Question 4 :

Pour obtenir l’état initial s2 du LFRS de taille 25 à partir de (z1, z2, z3) et (x1, x2, x3). On doit d’abord récupérer la valeur de y.

Pour cela vu que l’on connait les valeurs de x et z pour les 3 premiers octets et que l’on sait que z se calcul avec z = (x + y + c) % 256. Avec c = 1 si x + y > 255 et 0 sinon.

On peut poser une équation et obtenir : y = (z - x - c) % 256

Pour le calcul de y1 pas de problème car **c** est initialiser à 0 mais pour de y2 et y3, il faudra vérifier que dans la sortie précédente que x + y > 255. **c** étant la retenue de l’addition

Donc maintenant on connait les 3 premiers octets de y (y1, y2, y3) ce qui fait 24 bits donc on connait s2.

Question 5 :

En exploitant la méthode vue dans la question 4.

On génère toutes les 2^16 combinaison possible pour l’état initial s1 du LFSR 17. Ensuite pour chaque combinaison on génère les trois premiers octets x1, x2, x3.

Pour chaque x1, x2, x3 généré, on calcule y1, y2, y3 en utilisant y = (z - x - c) % 256, puis on les insère dans une liste.

Une fois la liste de tous les y1, y2, y3 possibles pour l’état initial s2 du LFSR 25. On génère le potentiel état initial de s2 à partir de y1, y2, y3 puis on vérifie si les octets z4, z5, z6 correspondent. Si oui alors on a trouvé l’état initial de s2, sinon on passe au prochain y1, y2, y3. On devra répété cela jusqu’à trouver l’état initial.

Ce qui coute le plus de ressource dans cette attaque est de générer les 2^16 état initial possible de s1. Sinon le travail de recherche de s2 à partir de la liste créer est beaucoup moins couteux.