
# SE - Блокировать 2 - Учебный блок 18 - Математика и прикладная физика

## Лаборатория 4

## Краткое описание проекта

Camarades Ludewisky Françoisov, Arnaud-Aymerisku Giboltev-Nowak


— 1 —

# GOST

## 1.1 Introduction

Bonjour camarade. Vous n'êtes pas sans savoir que ces enfoirés de capitalistes ont développé un processus de chiffrement symétrique qui nous semble inviolable pour le moment.

Une équipe de nos meilleurs mathématiciens a été chargée de trouver un système encore meilleur. C'est maintenant chose faite et ils ont été remerciés en étant envoyé ~~au goulag~~ en vacances.

Maintenant que l'idée est là, on vous demande à vous, nos meilleurs programmeurs, de l'implémenter pour permettre à nos services d'échanger des données confidentielles.

## 1.2 Objectifs

L'objectif de ce laboratoire est la manipulation des concepts mathématiques du cours théorique au travers de la réalisation de programmes en Python.

A la fin du laboratoire le camarade étudiant sera capable de :

1. réaliser un chiffrement de type symétrique moderne.

## 1.3 Énoncé

Le projet consiste à rendre un programme codé en Python qui permet d'implémenter un chiffrement et un déchiffrement GOST suivant un mode d'opération ECB, CBC et CTR (choisit au chiffrement). Le programme doit pouvoir être appelé sur des fichiers ".txt".

Afin de guider l'étudiant dans la programmation, un ensemble de stubs[1] est fourni. Les stubs sont organisés par fonctionnalité, chacune correspondant à un fichier python :

1. gost : reprend les fonctions permettant un chiffrement et un déchiffrement GOST suivant différents modes d'opération.
2. gost_feistel_function : reprend les transformations correspondant à $F(R_{i-1}, K_i)$ (permutation, S-box, etc.) appliquées au cas particulier du GOST.
3. key_generator : reprend la génération de clé pour chacun des round du GOST.
4. permutation : ... ça permute

En supplément de ces fichiers, un fichier utilities reprend les fonctions utilitaires utilisées par le programme (opérations sur fichiers et bytearray). Les fonctions de ce fichier peuvent être utilisées par l'étudiant pour se faciliter la vie.

1. Fonction sans code ne contenant que la documentation de la fonction


Il est demandé à l'étudiant de remplir chacun des stubs présents.

Étant donné la structure de l'algorithme, certaines fonctionnalités sont à réaliser en priorité. L'ordre suivant est suggéré :

1. utilities
2. permutation
3. key_generator
4. gost_feistel_function
5. gost

Chacun des fichier est accompagné d'un fichier test (foo.py a le fichier associé foo_test.py). Ce fichier reprend un ensemble de tests unitaire pour chacune des stubs. Ces tests permettent de vérifier le fonctionnement correct de chacune des fonctions demandées avant de continuer la progression. Il est primordial que tous les tests d'un fichier réussissent avant de passer au suivant.

Pour rappel, pour faire tourner un test, il faut placer le fichier de test dans le même dossier que votre code et faire un run via PyCharm. Si vous voulez exécuter UN SEUL test, vous pouvez appeler en ligne de commande

```
> python3 test_foo.py Test_foo.test_XXX
```

avec XXX le nom de la fonction à tester. Vous pouvez aussi cliquer sur ▶ à côté du test dans PyCharm.

## 1.4 Évaluation

L'évaluation portera sur le code complété. Le critère de réussite minimal est le passage de tous les tests proposés et une implémentation correcte des différents principes impliqué dans le GOST.

La qualité du travail (utilisation intelligente de librairies et clarté/concision efficacité du code) permettra d'augmenter la note.

L'implémentation du mode d'opération CTR et de la génération améliorée des clés (voir rappels) sont à faire en dernier lieu et la réussite est assurée qu'elles soient faites ou non ! Par contre si les autres stubs ne sont pas rempli et fonctionnels le projet sera en échec.

Si jugé nécessaire, des questions sur le projet peuvent être posées lors de l'oral.

## 1.5 Modalités

Les fichiers contenants les stubs complétés sont à rendre pour le dimanche suivant la dernière séance de laboratoire à 23h59, dans l'espace du cours.

## 1.6 Rappels

L'algorithme Gosudarstvennyi Standard Soyuza SSR (GOST) permet de chiffrer des données de manière symétrique.

L'algorithme est apparu dans les années 1970 et a été développé sur le modèle d'un réseau de Feistel. D'abord d'un niveau top-secret, il a été dé-classifié après la chute de l'URSS en 1994.

Il a été le pendant soviétique du chiffrement DES développé aux états-unis.

Les sections suivantes rappellent les éléments théoriques permettant d'appliquer le GOST.

Feistel

Les schémas de Feistel consistent à appliquer de manière répétée des opérations de substitution et de permutation aux données à chiffrer.

Le schéma général est représenté à la Figure 1.1.

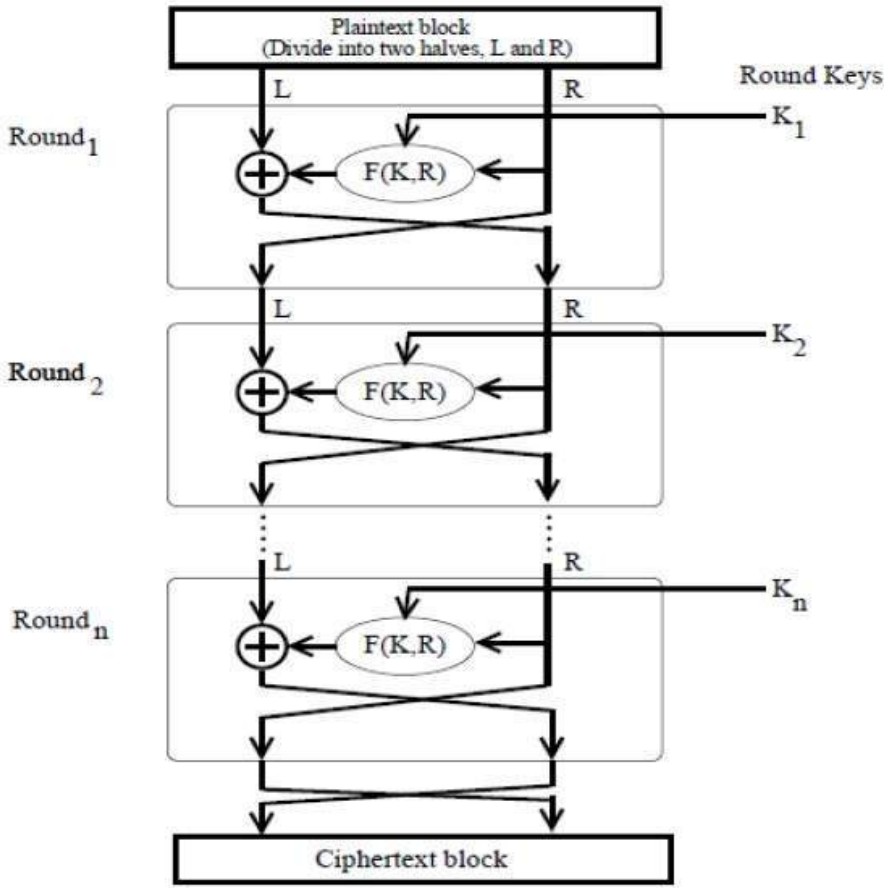


Figure 1.1 - Représentation du schéma de Feistel.

Les données à chiffrer sont tout d'abord séparées en blocs de 64 bits. Ces blocs sont ensuite tous traités de manière identique :

1. On sépare les blocs en deux parties (Left L et Right R) de 32 bits.
2. On effectue une opération de modification de la partie de droite qui utilise une clé K particulière(l'opération utilisée pour le GOST est détaillée dans les sections suivantes). On note la fonction qui effectue cette opération f.
3. On utilise ce résultat pour effectuer un XOR avec la partie L (substitution).
4. On inverse les parties L et R (Permutation). [2]

On peut ensuite répéter ces opérations un nombre quelconque de fois.
Mathématiquement, on peut écrire :

$$L_i = R_{i-1} \tag{1.1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \tag{1.2}$$

2. A l'exception du dernier round.

avec $i$ l'indice de l'étape.

GOST

L'algorithme GOST est construit directement sur le schéma de Feistel. Son principe de fonctionnement est représenté à la Figure 1.2.

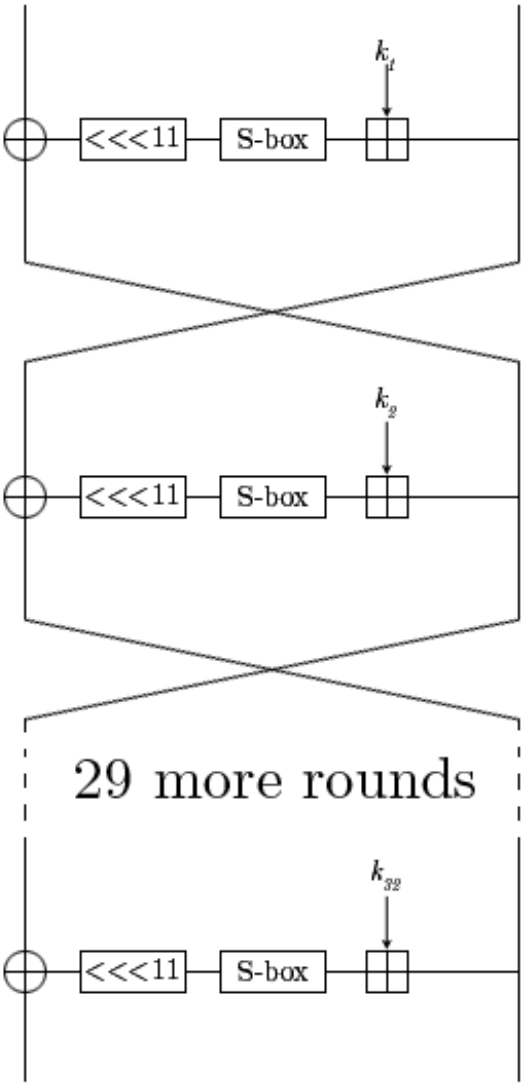


Figure 1.2 - Représentation du principe du GOST.

Il apporte ceci de spécifique par rapport au schéma général de Feistel :
- Il existe 32 rounds successifs.
- La fonction réalisée dans chacun des blocs utilise une sous-clé de 32 bits obtenue à partir d'une clé globale de 256 bits.

Chacune de ces étapes est décrite plus en détails dans les sections suivantes.


### Fonction de Feistel

Dans chacun des round, on applique une fonction de Feistel particulière $F(R_i, K_i)$ qui consistent en les opérations représentées à la Figure 1.2, décrites ci-dessous. Le résultat pourra être utilisé pour le "xor" avec la partie $L_i$.

1. La première étape est une addition modulo $2^{32}$ entre la partie de droite $R_i$ et la sous-clé $K_i$. La manière dont on extrait chaque sous-clé est décrite à la section 1.6.
2. On divise les 32 bits de $R_i$ en groupes de 4 bits. Chaque groupe de 4 bits détermine la position d'une table qui définit une substitution. Les tables sont définies dans la section 1.6.
3. Le résultat obtenu subi une permutation cyclique de 11 bits vers la gauche.

### Génération des sous-clé

Pour générer les 32 sous-clés $K_i$ nécessaires à chacun des rounds, on sépare la clé globale en 8 sous-clés $k_i$ de 32 bits ($k_1$ correspond aux 8 bits les plus faibles). On les applique comme décrit à la table ci-dessous.

| Round | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # sous-clé ($k_i$) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Round | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| # sous-clé ($k_i$) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

Pour déchiffrer, les clés sont appliquées dans l'ordre inverse.

### Tables de substitution (S-box)

Historiquement, chaque service avait une S-box qui lui était attribuée.[3] On peut par exemple trouver la S-box utilisée par la Banque centrale de la fédération de Russie à la table 1.1

Table 1.1 - Table de substitution (S-box) utilisée par la Banque centrale de la fédération de Russie. Les 4 bits d'entrée définissent la position de lecture dans la colonne. La valeur de sortie est exprimée en base hexadécimale.

| # | S-box |
|---|---|
| 1 | 4 A 9 2 D 8 0 E 6 B 1 C 7 F 5 3 |
| 2 | E B 4 C 6 D F A 2 3 8 1 0 7 5 9 |
| 3 | 5 8 1 D A 3 4 2 E F C 7 6 0 9 B |
| 4 | 7 D A 1 0 8 9 F E 4 6 C B 2 5 3 |
| 5 | 6 C 7 1 5 F D 8 4 A 9 E 0 3 B 2 |
| 6 | 4 B A 0 7 2 1 D 3 6 8 5 9 C F E |
| 7 | D B 4 1 3 F 5 9 0 A E 7 6 8 2 C |
| 8 | 1 F D 0 5 7 A 4 9 2 3 E 6 B 8 C |

On peut aussi trouver une table définit dans le standard GOST R 34.12-2015, représentée à la table 1.2.

### Mode d'opération

Le mode d'opération le plus simple consiste simplement à appliquer le GOST à des blocs successifs de 64 bits de données à chiffrer. Ce mode d'opération est connu sous le nom Electronic Code Book (ECB) et est représenté à la Figure 1.6.

3. On raconte que les services les moins appréciés obtenait une S-box dont on connaissait la faiblesse pour des facilités d'espionnage.

Table 1.2 - Table de substitution (S-box) utilisée par le standard R 34.12-2015. Les 4 bits d'entrée définissent la position de lecture dans la colonne. La valeur de sortie est exprimée en base hexadécimale.

| # | S-box |
|---|---|
| 1 | C 4 6 2 A 5 B 9 E 8 D 7 0 3 F 1 |
| 2 | 6 8 2 3 9 A 5 C 1 E 4 7 B D 0 F |
| 3 | B 3 5 8 2 F A D E 1 7 4 C 9 6 0 |
| 4 | C 8 2 1 D 4 F 6 7 0 A 5 3 E 9 B |
| 5 | 7 F 5 A 8 1 6 D 0 9 3 E B 4 2 C |
| 6 | 5 D F 6 9 2 C A B 7 8 1 4 3 E 0 |
| 7 | 8 E 2 5 6 9 1 C F 4 B 0 D A 3 7 |
| 8 | 1 7 E D 0 5 8 3 4 F A 6 9 C B 2 |

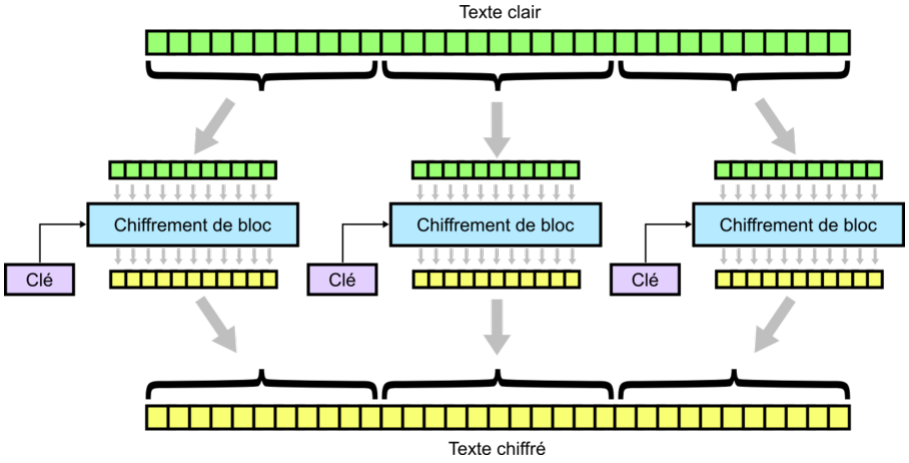


Figure 1.3 - Mode d'opération ECB.

Néanmoins, lorsque ce mode est utilisé, il est possible de voir apparaître certains motifs, ce qui est en contradiction avec les principes de la cryptographie. Ceci vient du fait que des blocs initiaux identiques donneront des blocs chiffrés identiques.

Plusieurs modes d'opérations ont été développés afin de palier à ce problème. On pense notamment au Cipher Block Chaining (CBC).

Le mode d'opération CBC consiste à effectuer sur chacun des blocs à chiffrer une opération de XOR avec le bloc chiffré précédemment avant de passer dans la "moulinette" GOST, comme représenté à la Figure 1.6. Pour le premier bloc à coder, un vecteur d'initialisation de 64 bits est utilisé.

Le mode d'opération CTR consiste à effectuer sur chacun des blocs à chiffrer un nombre issu d'un compteur spécifique. On chiffre ce nombre via GOST avant de faire un XOR avec le texte clair, comme représenté à la Figure 1.6. L'énorme avantage de ce mode est de pouvoir travailler en parallèle, chose qui amènerait un bonus non négligeable si elle était implémentée. On propose ici de combiner un vecteur d'initialisation (nonce) avec le compteur en effectuant un simple XOR. Ce vecteur d'initialisation est fourni premier élément, en ayant été chiffrer.

### Génération améliorée des clés

Le camarade Brezhnev trouve que la génération de clé est trop simple comparée au schéma utilisé par ces porcs de capitalistes. Il aimerait pouvoir choisir entre la manière décrite

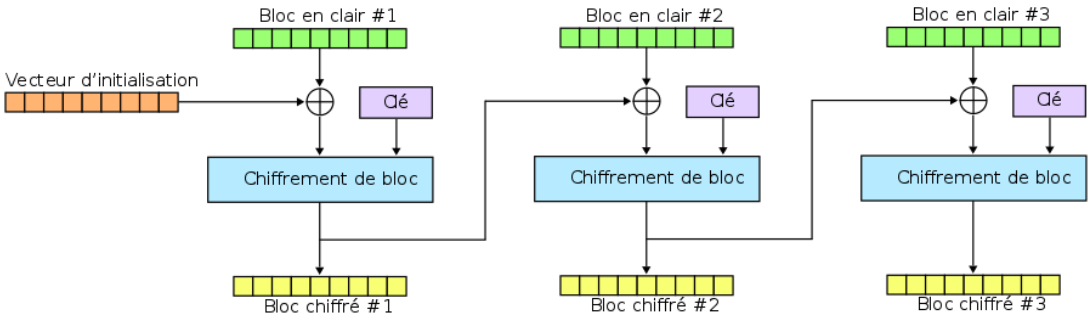


Figure 1.4 - Mode d'opération CBC.

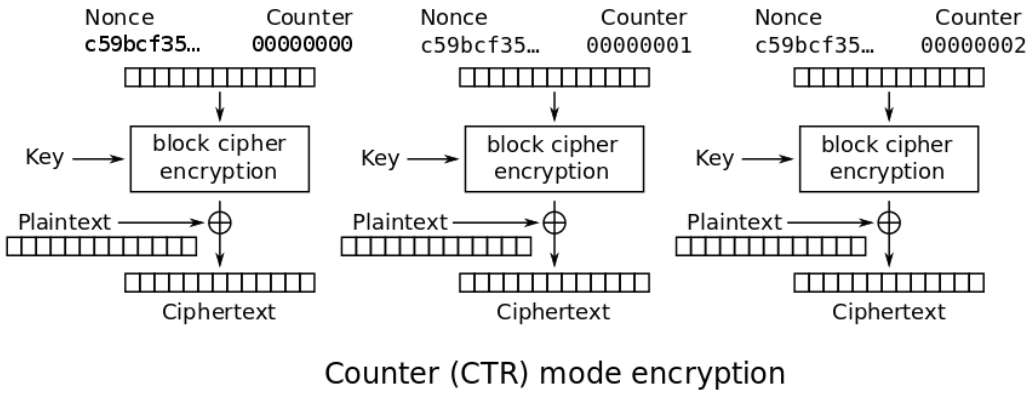


Figure 1.5 - Mode d'opération CTR.

à la section 1.6 et une qui s'apparente à celle utilisée pour le DES, décrite dans cette section. Il vous faut donc implémenter cette autre manière.

Les clés pour chacun des rounds sont obtenues à partir d'une clé globale de 128 bits. Pour les obtenir, une opération de shift suivi d'une permutation est effectuée, comme représenté à la Figure 1.6.

Les étapes suivantes sont effectuées préliminairement :

1. De la clé globale de 256 bits, dont on ne considère que les 128 premiers bits (à droites) séparée en deux parties de 64 bits (droite = 64 bits de poids faible).
2. On enlève 8 bits de parité de chaque partie.
3. On sépare les 56 bits restants en deux parties de 28 bits.

Pour chacun des rounds on effectue alors les opérations suivantes en parallèle sur les deux groupes de 56 bits :

1. On décale les groupes de 28 bits de 1 ou 2 bits vers la gauche. (1 pour les rounds 1, 2, 9, 16 et 2 pour les autres)
2. On permute les bits obtenus suivant une table donnée. Cette table ne comprend que 48 entrées et laisse donc tomber 8 bits.
3. On conserve les 32 premiers bits comme clé du round.

Ces opérations sont répétées 16 fois et on utilise en alternance le résultat de droite et de gauche dans chacun des rounds d'encodage.

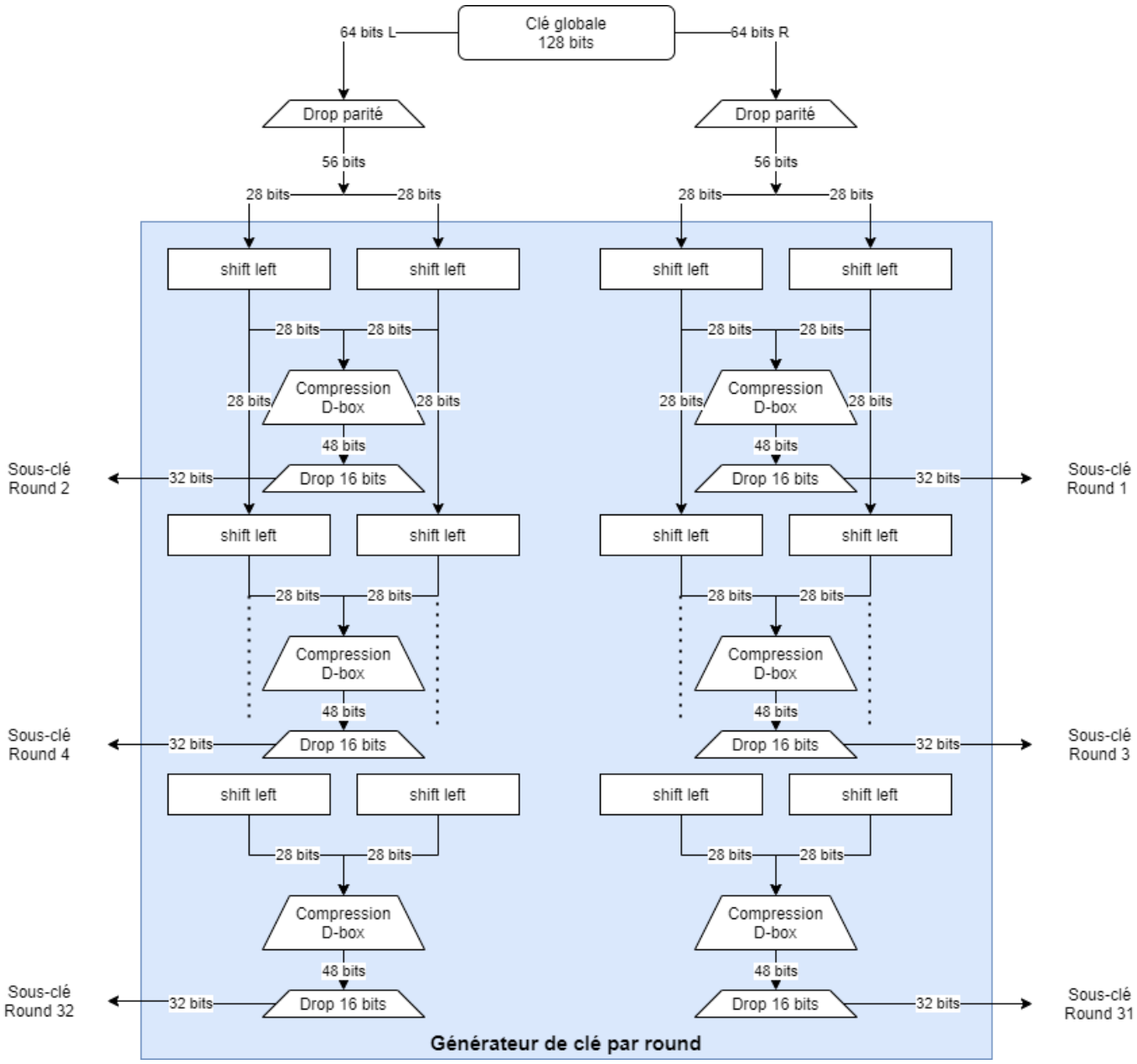


Figure 1.6 - Génération des clés pour chacun des 32 rounds.

Les tables de drop de parité et de compression sont disponibles à cette page wikipédia .

## 1.7 Références

Afin de compléter le bref rappel théorique, un ensemble de références sont proposées à l'étudiant :

- Page RFC décrivant l'algorithme de chiffrement/déchiffrement.

C C  C P

- Le livre Applied Cryptography (second edition) dont le rappel est inspiré, disponible sur demande.
- La page wikipédia reprenant toutes les tables utilisées pour le DES, on utilise les même tables de 'parity drop' et de compression pour la version améliorée de génération de clé.
- La page wikipédia du GOST. (quelques images et les S-box viennent d'ici)
- L'autre page wikipédia qui explique les modes d'opérations, disponible aussi en anglais. (les images relatives viennent d'ici)