ALGORITHME

C A J S

INTRODUCTION

Dans le cadre du cours de cryptographie, nous avons été chargés de concevoir un algorithme a clé symétrique avec les paramètres suivants:

- L’algorithme doit prendre en entrée un message et une clé secrète partagée entre les deux parties
- L’algorithme doit produire en sortie le message chiffré correspondant
- Doit être capable de résister à des techniques de cryptanalyse tel que la brute force et l’analyse différentielle.

DÉFINITION

Cryptanalyse: La cryptanalyse est l'art de déchiffrer un message chiffré sans connaître la clé. Elle regroupe toutes les méthodes utilisées pour casser un système cryptographique. Parmi ces méthodes, le brute force consiste à essayer toutes les combinaisons possibles de clés jusqu'à trouver celle qui permet de déchiffrer le message. C’est la méthode la plus simple, mais souvent la plus coûteuse en temps et en ressources.

ALGORITHME

Nous vous présentons maintenant notre algorithme CAJS TRIBE, voici le processus de développement.

ALGORITHME 1.0

- Clé avec trois lettres (24 bits)
- Convert to binary
- Permutation de type inverse
- Xor

Problème: L'algorithme ne passe pas la force brute ni l'analyse différentielle. D'autant plus, clé n'était pas sécuritaire.

ALGORITHME 2.0

ALGORITHME 3.0

ALGORITHME 1.0

ALGORITHME 2.0

ALGORITHME 3.0

- Clé 128 bits (16 caractères)
- Convert to binary
- Permutation de type inverse
- Xor
- Shift left (5)

Problème: L'algorithme passe la force brute, mais n'est toujours pas en mesure de résister à l'analyse différentielle.

ALGORITHME 1.0

ALGORITHME 2.0

ALGORITHME 3.0

- Clé 128 bits (16 caractères)
- Convert to binary
- Permutation Rot de 8 bits personnalisée + swap
- Substitution (S_box) par blocs de 4 bits
- Shift left (5)
- Mix column (rotation circulaire par bloc de 4 nibbles)
- Effectuer XOR avec la sous-clé de chaque ronde
- Mix column
- Xor (12 rondes)

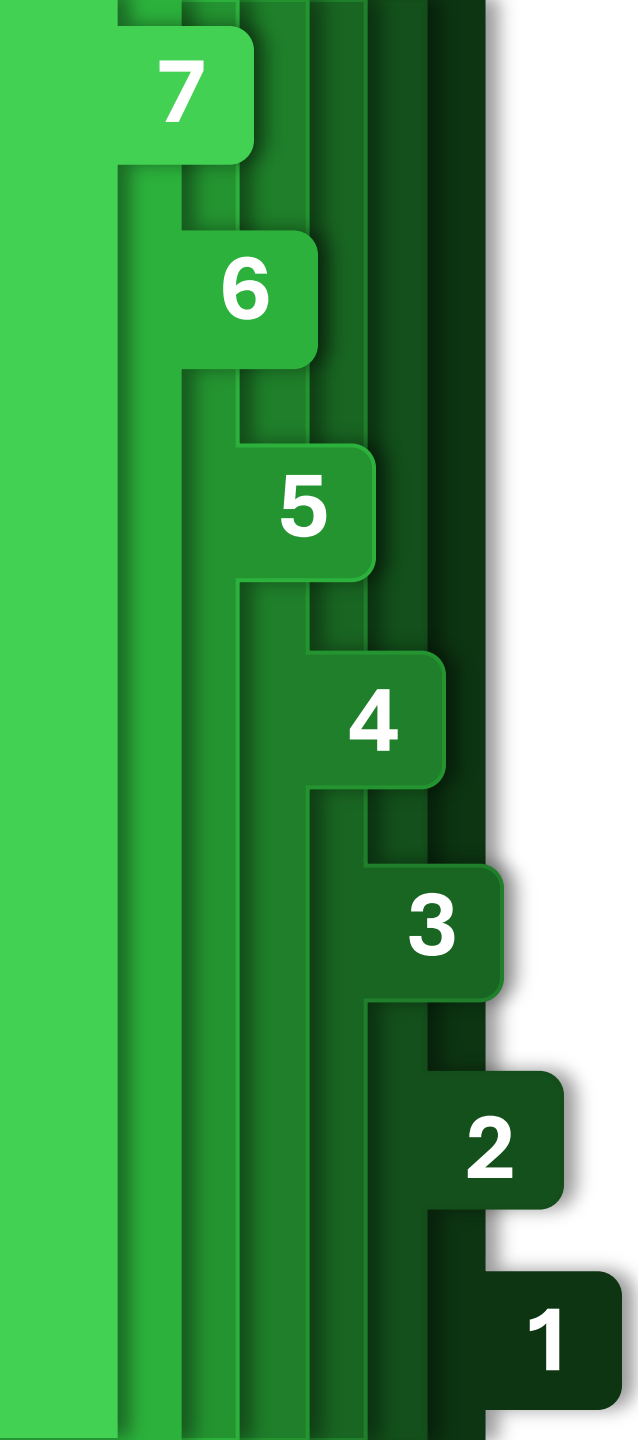
Résistance à la force brute et l'analyse différentielle et une clé de 128 bits.

ALGORITHME 3.0

- Conversion de la clé en binaire (128 bits): La clé fournie est transformée en une chaîne binaire. Si elle est trop courte, elle est complétée (padding) pour atteindre exactement 128 bits. Cela garantit une base uniforme pour le chiffrement.
- Conversion du texte clair en binaire: Le texte à chiffrer est converti caractère par caractère en binaire (chaque caractère devient 8 bits). Ce texte binaire est ensuite découpé en blocs de 128 bits. Si un bloc est incomplet, il est aussi complété.
- Génération des sous-clés: À partir de la clé principale, l'algorithme génère une série de sous-clés. Chaque sous-clé est utilisée à une ronde spécifique du chiffrement. Cela renforce la sécurité en rendant chaque étape unique.

ALGORITHME 3.0

- Chiffrement par rondes successives: Chaque bloc de texte passe par plusieurs transformations, répétées pour chaque ronde .
  - Permutation hybride : réorganise les bits du bloc selon un schéma complexe.
  - Substitution : remplace certains bits par d'autres selon une table ou une logique définie.
  - Décalage à gauche (shift left) : les bits sont déplacés vers la gauche, ce qui brouille leur position.
  - Mélange des colonnes (mix columns) : combine les bits entre eux pour créer des dépendances internes.
  - XOR avec la sous-clé : applique une opération logique entre le bloc et la sous-clé, deux fois par ronde.
  - Assemblage du texte chiffré final Une fois tous les blocs chiffrés, ils sont réunis pour former le texte chiffré complet. Ce résultat est une longue chaîne binaire, que tu peux ensuite encoder (par exemple en base64) pour le stocker ou le transmettre.



Algorithme

CAJS TRIBE

7

6

5

4

3

2

1

Pourquoi la clé de 128-Bits?

- Sécurité de chiffrement renforcée

Une clé de 128 bits offre une protection robuste contre les attaques par force brute, rendant l'accès non autorisé extrêmement difficile

- Séquence binaire unique

Chaque clé de 128 bits est une séquence unique de chiffres binaires, créant un nombre immense de combinaisons possibles.

- Représentation cryptographique

Les visuels incluent souvent des clés numériques ou métalliques avec des motifs binaires pour illustrer la puissance et la complexité du chiffrement.

Le niveau de sécurité est élevé car il faudrait un nombre astronomique d'années pour trouver la clé par force brute, même avec des superordinateurs

7

6

5

4

3

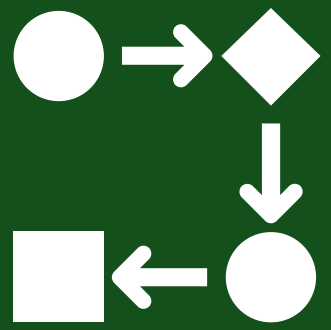
Conversion en binaire

- Conversion de la clé ainsi que du message.

Clé: 3Pv@hthYo8w8j22F

Message: Celui qui détient la clé, détient le message.

- Séparer le message en bloc de 128 bits. Si un bloc ne fait pas 128 bits, il a un ajout de zéros à la fin du dernier bloc.



2

1

7

6

5

4

Permutation Rot 8 bits + Swap

- Séparation des 128 bits en 8 bits
- Décalage de chaque bloc par la gauche
- Division du bloc en deux. La partie gauche vers la droite et vice-versa.

3

2

1

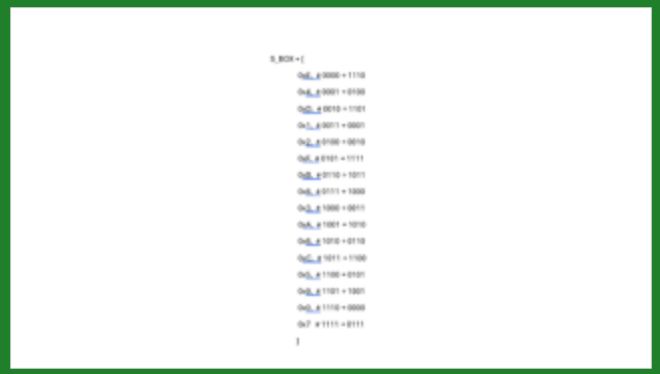
7

6

5

Substitution S_box par bloc de 4-bits

- Substituer les blocs de 4 bits par les valeurs de substitution choisies.



4

3

2

1

7

6

Shift left (5)

5

- Décalage des bits de 5 vers la gauche.

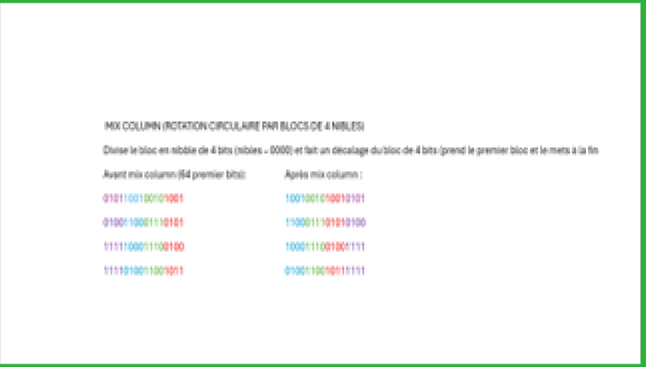
4

3

2

1

Mix Column – rotation circulaire par bloc de 4 nibles



MIX COLUMN (ROTATION CIRCULAIRE PAR BLOCS DE 4 NIBLES)

Divise le bloc en nibble de 4 bits (nibles – 0000) et fait un décalage du bloc de 4 bits (prend le premier bloc et le mets à la fin

Avant mix column (64 premier bits):	Après mix column :
0101100100101001	1001001010010101
0100110001110101	1100011101010100
1111100011100100	1000111001001111
1111010011001011	0100110010111111

Effectuer XOR avec la sous clé de chaque ronde

- **Avant toutes les permutations, 12 sous-clés ont été crées. Une qui sera utilisée à chaque ronde.**

7

6

5

4

3

2

1

RÉPÉTITION

Des deux dernières étapes soient le mix column ainsi que le Xor, car il a été déterminé qu'il y a un meilleur résultat dans la diffusion des bits.

PRINCIPE DE SÉCURITÉ

Confidentiel – illisible sans la clé correspondante

Robuste:

- Résistance à l’analyse différentielle
- Résistance à la force brute

DÉCHIFFREMENT

- Xor avec la sous-clé 12 jusqu'à 1
- Mix column (inverse)
- Xor à nouveau
- Mix column (inverse)
- Shift right (5)
- Substitution S_box, revenir avec les bits du départ
- Swap + permutation rot de 8 bits inversé