CM1

Histoire & fondamentaux de la cryptographie

Module Cryptographie

Aymeric DELIENCOURT
aymeric.deliencourt@yncrea.fr

- Diplômé de l'**ISEN Méditérranée**, option Cybersécurité (2021)
- Architecte Cybersécurité @**Naval Group**
- CTF Maker @**Barbhack** & @**LeHack**
- Open Source enthusiast
- Diver 🤿

Kahoot

https://rahoot.isen-cyber.ovh

🤔 Pourquoi la cryptographie ?

🛡️ La cryptographie, c’est quoi ?

L’art et la science de **protéger l’information**

Trois objectifs fondamentaux :

- 🔒 **Confidentialité** : seul le destinataire peut lire le message
- ✅ **Intégrité** : le message n’a pas été modifié en transit
- 🪪 **Authenticité** : le message vient bien de l’expéditeur annoncé

🌍 La crypto est partout

Vous l'utilisez **tous les jours** sans le savoir

- 🌐 **HTTPS** : chaque site web avec un cadenas
- 💬 **WhatsApp / Signal** : chiffrement de bout en bout
- 💳 **Carte bancaire** : puce cryptographique
- 🛂 **Passeport biométrique** : signature numérique
- 📶 **WiFi WPA** : chiffrement de votre réseau
- 🔗 **VPN** : tunnel chiffré

⚠️ Sans cryptographie...

- 👁️ Vos mots de passe seraient lisibles par votre FAI
- 🛒 Vos achats en ligne seraient interceptables
- 📨 Vos messages privés seraient publics
- 💳 Votre carte bancaire serait copiable à distance

📊 HTTPS : avant / après

📈 Évolution du pourcentage de pages chargées en HTTPS (Chrome) :

Année	% HTTPS
2014	~30%
2016	~50%
2018	~70%
2020	~85%
2025	**~95%**

🔒 Grâce à **Let’s Encrypt** (certificats gratuits, 2015) et aux navigateurs qui affichent "Non sécurisé" pour HTTP.

🔐 La CIA… de la crypto

Non, pas l'agence américaine ! Le triptyque **CIA** en sécurité informatique :

C : Comment s'assurer que seul le destinataire peut **lire** le message ?

🔒 → **Confidentialité** (*Confidentiality*)

I : Comment vérifier que le message n'a pas été **modifié** ?

✅ → **Intégrité** (*Integrity*)

A : Comment prouver **qui** a envoyé le message ?

🪪 → **Authenticité** (*Authentication*)

Et un 4ᵉ bonus : comment empêcher l'expéditeur de **nier** avoir envoyé le message ?

🚫 → **Non-répudiation**

🔐 Exemples : quelle propriété est violée ?

✏️ Un virement bancaire de 100€ est modifié en 10 000€ en transit

→ ✅ **Intégrité** violée

👁️ Quelqu'un lit vos messages WhatsApp à votre insu

→ 🔒 **Confidentialité** violée

📧 Vous envoyez un mail insultant et prétendez ne jamais l'avoir fait

→ 🚫 **Non-répudiation** nécessaire

🎭 Quelqu'un envoie un email en se faisant passer pour votre patron

→ 🪪 **Authenticité** violée

📜 Histoire de la cryptographie

3000 ans de secrets

⚔️ ~700 av. J.-C.

La scytale spartiate

Premier système de chiffrement connu · 🔒 Confidentialité

- Un bâton de bois + une bande de cuir enroulée
- Le message est écrit **horizontalement** sur le cuir enroulé
- Déroulé, le cuir ne montre que des lettres mélangées
- Pour lire : enrouler sur un bâton du **même diamètre**

Chiffrement par **transposition** : les lettres sont réarrangées, pas remplacées.

📖 Scytale : anecdote

🏛️ Les généraux spartiates l’utilisaient pour communiquer avec Athènes pendant les campagnes militaires.

🔑 Chaque général et le gouvernement possédaient un bâton de **même diamètre** : la "clé" du système.

🏃 Un messager intercepté ne pouvait pas lire le message sans le bon bâton.

🏺 Autres techniques antiques

La cryptographie ancienne ne se limitait pas à la scytale :

- 🇪🇬 **Hiéroglyphes modifiés** (~1900 av. J.-C.) : les scribes égyptiens remplaçaient certains hiéroglyphes par des symboles inhabituels dans les inscriptions funéraires
- 🇬🇷 **Polybe** (~200 av. J.-C.) : carré de 5×5 pour coder chaque lettre en un couple de chiffres

💡 Le besoin de **communiquer en secret** est universel et traverse toutes les civilisations.

📖 Le carré de Polybe (~200 av. J.-C.)

🇬🇷 **Polybe** invente un système pour transmettre des messages à distance avec des torches.

Chaque lettre est codée par sa **ligne** et sa **colonne** dans un carré 5×5 :

```
    1  2  3  4  5
1 | A  B  C  D  E
2 | F  G  H  IJ K
3 | L  M  N  O  P
4 | Q  R  S  T  U
5 | V  W  X  Y  Z
```

📝 **HELLO** → 23 15 31 31 34

📝 **Decode** → 24 43 15 33

💡 Ce principe est réutilisé dans de nombreux chiffrements ultérieurs, notamment le chiffre **ADFGVX** de la Première Guerre mondiale.

🏛️ ~50 av. J.-C.

Le chiffre de César

Substitution monoalphabétique par décalage fixe · 🔒
Confidentialité violée (25 clés seulement)

Chaque lettre est remplacée par la lettre située **k positions** plus loin dans l'alphabet.

```
Clair  : A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Chiffré: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

Avec la clé **k = 3** :

`HELLO` → `KHOOR`

📖 César : anecdotes

🗡️ Jules César utilisait un décalage de **3** pour ses communications militaires.

👑 Son neveu Auguste César utilisait un décalage de **1** : encore moins sécurisé !

🔓 Avec seulement **25 clés possibles**, ce chiffrement est trivial à casser aujourd'hui.

🔀 Le chiffrement par substitution générale

On associe maintenant **à chaque lettre une autre lettre**, sans ordre fixe ni règle générale :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
F	Q	B	M	X	I	T	E	P	A	L	W	H	S	D	O	Z	K	V	G	R	C	N	Y	J	U

- **ETRE OU NE PAS ETRE TELLE EST LA QUESTION**
- E → X, T → G, R → K ...
- **XGKX DR SX OFV XGKX GXWWX XVG WF ZRXVGPDS**

Pour le décrypter, on fait les substitutions **inverses**.

Avantage : espace des clés **gigantesque**.

→ Quelle est le nombre de clé possible ?

📐 Espace de clés

Une clé correspond à une **bijection** :

$$C : \{A, B, \ldots, Z\} \longrightarrow \{A, B, \ldots, Z\}$$

Il y a **26!** choix possibles :

- Pour crypter la lettre A, il y a **26** choix
- Pour B, il reste **25** choix
- Pour C, il reste **24** choix…
- Pour Z, **une seule** possibilité
- $26 \times 25 \times 24 \times \cdots \times 2 \times 1 = 26!$ choix de clés

📐 Espace de clés : 26!

$$26! = 403\,291\,461\,126\,605\,635\,584\,000\,000 \simeq 4 \times 10^{26} \text{ clés}$$

🏖️ C'est **plus que le nombre de grains de sable sur Terre** !

⏱️ Si un ordinateur pouvait tester **1 000 000** de clés par seconde, il lui faudrait alors plus de **12 millions d'années** pour tout énumérer.

→ Mais il y a une faiblesse...

⚠️ Substitution : espace immense, mais...

Faiblesse : une même lettre est toujours cryptée de la même façon.

$$\mathrm{E} \longmapsto \mathrm{X}$$

Les lettres n'apparaissent **pas avec la même fréquence** dans une langue.

Lettres les plus fréquentes en français :

E	S	A	I	N	T	R	U	L
14.6%	8.0%	7.5%	7.1%	6.8%	6.8%	6.4%	6.1%	5.6%

📊 La lettre la plus fréquente dans le chiffré = probablement l'image de **E**.

On reconstitue le message par **approche successive** (texte à trous).

💥 L'analyse fréquentielle casse la substitution, **quel que soit** l'espace de clés.

🧮 IXe siècle

Al-Kindi

Abu Yusuf Al-Kindi : mathématicien arabe (801-873) · 🔓 Casse la confidentialité

Premier à décrire la technique de l'**analyse fréquentielle** dans son *Manuscrit sur le déchiffrement des messages cryptographiques*.

📊 **Principe** : dans une langue donnée, certaines lettres apparaissent plus souvent que d'autres.

En français : **E** (~15%), **A** (~8%), **I** (~7%), **S** (~7%)

💡 Al-Kindi : anecdote

🌍 Un mathématicien arabe casse le chiffre de César… **1000 ans** avant que l'Europe ne le comprenne vraiment.

💥 L'analyse fréquentielle rend **tous** les chiffrements par substitution monoalphabétique vulnérables.

➡️ Pour résister, il faut un chiffrement **polyalphabétique**.

🔤 XVIe siècle

Le chiffre de Vigenère

Substitution polyalphabétique avec clé répétée · 🔒 Renforce la confidentialité

Chaque lettre est décalée d'une valeur **différente**, déterminée par une clé.

```
Message : A T T A Q U E
Clé     : C L E C L E C
Décalage: 2 11 4 2 11 4 2
Chiffré : C E X C B Y G
```

La même lettre `T` est chiffrée différemment selon sa position : `E` puis `X` ! 🤯

🏆 Vigenère : "le chiffre indéchiffrable"

Surnommé ainsi pendant **300 ans** !

L'analyse fréquentielle simple ne fonctionne plus car une même lettre n'est pas toujours chiffrée de la même façon.

🧠 **Charles Babbage** (l'inventeur de l'ordinateur mécanique) le casse en secret en **1854**... mais ne publie pas.

📝 C'est **Friedrich Kasiski** qui publie la méthode en **1863**.

🔑 Pourquoi Vigenère résiste à l'analyse fréquentielle ?

Avec César (monoalphabétique), **E** est toujours chiffré par la même lettre :

```
E → H (clé 3), toujours H
```

Avec Vigenère (polyalphabétique), **E** est chiffré différemment selon sa position :

```
E en position 1 → G (décalage 2)
E en position 2 → P (décalage 11)
E en position 3 → I (décalage 4)
```

📊 Les fréquences sont **lissées** : chaque lettre chiffrée apparaît à peu près autant.

💡 Pour casser Vigenère, il faut d'abord trouver la **longueur de la clé**, puis traiter chaque position comme un César indépendant.

🔤 Comment fonctionne Vigenère ?

Pour chaque lettre, on applique un **décalage différent** selon la position dans la clé :

$$C_i = (M_i + K_{i \bmod \ell}) \mod 26$$

où ℓ est la longueur de la clé.

Exemple avec clé = "CLE" (longueur 3) :

```
Position:  0  1  2  3  4  5  6  7
Message :  B  O  N  J  O  U  R  !
Clé     :  C  L  E  C  L  E  C
Décalage:  2  11 4  2  11 4  2
Chiffré :  D  Z  R  L  Z  Y  T  !
```

🔢 Vigenère : espace des clés et attaque

Pour une clé de longueur k, il y a 26^k choix possibles.

Exemple : $k = 4, \quad 26^4 = 456\,976$ clés

Même faiblesse que le chiffrement par substitution :

- **ALPH ABET** : les deux **A** sont à la même position → cryptés par la **même lettre**
- **ALPH ABET** ⟶ **DMUJ DCJV**

Attaque :

- On découpe le message en plusieurs listes : les 1ères lettres de chaque bloc, les 2èmes lettres…
- Chaque regroupement est chiffré par le **même décalage** (= un César)
- Attaque statistique sur chacun de ces regroupements

⚠️ Cette attaque n'est possible que si la taille des blocs est **petite devant la longueur du texte**.

👸 1587

Marie Stuart

Marie Stuart, reine d’Écosse, prisonnière d’Élisabeth Ire.

📨 Elle utilise un **chiffre de substitution** avec des symboles pour comploter l’assassinat d’Élisabeth.

🕵️ **Thomas Phelippes**, cryptanalyste, déchiffre toute la correspondance.

⚖️ Marie est **condamnée à mort et exécutée** en 1587.

👸 Marie Stuart : le piège

🕵️ Phelippes ne se contente pas de déchiffrer : il **ajoute un post-scriptum forgé** au message de Marie pour lui faire demander les noms des conspirateurs.

📝 Marie répond et donne les noms.

💀 Ce post-scriptum falsifié scelle le sort de Marie Stuart et de ses alliés.

> La cryptanalyse comme outil de **manipulation active** : on ne se contente pas de lire, on modifie.

> 🔒 **Confidentialité** violée (chiffre cassé) + ✅ **Intégrité** violée (post-scriptum forgé par Phelippes)

📨 1917

Le télégramme Zimmermann

🇩🇪 L'Allemagne envoie un message **chiffré** au Mexique :

> "Si les USA entrent en guerre, proposez au Mexique une alliance. En échange : le Texas, le Nouveau-Mexique et l'Arizona."

🇬🇧 Les Britanniques (Room 40) le **déchiffrent**.

📰 Publié dans la presse → indignation américaine → 🇺🇸 les USA entrent en guerre.

> 🔒 **Confidentialité** violée → ce déchiffrement a probablement **raccourci la guerre de 2 ans**.

📨 Zimmermann : les leçons

🔑 Le télégramme était chiffré avec le **code 0075** : un code de substitution allemand.

📡 Transmis par **câble transatlantique** via les ambassades… un câble que les Britanniques avaient **mis sur écoute** dès 1914.

🧠 Le déchiffrement a pris **plusieurs semaines** par l'équipe de la Room 40 de l'Amirauté britannique.

🎭 Le plus difficile n'était pas de déchiffrer, mais de **publier** sans révéler que les Britanniques lisaient les communications diplomatiques allemandes.

🌍 1914-1918 : la cryptographie dans les tranchées

La Première Guerre mondiale a été un tournant pour la cryptographie :

- 📡 Le **télégraphe** et la **radio** remplacent les messagers → les messages traversent l'espace ouvert
- 👂 N'importe qui peut intercepter une transmission radio
- 🔑 Besoin urgent de **chiffrements plus solides**

🇫🇷 La France utilise le **chiffre ADFGVX** (1918) : combinaison de substitution et transposition, l'un des plus complexes de la guerre.

🇩🇪 L'Allemagne commet l'erreur de **réutiliser des clés** et de suivre des formats prévisibles.

⚙️ 1939-1945

Enigma

Machine électromécanique utilisée par l'Allemagne nazie · 🔒 Confidentialité violée malgré la complexité

Composée de :

- 🔄 **3 à 5 rotors** : chaque rotor effectue une substitution différente
- 🔌 **Tableau de connexion** : échange des paires de lettres
- 🪞 **Réflecteur** : renvoie le signal à travers les rotors

Résultat : une substitution **polyalphabétique dynamique** avec un espace de clés astronomique.

😤 Enigma : les Allemands confiants

Les Allemands pensaient Enigma **incassable**.

🔢 Espace de clés : environ **158 962 555 217 826 360 000** combinaisons possibles.

💀 **Erreur fatale** : la confiance absolue les rend négligents dans les procédures.

⚠️ Enigma : les failles de procédure

Enigma aurait pu être bien plus difficile à casser. Les Allemands ont commis des erreurs **humaines** :

- 📋 Les messages météo commençaient toujours par **"WETTER"** (météo en allemand)
- 💬 Les opérateurs commençaient souvent par **"HEIL HITLER"** ou leur nom
- 🔤 Une lettre ne pouvait **jamais** se chiffrer en elle-même (faille du réflecteur)
- 🔑 Certains opérateurs choisissaient des clés prévisibles : "AAA", "ABC", les initiales de leur petite amie…

> La crypto la plus robuste est inutile si les **procédures** sont mauvaises. 🤦

🇵🇱 Les Polonais : premiers à attaquer Enigma

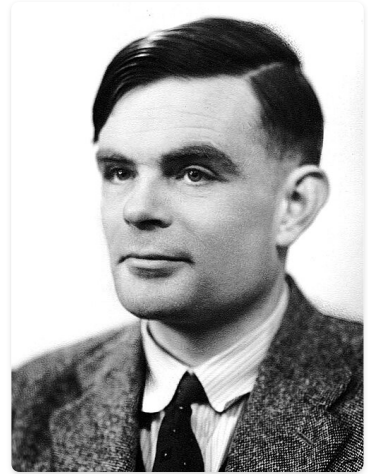
🧠 **Marian Rejewski**, mathématicien polonais, commence à casser Enigma dès **1932**.

⚙️ Avec Jerzy Rozycki et Henryk Zygalski, ils construisent la **"Bomba kryptologiczna"** (bombe cryptologique) : une machine pour automatiser le test des positions de rotors.

🤝 En 1939, la Pologne partage ses travaux avec la France et la Grande-Bretagne.

🇬🇧 Bletchley Park

Alan Turing

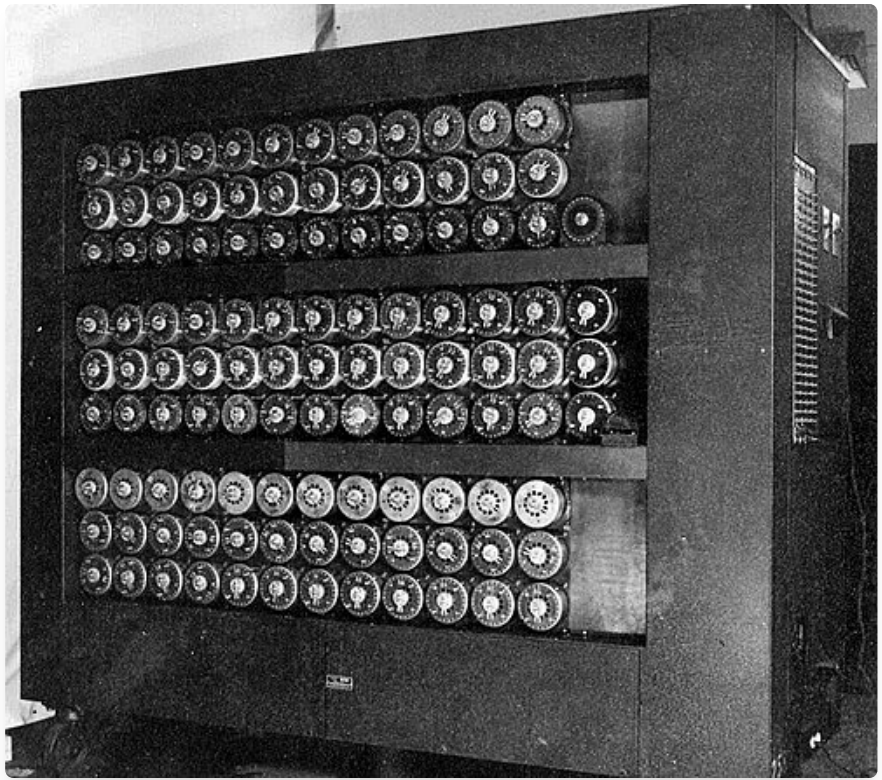


👩‍💻 **Alan Turing** et son équipe reprennent le travail des Polonais.

⚙️ Turing conçoit la **Bombe** (inspirée de la *Bomba* polonaise, mais avec une approche différente) : une machine électromécanique testant des milliers de configurations par jour.

🎯 Exploite les **failles de procédure** : messages météo, salutations stéréotypées, lettres qui ne se chiffrent jamais en elles-mêmes.

🏠 Bletchley Park, Buckinghamshire, Angleterre



⚙️ La Bombe de Turing, Bletchley Park

"Novembre 1940 : Churchill aurait décidé de ne pas évacuer Coventry pour ne pas révéler qu’Enigma était cassé. 568 morts. 😔"

- Anecdote historique (débattue)

🏅 Enigma : impact

📉 Le déchiffrement d'Enigma aurait **raccourci la guerre de 2 à 4 ans** selon les historiens.

💡 Des milliers de vies sauvées grâce aux mathématiques et à la persévérance.

🤫 Turing est considéré comme le **père de l'informatique** mais son travail à Bletchley Park est resté secret pendant 30 ans.

🖥️ Démo : simulateur Enigma

🌐 Démo en ligne : simulateur Enigma

- 🔄 Réglage des rotors et de leur position initiale
- 🔌 Configuration du tableau de connexion
- ⌨️ Chiffrement d’un message lettre par lettre
- 👀 Observer comment les rotors tournent à chaque frappe
- 🔍 Constater qu’une lettre ne se chiffre **jamais** en elle-même

🇺🇸 1942-1945

Les code talkers

🪶 Les **Navajo code talkers** : des soldats amérindiens utilisent leur langue maternelle comme code militaire dans le Pacifique.

💡 Le navajo est une langue :

- 🌍 Aucune écriture publiée à l'époque
- 🧠 Extrêmement complexe (tons, grammaire)
- 🔒 Aucun locuteur parmi les forces de l'Axe

✅ Le code navajo n'a **jamais été cassé** pendant la guerre.

🎬 Film : *Windtalkers* (2002) avec Nicolas Cage.

🪶 Code talkers : comment ça marchait ?

Les Navajo utilisaient un **double système** :

1️⃣ **Alphabet codé** : chaque lettre = un mot navajo (ex: A = "Wol-la-chee" = fourmi 🐜)

2️⃣ **Vocabulaire militaire** : des mots dédiés pour les termes tactiques

Terme militaire	Mot navajo	Traduction littérale
Avion de chasse	Da-he-tih-hi	Colibri 🐦
Sous-marin	Besh-lo	Poisson de fer 🐟
Bombe	A-ye-shi	Œuf 🥚
Général	Bi-keh-he	Celui qui commande

📊 Frise chronologique : résumé

- **~700 av. J.-C.** — Scytale spartiate
- **~200 av. J.-C.** — Carré de Polybe
- **~50 av. J.-C.** — Chiffre de César
- **IXe siècle** — Al-Kindi, analyse fréquentielle
- **XVIe siècle** — Chiffre de Vigenère
- **1587** — Marie Stuart
- **1917** — Télégramme Zimmermann
- **1939-1945** — Enigma, Bletchley Park
- **1942-1945** — Navajo code talkers

📊 Bilan : propriétés violées dans l'histoire

- 🏛️ **Jules César** : 🔒 Confidentialité violée — seulement 25 clés → cassable à la main
- 👸 **Marie Stuart** : 🔒 Confidentialité violée (chiffre cassé) + ✅ Intégrité violée (post-scriptum forgé) → ☠️ exécution
- 🇩🇪 **Zimmermann** : 🔒 Confidentialité violée — code réutilisé, câble sous écoute → 🇺🇸 entrée en guerre des USA
- ⚙️ **Enigma** : 🔒 Confidentialité violée — failles de procédure → cassé par Turing

💡 **Point commun** : une confiance excessive dans la sécurité du système. Confidentialité, intégrité et authenticité sont **indissociables**.

📏 Le principe de Kerckhoffs

👤 Auguste Kerckhoffs (1883)

Jean Guillaume Hubert Victor François Alexandre Auguste Kerckhoffs von Nieuwenhof (oui, tout ça !)

🇳🇱 Né aux Pays-Bas, professeur de langues à Paris.

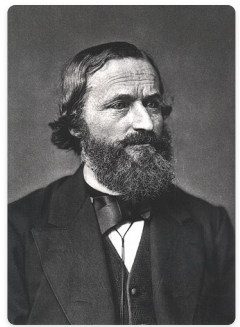
📜 Publie **6 principes** pour la conception de systèmes cryptographiques dans le *Journal des sciences militaires* (1883).

🏆 Son **2e principe** est devenu la règle d'or de la cryptographie moderne.

⚠️ Ne pas confondre !

🔐 **Auguste Kerckhoffs**

🇳🇱 Néerlandais (1835-1903)



⚡ **Gustav Kirchhoff**

🇩🇪 Allemand (1824-1887)

À votre avis, qu'ont-ils fait chacun ?

	🔐 Kerck**hoffs**	⚡ Kirch**hoff**
Connu pour	6 principes de crypto (1883)	Lois des circuits (1845)

⚡ Petit rappel : les lois de Kirchhoff

Vous les avez vues en cours d'électronique ! Lesquelles ?

Loi des nœuds :

$$\sum I_{\text{entrants}} = \sum I_{\text{sortants}}$$

Loi des mailles :

$$\sum U = 0 \quad \text{dans une maille fermée}$$

Ici, on s'intéresse à l'autre : **Kerckhoffs** et ses principes de cryptographie ! 🔐

📋 Les 6 principes de Kerckhoffs

1. Le système doit être **matériellement**, sinon mathématiquement, indéchiffrable
2. ⭐ **Il faut qu'il n'exige pas le secret**, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi
3. La clé doit pouvoir en être communiquée et retenue sans le secours de notes écrites
4. Il faut qu'il soit applicable à la correspondance télégraphique
5. Il faut qu'il soit portatif
6. Il faut que le système soit facile à utiliser

"La sécurité d’un système ne doit pas dépendre du secret de l’algorithme, mais **uniquement du secret de la clé.**"

- Auguste Kerckhoffs, 1883 · Principe n°2

🙈 Security through obscurity

L'opposé de Kerckhoffs : cacher l'algorithme pour assurer la sécurité.

Exemples d'échecs :

- 📀 **CSS** (Content Scramble System) pour les DVD : algorithme secret → cassé en 1999
- 📱 **A5/1** pour le GSM : algorithme secret → cassé quand il a fuité
- 📡 **RC4** dans WEP : implémentation secrète → catastrophe de sécurité

🙈 Autre exemple : le chiffrement ROT13

ROT13 : un "chiffrement" César avec la clé **k = 13**.

```
A B C D E F G H I J K L M → N O P Q R S T U V W X Y Z
```

💡 Particularité : appliquer ROT13 **deux fois** redonne le texte original (car 13 + 13 = 26).

⚠️ ROT13 a été utilisé **sérieusement** dans certains logiciels pour "protéger" des données :

- 📧 Netscape : stockage de mots de passe email
- 🔧 Certains fichiers de configuration Unix

❌ Ce n'est **pas de la cryptographie** : il n'y a pas de clé secrète !

🙈 L’algorithme de chiffrement de César dans Microsoft Office 97

📝 Microsoft Office 97 utilisait un algorithme **propriétaire** pour protéger les documents par mot de passe.

💀 En 2005, des chercheurs montrent que l’algorithme était si faible qu’il pouvait être cassé en **quelques millisecondes**.

🔑 Le mot de passe n’était pas utilisé comme clé, mais comme **vérification** : le document restait accessible.

📊 Des sites web proposaient de "déverrouiller" n’importe quel fichier Office 97 **instantanément**.

> **Sécurité par l’obscurité** : tant que personne ne regarde, ça "fonctionne". Dès qu’un chercheur regarde… 💥

✅ Kerckhoffs en pratique

Aujourd'hui, **tous les algorithmes modernes sont publics** :

- 🔐 **AES** : publié, analysé par des milliers de chercheurs
- 🔑 **RSA** : publié en 1977, toujours utilisé
- #️⃣ **SHA-256** : spécification publique, standard NIST

La sécurité repose **uniquement** sur la clé secrète.

Un algorithme public est **plus sûr** qu'un algorithme secret : il est scruté par la communauté. 🔍

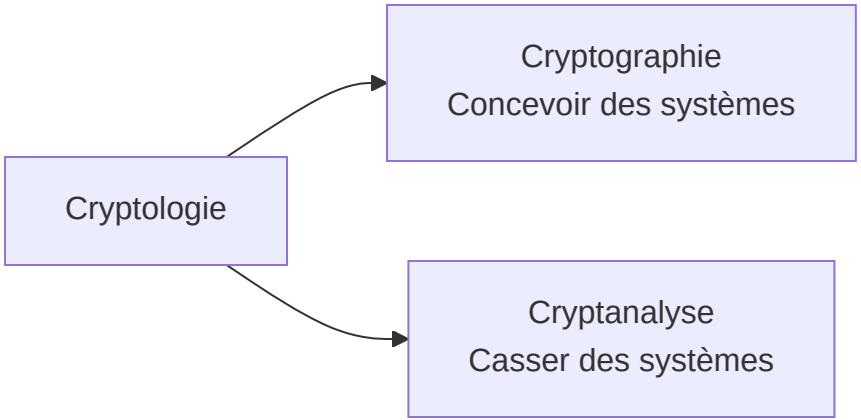
📖 Vocabulaire et notions de base

🗣️ Attention au vocabulaire !

Terme	Existe ?	Signification
🔒 **Chiffrer**	✅ Oui	Transformer un message avec une clé
🔓 **Déchiffrer**	✅ Oui	Retrouver le clair **avec** la clé
🔍 **Décrypter**	✅ Oui	Retrouver le clair **sans** la clé (= casser)
❌ **Crypter**	❌ **Non**	Ce mot **n'existe pas** !
❌ **Encrypter**	❌ **Non**	Anglicisme (*to encrypt*)
❌ **Chiffrage**	⚠️ Autre sens	= évaluer un **coût** (rien à voir !)

> "On **chiffre** avec une clé, on **déchiffre** avec la clé, et on **décrypte** sans la clé."

🔬 Cryptologie = Cryptographie + Cryptanalyse



- 🔒 **Cryptographe** : celui qui construit les systèmes
- 🔓 **Cryptanalyste** : celui qui essaie de les casser

Les deux travaillent ensemble pour améliorer la sécurité ! 🤝

🖼️ Stéganographie ≠ Cryptographie

	🔒 Cryptographie	🖼️ Stéganographie
Principe	Rendre le message **illisible**	**Cacher** l'existence du message
Le message est visible ?	✅ Oui (mais incompréhensible)	❌ Non (invisible)
Exemple ancien	Chiffre de César	Message tatoué sur un crâne rasé (Hérodote)
Exemple moderne	AES, RSA	Message caché dans les pixels d'une image

💡 On peut combiner les deux : **cacher** un message **chiffré** dans une image !

🧮 Bits et octets : rappel

- 🔢 **Bit** (*binary digit*) : la plus petite unité d'information (0 ou 1)
- 📦 **Octet** (*byte*) : 8 bits (ex: `01001101` = un caractère)
- 🔑 **Taille de clé** : nombre de bits de la clé

Exemples concrets :

Donnée	Taille
Un caractère ASCII	8 bits (1 octet)
Un mot de passe "abc123"	48 bits (6 octets)
Clé AES standard	128 bits (16 octets)

📖 Les termes essentiels

- 📝 **Texte clair** (*plaintext*) : le message original, lisible
- 📦 **Texte chiffré** (*ciphertext*) : le message transformé, illisible sans la clé
- 🔑 **Clé** (*key*) : le paramètre secret qui contrôle le chiffrement
- ⚙️ **Algorithme** (*cipher*) : la méthode de chiffrement (César, AES, RSA…)
- 🔒 **Chiffrement** (*encryption*) : transformer le clair en chiffré
- 🔓 **Déchiffrement** (*decryption*) : retrouver le clair à partir du chiffré **avec la clé**
- 🔍 **Décryptage** (*breaking*) : retrouver le clair **sans** la clé

🔄 Le processus de chiffrement

- ⚙️ L'**algorithme** est la recette (publique)
- 🔑 La **clé** est l'ingrédient secret
- 📦 Le **texte chiffré** est le résultat

Sans la clé, le texte chiffré est **incompréhensible**. 🤷

🔀 Deux grandes familles

🔄 Chiffrement symétrique

- **Une seule clé** pour chiffrer et déchiffrer
- 🔑 Alice et Bob partagent le même secret
- ⚡ Rapide
- 📦 Exemples : César, AES, DES

Problème :

Comment partager la clé ? 🤔

🔑 Chiffrement asymétrique

- **Deux clés** : une publique, une privée
- 🔓 Tout le monde peut chiffrer
- 🔐 Seul le propriétaire peut déchiffrer
- 📦 Exemples : RSA, ECC

Avantage :

Pas besoin de secret partagé ! ✅

🔀 Substitution vs Transposition

Deux techniques fondamentales pour transformer un message :

	🔤 Substitution	🔀 Transposition
Principe	Remplacer chaque lettre par une autre	Réarranger l'ordre des lettres
Exemple	A→D, B→E, C→F (César)	HELLO → LOLHE (mélange)
Les lettres	Changent	Restent les mêmes
L'ordre	Reste le même	Change
Exemple historique	Chiffre de César	Scytale spartiate

💡 Les algorithmes modernes (AES) combinent **les deux** !

🧮 XOR : l'opération magique de la crypto

L'opération **XOR** (OU exclusif, noté ⊕) est au cœur de la cryptographie moderne :

A	B	A ⊕ B
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

✨ Que se passe-t-il si on applique XOR **deux fois** avec la même valeur ?

$$M \oplus K \oplus K = M$$

🧮 XOR : exemple concret

Chiffrons "Hi" (en binaire) avec la clé `0xA5` :

```
Message  :  H        i
ASCII    :  01001000  01101001
Clé (×2) :  10100101  10100101
          ⊕ XOR     ⊕ XOR
Chiffré  :  11101101  11001100
```

Déchiffrement (même opération avec la même clé) :

```
Chiffré  :  11101101  11001100
Clé (×2) :  10100101  10100101
          ⊕ XOR     ⊕ XOR
Message  :  01001000  01101001 = "Hi" ✅
```

Notions fondamentales

🔑 La clé

Définition : paramètre secret qui contrôle le fonctionnement d'un algorithme cryptographique.

- Même algorithme + même message + **clé différente** → chiffré différent
- L'algorithme est public, la clé est secrète (Kerckhoffs)
- La clé doit être **suffisamment longue** pour résister aux attaques

🔢 L'espace de clés

Nombre total de clés possibles pour un algorithme donné.

Algorithme	Taille de clé	Espace de clés
César	5 bits	26
DES	56 bits	72 057 594 037 927 936
AES-128	128 bits	340 282 366 920 938 463 463 374 607 431 768 211 456

Plus l'espace de clés est grand, plus la force brute est difficile. 💪

📏 Comment calculer la taille de clé ?

Une clé est une suite de **bits** (0 ou 1). Chaque bit double le nombre de possibilités.

$$\text{Espace de clés} = 2^n \quad \text{où } n = \text{nombre de bits}$$

Bits	Calcul	Nombre de clés
1 bit	2^1	2
8 bits	2^8	256
56 bits	2^{56}	~72 × 10^{15}
128 bits	2^{128}	~3.4 × 10^{38}
256 bits	2^{256}	~1.2 × 10^{77}

⏱️ Combien de temps pour tout tester ?

Si un ordinateur teste **1 milliard de clés par seconde** (10^9 clés/s) :

$$T = \frac{2^n}{10^9} \text{ secondes}$$

Taille de clé	Nombre de clés	⏱️ Temps estimé
32 bits	$2^{32} \approx 4 \times 10^9$	~4 secondes
56 bits	$2^{56} \approx 7 \times 10^{16}$	~2.3 ans
128 bits	$2^{128} \approx 3.4 \times 10^{38}$	~10^{22} années 🌌
256 bits	$2^{256} \approx 1.2 \times 10^{77}$	~10^{60} années 🌌🌌

🌌 L'âge de l'univers : ~1.4×10^{10} années

🔥 En perspective : limites physiques

Le physicien **Bruce Schneier** a calculé :

> Même si on transformait toute l'énergie du Soleil ☀️ en calculs, et que chaque opération consommait l'énergie minimale théorique (principe de Landauer), on ne pourrait pas compter jusqu'à 2^{256}.

🌡️ Le **principe de Landauer** : effacer un bit coûte au minimum kT ln 2 joules d'énergie.

☀️ Le Soleil produit ~3.8×10^{26} watts.

🌌 En 1 milliard d'années, le Soleil ne fournirait pas assez d'énergie pour tester 2^{256} clés.

➡️ AES-256 est sûr **par les lois de la physique**, pas seulement par la technologie actuelle ! 🏆

🔑 La force d'un mot de passe

Un mot de passe a aussi un **espace de clés** : Espace = N^L (N = alphabet, L = longueur)

Type	N	8 caractères	Bits
Chiffres seuls	10	100 millions	~27
Minuscules	26	~2×10^{11}	~37
Min. + chiffres	36	~2.8×10^{12}	~41
Tout (maj., symboles)	95	~6.6×10^{15}	~53

⚠️ 8 caractères = au mieux **53 bits** de sécurité. C'est **moins que DES** !

💡 Utilisez des mots de passe **longs** (>12 car.) ou un **gestionnaire**. 🔐

💣 La force brute

Principe : essayer **toutes les clés possibles** jusqu'à trouver la bonne.

- César (26 clés) → ⚡ instantané
- DES (2^{56} clés) → 💀 cassé en 22h en 1999
- AES-128 (2^{128} clés) → 🌌 l'univers entier n'y suffirait pas

La force brute est l'attaque la plus **simple** mais la plus **coûteuse**.

👨‍💻 Démo : César chiffrement

```python
def cesar(texte, cle):
    resultat = ""
    for c in texte:
        if c.isalpha():
            base = ord('A') if c.isupper() else ord('a')
            resultat += chr((ord(c) - base + cle) % 26 + base)
        else:
            resultat += c
    return resultat

message = "Hello World"
chiffre = cesar(message, 3)
print(f"Message : {message}")
print(f"Chiffré : {chiffre}")
```

🔓 Démo : César force brute

```python
def cesar(texte, cle):
    resultat = ""
    for c in texte:
        if c.isalpha():
            base = ord('A') if c.isupper() else ord('a')
            resultat += chr((ord(c) - base + cle) % 26 + base)
        else:
            resultat += c
    return resultat

message_chiffre = "Khoor Zruog"
for cle in range(26):
    dechiffre = cesar(message_chiffre, -cle)
    marqueur = " <--" if "Hello" in dechiffre else ""
    print(f"Clé {cle:2d}: {dechiffre}{marqueur}")
```

📊 Analyse fréquentielle : pourquoi ça marche

Les lettres n'apparaissent pas avec la même fréquence. Quelle est la lettre la plus fréquente en français ?

🇫🇷 **En français** : E (14.7%), A (7.6%), I (7.4%), S (7.3%), T (7.2%)

Et en anglais ?

🇬🇧 **En anglais** : E (12.7%), T (9.1%), A (8.2%), O (7.5%), I (7.0%)

🔍 **Méthode** : si la lettre la plus fréquente dans le chiffré est **X**, alors X représente probablement **E**, donc :

$$\text{clé} = X - E$$

📊 Force brute : bilan

26 clés → on peut toutes les essayer **à la main** ✋

Mais si on avait **2^128** clés ?

Même en testant **1 milliard de clés par seconde**, il faudrait **10^22 années** : bien plus que l'âge de l'univers 🌌

➡️ La taille de la clé est **cruciale**.

💀 Échecs célèbres de la cryptographie

📡 WEP : le WiFi grand ouvert (1999-2004)

📶 **WEP** (Wired Equivalent Privacy) : premier protocole de sécurité WiFi.

❌ Problèmes multiples :

- 🔑 Clé trop courte (40 ou 104 bits)
- 🔄 Réutilisation des **vecteurs d'initialisation** (IV de 24 bits seulement)
- 🧮 Utilisation incorrecte de l'algorithme RC4

💀 **2001** : Fluhrer, Mantin et Shamir publient une attaque dévastatrice.

⚡ Aujourd'hui : un réseau WEP se casse en **moins de 5 minutes** avec `aircrack-ng`.

➡️ Remplacé par **WPA2** puis **WPA3**.

🎮 Sony PS3 : le fail du nombre aléatoire (2010)

🎮 Sony signe ses logiciels PS3 avec l'algorithme **ECDSA** (signature sur courbes elliptiques).

⚠️ ECDSA nécessite un nombre aléatoire **unique** *k* pour chaque signature :

$$s = k^{-1}(z + r \cdot d) \bmod n$$

💀 Sony utilise le **même *k*** à chaque fois ! (un nombre constant au lieu d'un aléatoire)

🧮 Avec deux signatures et le même *k*, on peut retrouver la **clé privée** :

$$d = \frac{s_2 \cdot z_1 - s_1 \cdot z_2}{s_1 \cdot r_2 - s_2 \cdot r_1} \bmod n$$

🏴‍☠️ Résultat : n'importe qui peut signer des logiciels "officiels" pour PS3.

💔 Heartbleed : la faille OpenSSL (2014)

🐛 Un bug dans **OpenSSL** (la bibliothèque utilisée par ~66% des serveurs web).

🔍 La fonctionnalité "heartbeat" de TLS permettait de lire **64 Ko de mémoire** du serveur à chaque requête.

💀 Dans cette mémoire : des **clés privées**, mots de passe, cookies de session…

📊 Impact :

- 🌐 ~500 000 serveurs vulnérables
- 🔑 Les clés privées de serveurs majeurs potentiellement compromises
- 📅 Le bug existait depuis **2 ans** avant d'être découvert

> Le code est écrit par des humains. Les humains font des erreurs. 🤷

🕵️ Dual EC DRBG : la backdoor de la NSA (2013)

🎲 **Dual EC DRBG** : un générateur de nombres pseudo-aléatoires standardisé par le NIST en 2006.

🤔 Dès 2007, des chercheurs soupçonnent une **porte dérobée** (backdoor).

📜 **2013** : les documents Snowden révèlent que la **NSA a payé 10 millions de $** à RSA Security pour que Dual EC DRBG soit le générateur par défaut de leurs produits.

💀 Avec la backdoor, la NSA pouvait **prédire** les nombres "aléatoires" et donc casser le chiffrement.

> Même un **standard officiel** peut contenir une backdoor. 😱

📱 Telegram : "on fait notre propre crypto" (2013+)

💬 **Telegram** a décidé de créer son **propre protocole** cryptographique : **MTProto**.

⚠️ Problèmes identifiés par les chercheurs :

- 🏗️ Construction non standard (SHA-1 + AES-IGE, un mode abandonné)
- 🔓 Le chiffrement de bout en bout n'est **pas activé par défaut**
- 📊 Les messages "normaux" sont lisibles par les serveurs Telegram

🆚 En comparaison, **Signal** utilise le protocole Signal, audité publiquement et adopté par WhatsApp, Google Messages…

> **Ne jamais inventer sa propre cryptographie.** Utiliser des standards éprouvés. 📏

📋 Leçons des échecs

- 🔑 **La taille de clé compte** : WEP (40 bits) trop faible
- 🎲 **L'aléatoire est crucial** : Sony PS3 (même *k* réutilisé)
- 🐛 **L'implémentation compte** : Heartbleed (bug dans le code)
- 🕵️ **La confiance est fragile** : Dual EC DRBG (backdoor NSA)
- 📏 **Utiliser les standards** : Telegram (crypto "maison" dangereuse)
- 🔒 La crypto est un **système** : une seule erreur suffit à tout compromettre

Ouverture sur la crypto moderne

🏛️ DES (1977)

Premier standard de chiffrement adopté par le gouvernement américain.

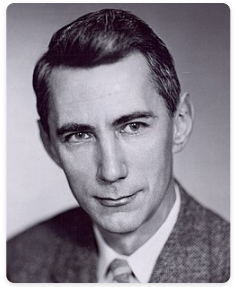
- 🔑 Clé de **56 bits**
- 🏢 Conçu par IBM, validé par la NSA
- 📅 Standard pendant 20 ans

💀 **1999** : cassé en **22 heures** par la machine "Deep Crack" de l'EFF (250 000 $).

➡️ 56 bits ne suffisent plus.

🧠 1949

Claude Shannon



Claude Shannon publie *"Communication Theory of Secrecy Systems"*.

📜 Il fonde la **théorie mathématique** de la cryptographie moderne.

Deux concepts clés :

- 🎲 **Confusion** : rendre la relation entre la clé et le chiffré aussi complexe que possible
- 🌊 **Diffusion** : répandre l'influence de chaque bit du clair sur tout le chiffré

💡 Ces deux principes sont à la base d'**AES** (que vous verrez en CM2).

🧠 Shannon : le chiffrement parfait

Shannon démontre mathématiquement qu'un chiffrement est **parfaitement sûr** si et seulement si :

$$H(K) \geq H(M)$$

🔑 La clé doit être **au moins aussi longue** que le message.

📝 C'est le principe du **masque jetable** (*one-time pad*) :

- Clé aléatoire de même longueur que le message
- Utilisée **une seule fois**
- XOR entre le message et la clé
- ✅ **Mathématiquement incassable** !

⚠️ Problème : il faut échanger une clé **aussi longue que le message**... pas pratique.

🆕 AES & RSA

🔐 AES (2001)

- Remplaçant de DES
- Compétition internationale NIST
- Clé **128/192/256 bits**
- Algorithme **Rijndael** 🇧🇪
- Standard mondial actuel

🔑 RSA (1977)

- Cryptographie **asymétrique**
- Deux clés : **publique** et **privée**
- Pas besoin de secret partagé
- Par Rivest, Shamir, Adleman
- Résout l'échange de clé

🤝 Diffie-Hellman (1976) : la révolution

🧑‍🔬 **Whitfield Diffie** et **Martin Hellman** publient *"New Directions in Cryptography"*.

💡 Idée révolutionnaire : deux personnes peuvent se mettre d'accord sur un **secret partagé** en communiquant sur un canal **public** !

🔑 C'est l'**échange de clé Diffie-Hellman** :

- Alice et Bob n'ont jamais besoin de se rencontrer
- Eve peut observer tout l'échange sans compromettre le secret
- Résout le problème vieux de 3000 ans ! 🎉

➡️ Détails au CM3.

Frise de la crypto moderne

- **1949** — Shannon, théorie de l'information
- **1976** — Diffie-Hellman, échange de clé
- **1977** — DES (standard) et RSA (asymétrique)
- **1991** — PGP, chiffrement email pour tous
- **1999** — DES cassé en 22h
- **2001** — AES, nouveau standard
- **2013** — Snowden, révélations NSA
- **2015** — Let's Encrypt, HTTPS gratuit
- **2018** — TLS 1.3

🌐 La crypto aujourd'hui : quelques chiffres

📊 En 2025 :

- 🔒 **95%** du trafic web mondial est chiffré (HTTPS)
- 💬 **2 milliards** d'utilisateurs WhatsApp avec chiffrement de bout en bout
- ₿ **Bitcoin** : 500 000+ transactions/jour sécurisées par SHA-256
- 📜 **Let's Encrypt** : 400+ millions de certificats actifs
- ⚛️ La course à la **cryptographie post-quantique** est lancée (NIST a standardisé ML-KEM en 2024)

La cryptographie est l'**infrastructure invisible** d'Internet. 🏗️

⚛️ Et demain ? La menace quantique

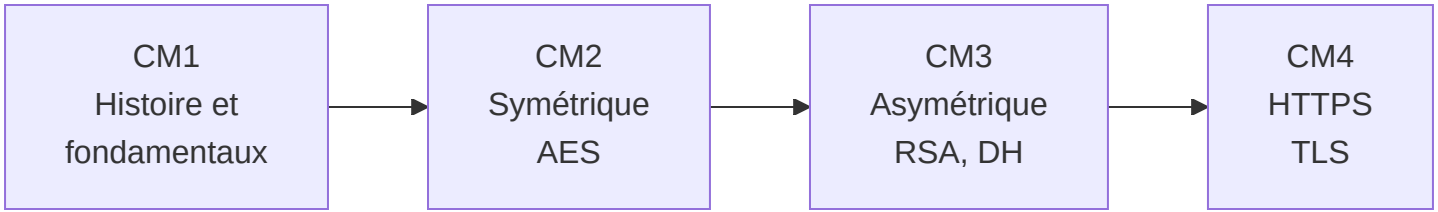
🖥️ Les **ordinateurs quantiques** menacent certains algorithmes actuels :

Algorithme	Menace quantique
🔑 RSA	☠️ **Cassé** par l'algorithme de Shor
🤝 Diffie-Hellman	☠️ **Cassé** par l'algorithme de Shor
🔐 AES-128	⚠️ Affaibli (Grover : 64 bits effectifs)
🔐 AES-256	✅ Résistant (128 bits effectifs)

🛡️ Le NIST a standardisé de nouveaux algorithmes **post-quantiques** en 2024 (ML-KEM, ML-DSA).

⏰ Les ordinateurs quantiques capables de casser RSA ne sont pas encore là, mais il faut se préparer **maintenant** (attaque "harvest now, decrypt later").

🗺️ Ce qu'on va voir dans ce module



- 🔐 **CM2** : Comment AES fonctionne, modes ECB/CBC, le problème de la clé
- 🔑 **CM3** : RSA, signatures numériques, Diffie-Hellman
- 🌐 **CM4** : Comment tout ça protège vos connexions web (TLS/HTTPS)
- 🧪 **TDs** : Vous allez casser des chiffrements, chiffrer des images, intercepter du trafic

📝 À retenir

- 🛡️ La cryptographie protège la **confidentialité**, l'**intégrité** et l'**authenticité**
- 📏 Le principe de **Kerckhoffs** : la sécurité repose sur la **clé**, pas sur l'algorithme
- 🔢 L'**espace de clés** détermine la résistance à la force brute
- 📜 L'histoire montre que les systèmes "incassables" finissent par être cassés
- 🧮 La crypto moderne (AES, RSA) repose sur des **problèmes mathématiques difficiles**

🧪 TD1 : la semaine prochaine

Vous allez :

- ✍️ Implémenter le **chiffre de César** en Python
- 💣 Le **casser par force brute**
- 📊 Utiliser l'**analyse fréquentielle** d'Al-Kindi
- ⭐ (Bonus) Attaquer **Vigenère**

📚 Bibliographie et sources

📖 Livres recommandés

- 📕 **Hervé Lehning**, *La Bible des codes secrets*, Flammarion, 2019
  - Histoire complète de la cryptographie, des hiéroglyphes à l'ère numérique
  - Très accessible, nombreuses anecdotes
- 📗 **Simon Singh**, *Histoire des codes secrets*, Le Livre de Poche, 1999
  - De César à l'informatique quantique
  - Le récit de Marie Stuart, Enigma, RSA
- 📘 **Bruce Schneier**, *Applied Cryptography*, Wiley, 1996
  - La référence technique en cryptographie appliquée
- 📙 **Christof Paar & Jan Pelzl**, *Understanding Cryptography*, Springer, 2010

🔗 Sources des anecdotes

- 📜 **Scytale, César, Al-Kindi** : Lehning, *La Bible des codes secrets*, ch. 1-3
- 👸 **Marie Stuart** : Singh, *Histoire des codes secrets*, ch. 1
- 🔤 **Vigenère et Babbage** : Singh, ch. 2 ; Lehning, ch. 5
- 📨 **Télégramme Zimmermann** : Singh, ch. 4 ; Barbara Tuchman, *The Zimmermann Telegram*, 1958
- ⚙️ **Enigma et Bletchley Park** : Andrew Hodges, *Alan Turing: The Enigma*, 1983
- 🇵🇱 **Polonais et la Bomba** : Dermot Turing, *X, Y & Z: The Real Story of How Enigma Was Broken*, 2018
- 🪶 **Navajo code talkers** : Chester Nez, *Code Talker*, 2011
- 🧠 **Claude Shannon** : Shannon, *"Communication Theory of Secrecy Systems"*, Bell System Technical Journal, 1949

🔗 Sources des échecs modernes

- 📡 **WEP** : Fluhrer, Mantin, Shamir, *"Weaknesses in the Key Scheduling Algorithm of RC4"*, 2001
- 🎮 **Sony PS3** : fail0verflow, présentation au 27C3, *"Console Hacking 2010"*
- 💔 **Heartbleed** : CVE-2014-0160, heartbleed.com
- 🕵️ **Dual EC DRBG** : Joseph Menn, Reuters, *"Exclusive: Secret contract tied NSA and security industry pioneer"*, 2013
- 📱 **Telegram MTProto** : Jakob Jakobsen & Claudio Orlandi, *"On the CCA (in)security of MTProto"*, 2015

🎓 Pour aller plus loin

🎬 **Vidéos et documentaires** :

- 🎥 *The Imitation Game* (2014) : film sur Turing et Enigma
- 📺 *Numberphile* (YouTube) : excellentes vidéos sur la crypto
- 📺 *Computerphile* (YouTube) : aspects informatiques de la crypto

🌐 **Sites interactifs** :

- 🔧 CrypTool (cryptool.org) : outil éducatif de cryptographie
- 🔐 CyberChef (gchq.github.io/CyberChef) : couteau suisse du chiffrement
- ⚙️ Simulateur Enigma en ligne

📰 **Suivre l'actualité crypto** :

- 🐦 Schneier on Security (schneier.com)
- 📄 IACR ePrint Archive (eprint.iacr.org)

📝 Résumé des concepts clés du CM1

Concept	Définition
🔒 Chiffrement	Transformer un message clair en message illisible
🔓 Déchiffrement	Retrouver le clair **avec** la clé
🔍 Décryptage	Retrouver le clair **sans** la clé
🔑 Clé	Paramètre secret qui contrôle le chiffrement
📏 Kerckhoffs	La sécurité repose sur la clé, pas l'algorithme
💣 Force brute	Tester toutes les clés (2^n)
📊 Analyse fréquentielle	Exploiter les statistiques de la langue

❓ Questions ?

CM1 · Histoire & fondamentaux de la cryptographie