
URL http://doi.acm.org/10.1145/22145.22185

[7] Impagliazzo, Russell and A. Levin, Leonid and Luby, Michael, Pseudorandom generation from one-way functions (1989) 12–24.
URL https://www.researchgate.net/profile/Michael_Luby/publication/235008680_Pseudorandom_generation_from_one-way_functions

[8] Goldreich, Oded and Goldwasser, Shafi and Micali, Silvio, How to construct random functions, J. ACM 33 (4) (1986) 792–807. doi:10.1145/6490.6503.
URL https://groups.csail.mit.edu/cis/pubs/shafi/1986-jacm.pdf

[9] Konheim, Alan G., The impetus to creativity in technology.
URL https://www.cs.ucsb.edu/~konheim/Cryptologia%20Paper.pdf

[10] Backes, Michael, Lecture notes for cs-578 cryptography (ss2007) : 3. block ciphers.
URL http://web.cs.du.edu/~ramki/courses/security/2011Winter/notes/feistelProof.pdf

[11] Y. Dodis, P. Puniya, Feistel networks made public, and applications.
URL https://www.semanticscholar.org/paper/Feistel-Networks-Made-Public%2C-and-Applications-Dodis-Puniya/7b4bb870ab19225dbad1cd4e450a2b5b57fefbbd

[12] Z. A. Ramzan, A study of Luby-Rackoff ciphers, Phd Thesis.
URL https://groups.csail.mit.edu/cis/theses/ramzan-phd.pdf

[13] Maurer, Ueli M., A simplified and generalized treatment of Luby-Rackoff pseudorandom permutation generators.
URL https://link.springer.com/content/pdf/10.1007%2F3-540-47555-9_21.pdf

[14] Lucks, Stefan, Faster Luby-Rackoff ciphers, Fast Software Encryption (1996) 189–203.

[15] Patarin, Jacques, Luby-Rackoff: 7 Rounds Are Enough for $2^{n(1-\epsilon)}$ Security, Advances in Cryptology, CRYPTO 2003 (2003) 513–529.
URL https://www.iacr.org/archive/crypto2003/27290510/27290510.pdf

[16] D. Boneh, CRYPTO., Advances in Cryptology – CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings, no. v. 23 in Lecture Notes in Computer Science, Springer, 2003.
URL https://books.google.ca/books?id=GzAPyE6hfxIC