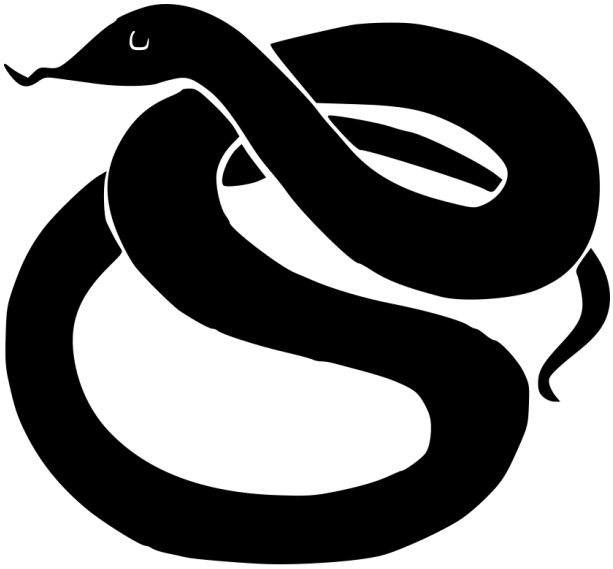
Algorithme de chiffrement par bloc : Serpent

Rapport

NGUYEN Alexandre - MILLOT Nathan



Licence Informatique TREC 7 Semestre 6

Date : 26 septembre 2025

Table des matières

Espaces de définition

1 Messages en clair et chiffrés

L'algorithme Serpent est un chiffrement par bloc avec :

— Espace des messages en clair $M = \{0,1\}^{128}$;

— Espace des messages chiffrés $C = \{0,1\}^{128}$.

Chaque bloc est donc constitué de 128 bits.

2 Clés

L'espace des clés dépend de la taille choisie :

— $K = \{0,1\}^{128}$;
— $K = \{0,1\}^{192}$;
— $K = \{0,1\}^{256}$.

En pratique, Serpent est défini pour ces trois tailles de clé.

Description du schéma Serpent

1 Structure générale

Serpent est basé sur un **réseau de substitution-permutation (SPN)**. Il comporte 32 tours successifs :

1. Ajout de sous-clé (XOR avec la sous-clé de tour) ;
2. Substitution via une des 8 S-boxes (non-linéarité) ;
3. Transformation linéaire (permutation de bits).

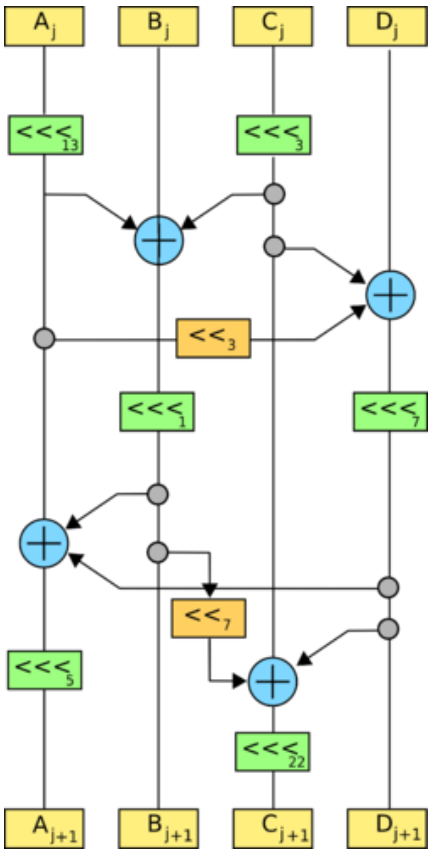


FIGURE 1 – Schéma général de Serpent

2 Sous-fonctions

Génération de sous-clés

À partir de la clé initiale, Serpent dérive 33 sous-clés de 128 bits chacune, notées $K_0, K_1, \ldots, K_{32}$.

Substitution (S-boxes)

Huit S-boxes différentes sont utilisées, notées $S_0, \ldots, S_7$. Elles transforment des blocs de 4 bits en 4 bits, et sont choisies de manière cyclique au fil des tours.

Transformation linéaire

Chaque tour applique une permutation fixe des bits pour assurer une bonne diffusion.

Dernier tour

Le dernier tour est légèrement différent : après la substitution, il n'y a pas de transformation linéaire, seulement l'ajout de sous-clé.

Exemple de chiffrement simplifié

Pour illustrer le fonctionnement de l'algorithme Serpent, nous présentons un exemple simplifié avec un bloc de données réduit à 8 bits (au lieu de 128 bits) et deux tours (au lieu de 32). Cet exemple suit la structure réelle de Serpent : ajout de sous-clé, substitution via S-box, et transformation linéaire, sauf pour le dernier tour où la transformation linéaire est omise, suivie d'un XOR final avec la dernière sous-clé. Nous inclurons également le déchiffrement pour montrer comment inverser le processus. Dans la réalité, Serpent traite des blocs de 128 bits avec 32 tours complets et 33 sous-clés de 128 bits.

3 Données initiales

Prenons les données suivantes :

- **Texte clair** : $B = 10101010$ (8 bits, équivalent à `0xAA` en hexadécimal).
- **Sous-clés** (simplifiées, 8 bits chacune) :
  - $K_0 = 11110000$ (`0xF0`),
  - $K_1 = 10101010$ (`0xAA`),
  - $K_2 = 11001100$ (`0xCC`).
- **S-boxes** : Nous utilisons deux S-boxes inspirées des S_0 et S_1 réelles de Serpent, définies pour des entrées de 4 bits (voir Tableaux 1 et 2). Pour simplifier, seules les

entrées utilisées dans l'exemple sont listées. Pour le déchiffrement, nous utilisons leurs inverses (voir Tableaux 3 et 4).

Entrée (décimal)	Entrée (binaire)	Sortie (décimal)	Sortie (binaire)
0	0000	3	0011
1	0001	8	1000
5	0101	6	0110
10	1010	4	0100
12	1100	7	0111

TABLE 1 – S-box S_0 simplifiée pour le tour 0.

Entrée (décimal)	Entrée (binaire)	Sortie (décimal)	Sortie (binaire)
0	0000	15	1111
1	0001	12	1100
8	1000	1	0001
12	1100	6	0110

TABLE 2 – S-box S_1 simplifiée pour le tour 1.

Entrée (décimal)	Entrée (binaire)	Sortie (décimal)	Sortie (binaire)
3	0011	0	0000
8	1000	1	0001
6	0110	5	0101
4	0100	10	1010
7	0111	12	1100

TABLE 3 – S-box inverse S_0^{-1} simplifiée.

4 Étape 1 : Tour 0 (chiffrement)

1. **Ajout de sous-clé** : Calculons $B \oplus K_0$.

$$B = 10101010 \oplus 11110000 = 01011010$$

2. **Substitution (S-box S_0)** : Divisons le bloc en deux groupes de 4 bits :
   — Premier groupe : 0101 (5 en décimal) $\rightarrow S_0(5) = 0110$ (6).
   — Deuxième groupe : 1010 (10 en décimal) $\rightarrow S_0(10) = 0100$ (4).
   Recombinons : $B = 01100100$.

3. **Transformation linéaire** : Pour simplifier, supposons que la transformation linéaire inverse l'ordre des bits.

$$01100100 \rightarrow 00100110$$

Entrée (décimal)	Entrée (binaire)	Sortie (décimal)	Sortie (binaire)
15	1111	0	0000
12	1100	1	0001
1	0001	8	1000
6	0110	12	1100

TABLE 4 – S-box inverse S_1^{-1} simplifiée.

5 Étape 2 : Tour 1 (dernier tour, chiffrement)

Dans Serpent, le dernier tour (tour 31) omet la transformation linéaire et est suivi d'un XOR avec K_{32}. Dans notre exemple simplifié avec deux tours, le tour 1 est considéré comme le dernier tour.

1. **Ajout de sous-clé** : Calculons $B \oplus K_1$.

$$B = 00100110 \oplus 10101010 = 10001100$$

2. **Substitution (S-box S_1)** : Divisons en deux groupes :
   — Premier groupe : 1000 (8) $\rightarrow S_1(8) = 0001$ (1).
   — Deuxième groupe : 1100 (12) $\rightarrow S_1(12) = 0110$ (6).
   Recombinons : $B = 00010110$.
3. **XOR final avec K_2** : Puisque c'est le dernier tour, nous omettons la transformation linéaire et effectuons un XOR avec K_2.

$$B = 00010110 \oplus 11001100 = 11011010$$

6 Résultat du chiffrement

Le **texte chiffré** est donc 11011010 (`0xDA` en hexadécimal).

7 Étape 3 : Déchiffrement

Le déchiffrement inverse les opérations du chiffrement, en commençant par le XOR final et en appliquant les transformations en ordre inverse, avec les S-boxes inverses et la transformation linéaire inverse (dans notre cas, l'inversion des bits est son propre inverse).

Partons du **texte chiffré** : $B = 11011010$.

Tour 1 inverse (dernier tour inverse)

1. **XOR inverse avec K_2** : $B \oplus K_2$.

$$B = 11011010 \oplus 11001100 = 00010110$$

2. **Substitution inverse (S-box S_1^{-1})** : Divisons en deux groupes :
   — Premier groupe : 0001 (1) $\rightarrow S_1^{-1}(1) = 1000$ (8).
   — Deuxième groupe : 0110 (6) $\rightarrow S_1^{-1}(6) = 1100$ (12).
   Recombinons : $B = 10001100$.
3. **XOR inverse avec K_1** : $B \oplus K_1$.

$$B = 10001100 \oplus 10101010 = 00100110$$

Tour 0 inverse

1. **Transformation linéaire inverse** : Inversion des bits (inverse de LT simplifiée).

$$00100110 \rightarrow 01100100$$

2. **Substitution inverse (S-box S_0^{-1})** : Divisons en deux groupes :
   — Premier groupe : 0110 (6) $\rightarrow S_0^{-1}(6) = 0101$ (5).
   — Deuxième groupe : 0100 (4) $\rightarrow S_0^{-1}(4) = 1010$ (10).
   Recombinons : $B = 01011010$.
3. **XOR inverse avec K_0** : $B \oplus K_0$.

$$B = 01011010 \oplus 11110000 = 10101010$$

8 Résultat du déchiffrement

Le **texte clair récupéré** est 10101010 (`0xAA` en hexadécimal), correspondant au texte clair initial.

9 Remarques

Cet exemple simplifié utilise un bloc de 8 bits et deux tours pour illustrer les opérations de Serpent. Dans l'algorithme réel :

— Le bloc est de 128 bits, divisé en 32 groupes de 4 bits pour les S-boxes.
— Les 32 tours utilisent les S-boxes $S_0, S_1, \ldots, S_7$ dans l'ordre cyclique (i mod 8).
— La transformation linéaire est plus complexe, impliquant des rotations et des XOR sur quatre mots de 32 bits, et son inverse est défini en conséquence.
— Le déchiffrement inverse toutes les opérations, en commençant par le XOR avec K_{32}, et en appliquant les S-boxes inverses et la transformation linéaire inverse dans l'ordre inverse des tours.

Sécurité et limites

1 Résistance aux attaques

Serpent a été conçu avec une marge de sécurité très élevée :

— Résistant à la cryptanalyse différentielle et linéaire ;

— Résistant aux attaques par clé liée.

Ses 32 tours sont considérés comme surdimensionnés par rapport au minimum nécessaire (16–24 tours auraient suffi).

2 Limites actuelles

Malgré sa robustesse, Serpent n'a pas été choisi comme AES en raison de sa relative lenteur par rapport à Rijndael (AES). Aujourd'hui, il est toujours considéré comme sûr, mais il est moins utilisé que AES ou ChaCha20, qui sont devenus les standards de fait.

Conclusion

Serpent est un algorithme de chiffrement robuste et bien conçu, qui mise sur la prudence et la sécurité. S'il n'a pas été choisi comme AES, il reste un chiffrement respecté, encore pertinent pour l'enseignement et certaines applications pratiques.