Entropie - Exemple

Entropie définition

Définition formelle

- L'entropie de Shannon se définit ainsi pour une variable aléatoire X avec un ensemble de symboles x_i et leurs probabilités $p(x_i)$:
- $H(X) = -\sum_i p\,(x_i) log_2(p(x_i))$
- $H(X)$ est exprimée en **bits**.
- Si chaque symbole est équiprobable, l'entropie est maximale.
- Exemple :
- Un bit totalement aléatoire (0 ou 1 avec $p = 0{,}5$) → entropie de 1 bit.
- Un bit biaisé (0 avec 90 %, 1 avec 10 %) → entropie < 1 bit (moins imprévisible).

Formule entropie

$$H_b(X) = \sum_{i=1}^{n} P_i \log_b \left(\frac{1}{P_i} \right) = -\sum_{i=1}^{n} P_i \log_b P_i .$$

$\log_2\left(\frac{1}{P}\right) = -\log_2(P)$ $\log_2(1) = 0$ $\log_2(1) = x \quad \Leftrightarrow \quad 2^x = 1$

On utilise en général un logarithme à base 2 car l'entropie possède alors les unités de bit/symbole.

$$H(X) = H_2(X) = -\sum_{i=1}^{n} P_i \log_2 P_i .$$

Entropie - Pile ou face

- On lance une pièce à pile ou face en connaissant les probabilités des deux résultats ; le modèle classique est celui d'un processus de Bernoulli.
- L'entropie du résultat du lancer à venir est maximale pour une pièce équilibrée. Il s'agit bien du cas d'incertitude maximale car c'est là qu'il est le plus difficile de prédire l'issue du lancer ; le résultat de chaque lancer de la pièce donne un bit complet d'information.

$$H_2(X) = -\sum_{i=1}^{n} P_i \log_2 P_i.$$

$$H(X) = -\sum_{i=1}^{} \frac{1}{2} \log_2 \frac{1}{2}$$

$$= -\sum_{i=1}^{2} \frac{1}{2} \cdot (-1) = 1$$

- $H(S) = \Sigma_i \; p_i \log_2 1/p_i$
  - `pile` : ½ $\log_2$ (1 / ½) = ½ $\log_2$ 2 = ½ * 1 = ½
  - `face` : ½ $\log_2$ (1 / ½) = ½ $\log_2$ 2 = ½ * 1 = ½
  - $H(S)$ = ½ + ½ = 1 ***bit***

Entropie –alphabet de 26 symboles équiprobables (A – Z)

Étape 1 : probabilité de chaque symbole

Puisque tout est **équiprobable** :

$$p_i = \frac{1}{26}, \quad \text{pour } i = 1, 2, \ldots, 26$$

Étape 2 : formule de l'entropie

$$H(X) = -\sum_{i=1}^{26} p_i \log_2(p_i)$$

Étape 3 : simplification

Comme toutes les probabilités sont identiques :

$$H(X) = -26 \cdot \frac{1}{26} \log_2\left(\frac{1}{26}\right)$$

$$H(X) = -\log_2\left(\frac{1}{26}\right)$$

$$H(X) = \log_2(26)$$

Étape 4 : calcul numérique

$$\log_2(26) = \frac{\ln(26)}{\ln(2)}$$

$$\ln(26) \approx 3.2581, \quad \ln(2) \approx 0.6931$$

$$\log_2(26) \approx \frac{3.2581}{0.6931} \approx 4.7004$$

Cela signifie qu'en moyenne, **il faut environ 4,7 bits pour coder un symbole de l'alphabet latin sans perte**.

Entropie –alphabet de 52 symboles équiprobables (a – Z, a – z)

Étape 1 : probabilité de chaque symbole

$$p_i = \frac{1}{52}, \quad \text{pour } i = 1, 2, \ldots, 52$$

Étape 2 : entropie de Shannon

$$H(X) = -\sum_{i=1}^{52} p_i \log_2(p_i)$$

Puisque les p_i sont identiques :

$$H(X) = -52 \cdot \frac{1}{52} \log_2\left(\frac{1}{52}\right)$$

$$H(X) = \log_2(52)$$

Étape 3 : calcul numérique

$$\log_2(52) = \frac{\ln(52)}{\ln(2)}$$

- $\ln(52) \approx 3.9512$
- $\ln(2) \approx 0.6931$

$$\log_2(52) \approx \frac{3.9512}{0.6931} \approx 5.7004$$

Cela signifie qu'en moyenne, **il faut environ 5,7 bits pour coder un symbole de l'alphabet latin sans perte**.

Entropie – alphabet de 72 symboles équiprobables (A – Z, a – z, 0 – 9, !@#$%?&*-+)

Étape 1 : probabilité de chaque symbole

$$p_i = \frac{1}{72}, \quad \text{pour } i = 1, 2, \ldots, 72$$

Étape 2 : entropie de Shannon

$$H(X) = -\sum_{i=1}^{72} p_i \log_2(p_i)$$

Comme toutes les probabilités sont identiques :

$$H(X) = -72 \cdot \frac{1}{72} \log_2\left(\frac{1}{72}\right)$$

$$H(X) = \log_2(72)$$

Étape 3 : calcul numérique

$$\log_2(72) = \frac{\ln(72)}{\ln(2)}$$

- $\ln(72) \approx 4.2767$
- $\ln(2) \approx 0.6931$

$$\log_2(72) \approx \frac{4.2767}{0.6931} \approx 6.1699$$

Cela signifie qu'en moyenne, **il faut environ 6,17 bits pour coder un symbole**

Shannon – sécurité informatique

Applications en sécurité informatique

- **Mots de passe** :
  - Un mot de passe aléatoire de 12 caractères parmi 62 symboles possibles (A-Z, a-z, 0-9) → entropie ≈ $log_2(62^{12})$ ≈**71 bits**.
  - Un mot de passe commun comme "password123" → entropie très faible, car prévisible.

Étape 1 : entropie d'un seul caractère

Chaque caractère est choisi **uniformément** parmi 62 symboles.

$$H_{\text{caractère}} = \log_2(62)$$

- $\ln(62) \approx 4.1271$
- $\ln(2) \approx 0.6931$

$$\log_2(62) = \frac{4.1271}{0.6931} \approx 5.954 \text{ bits}$$

Donc **un caractère apporte environ 5.95 bits d'entropie**.

Étape 2 : entropie du mot de passe de 12 caractères

Puisque les 12 caractères sont indépendants et équiprobables :

$$H_{\text{mot de passe}} = 12 \times H_{\text{caractère}}$$

$$H_{\text{mot de passe}} \approx 12 \times 5.954 = 71.45 \text{ bits}$$

Shannon – sécurité informatique

Applications en sécurité informatique

- **Mots de passe** :
  - Un mot de passe aléatoire de 12 caractères parmi 62 symboles possibles (A-Z, a-z, 0-9) → entropie ≈ $log_2(62^{12})$ ≈**71 bits**.
  - Un mot de passe commun comme "password123" → entropie très faible, car prévisible.
- **Clés cryptographiques** :
  - Une clé AES de 128 bits générée aléatoirement a une entropie de 128 bits.
  - Si elle est générée à partir d'un mauvais générateur pseudo-aléatoire → entropie plus basse, vulnérable aux attaques.

Entropie mot de passe et clé

Qu'est-ce que ça mesure ?

- L'entropie en bits mesure la **quantité d'incertitude** ou d'**imprévisibilité**.
- En sécurité, cela correspond au **nombre de tentatives qu'un attaquant doit faire, en moyenne, pour deviner le mot de passe** s'il ne connaît rien d'autre.

2. Exemple simple

- Un mot de passe de **1 bit d'entropie** → 2 possibilités (comme pile ou face).
- 2 bits d'entropie → 4 possibilités.
- 10 bits d'entropie → environ $2^{10} = 1024$ possibilités.
- 20 bits d'entropie → environ $2^{20} \approx 1 million$ possibilités.
- 40 bits d'entropie → environ $2^{40} \approx 1 trillion$ possibilités.
- Donc, **plus le nombre de bits est élevé, plus le mot de passe est difficile à deviner par force brute**.

Entropie mot de passe et clé

Application aux mots de passe

- Prenons un alphabet de **62 symboles possibles** (A–Z, a–z, 0–9).
- Entropie maximale par caractère = $log_2(62) \approx 5{,}95$ bits.

- Un mot de passe de 8 caractères **totalement aléatoire** → $8 \times 5{,}95 \approx 47{,}6$ bits.
- Un mot de passe de 12 caractères aléatoires → $12 \times 5{,}95 \approx 71{,}4$ bits.
- Cela veut dire que le nombre de combinaisons possibles est $2^{71} \approx 2{,}36 \times 10^{21}$,donc presque impossible à casser par force brute.

- Les **bits d'entropie** indiquent la **force théorique** d'un mot de passe.
- Plus il y a de bits, plus le mot de passe est difficile à deviner.
- Chaque bit supplémentaire **double** l'espace de recherche pour un attaquant.

Force brute - temps

- **Hypothèses**

- Attaquant capable de tester **10^9 (1 milliard) mots de passe par seconde** (ce qui est réaliste avec du matériel moderne type GPU/ASIC).
- Temps $\approx \frac{2^H}{10^9}$ secondes, où H est l'entropie en bits.

Entropie (bits)	Nombre de combinaisons possibles	Temps pour casser (1 milliard essais/s)
20 bits	≈ 1 million (2^{106})	≈ 1 ms (instantané)
30 bits	≈ 1 milliard (2^{30})	≈ 1 seconde
40 bits	≈ 1 trillion (2^{40})	≈ 12 jours
50 bits	≈ 1 quadrillion (2^{50})	≈ 35 ans
60 bits	≈ 1 quintillion (2^{60})	≈ 36 000 ans
70 bits	≈ 1 sextillion (2^{70})	≈ 36 millions d'années
80 bits	≈ 1 septillion (2^{80})	≈ 36 milliards d'années

Entropie – sécurité informatique - bits

- En dessous de **40 bits**, le mot de passe est faible (facilement cassable).
- Entre **50–60 bits**, on atteint une bonne sécurité pour usage courant.
- À **70–80 bits et plus**, on est au niveau d'une **sécurité cryptographique** (impraticable à casser par force brute).

Résumé

Entropie d'une clé (en bits) cryptographique. Source markovienne d'ordre 0 (chaque bit est indépendant).

Entropie d'un bit

$H_2(\text{bit}) = \quad H_2(X) = -\sum_{i=1}^{n} P_i \log_2 P_i. \quad = -\sum_{i=1}^{2} \frac{1}{2} \log_2 \quad = -\sum_{i=1}^{2} \frac{1}{2} \cdot (-1) = 1$

Entropie d'une clé (en bits) de longueur N : $H_2(\text{clé\_bits}) = H_2(\text{bit}) * \text{longueur\_clé} = 1 * N = N$

Entropie d'un mot de passe (symboles en octets). Source markovienne d'ordre 0 (chaque symbole est indépendant).

Entropie d'un symbole d'un alphabet de N symboles (i.e. alphabet = {A, B, C, …, Z}, N = 26) :

$$H = -\sum_{i=1}^{N} p_i \log_2(p_i) \quad = -N \cdot \frac{1}{N} \cdot \log_2\left(\frac{1}{N}\right) \quad = \log_2(N)$$

Entropie d'un mot de passe de longueur M composé des symboles d'un alphabet de taille N

$H_{\text{mot\_de\_passe}}$ = longueur_de_mot_de_passe * entropie_un_symbole = M * $\log_2$(N)

sources markoviennes vs sources non-markoviennes

- Une source markovienne est une source stochastique d'information dont la suite des symboles émis suit un processus de Markov. La probabilité d'émettre un symbole à l'instant t dépend seulement d'un nombre fini d'états ou symboles précédents

Chaîne de Markov d'ordre 1

La probabilité du prochain symbole dépend seulement du symbole précédent :

$$P(X_n = x_n \mid X_{n-1}, X_{n-2}, \dots) = P(X_n = x_n \mid X_{n-1})$$

Une source sans mémoire est un cas particulier : elle correspond à une source markovienne d'ordre 0 (chaque symbole est indépendant).

Si **chaque caractère du mot de passe est choisi totalement au hasard, indépendamment des autres**, alors on modélise avec une **source markovienne d'ordre 0** (*sans mémoire*).
Si **les caractères dépendent les uns des autres** (par exemple dans des mots de passe créés par des humains, avec des régularités comme "mablonde2025"), alors il est plus réaliste d'utiliser une **source markovienne d'ordre 1 (ou plus)**.

Sources non-markoviennes

- Une **source non-markovienne** est une source d'information **avec mémoire**, mais dont les dépendances ne peuvent pas être décrites uniquement par un processus de Markov d'ordre fini.

$$P(X_n \mid X_{n-1}, X_{n-2}, \dots) \neq P(X_n \mid X_{n-1}, \dots, X_{n-m})$$

pour tout m fini.
Donc le futur peut dépendre de **tout l'historique**, pas seulement des m derniers symboles.

Exemples de sources non-markoviennes
Langues naturelles (texte écrit par des humains)

Le choix d'un mot peut dépendre du contexte lointain (pas seulement du mot précédent).
Exemple : dans "Il était une fois... une", le mot suivant est très probablement "princesse", et cette dépendance remonte à plusieurs mots en arrière.

Propriétés

Confusion

- La confusion vise à rendre la relation entre la clé et le texte chiffré aussi compliquée que possible.
- Autrement dit, si on modifie un seul bit de la clé, le texte chiffré doit changer de façon imprévisible.
- But : masquer la structure de la clé dans le chiffré.

Exemple :

- Dans AES ou DES, les substitutions non linéaires (S-Boxes) créent de la confusion.
- Si on change un seul bit de la clé, le texte chiffré devient radicalement différent.

Diffusion

- La diffusion vise à répartir l'influence d'un seul symbole du texte clair sur le plus grand nombre de symboles du texte chiffré.
- Autrement dit, un petit changement dans le texte clair (ex. une lettre) doit affecter beaucoup de bits du texte chiffré.
- But : effacer les redondances statistiques de la langue (fréquences des lettres, mots fréquents…).

Exemple :

- Dans AES, les permutations linéaires réalisent la diffusion.
- Modifier 1 bit du texte clair entraîne des changements dans de nombreux bits du texte chiffré (effet avalanche).

Complémentarité

- **Confusion → complexifie la dépendance entre clé et chiffré.**
- **Diffusion → complexifie la dépendance entre clair et chiffré.**

Entropie et fréquences des symboles

En théorie de l'information, l'**entropie de Shannon** d'une source discrète est :

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

où p_i est la **probabilité d'apparition** du symbole i.

En pratique, si on observe une séquence (texte, signal, etc.), on estime p_i par la **fréquence relative** du symbole :

$$p_i \approx \frac{\text{nombre d'occurrences du symbole } i}{\text{taille totale de la séquence}}$$

Si on observe un texte de 1000 caractères : Lettre E = 150 fois, Lettre A = 100 fois, …

On peut calculer une **pseudo-entropie empirique** où f_i = fréquence relative observée. f_E = 150/1000, f_A = 100/1000,…

$$\hat{H} = -\sum_{i} f_i \log_2(f_i)$$

Exemple d’entropie et fréquences des symboles

Supposons un alphabet $\{A, B, C\}$ avec fréquences dans un texte de 100 symboles :

- A : 50 occurrences → $p(A) = 0.5$
- B : 30 occurrences → $p(B) = 0.3$
- C : 20 occurrences → $p(C) = 0.2$

Alors :

$$H(X) = -(0.5 \log_2 0.5 + 0.3 \log_2 0.3 + 0.2 \log_2 0.2)$$

$$= -(0.5 \cdot -1 + 0.3 \cdot -1.737 + 0.2 \cdot -2.322)$$

$$= 0.5 + 0.521 + 0.464 \approx 1.485 \text{ bits}$$

$$\hat{H} = -\sum_i f_i \log_2(f_i) \quad = \quad SN \quad \Psi(S_N) = \sum_i f_i(S_N) \log \frac{1}{f_i(S_N)}$$

Entropie en bloc - exemple

L'entropie en bloc est une généralisation de l'entropie de Shannon appliquée non pas à un seul symbole, mais à des séquences (blocs) de symboles.

- Si on a une source d'alphabet A, on définit l'entropie pour un bloc de longueur n :

$$H_n = -\sum_{x_1,x_2,\ldots,x_n \in A^n} P(x_1, x_2, \ldots, x_n) \log_2 P(x_1, x_2, \ldots, x_n)$$

Où P est la probabilité de la séquence $x_1,x_2,x_3,\ldots,x_n$

- $H_1 = H(S)$ entropie par symbole (cas classique).
- $H_2 = H(S^2)$ entropie des paires de symboles.
- $H_n = H(S^n)$ entropie des séquences de longueur n.

Entropie en bloc

Alphabet binaire $A = \{0, 1\}$.

Source aléatoire uniforme et indépendante :

- $P(0) = P(1) = 0.5$
- Pour un bloc de taille 2 (S^2 → entropie H_2):
  - Probabilités des séquences :
    $(00,01,10,11) = 0.25$ chacune $= \frac{1}{4}$.
  - Donc :

$$H_2 = -4 \times 0{,}25 \times log_2(0{,}25) = -4 \times \frac{1}{4} \times log_2(\frac{1}{4}) = 2\ bits$$

→ Chaque symbole apporte 1 bit → indépendance.

$$H_n = -\sum_{x_1, x_2, \ldots, x_n \in A^n} P(x_1, x_2, \ldots, x_n) \log_2 P(x_1, x_2, \ldots, x_n)$$

Entropie en bloc

Alphabet binaire $A = \{0, 1\}$.

Source aléatoire uniforme et indépendante :

- $P(0) = P(1) = 0.5$
- Pour un bloc de taille 3 (S^3 → entropie H_3):
  - Probabilités des séquences :
    $(000,001,010,011,100,101,110,111) = 0.125$ chacune $= \frac{1}{8}$.
  - Donc :

$$H_3 = -8 \times 0{,}125 \times log_2(0{,}125) = -8 \times \frac{1}{8} \times log_2(\frac{1}{8}) = 3\,bits$$

→ Chaque symbole apporte 1 bit → indépendance.

$$H_n = -\sum_{x_1, x_2, \ldots, x_n \in A^n} P(x_1, x_2, \ldots, x_n) \log_2 P(x_1, x_2, \ldots, x_n)$$

Entropie en bloc – S^3

Alphabet $= \{A, B, \dots, Z\}$.

Source aléatoire uniforme et indépendante :

- $P(\mathrm{A}) = P(\mathrm{B}) = \cdots = P(\mathrm{Z}) = 1/26$
- Pour un bloc de taille 3 (S^3 → entropie H_3):
  - Probabilités des séquences – toutes équiprobables: $(AAA, AAB, AAC, \dots, ZZZ), nombre\ de\ séquences.$ N=26^3=17576,
  - Probabilité d'une séquence $S^3_i = 1/17576$
  - Donc :

$$H = \log_2(N) = \log_2(26^3)$$

$$H = 3 \cdot \log_2(26)$$

H=3×4,7≈14,1 bits

Entropie en bloc langue française – S^3

Langue française

- les lettres **ne sont pas équiprobables** (E, A, I, S... plus fréquentes) ;
- il y a **fortes corrélations** entre lettres (digrammes/trigrammes courants : « que », « ent », « ion », etc.) ;

  → conséquence : **l'entropie empirique des trigrammes est bien plus faible** que 14,1 bits.

On distingue :

- H_1 : entropie par lettre (si on ignore dépendances), ~ $-\sum p(a)\log_2 p(a)$.
- H_3 : entropie des trigrammes (bloc de taille 3) :

$$H_3 = -\sum_{x \in A^3} P(x)\log_2 P(x).$$

Les langues naturelles ont un taux d'entropie par lettre typiquement dans l'ordre de ≈ 1,0 – 1,5 bits/lettre ; le français est de l'ordre de h = 1,0–1,4.

Pour des trigrammes : $H_3 \approx 3\times h \approx 3{,}0$ *à* $4{,}5$ *bits*

Comparaison numérique résumé

Équiprobable : $H_3 \approx 14{,}1$ bits.

Français réel (ordre de grandeur) : $H_3 \approx 3$ à 4,5 bits (typiquement ≈ 3.3 bits, $h \approx 1.1$ bits/lettre).

Conclusion : la langue naturelle a une énorme redondance : l'entropie réelle d'un trigramme est beaucoup plus faible que si toutes les combinaisons étaient possibles et équiprobables.

Force brute et John the Ripper

- 123456
- password
- qwerty
- azerty
- welcome
- iloveyou
- admin
- dragon
- football
- monkey
- shadow
- princess
- 123456789
- letmein

John the Ripper va tester ces mots tels quels.
Ensuite, il peut appliquer des règles de transformation (par exemple ajouter 123, mettre en majuscule la première lettre, remplacer a par @, etc.).
Exemple : password → Password1 → P@ssword!

Lien avec l'entropie

Si un mot de passe figure dans un dictionnaire, son entropie réelle est très faible : ce n'est pas un mot de passe aléatoire.
L'attaque par dictionnaire exploite précisément le fait que les humains ne choisissent pas leurs mots de passe comme une source markovienne d'ordre 0 uniforme, mais plutôt en fonction de mots communs → faible entropie effective.

Analyse fréquentielle

Lettre	Anglais	Français[3]
a	8.167%	7.636%
b	1.492%	0.901%
c	2.782%	3.260%
d	4.253%	3.669%
e	12.702%	14.715%
f	2.228%	1.066%
g	2.015%	0.866%
h	6.094%	0.737%
i	6.966%	7.529%
j	0.153%	0.613%
k	0.772%	0.074%
l	4.025%	5.456%
m	2.406%	2.968%
n	6.749%	7.095%

Lettre	Anglais	Français[3]
o	7.507%	5.796%
p	1.929%	2.521%
q	0.095%	1.362%
r	5.987%	6.693%
s	6.327%	7.948%
t	9.056%	7.244%
u	2.758%	6.311%
v	0.978%	1.838%
w	2.360%	0.049%
x	0.150%	0.427%
y	1.974%	0.128%
z	0.074%	0.326%

Mots les plus utilisées en français

1.	de	16.	qui	31.	lui
2.	la	17.	se	32.	leur
3.	le	18.	ce	33.	mais
4.	est	19.	dans	34.	plus
5.	que	20.	ne	35.	comme
6.	et	21.	pas	36.	ou
7.	à	22.	pour	37.	si
8.	les	23.	sur	38.	bien
9.	des	24.	avec	39.	mon
10.	en	25.	par	40.	ton
11.	un	26.	au	41.	sa
12.	une	27.	je	42.	ses
13.	du	28.	tu	43.	qui
14.	il	29.	nous	44.	où
15.	elle	30.	vous	45.	quand

Langue française

Longueur moyenne mot

- Dans des textes d'usage courant, la longueur moyenne d'un mot est d'environ **4,8 lettres**.
- Dans un dictionnaire (donc tous les mots "théoriquement possibles", y compris rares ou longs), la longueur moyenne monte autour de **10,09 lettres**

Nombre mots de la langue

- Un dictionnaire "courant" contient environ **60 000** mots.
- Le *Trésor de la langue française informatisé* (TLFi) recense environ **100 000 mots** (avec définitions, usages, etc.).
- Si on inclut les mots techniques, scientifiques, dialectes, etc., le nombre pourrait être **200 000 mots** ou plus

Cryptanalyse fréquentielle d'un chiffrement par substitution mono-alphabétique

Étape 1 : Texte chiffré

- Le message intercepté:

  UIJT JT B NFTTBHF

Étape 2 : Analyse des fréquences

- On compte la fréquence des lettres dans le texte chiffré.
  - J apparaît 3 fois.
  - T apparaît 3 fois.
  - D'autres lettres apparaissent moins souvent.
- En français, les lettres les plus fréquentes sont en général :
  - E, A, I, S, N, R, T, O, L, U.
- Ici, J et T sont très fréquents → possible correspondance avec E ou T ou S.

Étape 3 : Essai de substitution

- On essai si ce chiffrement est peut-être un simple **décalage de César**.
  Test un décalage de −1 (retirer 1 à chaque lettre).
- UIJT JT B NFTTBHF
- ↓ décalage -1
- THIS IS A MESSAGE

Cryptanalyses méthode classique transposition

Chiffrons ce texte avec un « bâton » de taille 6 et commençons au 3e « trou de ceinture » (3e caractère)

Texte original ➡ I L E T A I T U N E F O I S L H I S T O I R E D U N P E T I T C H A P E R O N R O U G E

Texte chiffré ➡ e n l i p h n i t e h r e a r l a f i e t p o i o s d i e u t i t u t r g u s o n c o e

e	n	l	i	p	h	n	i
t	e	h	r	e	a	r	l
a	f	i	e	t	p	o	
i	o	s	d	i	e	u	
t	i	t	u	t	r	g	
u	s	o	n	c	o	e	

Texte chiffré	e	n	l	i	p	h	n	i	t	e	h	r	e	a	r	l	a	f	i	e	t	p
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
	1	9	17	24	31	38	2	10	18	25	32	39	3	11	19	26	33	40	4	12	20	27
Cryptanalyses bâton	e	t	a	i	t	u	n	e	f	o	i	s	l	h	i	s	t	o	i	r	e	d
Cryptanalyses trou	i	l	e	t	a	i	t	u	n	e	f	o	i	s	l	h	i	s	t	o	i	r

Texte chiffré	o	i	o	s	d	i	e	u	t	i	t	u	t	r	g	u	s	o	n	c	o	e
	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
	34	41	5	13	21	28	35	42	6	14	22	29	36	43	7	15	23	30	37	44	8	16
Cryptanalyses bâton	u	n	p	e	t	i	t	c	h	a	p	e	r	o	n	r	o	u	g	e	i	L
Cryptanalyses trou	e	d	u	n	p	e	t	i	t	c	h	a	p	e	r	o	n	r	o	u	g	e