Rumba20

Présentation Rumba20

La fonction de hachage Rumba20 a été présentée en 2007 dans le cadre d’étude des attaques d'anniversaire généralisées. Elle réutilise une grande partie de la fonction hachage Salsa20 sauf qu' à l’entrée elle prend une valeur de 1536 et fournie une sortie de 512 bits.

Présentation Salsa20

Salsa20 est un chiffrement de flux proposé par Daniel Bernstein, elle est basée sur des opérations d'addition sur 32 bits, d'addition exclusive (XOR) et de rotations sur les bits, qui transforme une clé de 256 bits, un nombre pseudo aléatoire de 64 bits, et une position de flux sur 64 bits en un bloc de données en sortie de 512 bits

Salsa20 core

Cette fonction est utilisée par salsa20 pour crypter les données , et prend en entrée des textes sur 64 byte et fournie une sortie de 64 byte.
Rumba utilise salsa core pour compresser des chaines de caractères de 192 byte en 64 byte.

```
#define R(a,b) (((a) << (b)) | ((a) >> (32 - (b))))
    void salsa20_word_specification(uint32 out[16],uint32 in[16])
    {
      int i;
      uint32 x[16];
      for (i = 0;i < 16;++i) x[i] = in[i];
      for (i = 20;i > 0;i -= 2) {
        x[ 4] ^= R(x[ 0]+x[12], 7);          x[ 8] ^= R(x[ 4]+x[ 0], 9);
        x[12] ^= R(x[ 8]+x[ 4],13);          x[ 0] ^= R(x[12]+x[ 8],18);
        x[ 9] ^= R(x[ 5]+x[ 1], 7);          x[13] ^= R(x[ 9]+x[ 5], 9);
        x[ 1] ^= R(x[13]+x[ 9],13);          x[ 5] ^= R(x[ 1]+x[13],18);
        x[14] ^= R(x[10]+x[ 6], 7);          x[ 2] ^= R(x[14]+x[10], 9);
        x[ 6] ^= R(x[ 2]+x[14],13);          x[10] ^= R(x[ 6]+x[ 2],18);
        x[ 3] ^= R(x[15]+x[11], 7);          x[ 7] ^= R(x[ 3]+x[15], 9);
        x[11] ^= R(x[ 7]+x[ 3],13);          x[15] ^= R(x[11]+x[ 7],18);
        x[ 1] ^= R(x[ 0]+x[ 3], 7);          x[ 2] ^= R(x[ 1]+x[ 0], 9);
        x[ 3] ^= R(x[ 2]+x[ 1],13);          x[ 0] ^= R(x[ 3]+x[ 2],18);
        x[ 6] ^= R(x[ 5]+x[ 4], 7);          x[ 7] ^= R(x[ 6]+x[ 5], 9);
        x[ 4] ^= R(x[ 7]+x[ 6],13);          x[ 5] ^= R(x[ 4]+x[ 7],18);
        x[11] ^= R(x[10]+x[ 9], 7);          x[ 8] ^= R(x[11]+x[10], 9);
        x[ 9] ^= R(x[ 8]+x[11],13);          x[10] ^= R(x[ 9]+x[ 8],18);
        x[12] ^= R(x[15]+x[14], 7);          x[13] ^= R(x[12]+x[15], 9);
        x[14] ^= R(x[13]+x[12],13);          x[15] ^= R(x[14]+x[13],18);
      }
      for (i = 0;i < 16;++i) out[i] = x[i] + in[i];
    }
```

Relation entre Salsa20 et Rumba20

Rumba20 est établie sur Salsa20 et est conçu pour fournir une meilleur résistance aux collisions.
Pour bien comprendre la relation entre les deux fonctions de fonctionnement à l'entrée de la fonction Rumba on a un message M qui est sur 1536 bits, le message est scindé en quatre blocs M1, M2, M3, M4 ; chaque Mi est sur 384 bits ; la fonction Rumba ainsi définie est donne comme suite :

Rumba (M) = F1 (M1) F2 (M2) F3 (M3) F4 (M4)

Où chacun Fi est un exemple de la fonction Salsa20 avec des constantes diagonales distincte;

Schéma de la relation

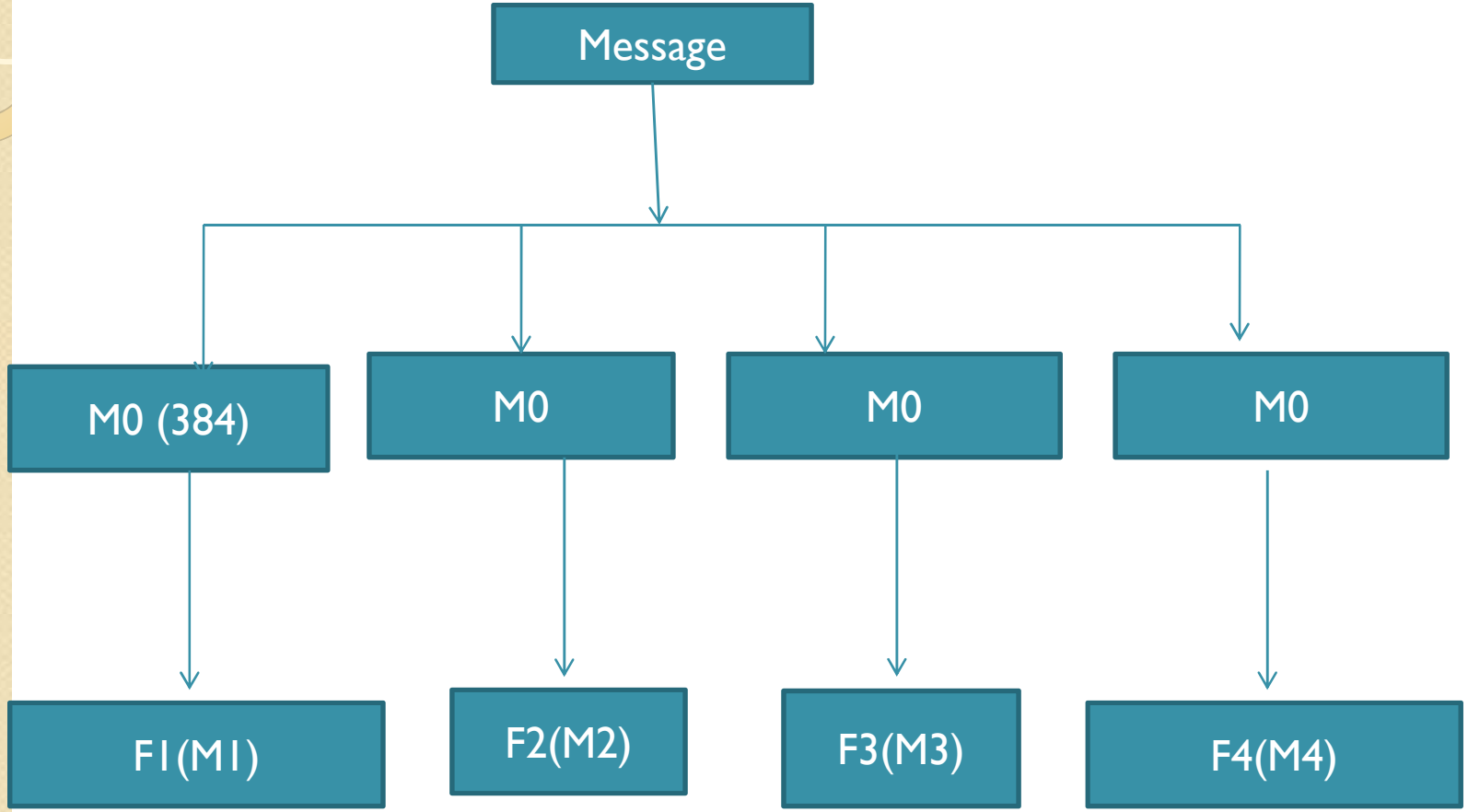
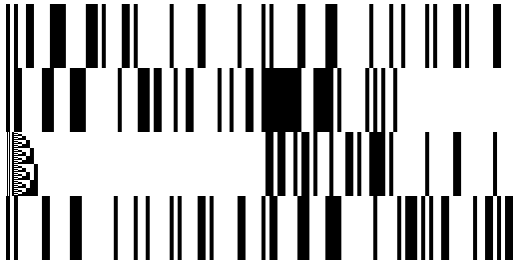
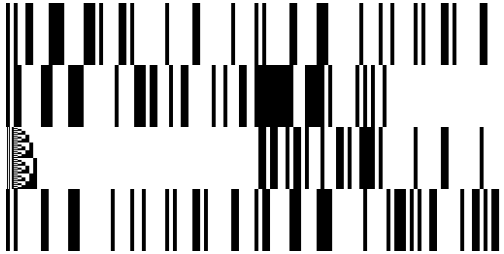


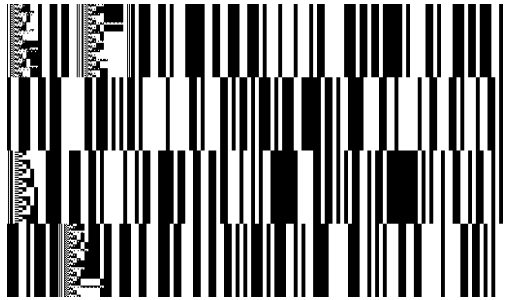
Tableau des constantes

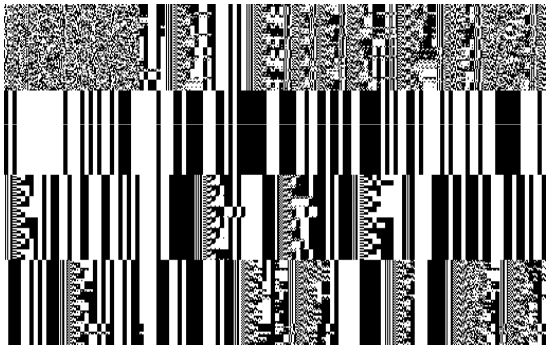
	Salsa20	Rumba20			
		F1	F2	F3	F4
C0	61707865	73726966	6f636573	72696874	72756f66
C1	3320646E	6d755274	7552646e	6d755264	75526874
C2	79622D32	30326162	3261626d	30326162	3261626d
C3	6B206574	636f6c62	6f6c6230	636f6c62	6f6c6230

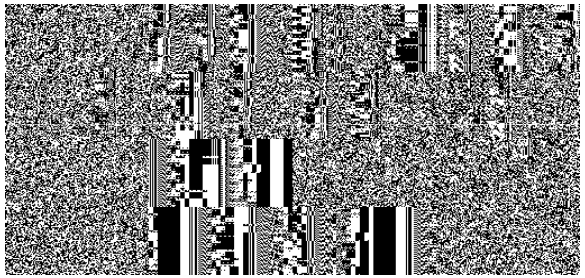
SALSA20 diffusion

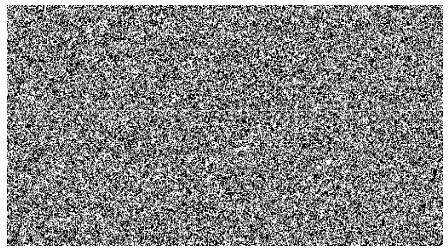


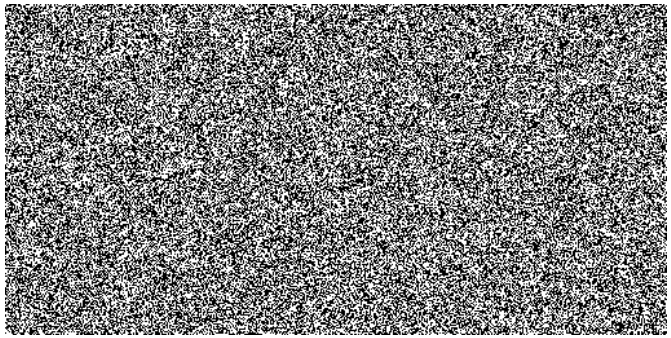


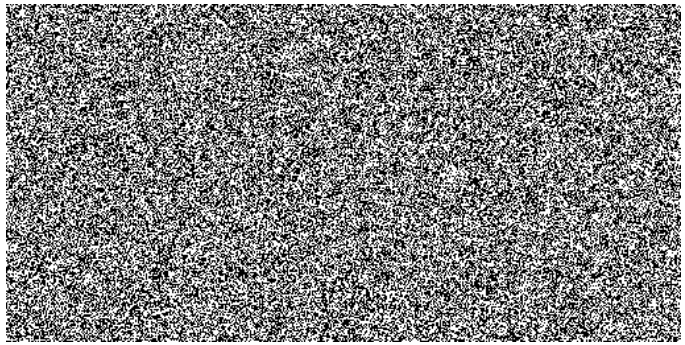




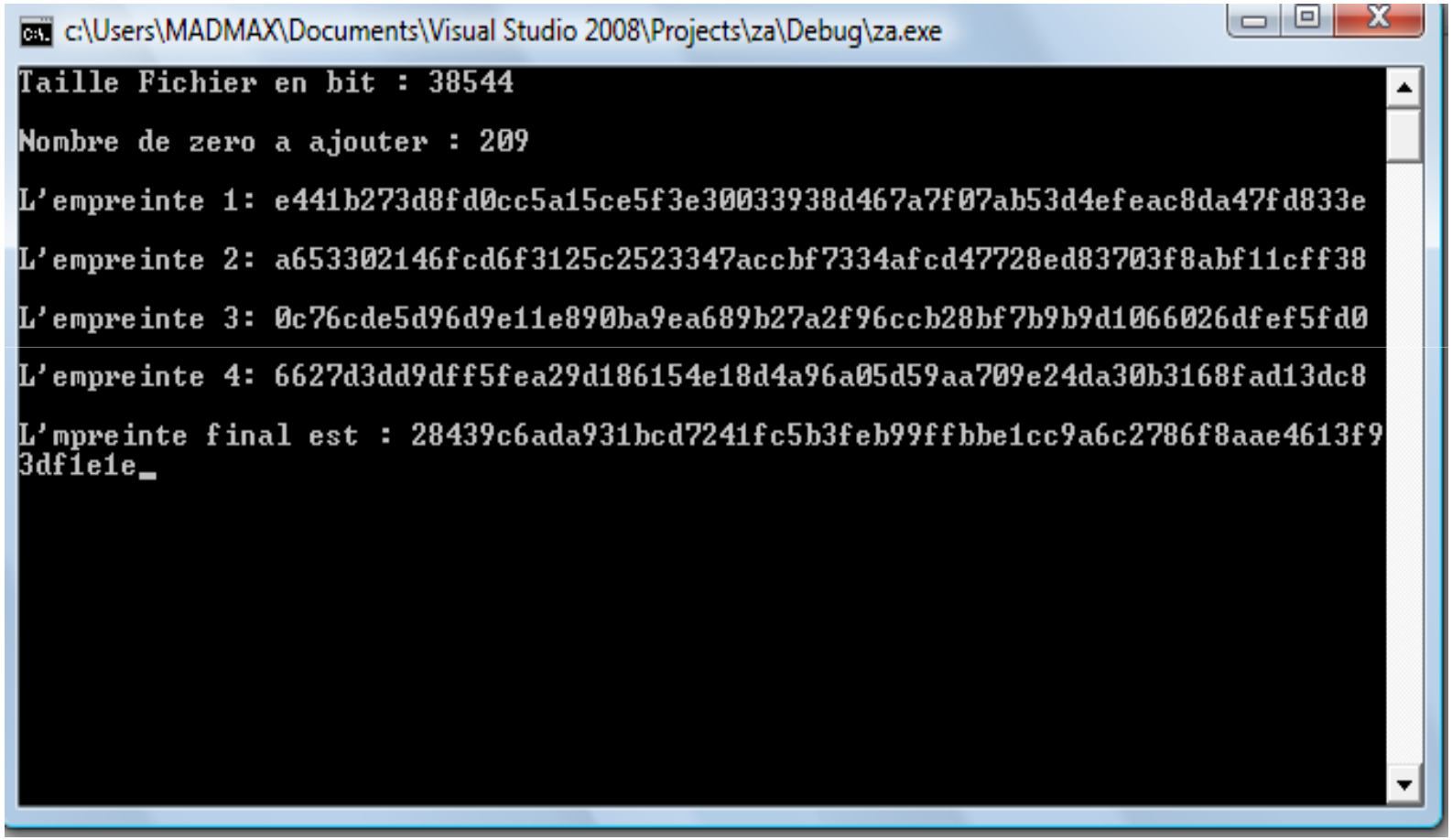








Implémentation



Conclusion

Bien que fonction du hachage rumba20 n'ait pas très utilisé actuellement car le noyau de la fonction n'est pas dur comme on aurait pu s'y attendre; Puisque la fonction salsa20 utilisée contient des Collisions facilement caractérisable, ainsi qu'un comportement linéaire indésirable . Cet étant indiqué, on peut considérer que la conception rumba20 est très innovatrice.

D'avantage de travail devrait être encouragé pour y apporte d'autre amélioration; En particulier, un nouveau, peut-être une définition plus complexe et plus longue de la fonction de quarterround Devrait mener à un primitif qui ne serait pas vulnérable à quelconque entrée et fournissent un algorithme à niveau élevé de sécurité.