الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique Et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement supérieur et de la Recherche scientifique
المدرسة الوطنية العليا لعلوم البحر و تهيئة الساحل
École Nationale Supérieure Des Sciences De La Mer Et De L'aménagement Du Littoral

Département : Environnement & Aménagement
Domaine : Sciences de la Nature et de la Vie (SNV)
Filière : Hydrobiologie continentale et marine
Spécialité : Ingénierie de l'Environnement Marin et Côtier

Année universitaire: (2024/25)

MEMOIRE DE FIN D'ÉTUDES

'' Une étude sur l'application de la cryptographie pour la protection des données environnementales marines ''

Réalisée par

Smaali ImadEddine

Encadré(e) par:	Mme BOUMEZBEUR. M	MCB	ENSSMAL
Co-encadré(e) par:	Mme ATTIA. N	MCA	ENSSMAL

Soutenu le 01/07/2025

Devant le jury

PRÉSIDENTE:	Mme BEDAIRIA. A	MCB	ENSSMAL
EXAMINATRICE :	Mme MOKHBI. D	MCB	ENSSMAL

Remerciements :

Je tiens avant tout à remercier très chaleureusement **Madame Boumezbeur. M**, mon encadrante, pour la qualité de son accompagnement tout au long de ce travail. Sa disponibilité, sa rigueur scientifique et ses conseils toujours pertinents ont été décisifs pour m'aider à structurer ma réflexion, à affiner mes analyses et à mener ce projet avec exigence et clarté. J'adresse également mes sincères remerciements à Madame **Attia. N**, ma co-encadrante, pour ses observations constructives et son soutien qui ont enrichi ce travail.

Je remercie également **mesdames BEDAIRIA. A** et **MOKHBI. D, les membres du jury**, pour l'attention qu'ils ont portée à ce travail, et pour la richesse de leurs remarques, qui constituent autant de pistes de progression que d'encouragements.

Enfin, je tiens à saluer **l'ensemble du personnel de l'École Nationale Supérieure des Sciences de la Mer et de l'Aménagement du Littoral**, dont l'engagement quotidien dans l'accompagnement administratif, pédagogique et humain des étudiants mérite toute notre gratitude. Leur présence discrète mais essentielle a largement contribué à rendre cette formation aussi formatrice que inspirante.

Dédicace :

À mes parents, qui sont les racines tranquilles de tout ce que je suis devenu. Ils m'ont transmis, à travers le silence et la constance, le goût de l'effort, l'amour du savoir et la patience du long chemin.

À mes frères et à mes sœurs, présences fidèles de mes premiers pas et de mes premiers combats, dont les regards et les gestes m'ont toujours rappelé que l'on avance mieux entouré de tendresse et de confiance.

À mon épouse, lumière douce dans mes nuits d'étude, dont la présence discrète, mais essentielle, a su me guider sans bruit, comme une boussole silencieuse dans les orages du doute.

À mes amis Akram, Taki et Idrisse, frères choisis du cœur et de l'esprit, avec qui j'ai partagé bien plus que des instants : des pensées vives, des rires vrais, et cette étrange alliance entre folie et lucidité.

Et à Madame Boumezbeur, dont la confiance, l'exigence bienveillante et les conseils discrets ont éclairé cette recherche comme une étoile qui rassure, même lorsque le ciel semble vide.

Table des matières:

Liste des abréviations:

Abréviation	Signification
AES	Advanced Encryption Standard
AIS	Automatic Identification System
AUV	Autonomous Underwater Vehicle
CBC	Cipher Block Chaining
ChaCha20	Stream cipher designed by Daniel J. Bernstein
CTR	Counter Mode
DGPS	Differential Global Positioning System
DoS	Denial of Service
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
GCM	Galois/Counter Mode
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HMAC	Hash-based Message Authentication Code
IoT	Internet of Things
IV	Initialization Vector
LWC	Lightweight Cryptography
MAC	Message Authentication Code
MIoT	Marine Internet of Things
NMEA	National Marine Electronics Association
OSNMA	Open Service Navigation Message Authentication
PKI	Public Key Infrastructure

- **PRNG** Pseudo-Random Number Generator
- **RSA** Rivest-Shamir-Adleman (algorithm)
- **RTK** Real-Time Kinematic
- **SCADA** Supervisory Control and Data Acquisition
- **SHA** Secure Hash Algorithm
- **SNR** Signal-to-Noise Ratio
- **S-Box** Substitution Box
- **TPM** Trusted Platform Module
- **UASN** Underwater Acoustic Sensor Network
- **VSAT** Very Small Aperture Terminal
- **XOR** Exclusive OR (opération logique)
- **ZEE** Zone Économique Exclusive
- **API** Application Programming Interface
- **CPU** Central Processing Unit
- **CRC** Cyclic Redundancy Check
- **DES** Data Encryption Standard
- **DH** Diffie-Hellman
- **DNS** Domain Name System
- **EEPROM** Electrically Erasable Programmable Read-Only Memory
- **EMC** Electromagnetic Compatibility
- **HTTPS** HyperText Transfer Protocol Secure
- **IDS** Intrusion Detection System
- **IP** Internet Protocol
- **LAN** Local Area Network
- **LED** Light Emitting Diode
- **MAC (réseau)** Media Access Control (réseau)
- **MQTT** Message Queuing Telemetry Transport

- **NIST** National Institute of Standards and Technology
- **NTP** Network Time Protocol
- **OTP** One-Time Pad
- **PWM** Pulse-Width Modulation
- **RAM** Random Access Memory
- **ROM** Read-Only Memory
- **RTOS** Real-Time Operating System
- **SSH** Secure Shell
- **SSL** Secure Sockets Layer
- **TLS** Transport Layer Security
- **UDP** User Datagram Protocol
- **UART** Universal Asynchronous Receiver-Transmitter
- **USB** Universal Serial Bus
- **WAN** Wide Area Network
- **WLAN** Wireless Local Area Network

Introduction générale :

À l'heure où les océans se numérisent à grande vitesse, la question de la protection des données environnementales marines s'impose avec une acuité nouvelle. Des milliers de capteurs, de balises et de dispositifs embarqués scrutent en permanence la température de l'eau, la qualité des écosystèmes, les mouvements des courants ou les migrations d'espèces. Cette formidable dynamique de collecte, facilitée par l'essor de l'Internet des objets et de la connectivité mondiale, génère des volumes de données dont la valeur scientifique et stratégique ne cesse de croître. Pourtant, cette richesse informationnelle s'accompagne d'un risque : celui de voir des données sensibles exposées, détournées ou manipulées, dans un contexte où les enjeux environnementaux, économiques et géopolitiques se conjuguent désormais en mer.

Dans ce paysage inédit, la cryptographie apparaît comme le socle indispensable de toute stratégie de sécurisation. Cette discipline, à la croisée des mathématiques et de l'informatique, s'est construite au fil des siècles pour répondre aux défis de la confidentialité, de l'intégrité et de l'authenticité des messages. Si ses premières traces remontent à près de quatre millénaires, avec des techniques primitives de codage utilisées par les civilisations antiques [**Kahn, D. (1996, December 5)**], elle a su traverser les âges, des alphabets secrets de Jules César [**Singh, S. (1999)**], **Stinson, D. R. (2005)**, **Paar, C., & Pelzl, J. (2010)**] aux systèmes modernes qui fondent aujourd'hui la sécurité de nos échanges numériques [**Schneier, B., & Diffie, W. (2015)**]. Ce long cheminement témoigne de sa capacité à s'adapter à la sophistication croissante des menaces, qu'il s'agisse d'espionnage, de sabotage ou simplement d'accès non autorisé à l'information.

Mais au-delà de cette histoire fascinante, la cryptographie se distingue désormais par la diversité de ses applications dans le monde maritime. Que ce soit pour la navigation, le monitoring environnemental, l'alerte précoce aux pollutions, la gestion des flottes ou les études scientifiques à grande échelle, elle s'invite partout où la donnée circule et doit être protégée. Les communications en mer, qu'elles reposent sur des réseaux radio, satellitaires ou acoustiques, sont soumises à des conditions extrêmes : isolation des équipements, variations de bande passante, contraintes énergétiques et nécessité de résister à des attaques sophistiquées. Otnes et al, à travers leurs études, ont mis en lumière la complexité technique de ces systèmes et la nécessité d'adapter les protocoles de communication aux réalités de l'environnement marin [**Qarabaqi, P., & Stojanovic, M. (2013)**, **Otnes, R., et al. (2012)**].

La convergence entre la cryptographie et les technologies marines soulève alors une question centrale : comment concevoir des mécanismes de protection à la fois robustes, efficaces et adaptés à l'hostilité du milieu océanique ? Plus précisément, comment garantir la confidentialité, l'intégrité et la traçabilité des données environnementales, tout en maîtrisant la consommation énergétique, la gestion des clés de chiffrement et la résilience des systèmes ? Cette problématique, au cœur des préoccupations actuelles de la communauté scientifique et des organismes internationaux, trouve une résonance particulière face à la montée des cybermenaces visant aussi bien la recherche que les intérêts économiques ou stratégiques [**Zhou, Q. et al. (2025)**, **Layode, N. et al. (2024).**].

L'originalité de ce travail réside dans l'exploration détaillée des solutions cryptographiques les mieux adaptées à la réalité du terrain marin. Il s'agit d'étudier non seulement les algorithmes eux-mêmes, qu'ils soient symétriques, asymétriques ou de nouvelle génération, mais aussi les protocoles de gestion des clés, l'intégration dans les dispositifs embarqués.

Cette approche se fonde sur une double exigence : garantir un haut niveau de sécurité et rendre ces solutions réellement opérantes, compte tenu des contraintes matérielles et environnementales spécifiques au monde marin.

Ce mémoire propose ainsi un parcours structuré, allant des fondements historiques et mathématiques de la cryptographie jusqu'aux applications concrètes dans la sécurisation des données environnementales marines. Il s'appuie sur les avancées récentes de la recherche scientifique, les recommandations institutionnelles, les normes de sécurité (comme l'ISO/IEC 27001 [**ISO. (2022)**]) et des études de cas illustrant les défis et les succès rencontrés sur le terrain.

Chapitre I Fondements mathématiques pour la cryptographie

Introduction

Avant de s'aventurer dans l'univers des algorithmes cryptographiques modernes et leur intégration dans les dispositifs embarqués en milieu marin, il est crucial de poser les bases : des fondations mathématiques solides, issues des mathématiques discrètes. Car la cryptographie ne se limite pas à des intuitions ingénieuses ou à des protocoles bien pensés, elle repose, de manière profonde, sur des structures rigoureuses telles que les groupes, les anneaux, les nombres premiers, ou encore les fonctions bijectives.

Ces notions ne sont pas des curiosités réservées aux mathématiciens : elles sont au contraire au cœur des mécanismes qui assurent la confidentialité, l'intégrité et l'authenticité des échanges numériques. Elles sont la trame invisible de nos communications sécurisées.

Prenons un exemple concret : imaginons Bob, un ingénieur chargé de superviser un réseau de capteurs marins. Il souhaite garantir que les données échangées entre la station côtière et un drone océanographique, nommé Alice, ne puissent être interceptées ni altérées. Pour cela, des algorithmes de chiffrement vont être déployés, et chacun d'eux repose sur des idées mathématiques précises : arithmétique modulaire, produits de nombres premiers, calculs d'inverses, ou encore opérations logiques binaires comme le XOR.

Sans une compréhension minimale de ces fondements, il devient difficile, voire impossible, de concevoir des systèmes sûrs, d'en évaluer les vulnérabilités ou d'en garantir la robustesse face aux attaques.

Ce premier chapitre poursuit donc une double ambition :

- Introduire pas à pas les outils mathématiques fondamentaux qui sous-tendent la plupart des systèmes cryptographiques actuels.

- Illustrer chaque concept à travers des exemples concrets, souvent issus du monde maritime ou de scénarios cryptographiques classiques, pour en montrer immédiatement la pertinence opérationnelle.

Nous commencerons par explorer les nombres premiers et la décomposition en facteurs, piliers des systèmes asymétriques tels que RSA. Nous poursuivrons avec l'arithmétique modulaire, une branche fascinante des mathématiques où les entiers "bouclent" sur eux-mêmes, formant des mondes numériques fermés. Nous introduirons ensuite la fonction indicatrice d'Euler $\varphi(n)$, essentielle pour appréhender la structure des groupes modulo, avant de plonger dans les opérations logiques binaires, avec un focus sur le XOR, pierre angulaire du chiffrement symétrique. Enfin, nous évoquerons le rôle des matrices, bien connues des ingénieurs, dans des systèmes de chiffrement avancés comme l'algorithme de Hill ou le AES, où l'algèbre linéaire devient une arme de protection des données.

I.1 – Nombres premiers et factorisation

Les nombres premiers constituent les briques essentielles de l'arithmétique et jouent un rôle central en cryptographie. Leur importance repose sur une propriété fondamentale : il est facile de multiplier deux grands nombres premiers, mais extrêmement difficile de retrouver ces facteurs à partir de leur produit, un problème connu sous le nom de factorisation, qui fonde la robustesse de nombreux systèmes de chiffrement.

I.1.1 Pourquoi est-ce important en cryptographie ?

Dans le monde de la cryptographie, notamment en cryptographie asymétrique, comme l'algorithme RSA, cette notion est indispensable. Par exemple, lorsqu'on choisit un exposant e qui doit être premier avec $\varphi(n)$, la fonction indicatrice d'Euler, où $n = p \times q$ est un produit de deux grands nombres premiers [**Rivest, R. L., Shamir, A., & Adleman, L. (1978)**].

Pourquoi ? Parce que cette condition garantit l'existence de l'inverse modulaire de e modulo $\varphi(n)$, ce qui est nécessaire pour pouvoir calculer la clé privée d. Et cet inverse n'existe que si e et $\varphi(n)$ sont premiers entre eux [**Rivest, R. L., Shamir, A., & Adleman, L. (1978)**, **Menezes, A., et al. (2018)**].

Mathématiquement, cela repose sur le théorème de Bézout :

> Il existe des entiers u et v tels que : $au + bv = 1 \Leftrightarrow a$ et b sont premiers entre eux.

Cette identité, que l'on peut trouver à l'issue de l'algorithme d'Euclide, est la pierre angulaire de nombreux algorithmes cryptographiques, comme l'algorithme de génération de clés RSA ou encore certains protocoles d'échange de clés comme Diffie-Hellman [**Menezes, A., et al. (2018).**, **Trappe, W., & Washington, L. C. (2006)**].

I.1.2 Théorème fondamental de l'arithmétique

Le théorème fondamental de l'arithmétique stipule :

> Tout entier naturel strictement supérieur à 1 peut être exprimé de manière unique (à l'ordre près) comme un produit de nombres premiers.

Par exemple :

$$84 = 2^2 \times 3 \times 7$$
$$315 = 3^2 \times 5 \times 7$$

Cela signifie que les nombres premiers sont à l'arithmétique ce que les atomes sont à la chimie.

I.1.3 Exemple, Alice et Bob :

Supposons qu'Alice choisisse deux grands nombres premiers, par exemple :

$$p = 61, q = 53$$

Elle calcule ensuite :

$$N = p \times q = 61 \times 53 = 3233$$

Alice envoie seulement le nombre N à Bob. Ce dernier peut utiliser N dans des opérations cryptographiques, sans jamais connaître p et q. Même si Eve intercepte N, elle ne pourra pas facilement retrouver p et q, car la factorisation de grands nombres comme N est computationnellement coûteuse.

C’est précisément cette asymétrie entre la facilité de multiplication et la difficulté de factorisation qui fonde la sécurité de RSA [**Rivest, R. L., Shamir, A., & Adleman, L. (1978)**].

I.1.4 La difficulté de la factorisation

La complexité du problème de la factorisation a été formalisée mathématiquement. À ce jour, aucun algorithme classique connu ne permet de factoriser un entier de plusieurs milliers de bits dans un temps raisonnable.

Par exemple, pour un entier N de 2048 bits, il faudrait des milliards d’années à un ordinateur classique pour retrouver ses facteurs premiers, même avec des optimisations modernes [**Bernstein, Daniel. J. (2009)**].

Cependant, certains algorithmes quantiques (comme Shor’s Algorithm) pourraient changer la donne à l’avenir, en cassant cette sécurité [**Shor, P. W. (1994)**].

I.2 – Arithmétique modulaire

L’arithmétique modulaire, repose sur une idée simple : on travaille avec des restes, comme sur une horloge. On dit que deux entiers a et b sont congruents modulo n , et on note :

$$a \equiv b \, mod \, n$$

Lorsque a et b laissent le même reste après division par n . Par exemple :

$$17 \equiv 5 \, mod \, 12 \, car \, 17 = 12 + 5$$

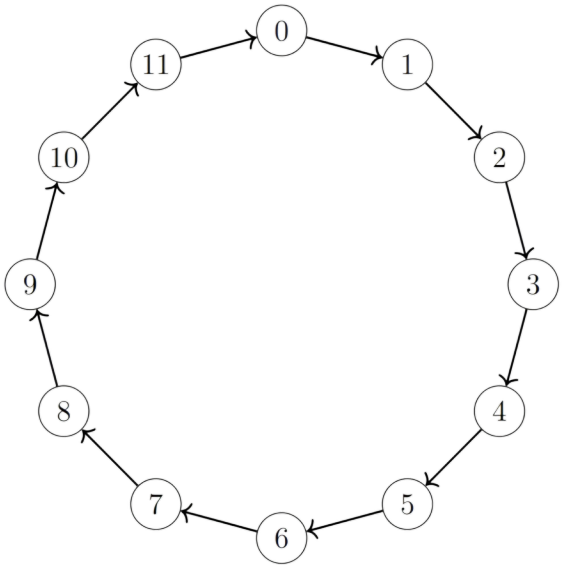
I.2.1 Exemple : l’horloge

Prenons une horloge à 12 heures. Si vous partez de 10h et ajoutez 5 heures :

$$10 + 5 = 15 \equiv 3 \, mod \, 12$$

C’est le principe du retour à zéro : on boucle dès qu’on atteint 12.

Voici une représentation circulaire de l’arithmétique modulaire modulo 12, souvent utilisée pour illustrer ce concept :



Chaque saut représente une addition ou multiplication mod. C'est ainsi que les valeurs reviennent à zéro lorsqu'elles dépassent la base.

En cryptographie, cette périodicité maîtrisée permet de restreindre les espaces de calcul et constitue la base de nombreux algorithmes de chiffrement tels que RSA, Diffie-Hellman ou ElGamal.

Elle s'appuie sur une propriété essentielle de l'arithmétique modulaire : ramener systématiquement les résultats dans une plage de valeurs bornée, qui permet d'éviter les débordements et d'alléger les calculs cryptographiques sur des processeurs embarqués, notamment dans les capteurs, drones ou balises marines, où la mémoire et l'énergie disponibles sont extrêmement restreintes.

I.3 – Définition de l'anneau ℤ/nℤ

Dans les mathématiques discrètes et en cryptographie, l'un des objets les plus fondamentaux, et souvent sous-estimés, est ce que l'on appelle l'anneau ℤ/nℤ.

Penser en termes de restes :

Quand vous divisez un entier *a* par un entier *n*, vous obtenez un quotient et un reste. Ce reste est toujours un entier compris entre 0 et $n - 1$. Par exemple, 14 divisé par 5 donne un reste de 4, car :

$$14 = 2 \times 5 + 4 \quad \text{← reste}.$$

(4 est le reste)

L'idée derrière ℤ/*n*ℤ, c'est précisément de ne conserver que ce reste, et d'ignorer le quotient. Autrement dit :

$$14 \equiv 4 \,(mod\, 5) \; et \; aussi \; 9 \equiv 4 \,(mod\, 5) \; et \; même - 1 \equiv 4 \,(mod\, 5).$$

Tous ces nombres appartiennent à la même classe modulo 5, qu'on note simplement :

$$[4] \in Z/5Z.$$

Qu'est-ce que $\mathbb{Z}/n\mathbb{Z}$?

L'anneau $\mathbb{Z}/n\mathbb{Z}$ est défini comme l'ensemble des classes d'équivalence de $\mathbb{Z}$ modulo n, c'est-à-dire :

$$Z/nZ = \{[0],[1],[2],\ldots,[n-1]\},$$

où chaque $[a]$ représente l'ensemble des entiers congrus à a modulo n.

Deux entiers a et b sont dits congrus modulo n si n divise leur différence, c'est-à-dire :

$$a \equiv b (mod\ n) \Leftrightarrow n \mid (a-b).$$

Les classes $[a]$ forment un anneau : on peut y **a**dditionner, multiplier, et cela donne encore une classe modulo n. C'est ce qui rend $\mathbb{Z}/n\mathbb{Z}$ si précieux dans les constructions cryptographiques : on reste dans le système, quoi qu'on fasse.

I.3.1 Structure de l'anneau $\mathbb{Z}/n\mathbb{Z}$: anneau ou corps ?

$\mathbb{Z}/n\mathbb{Z}$ est toujours un anneau, mais il est un corps seulement si n est premier.

Cela signifie que :

- Si n est premier (par exemple $n = 7$), alors chaque élément non nul possède un inverse et $\mathbb{Z}/7\mathbb{Z}$ est un corps fini, noté aussi $\mathbb{F}_7$.
- Si n n'est pas premier (par exemple $n = 15$), alors certains éléments n'ont pas d'inverse, et $\mathbb{Z}/15\mathbb{Z}$ n'est pas un corps, mais reste un anneau [**Douglas Robert Stinson, & Paterson, M. B. (2019)**].

En cryptographie (Chapitre 2), on exploite ces deux cas :

- RSA travaille dans $\mathbb{Z}/n\mathbb{Z}$ avec $n = p \times q$, un produit de deux grands nombres premiers, ce qui implique qu'il faut faire attention à l'existence d'inverses [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**].
- D'autres protocoles, comme ECC, se construisent sur des corps finis $\mathbb{Z}/p\mathbb{Z}$, où p est un grand nombre premier, pour bénéficier d'un cadre algébrique plus régulier [**Koblitz, N. (2012)**].

I.3.2 Définition du groupe multiplicatif $Z_n^{\times}$

L'ensemble des entiers premiers avec n (c'est-à-dire ayant un PGCD égal à 1 avec n) et strictement inférieurs à n forme un groupe multiplicatif mod n , noté [**Boneh, D., & Shoup, V. (2023, January)**] :

$$Z_n^{\times} = \{a \in Z \mid 1 \leq a < n \text{ et } pgcd(a,n) = 1\}$$

‘’Note:

pgcd est l'abréviation de **"plus grand commun diviseur"**, c'est-à-dire le plus grand entier qui divise deux nombres sans laisser de reste.

L'expression :

$$pgcd(a,n)=1$$

signifie que le seul diviseur commun positif entre les entiers *a* et *n* est 1. Autrement dit, *a* et *n* sont premiers entre eux (co-premiers).''

Exemple : soit $n=7$, alors :

$$Z_7^{\times}=\{1,2,3,4,5,6\}$$

Et chaque élément de cet ensemble a un inverse :

- $3\times 5\equiv 1 \, mod \, 7$
- $2\times 4\equiv 1 \, mod \, 7$

Inversibilité dans Z/nZ

Un élément $\underline{a}$ de Z/nZ est inversible s'il existe un entier $\underline{b}$ tel que :

$$\underline{a}\cdot\underline{b}=\underline{1}$$

Ce qui revient à dire que a et n sont premiers entre eux, c'est-à-dire $pgcd(a,n)=1$. . Cela joue un rôle crucial en cryptographie, car beaucoup d'algorithmes, comme RSA ou ElGamal, exigent des clés inversibles dans un anneau modulaire [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**, **Stinson, D. R. (2005)**].

Avantages pour l'implémentation embarquée

Dans les systèmes cryptographiques embarqués, notamment dans l'IoT marin, les opérations dans Z/nZ sont très avantageuses [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**, **Stinson, D. R. (2005)**] :

- **Efficacité** : les opérations comme l'addition ou la multiplication modulo *n* ont une complexité constante, ce qui les rend bien adaptées aux microcontrôleurs à ressources limitées [**Stinson, D. R. (2005)**].
- **Robustesse** : permet de modéliser des cycles, des trames périodiques ou des rotations [**Schneier, B. (1996)**].
- **Sécurité** : les propriétés algébriques des anneaux finis permettent de construire des protocoles résistants aux attaques structurelles, en masquant les comportements linéaires ou prévisibles [**Koblitz, N. (2012)**, **Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**].

I.3.3 Calcul d'un inverse modulaire

Méthode mathématique : algorithme d'Euclide étendu

Pour trouver l'inverse de $a \, mod \, n$, on cherche un entier x tel que :

$$a\cdot x\equiv 1 \, mod \, n$$

Cette équation est une équation diophantienne, et on peut la résoudre à l'aide de l'algorithme d'Euclide étendu [**Douglas Robert Stinson, & Paterson, M. B. (2019)**, **Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**]. Ce dernier permet non seulement de calculer le pgcd de deux nombres, mais aussi de retrouver les coefficients de Bézout :

$$a \cdot x + n \cdot y = p\,gcd(a, n)$$

Si $p\,gcd(a, n) = 1$, alors x est l'inverse de a modulo n.

Exemple :

Calculer l'inverse de $3 \, mod \, 7$.

- On applique l'algorithme d'Euclide étendu :
  - $7 = 2 \times 3 + 1$
  - $1 = 7 - 2 \times 3 \Rightarrow 1 = (-2) \cdot 3 + 1 \cdot 7$

Donc :

$$(-2) \cdot 3 + 1 \cdot 7 = 1 \Rightarrow -2 \cdot 3 \equiv 1 \, mod \, 7 \Rightarrow x = -2 \equiv 5 \, mod \, 7$$

L'inverse de $3 \, mod \, 7$ est 5, car $3 \cdot 5 = 15 \equiv 1 \, mod \, 7$.

I.4 – Le XOR :

L'opérateur XOR, abréviation de « exclusive OR », est sans doute l'une des opérations les plus fondamentales en cryptographie symétrique. Derrière sa simplicité logique se cache une propriété d'inversibilité élégante, qui en fait un outil central dans le chiffrement de nombreux algorithmes modernes, notamment dans les schémas de masquage de flux, les modes de chiffrement de blocs, ou les algorithmes Stream cipher [**Bernstein, D. (2008a)**].

L'opération XOR agit bit à bit, selon la règle suivante :

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Autrement dit, le XOR entre deux bits est 1 si les bits sont différents, et 0 s'ils sont identiques.

I.4.2 Une intuition mathématique et algébrique

D'un point de vue mathématique, le XOR est une opération dans le groupe ¿, où :

- L'ensemble $F_2 = \{0,1\}$.
- L'opération $+$¿ est définie modulo 2 (donc équivalente au XOR).

Cette propriété est très puissante :

$$a \oplus b = c \Leftrightarrow a = c \oplus b$$

Ce qui signifie que le XOR est **son propre inverse**, c'est-à-dire :

$$a \oplus b \oplus b = a$$

Cela rend le XOR particulièrement utile pour le chiffrement et le déchiffrement : il suffit d'appliquer la même opération avec la même clé pour retrouver le message d'origine.

I.4.3 Exemple simple de chiffrement avec XOR

Supposons qu'Alice veuille envoyer à Bob le message binaire $M = 10110110$ en le chiffrant à l'aide d'une clé $K = 01101001$.

Étape 1 – Chiffrement :

$$C = M \oplus K = 10110110 \oplus 01101001 = 11011111$$

Afin d'obtenir le message original, Bob doit déchiffrer le message reçu. Pour cela, il applique une opération XOR entre le message reçu et la clé K.

Étape 2 – Déchiffrement :

$$M = C \oplus K = 11011111 \oplus 01101001 = 10110110$$

Bob retrouve exactement le message original. Cet exemple montre que le XOR est parfaitement adapté pour chiffrer et déchiffrer, tant que la clé est identique et secrète [**Schneier, B. (1996)**].

I.5 – Comprendre la fonction indicatrice d'Euler *φ(n)*

Dans l'univers mathématique qui sous-tend la cryptographie, certaines fonctions jouent un rôle central dans la robustesse des systèmes de chiffrement. L'une des plus fondamentales est la fonction indicatrice d'Euler, notée *φ(n)*. Ce concept, bien qu'introduit il y a plusieurs siècles, continue d'alimenter les algorithmes de sécurité modernes, notamment RSA, où elle intervient dans le processus de génération des clés. Pour en mesurer la portée, commençons par en saisir la signification intuitive [**Boneh, D., & Shoup, V. (2023, January)**, **Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**].

I.5.1 Une fonction qui compte les entiers "co-premiers" à *n*

La fonction *φ(n)* désigne, pour un entier naturel positif *n*, le nombre d'entiers strictement inférieurs à *n* qui sont premiers avec lui, c'est-à-dire qui n'ont aucun diviseur commun avec *n* à part 1 [**Stallings, W. (2017)**].

Prenons un exemple simple :

> **Exemple 1 :** Soit n = 8.
>
> Les entiers strictement inférieurs à 8 sont : {1, 2, 3, 4, 5, 6, 7} Parmi eux, seuls {1, 3, 5, 7} n'ont pas de facteur commun avec 8. On a donc : φ(8) = 4.

I.5.2 Cas particulier : *n* est un nombre premier

Lorsqu'on choisit un nombre premier p, un phénomène intéressant se produit : tous les entiers de 1 à p−1 sont automatiquement premiers avec p. En effet, un nombre premier ne partage de diviseur qu'avec 1 et lui-même.

> Ainsi, pour tout p premier, on a : $\varphi(p) = p - 1$.

Cette propriété, bien que très simple, est extrêmement précieuse dans les algorithmes cryptographiques, car elle permet des simplifications élégantes dans de nombreux calculs [**Stallings, W. (2017)**].

> **Exemple 2 :** Pour p = 13, on obtient directement : $\varphi(13) = 12$, puisque 13 est premier.

I.5.3 Calcul de $\varphi(n)$ pour un nombre composé

Lorsque *n* n'est pas premier, on peut tout de même déterminer φ(n) en exploitant sa décomposition en facteurs premiers. Supposons que :

$$n = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_k^{e_k}$$

où les p_i sont des nombres premiers distincts et les e_i leurs exposants.

La fonction $\varphi(n)$ peut alors être calculée grâce à la formule suivante, issue de l'arithmétique modulaire :

$$\varphi(n) = n \cdot \left(1 - \frac{1}{p_1}\right) \cdot \left(1 - \frac{1}{p_2}\right) \cdots \left(1 - \frac{1}{p_k}\right)$$

Cette expression repose sur les propriétés structurelles du groupe multiplicatif modulo n, un concept clé en cryptographie [**Boneh, D., & Shoup, V. (2023, January)**].

> **Exemple 3:** Calculons $\varphi(20)$. On commence par factoriser : $20 = 2^2 \times 5$ Appliquons la formule :
>
> $$\varphi(20) = 20 \cdot \left(1 - \frac{1}{2}\right) \cdot \left(1 - \frac{1}{5}\right) = 20 \cdot \frac{1}{2} \cdot \frac{4}{5} = 8$$
>
> Ce résultat signifie qu'il existe exactement 8 entiers strictement inférieurs à 20 qui n'ont aucun facteur commun avec lui.

I.6 – L'usage des matrices

Bien avant que les algorithmes à clé publique ne révolutionnent la cryptographie, les chercheurs se sont appuyés sur une branche fondamentale de l'algèbre : la théorie des matrices. Derrière leur apparente simplicité, ces objets mathématiques se sont révélés particulièrement efficaces pour transformer des données de manière structurée et déterministe. Ils ont ainsi donné naissance à une classe de chiffrements appelés substitutions polyalphabétiques linéaires, ou plus simplement, chiffrements matriciels.

I.6.1 Ce que sont les matrices, et pourquoi elles intéressent la cryptographie

Les matrices sont capables de transformer une suite de lettres (ou d'octets) en une autre séquence, selon une règle mathématique précise. Cette transformation se fait dans un espace à arithmétique modulaire, c'est-à-dire où les calculs sont effectués "en boucle", modulo un

certain entier *n*. Typiquement, *n* = 26 pour l'alphabet latin (A–Z), ou *n* = 256 pour les données binaires [**Hill, L. S. (1929)**, **Joux, A. (2009)**].

L'intérêt de cette approche réside dans le fait que la transformation est réversible, à condition que la matrice utilisée soit inversible dans cet espace modulo n. Cela signifie que son déterminant doit être premier avec n, autrement dit, ne pas avoir de diviseur commun avec n, sauf 1 [**Douglas Robert Stinson, & Paterson, M. B. (2019)**, **Joux, A. (2009)**].

En cryptographie, ce type de transformation s'inscrit dans le cadre d'un chiffrement symétrique, où la même clé, ici, la matrice, est partagée entre l'expéditeur et le destinataire [**Hill, L. S. (1929)**]. Le message original est converti par multiplication avec cette matrice ; le destinataire utilise l'inverse de la matrice pour retrouver le message clair.

Bien que cette méthode ne soit plus utilisée seule dans les systèmes modernes, elle constitue une introduction essentielle à des techniques plus avancées fondées sur l'algèbre linéaire, comme celles employées dans certains composants de l'AES ou dans des mécanismes d'obscurcissement cryptographique.

I.6.2 Le chiffrement de Hill

Le chiffrement de Hill, proposé par Lester S. Hill en 1929 [**Hill, L. S. (1929)**], est l'exemple emblématique de l'usage des matrices dans les systèmes de chiffrement classiques. Il repose sur l'idée suivante :

- On représente chaque lettre d'un message par un nombre (A=0, B=1, ..., Z=25).
- On regroupe les lettres en blocs de taille n, que l'on transforme en vecteurs.
- On applique une matrice carrée de taille *n* x *n* pour transformer le vecteur, en calculant :

  $$C = A \cdot M \; mod \; 26$$

  où :

  - M est le vecteur du message clair (plaintext),
  - A est la matrice de chiffrement,
  - C est le vecteur chiffré.

Pour que le chiffrement soit réversible, la matrice A doit être inversible modulo 26, c'est-à-dire qu'il doit exister une matrice A^{-1} telle que :

$$A \cdot A^{-1} \equiv I \; mod \; 26$$

I.6.3 Exemple de chiffrement

Prenons un message simple **: "HELP"**

Étape 1 – Codage alphabétique du message

On commence par convertir chaque lettre en un entier selon la table classique : A = 0, B = 1, ..., Z = 25. Ainsi :

- H = 7,
- E = 4,
- L = 11,
- P = 15.

On regroupe ensuite ces valeurs en blocs de taille 2, ce qui donne deux vecteurs colonne :

$$M_1 = \begin{bmatrix} 7 \\ 4 \end{bmatrix} \qquad M_2 = \begin{bmatrix} 11 \\ 15 \end{bmatrix}$$

Étape 2 – Choix de la matrice de chiffrement

On choisit une **matrice carrée 2×2**, qui servira de clé de chiffrement :

$$A = \begin{vmatrix} 3 & 3 \\ 2 & 5 \end{vmatrix}$$

Cette matrice est choisie de manière à être **inversible modulo 26**, une condition nécessaire pour que le chiffrement soit réversible.

Étape 3 – Calcul du premier bloc chiffré

Le bloc chiffré C_1 est obtenu par le produit matriciel suivant :

$$C_1 = A.M_1 \ mod 26 = \begin{bmatrix} 3 & 3 \\ 2 & 5 \end{bmatrix} . \begin{bmatrix} 7 \\ 4 \end{bmatrix} = \begin{bmatrix} 3\times 7 + 3\times 4 \\ 2\times 7 + 5\times 4 \end{bmatrix} = \begin{bmatrix} 21 + 12 \\ 14 + 20 \end{bmatrix} = \begin{bmatrix} 33 \\ 34 \end{bmatrix} mod\ 26 = \begin{bmatrix} 7 \\ 8 \end{bmatrix}$$

En reprenant le codage alphabétique, on obtient : **7 = H, 8 = I** → Le premier bloc chiffré est donc : **"HI"**.

Étape 4 – Calcul du second bloc

De la même manière, on calcule :

$$C_2 = A.M_2 \ mod\ 26 = \begin{bmatrix} 3 & 3 \\ 2 & 5 \end{bmatrix} . \begin{bmatrix} 11 \\ 15 \end{bmatrix} = \begin{bmatrix} 3\times 11 + 3\times 15 \\ 2\times 11 + 5\times 15 \end{bmatrix} = \begin{bmatrix} 33 + 45 \\ 22 + 75 \end{bmatrix} = \begin{bmatrix} 78 \\ 97 \end{bmatrix} \ mod\ 26 = \begin{bmatrix} 0 \\ 19 \end{bmatrix}$$

Ce qui correspond à : **0 = A**, **19 = T** → Le second bloc chiffré est : **"AT"**

Message final chiffré :

HELP → HIAT

Ce petit exemple démontre la puissance du chiffrement matriciel, tout en soulignant l'importance de l'arithmétique modulaire et de l'inversibilité matricielle, deux piliers fondamentaux de la cryptographie classique.

I.6.4 Inversibilité et sécurité

La sécurité du chiffrement de Hill dépend de l'inversibilité de la matrice. Pour qu'une matrice 2x2 soit inversible modulo 26, son déterminant Δ doit être premier avec 26. On calcule $\Delta = ad - bc$, puis on s'assure que :

$$pgcd(\Delta, 26) = 1$$

Sinon, la matrice n'est pas inversible, et le message ne pourra pas être déchiffré. Ce principe illustre l'importance de la structure arithmétique modulaire dans la cryptographie [**Hill, L. S. (1929)**, **Joux, A. (2009)**].

I.6.5 Limites du chiffrement matriciel

Malgré son élégance, le chiffrement de Hill présente plusieurs limites :

- Il est sensible aux attaques par analyse fréquentielle si les blocs sont petits (taille 2 ou 3).
- Une attaque par texte clair connu permet de retrouver la matrice de chiffrement en résolvant un système linéaire modulaire [**Joux, A. (2009)**].
- Il n'est pas adapté à la cryptographie moderne, mais reste précieux pour comprendre des concepts clés comme l'algèbre linéaire modulaire et l'inversibilité.

I.6.6 Applications modernes des matrices

Aujourd'hui, les matrices ne sont plus au cœur des algorithmes de chiffrement modernes, mais elles réapparaissent dans d'autres contextes cryptographiques :

- Codes correcteurs d'erreurs (Reed-Solomon, Cycliques, MDS).
- Cryptographie post-quantique, avec les réseaux euclidiens et les systèmes à base de matrices aléatoires [**Joux, A. (2009)**, **Douglas Robert Stinson, & Paterson, M. B. (2019)**].
- Transformées linéaires dans AES, où la substitution et la diffusion sont exprimées en termes matriciels [**NIST. (2001)**].

I.7 – Synthèse des notions abordées

L'objectif de ce premier chapitre était de poser des bases solides, à la fois intuitives et rigoureuses, afin de permettre une compréhension approfondie des mécanismes cryptographiques que nous étudierons dans les chapitres suivants, notamment dans le contexte des systèmes embarqués marins.

Nous avons mis en évidence le rôle central des nombres premiers, dont la rareté et l'indivisibilité permettent de construire des systèmes robustes comme RSA [**Douglas Robert Stinson, & Paterson, M. B. (2019)**]. Leur factorisation constitue un problème

computationnel complexe, exploité pour sécuriser les algorithmes asymétriques. [**Stallings, W. (2017)**].

Les notions de modulo et de structure Z/nZ nous ont permis de comprendre comment l'arithmétique modulaire crée un espace fini dans lequel on peut faire des calculs essentiels pour concevoir des fonctions inversibles, des clés de chiffrement, ou des cycles de transformations cryptographiques [**Douglas Robert Stinson, & Paterson, M. B. (2019)**]. La fonction d'Euler $\varphi(n)$ a enrichi cette structure en quantifiant le nombre d'entiers inférieurs à n et premiers avec lui, une pièce maîtresse dans les formules comme celles d'Euler ou Fermat, appliquées à RSA [**Stallings, W. (2017)**].

Nous avons également introduit une opération élémentaire, mais omniprésente dans la cryptographie symétrique : le XOR (ou OU exclusif). Cette opération logique constitue la pierre angulaire de nombreux algorithmes, car elle offre une réversibilité naturelle et une bonne diffusion des bits, tout en étant extrêmement rapide à implémenter matériellement [**Stallings, W. (2017)**, **Smart, N. P. (2016)**].

Les matrices, quant à elles, permettent de transformer des blocs d'information par des combinaisons linéaires contrôlées. Même si les chiffrements comme celui de Hill ne sont plus utilisés en pratique pour des raisons de sécurité, leur compréhension permet de saisir la logique interne des opérations de substitution et de diffusion présentes dans les algorithmes modernes (ex : AES, S-box, transformations linéaires, etc.) [**NIST. (2001)**, **Washington, L. C. (2008)**].

Un socle indispensable pour la suite

Ces notions, loin d'être de simples outils mathématiques isolés, forment un vocabulaire commun indispensable pour aborder la cryptographie appliquée. Elles constituent le socle sur lequel reposent des algorithmes aussi divers que :

- Le chiffrement symétrique (AES, ChaCha20) [**NIST. (2001)**],
- Le chiffrement asymétrique (RSA, ECC) [**Rivest, R. L., Shamir, A., & Adleman, L. (1978)**, **Stallings, W. (2017)**],
- Les fonctions de hachage (SHA-2, SHA-3) [**Douglas Robert Stinson, & Paterson, M. B. (2019)**],
- Les signatures numériques [**Smart, N. P. (2016)**],
- Et plus globalement, les protocoles cryptographiques sécurisés que nous retrouverons dans des systèmes contraints comme les réseaux de capteurs sous-marins, les gliders autonomes, ou encore l'Internet des objets marins (MIoT) [**Joux, A. (2009)**].

Chapitre II : Fondamentaux de la cryptographie

II.1 – Origines et principes fondamentaux

La cryptographie, au sens le plus large, désigne l'ensemble des techniques visant à protéger une information contre l'accès non autorisé, en la rendant inintelligible à toute personne ne possédant pas les moyens légitimes pour la lire. Cette pratique ne date pas d'hier : son histoire remonte à l'Antiquité.

Les Égyptiens, par exemple, gravaient déjà des messages à contenu symbolique dans leurs hiéroglyphes, dont certains étaient volontairement obscurs. Les Grecs, eux, utilisaient la scytale, un dispositif simple composé d'un cylindre autour duquel on enroulait une bande de cuir pour dissimuler un message. L'un des systèmes les plus emblématiques reste cependant le chiffre de César, attribué à Jules César, qui utilisait un décalage alphabétique pour transmettre des ordres codés à ses troupes [**Kahn, D. (1996, December 5)**].

Avec le temps, cette pratique longtemps intuitive et artisanale s'est transformée en discipline scientifique rigoureuse. Les intuitions de l'Antiquité ont progressivement laissé place à des fondations solides, reposant sur des domaines mathématiques tels que la théorie des nombres, l'algèbre, et plus récemment, sur la complexité computationnelle.

Aujourd'hui, la cryptographie ne se limite plus à l'espionnage ou à la guerre : elle est devenue un pilier de la cybersécurité moderne, protégeant les échanges numériques du quotidien aussi bien que les systèmes d'information stratégiques, les transactions financières, ou encore les infrastructures critiques.

> « La cryptographie moderne est moins une affaire de cacher que de prouver que ce qui est caché ne peut être retrouvé sans l'autorisation voulue. » Bruce Schneier [**Schneier, B. (1996)**]

Définitions clés :

Terme	Définition
Message en clair (M)	Donnée originale, lisible, avant tout traitement cryptographique.
Message chiffré (C)	Version transformée du message en clair, conçue pour être incompréhensible sans la clé.
Clé (K)	Donnée secrète utilisée dans les opérations de chiffrement ou de déchiffrement.
Chiffrement (E)	Fonction mathématique transformant un message en clair en message chiffré : $C = E_K(M)$
Déchiffrement (D)	Fonction mathématique restituant le message original : $M = D_K(C)$ $M = D_K(C)$
Algorithme de chiffrement	Procédé mathématique spécifique pour chiffrer

Terme	Définition
	un message à l'aide d'une clé.
Algorithme de déchiffrement	Procédé inverse pour retrouver le message en clair.
Cryptosystème	Ensemble structuré de techniques cryptographiques (algorithmes, clés, protocoles) visant à fournir une ou plusieurs propriétés de sécurité (confidentialité, intégrité, etc.).

II.2 – Objectifs fondamentaux de la cryptographie

La cryptographie, dans ses fondements modernes, ne se limite pas à dissimuler des secrets. Elle est devenue un outil mathématique sophistiqué destiné à garantir des propriétés de sécurité précises dans les échanges numériques. On identifie généralement quatre objectifs principaux, chacun répondant à un besoin spécifique des systèmes informatiques sécurisés : la confidentialité, l'intégrité, l'authenticité et la non-répudiation [**Schneier, B. (1996)**].

- ### Confidentialité : Protéger le secret des données

La confidentialité assure que seules les personnes autorisées puissent accéder à une information. En d'autres termes, même si un adversaire intercepte le message, il ne doit rien pouvoir en déduire.

Prenons un exemple classique : Alice souhaite envoyer un message secret à Bob, mais Eve (l'espionne) peut écouter la communication. Pour protéger son message, Alice applique un algorithme de chiffrement à son message en utilisant une clé secrète K. Ce message chiffré est envoyé à Bob, qui utilise la même clé (dans un système symétrique) ou sa clé privée (dans un système asymétrique) pour effectuer l'opération inverse.

Ce processus garantit que l'information reste hors de portée d'Eve, même si elle intercepte le message chiffré, tant qu'elle ne connaît pas la clé secrète K [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**].

- ### Intégrité : Préserver l'exactitude des données

L'intégrité vise à garantir que l'information reçue est exactement celle qui a été envoyée — sans modification, volontaire ou accidentelle. Elle est essentielle dans les systèmes critiques, comme ceux déployés en mer, où une donnée altérée sur la température de l'eau ou la pression pourrait entraîner des décisions erronées du système embarqué.

Pour cela, on utilise généralement des fonctions de hachage cryptographiques (ex. SHA-2, SHA-3), qui transforment une donnée M en une empreinte unique $h=H(M)$. Cette empreinte est transmise avec le message. Si le message est modifié en cours de route, l'empreinte calculée par le récepteur ne correspondra plus, signalant une altération.

- **Authenticité : Identifier l’expéditeur légitime**

L’authenticité garantit que le message provient bien de l’expéditeur revendiqué. Cela empêche les attaques où un adversaire, tel qu’Eve, enverrait un faux message se faisant passer pour Alice.

Dans les systèmes symétriques, cela se fait par un MAC (Message Authentication Code) basé sur une clé partagée. Dans les systèmes asymétriques, on utilise des signatures numériques : Alice signe le message avec sa clé privée, et Bob vérifie la signature avec sa clé publique [**National Institute of Standards and Technology (NIST). (2013)**].

Dans un environnement marin, cela est critique pour assurer que les données reçues par une bouée, un glider ou un AUV ne soient pas des injections malveillantes.

- **Non-répudiation : Éviter le déni d’action**

La non-répudiation assure qu’un expéditeur ne peut pas nier avoir envoyé un message. Elle repose principalement sur les signatures numériques, qui lient mathématiquement l'identité de l'expéditeur à l'information.

Si Alice a signé un document électronique avec sa clé privée, elle ne peut pas ensuite prétendre qu'elle ne l'a pas fait, puisque personne d’autre n’a accès à cette clé. Ce principe est utilisé, par exemple, pour valider des transactions maritimes à distance, ou l’émission de rapports environnementaux automatisés [**Perrig, A., et al. (2002)**].

II.3 – Cryptographie symétrique

Imaginez deux chercheurs océanographes, Alice et Bob, communiquant entre une station côtière et un robot sous-marin. Leur message doit rester confidentiel, même s’il traverse des canaux radio exposés à d’éventuelles écoutes. Pour cela, ils partagent un secret : une même clé numérique, connue d’eux seuls. Grâce à cette clé, ce que l’un chiffre, l’autre peut déchiffrer. Ce principe, d’une simplicité presque élégante, est au cœur de ce qu’on appelle la cryptographie symétrique [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**].

La force de cette approche réside dans son efficacité : les opérations sont rapides, peu coûteuses en énergie, et facilement intégrables dans du matériel limité [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**]. C’est précisément ce qui la rend précieuse dans les systèmes embarqués marins, où les microcontrôleurs sont peu puissants, les communications intermittentes, et la fiabilité primordiale.

Cependant, cette efficacité cache un défi majeur : le partage sécurisé de la clé. Car si cette dernière venait à être interceptée, toute la confidentialité du système serait compromise [**Stallings, W. (2017)**]. C’est donc un équilibre subtil entre simplicité opérationnelle et sécurité cryptographique qu’il faut maintenir.

Au fil de cette section, nous allons explorer les fondements de la cryptographie symétrique, du classique AES au plus léger ChaCha20 [**Douglas Robert Stinson, & Paterson, M. B. (2019)**]. L’objectif n’est pas seulement de comprendre comment ces outils fonctionnent, mais aussi pourquoi ils continuent de jouer un rôle central dans la sécurité.

II.3.1 Comprendre la cryptographie symétrique : principe et mécanisme

La cryptographie dite *symétrique* regroupe un ensemble de techniques dans lesquelles une même clé secrète est utilisée à la fois pour chiffrer et pour déchiffrer un message [**Douglas Robert Stinson, & Paterson, M. B. (2019)**]. Ce modèle suppose qu'un lien de confiance préalable existe entre les deux parties qui communiquent, celles-ci devant avoir échangé cette clé confidentielle en amont, et ce, via un canal sécurisé.

Sur le plan formel, si l'on note M le message clair, C le texte chiffré, et K la clé partagée, le processus de chiffrement peut s'exprimer par la relation suivante :

$$C = E_K(M) \, et \, M = D_K(C)$$

où E_K représente la fonction de chiffrement, et D_K la fonction de déchiffrement, toutes deux dépendantes de la même clé secrète K [**Douglas Robert Stinson, & Paterson, M. B. (2019)**].

Cette approche, relativement directe, présente plusieurs atouts notables :

- Elle est rapide à exécuter,
- Elle consomme peu d'énergie,
- Et elle s'adapte facilement à des matériels embarqués limités [**Stallings, W. (2017)**].

Ces caractéristiques en font une solution idéale pour des environnements contraints, comme ceux des réseaux de capteurs océaniques, des bouées autonomes ou des systèmes de surveillance marine embarqués [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**, **Stallings, W. (2017)**].

> **Exemple** : Imaginons un capteur sous-marin chargé de mesurer la température dans une zone environnementale sensible. Ce dispositif chiffre les données qu'il collecte à l'aide d'une clé K déjà intégrée lors du déploiement. Une fois le message transmis à la station de contrôle, celle-ci utilise la même clé K pour retrouver l'information originale. Comme aucune clé n'est transmise pendant l'échange, le risque d'interception en cours de route est significativement réduit [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**].

> **‘’Note : Fonctionnement des capteurs sous-marin :**
>
> Un capteur sous-marin est un dispositif autonome destiné à mesurer certains paramètres physico-chimiques ou biologiques dans les milieux aquatiques, en immersion prolongée. Il peut être embarqué sur des structures fixes (stations d'observation, câbles sous-marins), des plateformes mobiles (AUVs, gliders), ou encore associé à des bouées intelligentes. Ces capteurs intègrent généralement une sonde spécifique (température, pression, salinité, oxygène dissous, bruit ambiant, etc.) couplée à une unité de traitement embarquée, capable d'effectuer un pré-traitement des données et parfois même un chiffrement local. Les données sont soit stockées en mémoire, soit transmises à distance, via des moyens adaptés à l'environnement : ondes acoustiques pour la transmission sous l'eau, liaisons radio ou satellite en surface, ou encore relais via des bouées. Ces systèmes sont le plus souvent alimentés par batterie et conçus pour fonctionner de manière

énergétiquement efficace, avec des ressources de calcul très limitées. Cela en fait des cibles vulnérables à l'interception, à la falsification ou à l'usurpation d'identité. D'où la nécessité d'employer des algorithmes de chiffrement symétriques légers et robustes (comme AES, ChaCha20 ou PRESENT) pour assurer la confidentialité, l'intégrité et l'authenticité des données qu'ils transmettent.''

Cependant, ce modèle soulève également une problématique critique : comment s'assurer que la clé secrète est transmise de façon sûre dès le départ ? Si un attaquant parvient à intercepter cette clé lors de son partage initial, il devient capable non seulement de lire tous les messages, mais aussi d'en injecter de faux, se faisant passer pour un dispositif légitime [**Stallings, W. (2017)**]. Ce point faible du canal de distribution de la clé constitue l'une des limites majeures de la cryptographie symétrique, et justifie l'existence de méthodes complémentaires basées sur des schémas asymétriques, que nous aborderons plus loin.

Enfin, dans un système bien conçu, la sécurité d'un algorithme symétrique ne dépend pas uniquement du secret de la clé, mais également de la robustesse de l'algorithme de chiffrement utilisé. Des standards comme AES (Advanced Encryption Standard) ou ChaCha20 ont fait l'objet d'évaluations approfondies par la communauté scientifique et sont aujourd'hui largement employés pour garantir la sécurité des données [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**, **Douglas Robert Stinson, & Paterson, M. B. (2019)**].

II.3.2 Principes de fonctionnement d'un algorithme symétrique

Dans un algorithme de cryptographie symétrique, le cœur du mécanisme repose sur une clé secrète partagée qui permet à la fois de chiffrer et de déchiffrer un message. Contrairement aux systèmes asymétriques où deux clés distinctes (publique et privée) sont utilisées, la cryptographie symétrique mise sur la simplicité algorithmique, la rapidité de traitement, et la légèreté computationnelle, des caractéristiques précieuses dans les environnements embarqués marins [**Schneier, B. (1996)**, **Stallings, W. (2017)**].

Étapes générales d'un chiffrement symétrique

Le fonctionnement typique d'un algorithme symétrique suit les étapes suivantes :

1. **Génération et partage de la clé secrète** : Avant toute communication, les deux parties (par exemple Bob et Alice) doivent disposer d'une même clé secrète K, qui est générée à l'aide de générateurs pseudo-aléatoires et transmise de manière sécurisée. Cette étape est cruciale : si la clé est compromise, toute la confidentialité du système est remise en question [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**].
2. **Préparation du message** : Le message original (ou texte en clair, noté M) est souvent découpé en blocs de taille fixe (ex. 128 bits) ou traité bit par bit [**Stallings, W. (2017)**].
3. **Chiffrement** : Le texte en clair M est transformé en un texte chiffré C, via une fonction de chiffrement E et la clé K :

$$C = E_K(M)$$

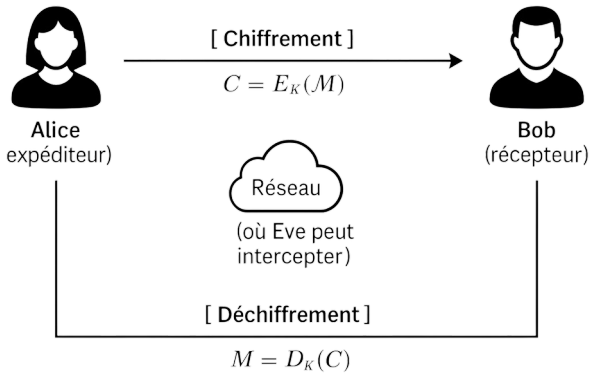
Cette transformation doit être déterministe, mais également non triviale : sans la clé, il doit être extrêmement difficile, voire impossible, de retrouver M à partir de $C[Menezes, A., C. van Oorschot, P., \wedge Vanstone, S. A. (2018)]$.

4. **Transmission sur un canal potentiellement non sécurisé** : Le message chiffré C peut alors être transmis à travers des canaux ouverts, comme une liaison radio marine, sans crainte que son contenu ne soit intercepté en clair [**Schneier, B. (1996)**].
5. **Déchiffrement** : Le destinataire (ex. Alice) utilise la même clé secrète K pour retrouver le message original :

$$M = D_K(C)$$

où D est la fonction de déchiffrement correspondant à $E[Stallings, W. (2017)]$.

Schéma explicatif : communication sécurisée entre Alice et Bob



Ce schéma illustre le rôle fondamental de la cryptographie symétrique : même si une espionne comme Eve intercepte le message chiffré, elle ne pourra en comprendre le contenu sans connaître la clé appropriée.

Caractéristiques essentielles

Un bon algorithme symétrique doit remplir plusieurs critères fondamentaux :

- **Confidentialité** : la sécurité repose uniquement sur la protection de la clé K, et non sur la dissimulation de l'algorithme [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**].
- **Inversibilité maîtrisée** : seule la personne disposant de la clé peut inverser le chiffrement.
- **Résistance aux attaques** : le texte chiffré doit apparaître aléatoire et ne pas révéler d'information sur le message original [**Schneier, B. (1996)**].
- **Efficacité** : les algorithmes symétriques doivent fonctionner avec un coût faible en ressources, ce qui les rend particulièrement adaptés aux capteurs marins, drones sous-

marins autonomes et autres systèmes embarqués à faible puissance de calcul [**Stallings, W. (2017)**].

Exemple concret de chiffrement symétrique en milieu océanique

Prenons un scénario concret : Bob souhaite transmettre à Alice une série de mesures environnementales enregistrées par un capteur océanique, telles que la température de l'eau ou sa salinité. Avant l'envoi, ces données sont chiffrées à l'aide d'une clé secrète K, déjà embarquée dans le dispositif de collecte. Une fois le message reçu, Alice, qui possède elle aussi la même clé K, est en mesure de déchiffrer le contenu et de retrouver les données originales. En revanche, tout individu interceptant la transmission sans connaître la clé ne pourra ni interpréter ni exploiter les informations [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**].

Exemple :

Imaginons que Bob, opérateur d'un drone marin autonome chargé de collecter des données environnementales, souhaite transmettre à Alice, responsable à la station côtière, la mesure suivante : **M = "TEMP:23°C"**

Pour garantir la confidentialité de l'échange, Bob et Alice conviennent à l'avance d'une clé secrète partagée, de même longueur que le message, générée lors de la mise en service de l'équipement. Par exemple : **K = "z7XA,1L&C"**

Le principe du chiffrement par XOR consiste à combiner chaque caractère du message avec celui de la clé, bit à bit, selon la table ASCII. Ce procédé illustre la simplicité et la réversibilité du chiffrement symétrique.

1. Encodage ASCII des caractères

Chaque caractère du message et de la clé est converti en binaire (ASCII) :

Message (M)	ASCII (décimal)	ASCII (binaire)	Clé (K)	ASCII (décimal)	ASCII (binaire)
T	84	01010100	z	122	01111010
E	69	01000101	7	55	00110111
M	77	01001101	X	88	01011000
P	80	01010000	A	65	01000001
:	58	00111010	,	44	00101100
2	50	00110010	1	49	00110001
3	51	00110011	L	76	01001100
°	176	10110000	&	38	00100110
C	67	01000011	C	67	01000011

2. Chiffrement par XOR (bit à bit)

Bob applique l'opérateur XOR entre chaque octet du message et de la clé :

M (binaire)	K (binaire)	Résultat (XOR)	ASCII (chiffré)
01010100	01111010	00101110	.
01000101	00110111	01110010	r
01001101	01011000	00010101	$^{N}_{AK}$
01010000	01000001	00010001	$^{D}_{C1}$
00111010	00101100	00010110	$^{S}_{YN}$
00110010	00110001	00000011	$^{E}_{TX}$
00110011	01001100	01111111	$^{D}_{EL}$
10110000	00100110	10010110	–
01000011	01000011	00000000	$^{N}_{UL}$

Le message chiffré obtenu est une suite de caractères non lisibles, typique d'un texte protégé.

3. Transmission

Ce texte chiffré est alors transmis via le canal radio du drone. Même si le message est intercepté, il demeure incompréhensible sans la clé.

4. Déchiffrement (réversibilité du XOR)

Arrivée à la station côtière, Alice reçoit le texte chiffré. Elle applique la même clé, caractère par caractère, grâce au même opérateur XOR : Chaque octet du texte chiffré, combiné à celui de la clé, restitue l'octet d'origine, permettant à Alice de retrouver instantanément le message : **"TEMP:23°C"**

Résumé pédagogique

Cet exemple illustre le principe du chiffrement symétrique : la sécurité repose entièrement sur le secret de la clé, qui doit être utilisée une seule fois et protégée. Dans la réalité, les systèmes marins modernes privilégient l'usage d'algorithmes avancés (comme AES ou ChaCha20) pour une gestion plus flexible et une robustesse accrue [**Stallings, W. (2017)**, 55].

II.3.3 Classes de chiffrements symétriques

La cryptographie symétrique englobe une diversité d'algorithmes, chacun se distinguant par sa structure interne, ses performances et sa capacité à s'adapter à des environnements techniques contraints, comme ceux des systèmes embarqués. On peut regrouper ces algorithmes en deux grandes catégories principales : les chiffrements par blocs et les chiffrements par flot (en anglais, *block ciphers* et *stream ciphers*).

Bien qu'ils partagent le même objectif, protéger la confidentialité des données, ces deux approches diffèrent fondamentalement dans la manière dont elles utilisent la clé secrète pour transformer le message en clair [**Stallings, W. (2017)**].

1. Chiffrement par blocs

Dans un chiffrement par blocs, le message est découpé en segments de taille fixe (généralement 64 ou 128 bits), appelés blocs, qui sont chiffrés un à un. Chaque bloc subit une série d'opérations algébriques complexes (substitutions, permutations, mélanges, etc.) pour produire un bloc chiffré.

Les plus célèbres :

1. **AES (Advanced Encryption Standard)** : standard international de chiffrement symétrique, très utilisé pour la protection des données sensibles dans les systèmes informatiques, y compris embarqués [**NIST. (2001)**].
2. **DES (Data Encryption Standard)** : aujourd'hui obsolète, mais historiquement important [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**].

Modes de fonctionnement : Les chiffrements par blocs nécessitent souvent un mode d'opération (CBC, ECB, CTR, etc.) pour chiffrer des messages plus longs qu'un bloc unique et pour introduire de l'aléa. Ces modes influencent fortement la sécurité et la diffusion des erreurs dans les messages [**Stallings, W. (2017)**].

2. Chiffrement par flot

Contrairement aux chiffrements par blocs, qui opèrent sur des ensembles de données fixes, le chiffrement par flot agit de manière plus granulaire : il transforme les données bit à bit ou octet par octet. Pour chaque élément du message, l'algorithme produit un bit de clé pseudo-aléatoire, appelé *keystream*, qui est ensuite combiné au message en clair par l'opération XOR (ou addition modulo 2).

Ce fonctionnement progressif offre plusieurs avantages :

- Il permet de commencer à chiffrer ou à déchiffrer dès la première donnée reçue, sans attendre l'ensemble du message.
- Il consomme peu de mémoire et de puissance, ce qui en fait une solution idéale pour les systèmes embarqués à ressources limitées, comme les capteurs marins ou les bouées intelligentes [Bernstein, D. (2008a)].

Exemple :

Imaginons qu'Alice veuille envoyer à Bob le message suivant : Message clair (en binaire) : `10101011` Elle utilise un flux de clés généré par un algorithme de chiffrement par flot : Keystream (en binaire) : `11000110` Elle applique l'opération XOR, bit à bit :

```
  10101011   ← message clair
⊕ 11000110   ← flux de clés
  --------
  01101101   ← message chiffré
```

Bob, de son côté, dispose du même flux de clés, qu'il utilise pour déchiffrer :

```
  01101101   ← message chiffré reçu
⊕ 11000110   ← même keystream
```

```
--------
10101011   ← message clair retrouvé
```

Cette opération est symétrique : le même XOR appliqué deux fois avec la même clé restaure le message original.

Ce type de chiffrement est notamment mis en œuvre dans des algorithmes modernes comme ChaCha20, réputé pour sa rapidité, sa robustesse, et son efficacité sur des matériels embarqués contraints. C'est l'une des raisons pour lesquelles il est recommandé dans les contextes marins, où la transmission se fait souvent en temps réel, avec des ressources limitées [Bernstein, D. (2008a)].

3. Comparaison et choix contextuel

Caractéristique	Chiffrement par blocs	Chiffrement par flot
Granularité	Bloc (64 ou 128 bits)	Bit ou octet
Mode de traitement	Par blocs (nécessite mode)	Flux continu
Robustesse	Très élevée (ex : AES)	Variable selon l'algorithme
Complexité de mise en œuvre	Plus élevée	Simpler et plus rapide
Adapté aux systèmes contraints	Moyennement	Oui (notamment ChaCha20)
Exemples	AES, Camellia	ChaCha20, Grain, Trivium

4. Enjeux pour l'embarqué marin

Dans les environnements marins, où la mémoire embarquée, l'autonomie énergétique, et la bande passante sont des ressources critiques, le choix entre chiffrement par blocs ou par flot dépend fortement du contexte d'usage :

- **Pour les transmissions continues** (capteurs, télémétrie), un algorithme léger à flot est généralement préférable [Bernstein, D. (2008a)].
- **Pour le stockage sécurisé** ou l'enregistrement de journaux de bord, un chiffrement par blocs (ex. AES-128 en mode CTR) reste un choix robuste et standardisé [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**].

II.3.4 Algorithmes de chiffrement symétrique

À mesure que les exigences de sécurité se sont accrues dans les systèmes numériques modernes, en particulier dans les environnements contraints comme les dispositifs embarqués, la cryptographie symétrique a connu des progrès considérables. Deux algorithmes s'imposent aujourd'hui comme références : AES, devenu la norme incontournable dans de

nombreux secteurs, et ChaCha20, une solution plus récente et particulièrement bien adaptée aux systèmes légers, notamment en milieu maritime [**Stallings, W. (2017)**, **Bernstein, D. (2008a)**].

1) AES – Une pierre angulaire de la cryptographie moderne

Depuis son adoption officielle par le NIST en 2001 sous le nom de FIPS 197, l'Advanced Encryption Standard (AES) est devenu le standard industriel en matière de chiffrement symétrique [**NIST. (2001)**]. Il repose sur l'algorithme Rijndael, conçu par les cryptographes belges Joan Daemen et Vincent Rijmen, dont les travaux ont marqué un tournant dans l'histoire de la cryptographie [**Daemen, J., & Rijmen, V. (2002)**].

Principes de fonctionnement

AES est un chiffrement par blocs : il traite les données par blocs fixes de 128 bits, en utilisant des clés symétriques de 128, 192 ou 256 bits. Le processus de chiffrement s'effectue à travers une série d'étapes répétées appelées tours (10, 12 ou 14 selon la taille de la clé), impliquant des opérations de substitution, de permutation, de transformation matricielle (MixColumns), et d'ajout de clé secrète à chaque tour [**Stallings, W. (2017)**].

Avantages majeurs

- **Solidité cryptographique** : À ce jour, aucune attaque pratique n'a compromis la sécurité d'AES, ce qui en fait un des algorithmes les plus fiables [**Stallings, W. (2017)**].
- **Performances élevées** : AES est optimisé matériellement sur la plupart des architectures modernes, notamment grâce aux instructions spécialisées AES-NI, disponibles sur de nombreux processeurs.
- **Adoption universelle** : Il est intégré à des standards de sécurité essentiels comme TLS, IPsec, ou encore dans les cartes bancaires et les documents d'identité électroniques (passeports biométriques, etc.).

Limites d'usage

- **Moins efficace sur matériel contraint** : Sans accélération matérielle, AES peut s'avérer lourd à exécuter sur des **microcontrôleurs** ou des systèmes embarqués à ressources limitées.
- **Rigidité du mode par blocs** : Dans les applications nécessitant un flux de données continu (streaming), le fonctionnement par blocs impose une gestion complexe ou nécessite l'adaptation via des modes comme CTR ou CFB [**NIST. (2001)**].

2) DES – Le prédécesseur emblématique d'AES

Avant l'émergence d'AES, un autre algorithme avait dominé le monde de la cryptographie symétrique pendant plusieurs décennies : le Data Encryption Standard, plus connu sous le nom de DES. Mis en place par le NIST, il s'agissait du tout premier standard cryptographique civil officiel aux États-Unis [**National Bureau of Standards. (1988, January 22)**].

Fonctionnement de DES

DES est un chiffrement par blocs qui opère sur des blocs de 64 bits avec une clé de 56 bits (bien que la clé soit encodée sur 64 bits, 8 bits sont réservés à la détection d'erreurs).

Le cœur de DES repose sur une structure en 16 tours appelée réseau de Feistel, où à chaque tour :

- le bloc de données est divisé en deux moitiés (gauche et droite),
- une fonction de transformation non linéaire est appliquée à l'une des moitiés,
- puis elle est combinée à l'autre moitié à l'aide de l'opération XOR.

Chaque tour utilise une sous-clé dérivée de la clé principale via un schéma de permutation et de décalage. À la fin du processus, les moitiés sont recombinées pour produire le bloc chiffré.

Faiblesses connues

Bien que novateur à son époque, DES est aujourd'hui considéré comme obsolète :

- **Clé trop courte** : avec seulement 56 bits utiles, DES est vulnérable aux attaques par force brute modernes.
- **Sensibilité aux attaques différentielles et linéaires**.

Il a été officiellement retiré en 2005 et remplacé par AES, bien que des variantes comme Triple-DES (3DES) soient encore en usage dans certains systèmes hérités [**Stallings, W. (2017)**, **National Bureau of Standards. (1988, January 22)**].

Exemple :

Pourquoi des modes de chiffrement ?

DES, comme AES, est un chiffrement par blocs. Cela signifie qu'il ne traite que des blocs de taille fixe. Pour chiffrer un message plus long, on doit combiner plusieurs blocs successifs à l'aide d'un mode opératoire, tel que le mode CBC.

Mode CBC – Enchaînement des blocs par chaînage

En CBC (Cipher Block Chaining), chaque bloc de texte clair est XORé avec le bloc chiffré précédent avant d'être chiffré lui-même.

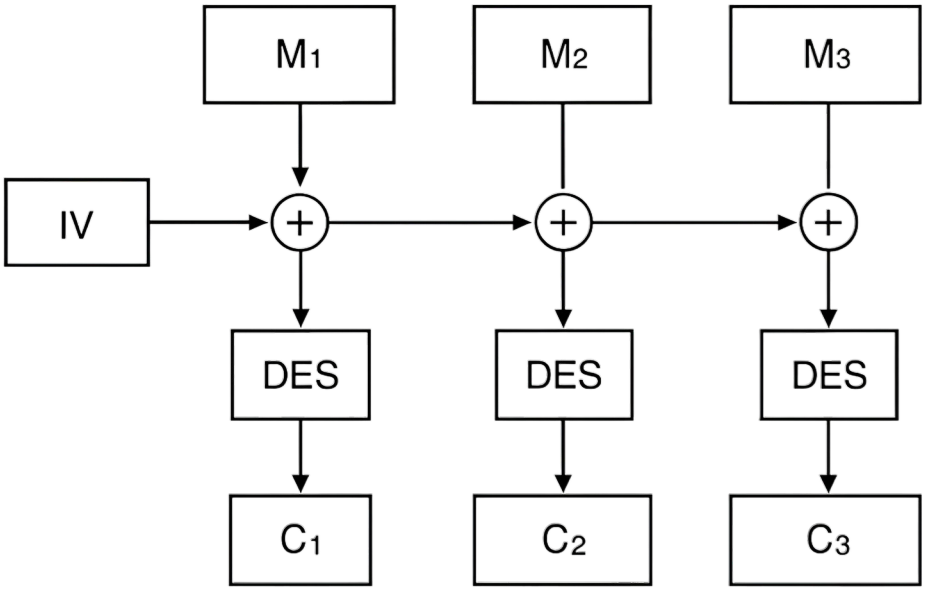
Le tout premier bloc, ne disposant pas d'un bloc chiffré précédent, est combiné avec une valeur aléatoire initiale appelée IV (*initialisation vector*).

Schéma du processus CBC :

Voici une illustration pédagogique du fonctionnement du mode CBC :

Illustration :

Message clair découpé:



- ⊕ : opération XOR (exclusif)
- **IV** : vecteur d'initialisation (aléatoire et connu des deux parties)
- **M1, M2, M3** : blocs de texte clair
- **C1, C2, C3** : blocs chiffrés

Exemple chiffré simplifié avec DES et CBC

Prenons un message en clair : M = "IMAGINE" (56 bits, soit un bloc pour DES).

Étapes :

1. **Message** : supposons que M soit converti en binaire : M = 01001001 01001101 01000001 01000111 01001001 01001110 01000101 (7 caractères ASCII → 56 bits)
2. **IV** (vecteur d'initialisation) : IV = 00011100 10101010 ... (64 bits aléatoires)
3. **Opération CBC** :
    - Bloc clair M1 est XORé avec IV → donne un bloc intermédiaire.
    - Ce bloc est ensuite chiffré avec DES → résultat : C1.
    - Bloc suivant (M2, si présent) est XORé avec C1, puis chiffré → C2.

Avantages du CBC :

- Renforce la sécurité : deux blocs identiques en clair ne donneront pas le même résultat chiffré, car chacun dépend du bloc précédent.
- Résiste mieux aux attaques que le mode ECB (Electronic Codebook).

Inconvénients

- **Pas parallélisable** : chaque bloc dépend du précédent, ce qui rend le traitement séquentiel.
- **Sensibilité à la perte de synchronisation** : une erreur dans un bloc se propage au suivant.

2) ChaCha20

Conçu par Daniel J. Bernstein, ChaCha20 est un chiffrement par flot à haute performance, pensé pour corriger certaines faiblesses de son prédécesseur Salsa20. Il s'impose aujourd'hui comme une solution de choix dans les environnements embarqués où les ressources sont limitées [Bernstein, D. (2008a)].

Fonctionnement :

ChaCha20 génère un flux pseudo-aléatoire de 512 bits, divisé en mots de 32 bits, qui est ensuite XORé avec le message clair. Le cœur de l'algorithme repose sur une série de transformations additives, XOR, et rotations (*ARX operations*) sur une matrice interne de 4×4 mots. Cette structure évite l'utilisation de tables S-box, ce qui le rend moins vulnérable aux attaques par canaux auxiliaires [Bernstein, D. (2008a)].

Forces :

- **Excellente performance logicielle**, même sans accélération matérielle.
- **Simplicité d'implémentation** et faible empreinte mémoire, idéales pour l'IoT marin [Bernstein, D. (2008a)].
- **Sécurité élevée**, validée par des années d'analyse cryptographique et recommandée par Google, OpenSSH et le protocole TLS/1.3 [**Nir, Y., & Langley, A. (2015)**].

Limites :

- Moins répandu dans les systèmes certifiés ou réglementés (car non standardisé FIPS) [**Nir, Y., & Langley, A. (2015)**].
- Nécessite un bon générateur de nonce pour garantir la sécurité.

II.3.5 Comparaison AES vs ChaCha20

Le choix entre AES et ChaCha20 repose sur plusieurs critères techniques et opérationnels. Les deux algorithmes sont robustes, largement utilisés, mais présentent des structures, des performances et des niveaux d'adaptation très différents aux contraintes de terrain. Le tableau ci-dessous synthétise les principales différences entre ces deux figures de proue de la cryptographie symétrique moderne.

Critère	**AES (Advanced Encryption Standard)**	**ChaCha20**
Type d'algorithme	Chiffrement par bloc (128 bits) [**NIST. (2001)**]	Chiffrement par flot (stream cipher) [Bernstein, D. (2008a)]
Structure interne	Substitution-permutation network (SPN) **[NIST. (2001).]**	ARX : Addition-Rotation-XOR [Bernstein, D. J. (2019)]
Longueurs de clé supportées	128, 192, 256 bits [**NIST. (2001)**]	256 bits uniquement [Bernstein, D. (2008a)]
Résistance aux attaques connues	Très sûr, mais vulnérable aux attaques par canal auxiliaire (side-channel) mal gérées [Mangard, S., Oswald, E., & Popp, T. (2007]	Résilience accrue aux attaques par timing/canal auxiliaire [**Meijer, C., et al**. **(2021)**]
Vitesse sur plateformes emb**2021**arquées	Bonne si support matériel (ex : AES-NI), sinon plus lente [**Gueron, S. (2010).**]	Très performant même sans accélération matérielle [**Meijer, C., et al. (2021)]**
Support matériel	Excellente prise en charge (processeurs modernes, IoT hardware sécurisé) [**Gueron, S. (2010)**	Support logiciel optimisé, peu de matériel dédié [Agosta, G., et al. (2015).]
Adapté aux systèmes contraints	Peut être lourd sans optimisation spécifique [**Koc, C. K., & Paar, C. (2003)]**	Conçu pour être rapide et léger en ressources [Bernstein, D. (2008a)]
Popularité / standardisation	Norme NIST (FIPS 197), utilisée mondialement [**NIST. (2001)**]	Standardisé par l'IETF, utilisé dans TLS 1.3, WireGuard, OpenSSH [**Rescorla, E. (2018)**]
Applications typiques	Stockage sécurisé, VPN, chiffrement de disque, fichiers [**Mangard, S., Oswald, E., & Popp, T. (2007)**]	IoT, mobiles, trafic réseau temps réel, VPN modernes [**Rescorla, E. (2018)**]

Analyse contextuelle

Le choix d'un algorithme de chiffrement n'est jamais anodin : il doit répondre à des contraintes spécifiques de performance, de consommation énergétique et de compatibilité matérielle. C'est dans ce contexte que l'AES et ChaCha20 occupent deux positions complémentaires.

D'un côté, AES demeure une solution de référence dès lors que le matériel embarqué bénéficie d'une accélération matérielle dédiée, comme les extensions AES-NI intégrées aux processeurs Intel, ou encore certaines puces cryptographiques spécialisées. Cette optimisation matérielle permet d'atteindre des vitesses de chiffrement très élevées, tout en maintenant une consommation d'énergie raisonnable. Ce sont ces qualités qui expliquent sa large adoption

dans les systèmes certifiés, les protocoles standardisés, et les infrastructures industrielles de cybersécurité [**Gueron, S. (2010)**], [**Rescorla, E. (2018)**].

De l'autre côté, dans les contextes plus contraints, capteurs océaniques autonomes, balises LoRa à longue portée, ou microcontrôleurs à très faible consommation, c'est ChaCha20 qui s'impose naturellement. Contrairement à AES, il ne nécessite aucun support matériel particulier pour offrir d'excellentes performances. Son algorithme, fondé sur des additions, des rotations et des opérations XOR, est simple à implémenter, rapide même sur des architectures modestes, et sa sécurité est aujourd'hui solidement éprouvée par la communauté scientifique. Ces caractéristiques font de ChaCha20 une option stratégique pour la cryptographie embarquée en milieu marin, où les ressources sont précieuses et la fiabilité essentielle [**Bernstein, D. (2008a)**, **Meijer, C., et al. (2021)**].

II.3.6 Forces et limites de la cryptographie symétrique

Bien que ses racines remontent aux premières heures de la sécurité de l'information, la cryptographie symétrique demeure aujourd'hui l'un des piliers essentiels de la protection des échanges numériques. Sa rapidité, sa fiabilité, et la simplicité de ses mécanismes en font un outil de choix, notamment dans le contexte des dispositifs embarqués en mer ou des environnements techniques restreints [**Stallings, W. (2017)**]. Cependant, son efficacité repose sur des hypothèses de sécurité spécifiques et elle n'est pas exempte de certaines faiblesses.

Forces majeures

1. **Efficacité computationnelle** Les algorithmes symétriques sont réputés pour leur rapidité aussi bien lors du chiffrement que du déchiffrement. Cette efficacité est particulièrement précieuse dans les systèmes à ressources limitées, comme les capteurs sous-marins à faible consommation ou les drones autonomes marins. Contrairement à la cryptographie asymétrique, ils n'imposent pas de lourds calculs sur de grands entiers ou d'opérations exponentielles, ce qui allège considérablement la charge informatique [**Douglas Robert Stinson, & Paterson, M. B. (2019)**].
2. **Simplicité d'implémentation matérielle** Grâce à leur structure déterministe, reposant souvent sur des opérations élémentaires (substitutions et permutations pour AES, additions modulaires et rotations pour ChaCha20), les algorithmes symétriques peuvent être facilement déployés sur des microcontrôleurs ou des circuits programmables (FPGA), avec une dépense énergétique minime [**Stallings, W. (2017)**].
3. **Taux de transfert élevé** La faible latence des chiffrements symétriques autorise le traitement en temps réel de volumes massifs de données, une caractéristique cruciale pour les applications marines telles que la télémétrie, l'acquisition continue de mesures océanographiques, ou la transmission de données acoustiques [**Schneier, B. (1996)**].
4. **Compatibilité avec les architectures à clés pré-distribuées** Dans des environnements isolés où la connectivité réseau est intermittente, comme sur des sondes dérivantes ou des véhicules sous-marins autonomes, il est généralement plus simple de gérer des systèmes basés sur des clés préalablement distribuées,

ce qui favorise l'usage du chiffrement symétrique [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**].

Limites et vulnérabilités

1. **Problème de distribution des clés** Le principal point faible de la cryptographie symétrique réside dans la nécessité de partager une clé secrète avant toute communication. Dans le cas de réseaux dispersés de capteurs ou de dispositifs autonomes sous-marins, l'établissement et le renouvellement sécurisé de ces clés deviennent une opération délicate [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**].
2. **Pas de non-répudiation** Puisque l'expéditeur et le destinataire utilisent la même clé, il n'est pas possible de démontrer l'origine d'un message de manière incontestable. Cette limite prive la cryptographie symétrique de certaines applications, notamment celles nécessitant des signatures numériques ou la preuve d'authenticité [**Douglas Robert Stinson, & Paterson, M. B. (2019)**].
3. **Exposition au risque en cas de compromission** Si une clé secrète venait à être compromise, l'ensemble des échanges, passés comme futurs, se retrouverait exposé. Ce danger est particulièrement critique dans les systèmes marins, où il est souvent impossible de mettre à jour les clés à distance de façon sécurisée [**Schneier, B. (1996)**].
4. **Réutilisation de clés ou vecteurs d'initialisation (IV)** Une mauvaise gestion des clés ou des vecteurs d'initialisation (IV) peut ouvrir la porte à des attaques telles que le known-plaintext ou le chosen-plaintext, en particulier dans certains modes opératoires par blocs comme ECB ou CBC [**Douglas Robert Stinson, & Paterson, M. B. (2019)**]. Une discipline stricte dans la génération et l'utilisation des clés et des IV est donc impérative pour préserver la sécurité du système.

Vers une intégration hybride

Dans le monde réel, la plupart des systèmes de sécurité modernes privilégient une stratégie hybride, mêlant astucieusement les avantages des deux grandes familles de cryptographie. On retrouve ainsi la cryptographie symétrique, appréciée pour sa rapidité de traitement, associée à des méthodes asymétriques qui permettent un échange de clés sécurisé, même entre parties distantes.

Prenons l'exemple d'un système de communication maritime : il n'est pas rare que le protocole commence par une négociation de clé, utilisant Diffie-Hellman pour que les deux interlocuteurs génèrent ensemble un secret partagé. Une fois cette clé établie en toute sécurité, le flux de données continue est alors protégé grâce à un algorithme symétrique performant comme AES ou ChaCha20 [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**, **Stallings, W. (2017)**].

Cette approche permet de concilier la robustesse et la praticité, en tirant le meilleur des deux mondes.

II.4 – La cryptographie asymétrique

Alors que la cryptographie symétrique repose sur une clé partagée entre les parties communicantes, la cryptographie asymétrique s'appuie sur une paire de clés distinctes : l'une publique, accessible à tous, et l'autre privée, connue uniquement de son propriétaire. Cette séparation permet d'éliminer l'un des problèmes majeurs de la cryptographie classique : la distribution sécurisée des clés [**Diffie, W., & Hellman, M. (1976)**].

II.4.1 Définition

La cryptographie asymétrique, également désignée sous le nom de cryptographie à clé publique, repose sur un principe fondamental : l'utilisation de deux clés distinctes mais mathématiquement liées, l'une publique, l'autre privée. Le processus fait intervenir deux fonctions complémentaires : une fonction de chiffrement, notée E, qui exploite la clé publique (K_{pub}), et une fonction de déchiffrement, notée D, qui utilise la clé privée (K_{priv}). Pour assurer la cohérence du système, il faut que :

$$D_{K_{priv}}\left(E_{K_{pub}}(M)\right)=M$$

Cela signifie que si Alice souhaite adresser un message confidentiel à Bob, elle emploie la clé publique de ce dernier pour chiffrer le message. Bob, détenteur de la clé privée associée, sera alors le seul capable de restaurer le message d'origine [**Douglas Robert Stinson, & Paterson, M. B. (2019)**].

L'intérêt et la sécurité de ce mécanisme reposent sur la difficulté de certains problèmes mathématiques bien connus, tels que la factorisation des grands entiers (au cœur de RSA), le calcul du logarithme discret (utilisé dans les systèmes ElGamal et DSA), ou encore le logarithme discret sur les courbes elliptiques, qui fonde la sécurité des algorithmes à courbes elliptiques comme ECDSA ou ECDH [**Diffie, W., & Hellman, M. (1976)**, **Miller, V. S. (1986)**, **Elgamal, T. (1985)**].

Parmi les algorithmes les plus emblématiques de la cryptographie asymétrique, on retrouve :

- **RSA** : basé sur la complexité de la factorisation des grands nombres premiers [**Douglas Robert Stinson, & Paterson, M. B. (2019)**].
- **ElGamal** : fondé sur la difficulté du logarithme discret dans un groupe multiplicatif [**Elgamal, T. (1985)**].
- **DSA (Digital Signature Algorithm)** : variante du schéma ElGamal, dédiée à la signature numérique [**Elgamal, T. (1985)**].
- **Algorithmes à courbes elliptiques** (ECC), comme ECDSA et ECDH : tirant parti du logarithme discret sur les courbes elliptiques, ces solutions offrent un haut niveau de sécurité pour des clés de taille relativement réduite [**Miller, V. S. (1986)**].

L'adoption de ces différentes familles d'algorithmes permet aujourd'hui de sécuriser la majorité des échanges numériques sensibles, qu'il s'agisse d'échanges maritimes, bancaires ou gouvernementaux.

II.4.2 RSA

Apparu en 1977 grâce aux travaux de Rivest, Shamir et Adleman, l'algorithme RSA s'est rapidement imposé comme l'une des pierres angulaires de la cryptographie contemporaine.

Sa robustesse s'appuie sur une propriété mathématique majeure : il est extrêmement difficile, d'un point de vue algorithmique, de retrouver les facteurs premiers à partir d'un grand nombre qui en est le produit. Ce mécanisme constitue la base de la sécurité du système. RSA illustre parfaitement la logique des systèmes à clé publique : la clé utilisée pour chiffrer un message n'est pas la même que celle employée pour le déchiffrer, chaque opération reposant sur une clé distincte [**Rivest, R. L., Shamir, A., & Adleman, L. (1978)**].

Pour mieux comprendre la logique de RSA, il est utile d'en détailler les étapes à travers un exemple concret.

Exemple illustratif : étapes de la génération et de l'utilisation des clés RSA

Commençons par rappeler une propriété mathématique centrale : si n est le produit de deux nombres premiers distincts, p et q, alors l'indicatrice d'Euler s'écrit $\varphi(n)=(p-1)(q-1)$. Cette valeur permet de construire la clé privée en calculant l'inverse modulaire de l'exposant de chiffrement, condition essentielle pour garantir la réversibilité des opérations dans le groupe $Z_n^{\times}$ [**Douglas Robert Stinson, & Paterson, M. B. (2019)**].

Voici, étape par étape, le déroulement d'un chiffrement RSA avec de petits nombres pour faciliter la compréhension :

1. **Choix des nombres premiers** Sélectionnons $p=61$ et $q=53$.
2. **Calcul du module** On obtient $n=p\times q=3233$, qui servira de base commune pour le chiffrement et le déchiffrement.
3. **Calcul de l'indicatrice d'Euler** $\varphi(n)=(p-1)\times(q-1)=60\times 52=3120$.
4. **Sélection de l'exposant de chiffrement** e On choisit un entier e premier avec $\varphi(n)$; prenons $e=17$.
5. **Détermination de la clé privée** d On calcule l'inverse de e modulo $\varphi(n)$: $d=2753$.

On obtient alors deux paires de clés :

- **Clé publique** : $(e,n)=(17,3233)$
- **Clé privée** : $(d,n)=(2753,3233)$

Pour chiffrer un message, supposons que le message d'Alice soit $M=123$. Le chiffrement s'effectue ainsi :

- $C=M^e \bmod n=123^{17} \bmod 3233=855$

Le destinataire Bob, muni de sa clé privée, déchiffre :

- $M=C^d \bmod n=855^{2753} \bmod 3233=123$

On voit ainsi que le message initial est bien récupéré, ce qui assure la confidentialité et la fiabilité de la communication. Ce mécanisme, simple dans cet exemple, devient redoutablement robuste avec des clés de grande taille, exploitant pleinement la difficulté de la factorisation pour garantir la sécurité des échanges.

La robustesse du chiffrement RSA s'appuie directement sur le fait qu'il est extrêmement ardu de décomposer le module n en ses facteurs premiers p et q. Tant que cette opération demeure hors de portée des moyens de calcul actuels, l'intégrité du système reste garantie. À ce jour, les attaques les plus efficaces contre RSA exploitent des techniques classiques de factorisation, mais celles-ci deviennent rapidement inadaptées lorsque la longueur des clés atteint ou dépasse 2048 bits [**Rivest, R. L., Shamir, A., & Adleman, L. (1978)**].

II.4.4 ECC : la cryptographie sur les courbes elliptiques

La cryptographie à base de courbes elliptiques, plus connue sous l'acronyme ECC (Elliptic Curve Cryptography), s'est imposée comme une alternative moderne à RSA. Plutôt que de s'appuyer sur la factorisation de grands nombres premiers, cette méthode exploite les propriétés mathématiques particulières des courbes elliptiques. L'un de ses atouts majeurs réside dans l'efficacité de sa sécurité : il suffit, par exemple, d'une clé de 256 bits en ECC pour atteindre le niveau de protection d'une clé RSA de 3072 bits [**Koblitz, N. (1987)**].

Au cœur du fonctionnement d'ECC se trouve l'opération dite de multiplication scalaire, qui consiste à additionner un point de la courbe à lui-même un certain nombre de fois. Ces opérations sont réalisées sur des courbes définies par des équations du type :

$$y^2 = x^3 + ax + b \,(mod\ p),$$

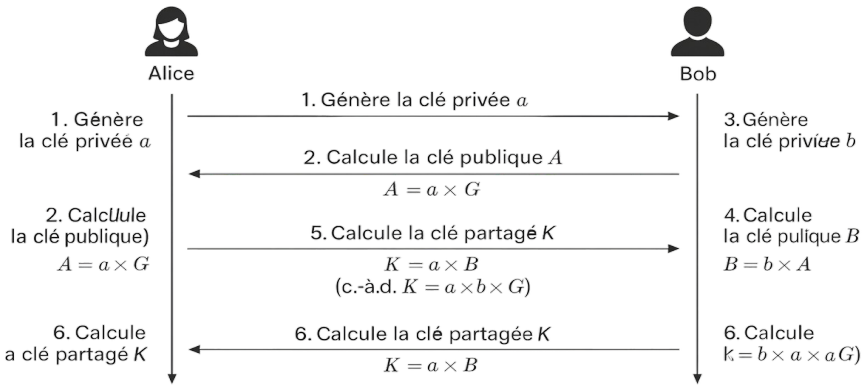
où les calculs s'effectuent dans un corps fini.

Ce cadre algébrique présente une double particularité : il rend l'addition de points relativement simple à calculer, alors qu'il est extrêmement difficile, en pratique, de déterminer le facteur initial à partir du résultat (problème du logarithme discret), ce qui fonde la robustesse du système.

Grâce à la légèreté de ses clés et à la rapidité de ses calculs, l'ECC est devenue la solution privilégiée pour les dispositifs embarqués, et particulièrement en milieu marin, où les ressources en mémoire et en puissance de traitement sont souvent très limitées [**Bernstein, D. J. (2006)**].

Illustration intuitive de l'échange de clé ECDH (Elliptic-Curve Diffie-Hellman)

L'échange de clé Diffie-Hellman sur courbe elliptique repose sur une élégante symétrie mathématique. Le schéma ci-dessous représente pas à pas le fonctionnement de l'algorithme,



tel qu'il est mis en œuvre par deux entités : Alice (à gauche) et Bob (à droite), jusqu'à l'établissement d'un secret partagé noté $K = abG$, un point sur la courbe.

Étapes de l'échange :

- **Étape 1 (Alice) & Étape 3 (Bob)** : *Choix des clés privées* : Chacun commence par générer un **entier secret** :
    - Alice choisit un scalaire a,
    - Bob choisit indépendamment un scalaire b. Ces deux valeurs sont gardées strictement confidentielles.
- **Étape 2 (Alice) & Étape 4 (Bob)** : *Calcul des clés publiques* : En multipliant leur secret respectif par le point générateur G de la courbe elliptique, chacun obtient une clé publique :
    - Alice calcule $A = aG$,
    - Bob calcule $B = bG$. Ces points, A et B, sont des éléments du groupe elliptique et peuvent être librement échangés sur le réseau.
- **Transmission sur le canal** : Seuls les points A et B circulent — les secrets a et b ne sont jamais divulgués.
- **Étape 5 (Alice) & Étape 6 (Bob)** – *Calcul de la clé partagée* : Grâce à la propriété commutative de la multiplication scalaire sur courbe elliptique, chacun peut combiner sa propre clé privée avec la clé publique reçue pour dériver la même clé partagée :
    - Alice calcule $K = aB = a(bG) = abG$,
    - Bob calcule $K = bA = b(aG) = baG$. Ainsi, tous deux aboutissent à la même valeur secrète K sans jamais l'échanger explicitement.

Propriété de sécurité fondamentale :

La robustesse de ce protocole repose sur la difficulté du problème du logarithme discret sur courbes elliptiques (ECDLP). En effet, même si un attaquant malveillant intercepte les clés publiques A et B, il lui est computationnellement impossible de retrouver la clé partagée K, tant qu'il n'a pas accès aux secrets a ou b.

II.4.5 Comparaison RSA vs ECC

Critère	RSA (2048 bits)	ECC (256 bits)
Sécurité équivalente	Oui	Oui
Taille des clés	Longue	Courte
Temps de chiffrement	Rapide	Très rapide
Temps de déchiffrement	Lent	Rapide
Espace mémoire requis	Élevé	Réduit
Adapté aux systèmes embarqués	Limité	Excellent

Dans les réseaux de capteurs sous-marins, l'ECC est fortement recommandée en raison de son efficacité énergétique et de sa faible empreinte mémoire [**Lauter, K. (2004)**].

II.5 – Fonctions de hachage et intégrité des données

La sécurité ne se limite pas à la confidentialité. Il est impératif de garantir que les données transmises ou stockées n'ont pas été altérées, volontairement ou accidentellement. C'est ce que l'on appelle l'intégrité des données. Les fonctions de hachage cryptographiques jouent un rôle fondamental : elles permettent de résumer un message par une empreinte unique, facilitant ainsi la détection de toute modification [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**, **May, W. (2015)**].

Cette section explore leur définition, leur rôle, leurs propriétés essentielles, ainsi que leurs limites et les mesures compensatoires.

II.5.1 Définition intuitive et rôle fonctionnel

Une fonction de hachage cryptographique est une opération mathématique qui transforme une entrée de longueur arbitraire (comme un fichier, un message, ou une mesure capteur) en une empreinte binaire de longueur fixe, appelée *digest*. On peut la comparer à une empreinte digitale : bien que deux individus puissent avoir des structures similaires leurs empreintes sont uniques. De la même manière, deux messages différents ont, en principe, des hachés différents [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**].

Mathématiquement, une fonction de hachage est une application de la forme :

$$h:\{0,1\}^{¿} \to \{0,1\}^{n}$$

où $\{0,1\}^{¿}$ est un message binaire de longueur quelconque et n est la taille fixe du résultat (souvent 256 ou 512 bits).

Cette fonction doit être rapide à calculer, mais inverser ou manipuler la sortie ne doit donner aucune information utile sur l'entré**e**, une propriété essentielle pour la sécurité des échanges [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**, **May, W. (2015)**].

> **Exemple :** Alice mesure une température océanique M, et en génère le haché $H=h(M)$ qu'elle transmet à Bob avec le message. Bob, de son côté, calcule à nouveau $h(M)$. Si le résultat est identique à H, il peut alors vérifier l'intégrité du message, autrement dit, qu'il n'a pas été modifié par Eve ou altéré lors du transit.

Cette opération est au cœur de nombreux protocoles cryptographiques modernes (TLS, SSH, DNSSEC…), où les fonctions de hachage sont utilisées pour vérifier l'intégrité, construire des signatures numériques, ou dériver des clés cryptographiques [**Rescorla, E. (2018)**, **Eastlake, D., & Hansen, T. (2011)**].

II.5.2 Propriétés fondamentales d'une bonne fonction de hachage

Une fonction de hachage cryptographique ne doit pas simplement produire un résumé ; elle doit respecter des **propriétés de sécurité** essentielles pour être fiable :

1. **Résistance à la préimage :** Il doit être incomputable de retrouver un message M à partir d'un haché $H = h(M)[May, W.(2015)]$.
2. **Résistance à la seconde préimage :** Il doit être difficile de trouver un autre message $M' \neq M$ tel que $h(M') = h(M)$ [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**].
3. **Résistance aux collisions :** Il doit être extrêmement improbable de trouver deux messages $M \neq M'$ produisant le même haché [**Stevens, M., Bursztein, E., et al. (2017)**].
4. **Effet avalanche :** Une petite variation de l'entrée entraîne un changement radical du haché (plus de 50 % des bits changés en moyenne) [**May, W. (2015)**].

II.5.3 Algorithmes de hachage populaires et sécurité

Algorithme	Taille du haché	Sécurité actuelle	Statut
MD5	128 bits	Complètement cassé	Obsolète
SHA-1	160 bits	Collisions démontrées [**Stevens, M., Bursztein, E., et al. (2017)**]	Déprécié par NIST
SHA-256	256 bits	Robuste à ce jour [**May, W. (2015)**]	Standard NIST
SHA-3 (Keccak)	256–512 bits	Robuste, conception différente [**Bertoni, G., Daemen, J., et al. (2011)**]	Norme FIPS 202
BLAKE2/ BLAKE3	Variable	Très rapide, sécurité formelle [**Aumasson, J.-P., Neves, S., et al. (2013)]**	Recommandé pour systèmes embarqués

> **Remarque :** SHA-256 est aujourd'hui largement utilisé dans les applications marines pour vérifier l'intégrité des messages, notamment dans les paquets de communication acoustique ou satellite [**May, W. (2015)**], [**Stevens, M., Bursztein, E., et al. (2017)**].

II.5.4 Limites et contre-mesures

Limites :

Même si les fonctions de hachage cryptographiques sont des outils puissants pour l'intégrité des données, elles présentent plusieurs limitations pratiques et théoriques :

- **Le hachage seul ne garantit pas l'authenticité**. Un attaquant comme Eve peut intercepter un message M, le modifier en M', puis recalculer un nouveau haché $H' = h(M')$. Si le système ne vérifie que l'intégrité via le haché, il sera trompé. C'est pourquoi il est indispensable d'associer les fonctions de hachage à des mécanismes d'authentification, comme les HMAC [**Krawczyk, H., Bellare, et al. (2025)**, **Eastlake, D., & Hansen, T. (2011)**].

- **Vulnérabilité aux collisions pour certains algorithmes** : Des attaques pratiques ont été démontrées sur MD5 [**Wang, X., & Yu, H. (2005)**] et SHA-1 [**Stevens, M., Bursztein, E., et al. (2017)**], où deux messages distincts peuvent produire le même haché. Cela ouvre la voie à des attaques de substitution de fichiers ou à la falsification de signatures.

- **Dépendance aux ressources de calcul** : Certains algorithmes comme SHA-512, bien que sécurisés, sont trop lourds pour être utilisés efficacement sur des dispositifs marins embarqués à faible puissance [A**umasson, J.-P. Meier, W., et al. (2015)**]. Cela rend nécessaire l'adoption d'algorithmes plus légers, comme BLAKE2 ou des versions tronquées de SHA-3.

- **Attaques par longueur étendue (Length Extension Attacks)** : Si le haché $h(M)$ est utilisé seul pour authentifier un message, certains algorithmes comme SHA-1 ou MD5 permettent à un attaquant de calculer $h(M \parallel M')$ à partir de $h(M)$, sans connaître M [**Koblitz, N., & Springerlink (Online Service). (1996)**]. Cela compromet gravement l'intégrité.

Contre-mesures :

Pour pallier ces faiblesses, plusieurs stratégies sont recommandées :

- **Utilisation de HMAC (Hash-based Message Authentication Code)** : HMAC ajoute un niveau de sécurité en combinant une clé secrète partagée K avec le message, ce qui empêche Eve de générer un haché valide sans connaître la clé. HMAC-SHA-256 est aujourd'hui une norme robuste [**Eastlake, D., & Hansen, T. (2011)**, **National Institute of Standards and Technology. (2008)**].

$$HMAC_K(M) = h((K \oplus opad) \parallel h((K \oplus ipad) \parallel M))$$

- **Abandon des fonctions vulnérables (MD5, SHA-1)** au profit de SHA-2 (SHA-256/512), SHA-3 ou BLAKE3 [30, **umasson, J.-P., Meier, W., et al. (2015)**].

- **Choix d'algorithmes optimisés pour les systèmes contraints** : BLAKE2, par exemple, offre une sécurité équivalente à SHA-3 mais avec une empreinte mémoire plus faible et une vitesse d'exécution adaptée aux microcontrôleurs [**BLAKE2 Team (Aumasson, J.-P., et al). (2015)**].

- **Validation par signature numérique** : Pour les données critiques, le haché peut être signé avec une clé privée asymétrique. Cela garantit à la fois l'intégrité et

l'authenticité du message (ex. : signature ECDSA sur $h(M)$) [**Johnson, D., Menezes, A., & Vanstone, S. (2001)**].

II.6 – Signatures numériques : authentification et non-répudiation dans les communications marines sécurisées

Lorsque Bob reçoit un message critique de la part d'Alice, par exemple une commande de plongée transmise à un drone sous-marin autonome, il ne suffit pas que le message soit confidentiel. Encore faut-il qu'il soit authentique, c'est-à-dire que Bob puisse vérifier qu'il provient bien d'Alice, et qu'il soit intègre, c'est-à-dire qu'il n'ait pas été altéré pendant la transmission. C'est là que les signatures numériques entrent en jeu [**Rivest, R. L., Shamir, A., & Adleman, L. (1978)**, **Cooper, D., et al. (2008)**].

II.6.1 Une analogie simple : la signature manuscrite, version mathématique

De la même manière qu'une signature manuscrite engage juridiquement son auteur dans le monde physique, une signature numérique permet à Alice de s'engager sur un message M qu'elle envoie à Bob. Si un litige survient, elle ne pourra pas nier l'avoir signé, c'est ce qu'on appelle la non-répudiation.

Mais à la différence d'une signature manuscrite, dont l'authenticité repose sur l'expertise d'un graphologue, ici tout repose sur des fondations mathématiques solides, reposant principalement sur les propriétés asymétriques des fonctions cryptographiques [**Rivest, R. L., Shamir, A., & Adleman, L. (1978)**].

II.6.2 Principe général d'une signature numérique

Imaginons la scène suivante :

1. Alice veut envoyer à Bob un message M, et prouver qu'il vient bien d'elle.
2. Elle commence par calculer l'empreinte du message via une fonction de hachage
$$h = H(M)$$
3. Elle signe cette empreinte à l'aide de sa clé privée K_{priv}^{Alice} :
$$\sigma = Sign_{K_{priv}}(h)$$
4. Elle envoie à Bob le couple (M, σ).
5. Bob, à la réception, recalcule l'empreinte $h' = H(M)$, puis vérifie la signature à l'aide de la clé publique d'Alice :
$$Verify_{K_{pub}^{Alice}}(h', \sigma)$$
6. Si la vérification réussit, Bob peut être sûr que :
    - Le message n'a pas été modifié ;
    - il provient bien d'Alice ;
    - Alice ne pourra pas nier son envoi ultérieur.

Ce mécanisme est la colonne vertébrale de la confiance dans la communication sécurisée, notamment dans les protocoles comme TLS/SSL, les certificats numériques (X.509), ou les échanges machine-à-machine dans l'IoT marin [**Cooper, D., et al. (2008)**].

Exemple :

Pour illustrer le fonctionnement des signatures numériques, concentrons-nous sur un des schémas les plus classiques et les mieux documentés : le schéma de signature RSA, basé sur la difficulté de factoriser de grands nombres premiers [**Rivest, R. L., Shamir, A., & Adleman, L. (1978)**].

Signature RSA – Étapes complètes

Supposons qu'Alice veuille signer un message M à l'aide de sa clé privée d.

1. Hachage du message :

Alice commence par calculer un condensé du message avec une fonction de hachage sécurisée (ex : SHA-256) :

$$h = H(M)$$

2. Signature proprement dite :

Elle élève le haché à la puissance d modulo N :

$$\sigma = h^d \bmod N$$

Le résultat σ est la signature. Elle envoie à Bob le couple (M, σ).

Vérification RSA – Côté Bob

Bob reçoit M et σ. Il effectue les opérations suivantes :

1. Il hache le message :

$$h' = H(M)$$

2. Il vérifie si :

$$\sigma^e \bmod N = h'$$

Si l'égalité est vraie, la signature est valide, ce qui garantit que :

- Le message n'a pas été modifié (intégrité),
- Il a été signé par quelqu'un possédant la clé privée (authenticité),
- Et l'émetteur ne peut pas nier sa responsabilité (non-répudiation).

Exemple numérique simplifié

Soit :

- $p=17, q=11 \Rightarrow N=187, \varphi(N)=160$
- $e=7 \Rightarrow d=23$, car $7 \cdot 23 \equiv 1 \, mod \, 160$

Alice veut signer le message M tel que $H(M)=88$. Elle calcule :

$$\sigma = 88^{23} \, mod \, 187 = 11$$

Bob vérifie :

$$\sigma^7 \, mod \, 187 = 11^7 \, mod \, 187 = 88$$

Il retrouve bien le haché $h' = 88$, donc la signature est valide.

À retenir

- L'opération $\sigma = h^d \, mod \, N$ est une fonction unidirectionnelle difficile à inverser sans d.
- La vérification $\sigma^e \, mod \, N = h'$ repose uniquement sur la clé publique, donc tout le monde peut vérifier la signature sans compromettre la sécurité.
- L'intégrité dépend du hachage ; l'authenticité, de la possession de la clé privée ; la non-répudiation, du lien mathématique fort entre message et signature.

II.6.3 Principe général d'une signature numérique

Algorithme	Fondement mathématique	Taille signature	Vitesse	Recommandé pour...
RSA	Arithmétique modulaire	Longue	Lent	Documents lourds, certificats
ECDSA	Courbes elliptiques	Courte	Rapide	IoT, systèmes embarqués
Ed25519	ECC sur courbe spéciale	Très courte	Très rapide	Applications modernes légères
SPHINCS+	Basé sur le hachage (post-quantique)	Longue	Modéré	Environnements quantiques

Les courbes elliptiques offrent un excellent compromis entre sécurité, légèreté et rapidité, ce qui en fait un choix naturel pour les systèmes embarqués marins, où les contraintes de calcul et d'énergie sont importantes [**Miller, V. S. (1986)**].

II.7 – Cryptographie légère

La cryptographie légère (ou *lightweight cryptography* en anglais) est une branche spécialisée de la cryptographie qui vise à fournir des mécanismes de sécurité robustes mais peu gourmands en ressources, que ce soit en termes de puissance de calcul, de mémoire, de consommation énergétique ou de bande passante. Ces solutions sont particulièrement adaptées aux systèmes embarqués contraints, tels que ceux utilisés dans les environnements marins, où les capteurs, balises, drones sous-marins ou bouées intelligentes sont souvent alimentés par batterie et fonctionnent dans des conditions extrêmes [**Eisenbarth, T., Kumar, S., et al. (2007)**].

Contrairement aux algorithmes classiques comme AES ou RSA, qui peuvent être très efficaces sur des ordinateurs standards ou des serveurs, la cryptographie légère s'adresse à des dispositifs où chaque bit d'énergie et chaque cycle d'horloge comptent. L'idée n'est pas de créer des algorithmes faibles, mais d'optimiser le compromis entre sécurité et efficacité, sans jamais compromettre les objectifs fondamentaux : confidentialité, intégrité, authenticité et, dans certains cas, anonymat [**Thakor, V. A., et al. (2021)**].

Exemple : Un capteur de température océanographique qui transmet ses données toutes les 10 minutes via une liaison satellite a besoin d'un chiffrement rapide, peu énergivore, mais suffisamment sûr pour empêcher l'interception ou la falsification des données environnementales.

II.7.1 Critères d'évaluation d'un algorithme léger

Un algorithme léger n'est pas simplement une version réduite d'un algorithme classique. Il doit répondre à des critères bien définis [**Eisenbarth, T., Kumar, S., et al. (2007)**]:

1. **Taille du code** : l'algorithme doit tenir dans quelques kilooctets de mémoire ROM.
2. **Mémoire RAM utilisée** : souvent inférieure à 200 octets.
3. **Cycles par octet** : doit permettre un traitement rapide même à basse fréquence.
4. **Consommation énergétique** : exprimée en nanojoules par bit.
5. **Sécurité** : doit résister aux attaques modernes sans dépendre d'une sécurité "par obscurité".

Un bon compromis est crucial : une sécurité trop élevée pourrait rendre le chiffrement trop coûteux pour les plateformes embarquées, une sécurité trop faible rend le dispositif vulnérable à des attaques physiques ou logicielles.

La cryptographie légère devient ainsi un composant stratégique de la souveraineté technologique et de la protection des données environnementales en milieu hostile [**Thakor, V. A., et al. (2021)**].

II.7.2 Algorithmes phares : ASCON et PRESENT

a) ASCON

ASCON est un algorithme de chiffrement authentifié (permettant à la fois la confidentialité et l'intégrité) sélectionné en 2023 par le NIST comme nouveau standard pour la cryptographie légère [**Turan, M. S., et al. (2023).**]. Il est basé sur des opérations logiques simples (XOR, rotations, AND) et offre une excellente efficacité sur les petits processeurs.

- **Mode de fonctionnement** :

  Le fonctionnement d'ASCON suit le modèle suivant :

  $$ASCON\,ext-128(K,N,A,M)\Rightarrow C,T$$

  Où :

  - K : clé secrète
  - N : nonce
  - A : données authentifiées (non chiffrées)
  - M : message clair
  - C : texte chiffré
  - T : tag d'authentification

- **Utilisation typique** : authentifier et chiffrer de courts messages dans des capteurs IoT ou marins

- **Avantages** :

  - Faible consommation énergétique [**Dobraunig, C., et al. (2021)**],
  - Résistance à des attaques connues (différentiel, linéaire, etc.)
  - Bonne performance sur architectures 8-bit ou 32-bit [**Turan, M. S., et al. (2023)**]

  Exemple : Un capteur de pression sous-marin envoie des données toutes les 15 minutes à un satellite relais. ASCON permet de chiffrer chaque message avec authentification, en consommant moins de 5 µJ par opération [**Dobraunig, C., et al. (2021)**].

b) PRESENT

Développé par des chercheurs européens en 2007, PRESENT est un algorithme de chiffrement par blocs de 64 bits, avec des clés de 80 ou 128 bits [**Bogdanov, A., et al. (2007)**]. Il est extrêmement léger, idéal pour le matériel embarqué à très faibles ressources.

- **Type** : chiffrement symétrique par blocs
- **Structure** : substitution-permutation network (SPN)
- **Nombre de rounds** : 31
- **Domaines d'application** :
  - RFID

- Smartcards
- Systèmes embarqués marins basse consommation

Son empreinte mémoire minimale et sa simplicité en font un bon candidat pour les systèmes marins intégrés dans des flotteurs autonomes ou des sondes déployées longtemps [**Bogdanov, A., et al. (2007)**, **Eisenbarth, T., Kumar, S., et al. (2007)**].

II.7.3 Comparaison de la complexité

Algorithme	Taille de bloc	Taille de clé	Nombre d'opérations logiques	Empreinte mémoire (RAM + ROM)	Niveau de sécurité estimé
AES-128	128 bits	128 bits	~10 000 cycles	>10 Ko	Élevé
ASCON	128 bits	128 bits	~1 500 cycles	~2 Ko	Élevé (pour usage IoT)
PRESENT	64 bits	80/128 bits	~1 000 cycles	~1 Ko	Moyen à élevé

II.8 – Protocoles cryptographiques

Lorsqu'on aborde la cryptographie appliquée, en particulier dans des systèmes embarqués comme ceux utilisés pour la surveillance environnementale marine, il ne suffit pas de posséder de bons algorithmes de chiffrement ou de hachage. Encore faut-il savoir comment les orchestrer correctement pour répondre à des objectifs précis : confidentialité, authentification, intégrité ou non-répudiation. C'est là qu'interviennent les protocoles cryptographiques.

Un protocole cryptographique, en quelque sorte, c'est une recette : une suite bien définie d'opérations, qui mobilise des primitives (chiffrement, hachage, signature, échange de clés, etc.) afin de permettre à deux (ou plusieurs) parties de communiquer de manière sécurisée, même sur un canal ouvert à l'écoute.

Pour illustrer cela de manière plus intuitive, imaginons encore une fois notre duo classique : Alice et Bob. Supposons qu'ils souhaitent échanger un message confidentiel, mais qu'ils ne partagent aucun secret commun préalable. Comment peuvent-ils établir un canal sécurisé, en présence de l'espionne Eve qui écoute toutes leurs communications ? C'est ici qu'un protocole devient essentiel.

II.8.1 Exemple fondamental : l'échange de clés Diffie-Hellman

Introduit en 1976, le protocole de Diffie-Hellman a révolutionné la cryptographie en introduisant le concept d'échange de clés sur un canal non sécurisé [**Diffie, W., & Hellman,**

M. (1976)]. Le génie de cette approche repose sur un principe mathématique simple, mais puissant : le problème du logarithme discret, réputé difficile à inverser dans de grands groupes cycliques.

Idée intuitive

Alice et Bob souhaitent établir une **clé secrète partagée** K sans jamais l'échanger directement. Pour cela, ils choisissent un groupe mathématique où l'opération d'exponentiation est facile, mais où l'inverse (trouver l'exposant) est difficile. Typiquement, on travaille dans le groupe multiplicatif des entiers modulo un grand nombre premier p.

Étapes du protocole

1. Alice et Bob se mettent d'accord sur deux paramètres publics : un grand nombre premier p et une base g (appelée générateur du groupe).
2. Alice choisit un secret a, calcule $A = g^a \bmod p$, et envoie A à Bob.
3. Bob choisit un secret b, calcule $B = g^b \bmod p$, et envoie B à Alice.
4. Alice calcule la clé : $K = B^a \bmod p$
5. Bob calcule aussi : $K = A^b \bmod p$

Et comme :

$$B^a = \left(g^b\right)^a = g^{ab} \bmod p \text{ et } A^b = \left(g^a\right)^b = g^{ab} \bmod p$$

Ils obtiennent exactement la même clé $K = g^{ab} \bmod p$, sans que Eve, qui n'a vu que g, A, et B, ne puisse retrouver a ou b, ni donc la valeur de K, à moins de résoudre un problème de logarithme discret — réputé intractable [**Diffie, W., & Hellman, M. (1976)**].

Limites pratiques

Bien que ce protocole soit élégant, sa version de base n'assure pas l'authenticité. Eve pourrait intercepter les messages et effectuer une attaque de type "man-in-the-middle". Pour y corriger, il faut le combiner à d'autres mécanismes cryptographiques, comme les signatures numériques [**Boyd, C., & Anish Mathuria. (2003)**].

II.8.2 Protocoles d'authentification mutuelle

Dans des environnements marins, souvent déconnectés ou intermittents, il est crucial de s'assurer que les entités qui communiquent sont bien celles qu'elles prétendent être. C'est ce que permettent les protocoles d'authentification. L'un des plus connus est Needham-Schroeder, adapté dans sa version publique par Lowe après la découverte d'une faille [**Gavin, L. (1996)**].

Protocole de Needham-Schroeder (clé publique)

1. Alice génère un nonce (nombre aléatoire) N_A et l'envoie à Bob, chiffré avec la clé publique de Bob.
2. Bob déchiffre, récupère N_A, le renvoie à Alice, accompagné d'un nonce N_B, le tout signé ou chiffré avec la clé publique d'Alice.
3. Alice vérifie N_A et renvoie N_B, prou vant qu'elle est bien l'initiatrice.

Ce mécanisme garantit que chaque participant réagit en temps réel, prouvant ainsi son identité et sa capacité à chiffrer/déchiffrer [**Needham, R. M., & Schroeder, M. D. (1978)**].

II.8.3 Protocoles hybrides : TLS, SSH et VPN

Dans les applications réelles, les protocoles cryptographiques sont composés de plusieurs briques. Le chiffrement asymétrique est souvent utilisé pour échanger une clé symétrique, qui servira ensuite à chiffrer les données. Cela combine le meilleur des deux mondes : sécurité des clés (asymétrique) et rapidité du chiffrement (symétrique) [**Rescorla, E. (2018)**, **Ylonen, T. (2006)**, **Frankel, S., & Krishnan, S. (2011)**].

Exemples notables :

- **TLS (Transport Layer Security)** : utilisé dans HTTPS. Établit une clé de session via RSA ou Diffie-Hellman, puis utilise AES ou ChaCha20 pour chiffrer les données.
- **SSH (Secure Shell)** : protocole d'accès distant sécurisé, avec authentification par clé publique ou mot de passe, puis chiffrement de la session.
- **VPN (Virtual Private Network)** : encapsule et chiffre le trafic réseau, souvent via IPsec ou OpenVPN, pour protéger les communications sur des réseaux non fiables.

II.8.4 Protocole Zero-Knowledge : prouver sans révéler

Les protocoles à preuve à divulgation nulle de connaissance (Zero-Knowledge Proofs) permettent de démontrer qu'on connaît une information, sans jamais révéler cette information elle-même. C'est une notion contre-intuitive, mais essentielle, notamment en cryptographie post-quantique ou pour l'authentification dans les environnements embarqués [**Sasson, E. B., et al. (2014)**, **Boneh, D., & Shoup, V. (2023, January)**].

Exemple illustratif :

Imaginons une situation concrète dans un contexte marin. Alice, ingénieure de bord sur un navire scientifique, prétend connaître le code secret qui ouvre une soute scellée contenant des instruments sensibles de mesure. Bob, responsable de la sécurité embarquée, doit s'assurer qu'elle connaît bien ce code… sans pour autant le lui faire révéler. C'est là qu'intervient un protocole de type preuve à divulgation nulle de connaissance.

Voici comment cela fonctionne, de manière intuitive :

La soute est construite de manière circulaire, avec deux entrées distinctes : l'une à bâbord, l'autre à tribord. Un mécanisme à l'intérieur, contrôlé par le code secret, permet de passer d'une entrée à l'autre sans sortir.

1. Première étape : Alice entre par l'une des deux portes au hasard, tandis que Bob reste à l'extérieur, sans savoir par laquelle elle est entrée.
2. Deuxième étape : Bob lui demande ensuite, au hasard également, de ressortir par l'entrée bâbord ou tribord.

3. Troisième étape : Si Alice connaît le code, elle peut traverser la soute intérieurement, peu importe la demande de Bob.
4. Répétition : En répétant cette opération de nombreuses fois (disons 20 fois ou plus), Bob acquiert une forte conviction probabiliste qu'Alice connaît bel et bien le secret, même s'il n'a jamais vu le code.

Ce mécanisme, malgré son apparente simplicité, repose sur des bases mathématiques rigoureuses. Il garantit que la preuve est valide, qu'elle ne divulgue aucune information sur le secret lui-même, et qu'elle est difficile à contrefaire [**Sasson, E. B., et al. (2014)**]. C'est exactement l'essence d'un protocole Zero-Knowledge : prouver que l'on sait sans révéler ce que l'on sait.

Dans les systèmes cryptographiques modernes, ce type de protocole est utilisé pour vérifier une identité ou une connaissance sans transmettre l'information sensible, ce qui est particulièrement utile dans les environnements embarqués où les communications sont intermittentes ou risquent d'être interceptées. On retrouve ces concepts dans les zk-SNARKs, utilisées par exemple dans les blockchains comme Zcash pour permettre des paiements anonymes mais vérifiables [**Goldwasser, S., et al. (1989)**].

II.9 – Comparaison : Cryptographie symétrique vs Cryptographie asymétrique

Imagine un sous-marin autonome effectuant des relevés océanographiques confidentiels. Comment peut-il communiquer avec la base sans que personne n'intercepte ses données ? Pour cela, il doit choisir entre deux approches cryptographiques fondamentales : la symétrique et l'asymétrique. Mais ce choix n'est pas anodin, surtout dans les environnements contraints comme les dispositifs marins embarqués, où les ressources (énergie, mémoire, bande passante) sont sévèrement limitées [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**, **Stallings, W. (2017)**].

Comprendre la différence entre ces deux familles d'algorithmes n'est pas seulement un exercice théorique. C'est un enjeu stratégique pour garantir la sécurité, la performance et la viabilité des communications dans des environnements complexes et hostiles comme les fonds marins, où la connectivité est rare et précieuse [**Stallings, W. (2017)**, **Boneh, D., & Shoup, V. (2023, January)**].

II.9.2 Principes fondamentaux comparés

Cryptographie symétrique : C'est comme si Bob et Alice partageaient une même boîte à serrure unique : pour chiffrer et déchiffrer un message, ils utilisent exactement la même clé. Rapide et efficace, cette approche est idéale pour les échanges fréquents ou les grands volumes de données. Mais elle pose un problème fondamental : comment échanger la clé secrète sans qu'un espion (Eve) ne l'intercepte [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**, **Barker, E. (2020)].**

Cryptographie asymétrique : Ici, Bob et Alice disposent chacun d'une paire de clés : une clé publique, qu'ils peuvent diffuser librement, et une clé privée, qu'ils gardent secrète. Si

Bob veut envoyer un message sécurisé à Alice, il chiffre le message avec la clé publique d'Alice. Seule Alice pourra le déchiffrer grâce à sa clé privée. Cette méthode est très sûre, surtout pour l'échange initial de clés, mais elle est beaucoup plus lente et gourmande en ressources [**Stallings, W. (2017)**, **Boneh, D., & Shoup, V. (2023, January)**].

Analogie intuitive :

- Clé symétrique : un cadenas partagé entre Bob et Alice, simple, mais encore faut-il se retrouver pour l'échanger.
- Clé asymétrique : une boîte aux lettres où tout le monde peut déposer un message (clé publique), mais seul le propriétaire (clé privée) peut l'ouvrir.

II.9.3 Tableau comparatif synthétique

Critère	**Cryptographie Symétrique**	**Cryptographie Asymétrique**
Nombre de clés	Une seule clé partagée	Deux clés : une publique, une privée
Rapidité	Très rapide	Plus lente (exponentiation modulaire)
Complexité algorithmique	Faible à modérée	Élevée
Taille typique des clés	128 à 256 bits	1024 à 4096 bits (voire plus)
Sécurité	Forte, mais dépend du secret partagé	Forte, repose sur problèmes math. Complexes
Consommation d'énergie	Faible	Élevée
Idéal pour	Gros volumes, environnements contraints	Échange de clés, signature, authentification
Exemples	AES, ChaCha20	RSA, ECC, ElGamal

II.9.4 Vers une cryptographie hybride

Dans la pratique, la plupart des systèmes modernes ne choisissent pas entre les deux… ils les combinent intelligemment. Cette approche, appelée cryptographie hybride, permet de tirer parti des avantages de chacun [**Stallings, W. (2017)**, **Barker, E. (2020)**]:

- On utilise la cryptographie asymétrique pour échanger en toute sécurité une clé symétrique,

- Puis, cette clé sert à chiffrer rapidement tout le reste des communications via un algorithme symétrique comme AES ou ChaCha20.

C’est le cas dans:

- Le protocole TLS (utilisé dans HTTPS),
- Les VPN,
- Et de plus en plus dans les systèmes MIoT (Marine IoT), où les échanges doivent rester légers mais sûrs [**Barker, E. (2020)**].

II.9.5 Synthèse et recommandation pour les systèmes embarqués

En résumé:

- La **cryptographie symétrique** est incontournable dans les systèmes contraints : rapide, efficace, économe en énergie [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**, **Bernstein, D. (2008a)**].
- La **cryptographie asymétrique**, bien que plus coûteuse, est précieuse pour la distribution de clés et les authentifications [**Stallings, W. (2017)**, **Boneh, D., & Shoup, V. (2023, January)**].
- Une **approche hybride** est fortement recommandée pour les systèmes embarqués marins, où la sécurité ne doit pas se faire au détriment de l’autonomie [**Barker, E. (2020)**].

Recommandation : Dans les environnements marins embarqués, commencer par une phase d’authentification asymétrique, puis basculer sur un chiffrement symétrique léger (comme ChaCha20 ou AES-128) est la stratégie optimale, tant en termes de sécurité que de performance [**Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018)**, **Stallings, W. (2017)**, **Bernstein, D. (2008a)**, **Boneh, D., & Shoup, V. (2023, January)**].

Chapitre III : Applications de la cryptographie aux systèmes marins

III.1 – Introduction

Au cœur des espaces marins, loin des relais terrestres et des réseaux conventionnels, assurer la sécurité des communications n'est pas une option : c'est une nécessité vitale. Qu'il s'agisse d'un cargo traversant les océans, d'un sous-marin discret naviguant à grande profondeur, ou d'un réseau de capteurs collectant des mesures dans les abysses, tous ces systèmes sont exposés à un risque croissant : celui d'attaques informatiques de plus en plus élaborées.

Pourtant, le milieu maritime est l'un des plus complexes lorsqu'il s'agit de déployer des solutions de cybersécurité. Les communications y sont souvent limitées, peu fiables, lentes, et très coûteuses en énergie. Les équipements embarqués, qu'il s'agisse de sondes autonomes, de balises intelligentes ou de drones océaniques, reposent généralement sur des microcontrôleurs peu puissants, fonctionnant sur batterie, et dans des conditions d'isolement extrême. Dans un tel contexte, garantir l'authenticité, la confidentialité et l'intégrité des données représente un véritable défi technologique.

C'est précisément dans ce cadre que la cryptographie prend tout son sens. Loin des démonstrations théoriques, elle devient un outil de terrain, indispensable à la fiabilité des opérations maritimes. Des algorithmes de chiffrement robustes comme AES ou ChaCha20, aux signatures numériques fondées sur les courbes elliptiques, en passant par des solutions ultra-légères telles que PRESENT ou ASCON, ces mécanismes permettent de créer un socle de confiance dans des environnements où l'instabilité, la latence et l'exposition aux menaces sont la norme.

Pourquoi cette exigence cryptographique est-elle si cruciale en mer ? Parce qu'une attaque réussie n'implique pas seulement la compromission de données : elle peut mettre en péril des missions scientifiques entières, provoquer des erreurs de navigation, voire fragiliser des systèmes de défense sensibles. En témoigne l'incident de 2011, lorsqu'un drone américain a été détourné via une attaque de type GPS spoofing dans la région du Golfe Persique, une démonstration flagrante de ce que peut entraîner l'absence de sécurisation des signaux [**Humphreys, T. (2012)**].

Ce chapitre poursuit donc un double objectif. Il s'agit, d'une part, de dresser un état des lieux des systèmes marins actuels pour lesquels la cryptographie joue, ou devrait jouer, un rôle stratégique. Et d'autre part, d'expliquer de manière accessible, à travers des exemples concrets et des éclairages pédagogiques, comment différents types d'algorithmes peuvent s'adapter, ou se heurter, aux contraintes maritimes.

Nous avancerons de façon progressive. Nous commencerons par explorer les principaux moyens de communication en mer et les équipements embarqués. Ensuite, nous analyserons les menaces spécifiques à cet environnement. Nous entrerons alors dans le vif du sujet avec l'étude des algorithmes symétriques, des approches légères conçues pour l'IoT marin, ainsi que des techniques asymétriques, notamment celles fondées sur les courbes elliptiques. Enfin, une synthèse visuelle et des recommandations pratiques viendront clôturer ce parcours.

Tout au long de cette analyse, nous conserverons une idée centrale : en environnement maritime, la cryptographie n'est pas un accessoire, c'est le garant indispensable de la résilience numérique.

III.2 – Systèmes de communication marine et matériels concernés

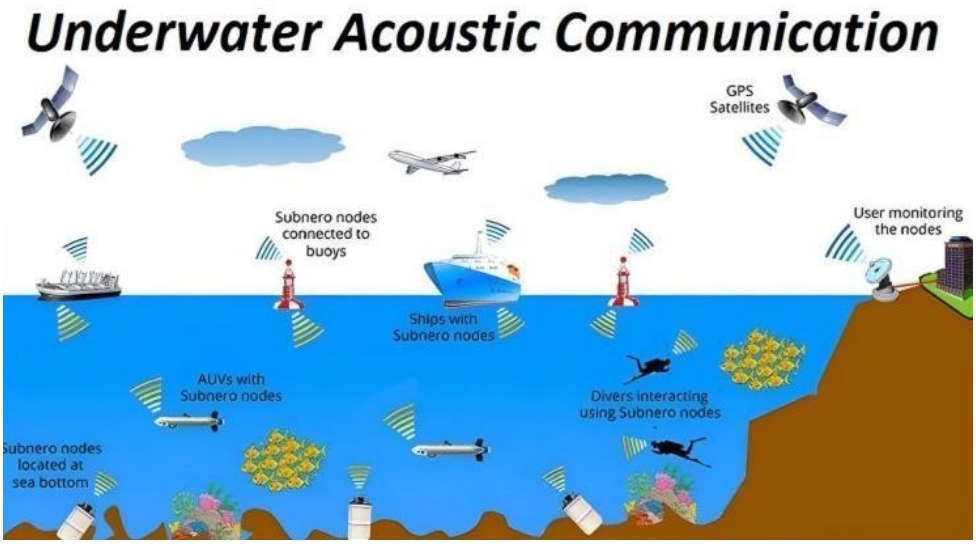
Dans l'univers complexe du maritime moderne, la transmission sécurisée des données repose sur une architecture technologique vaste, fragmentée et souvent hétérogène. Ces infrastructures ne servent pas uniquement à orienter les navires ou à optimiser la logistique : elles sont aussi indispensables à la surveillance de l'environnement, aux activités scientifiques, aux opérations militaires sensibles et au bon fonctionnement des réseaux de capteurs déployés en mer. Tous ces systèmes partagent une caractéristique essentielle : ils traitent des données critiques dans un environnement extrêmement contraignant, aussi bien sur le plan physique que numérique. Comprendre les vecteurs de communication utilisés dans ces milieux est donc une étape cruciale pour identifier précisément les besoins en matière de cryptographie.

III.2.1 Réseaux acoustiques sous-marins (UASNs)

Les Underwater Acoustic Sensor Networks (UASNs) représentent la principale solution pour la communication dans les environnements sous-marins. Comme les ondes radio sont fortement atténuées sous l'eau, ces réseaux reposent sur la transmission d'ondes acoustiques. Ils sont fréquemment utilisés pour :

- Surveiller les niveaux de pollution marine,
- Détecter les séismes sous-marins,
- Suivre les déplacements de bancs de poissons ou de véhicules autonomes (AUVs).

Cependant, la communication acoustique souffre de débits très bas, de latences importantes et d'une grande vulnérabilité à la détection. Dans ce contexte, les algorithmes cryptographiques doivent être conçus pour fonctionner avec des ressources limitées, tolérer les pertes de données, tout en garantissant la sécurité [**Akyildiz, I. F., et al. (2005)**].

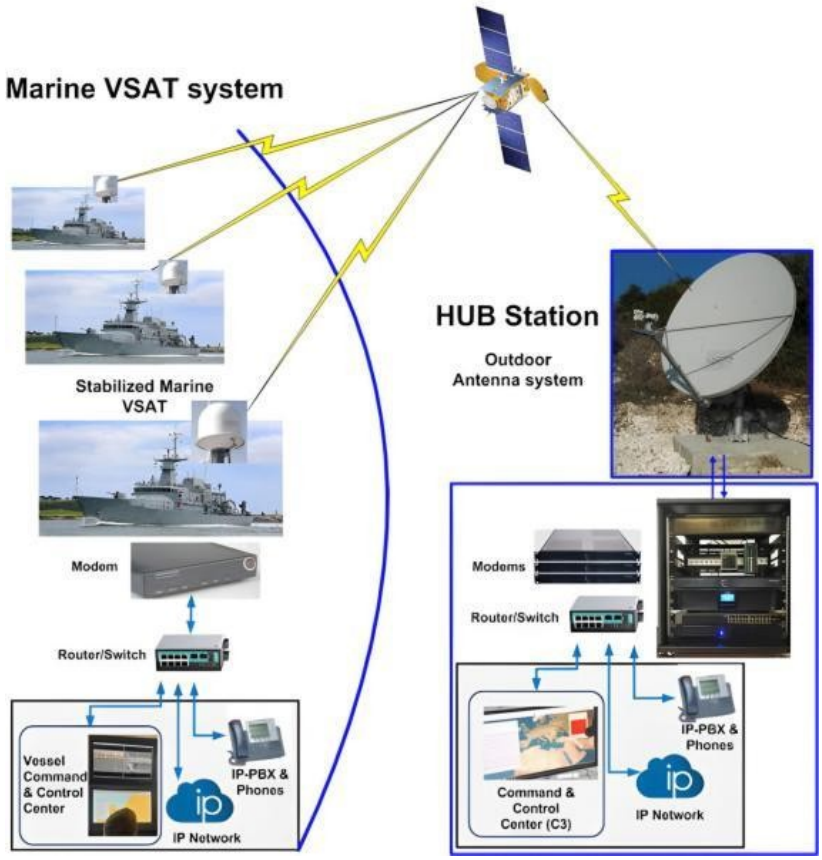


III.2.2 Communications satellites maritimes (VSAT)

À la surface, la plupart des navires établissent leur liaison avec la terre grâce aux Very Small Aperture Terminals (VSAT), qui utilisent les satellites pour transmettre :

- Des informations de navigation,
- Des flux internet,
- Des communications vocales ou vidéo.

Mais ces systèmes, fondés sur les protocoles classiques de l'internet (comme TCP/IP), sont exposés à de multiples menaces : écoutes illicites, attaques par déni de service (DoS), ou détournement de session. Pour s'en prémunir, on met en œuvre des tunnels chiffrés (VPN, IPSec) reposant sur des algorithmes robustes comme AES-256 ou ChaCha20-Poly1305 [**Romain, G., et al. (2021)**].



III.2.3 AIS (Automatic Identification System)

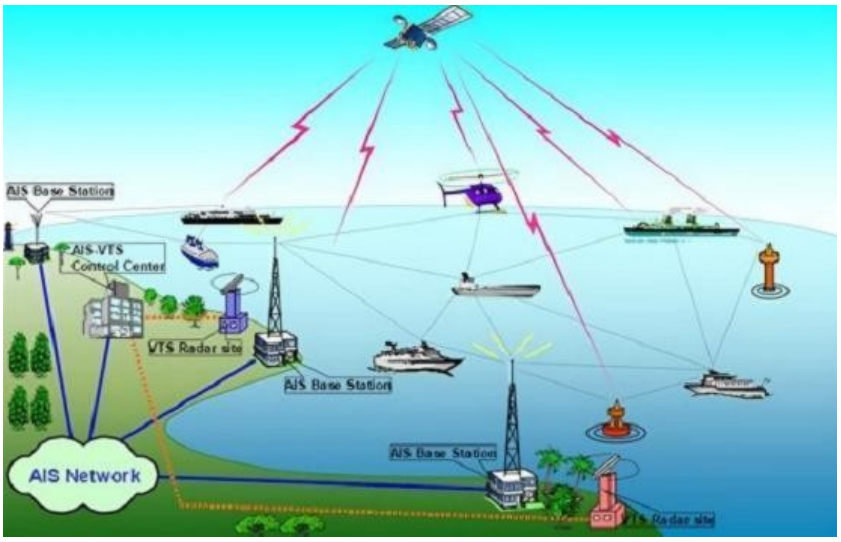
Le système d'identification automatique (AIS) est aujourd'hui largement déployé sur les navires commerciaux, où son usage est généralement obligatoire. Il permet la transmission automatique, par radio VHF, de données de navigation essentielles, telles que la position géographique, la vitesse, le cap ou encore la nature du navire, vers d'autres bâtiments en mer ou vers des stations terrestres. Cette fonctionnalité vise à améliorer la sécurité des trajets, la coordination du trafic maritime et la prévention des collisions.

Cependant, le protocole AIS, dans sa version initiale et la plus largement utilisée, n'intègre aucun mécanisme de protection cryptographique. Cette absence de sécurisation ouvre la voie

à de multiples formes d'exploitation malveillante. Plusieurs travaux, tant de la part de la communauté scientifique que d'acteurs mal intentionnés, ont démontré la possibilité :

- De falsifier les positions géographiques transmises,
- De faire disparaître un navire des écrans radar,
- Ou encore de générer des signaux d'alerte artificiels, perturbant ainsi le trafic maritime [**Balduzzi, et al. (2014, December 8)**].

Face à ces risques croissants, des efforts sont actuellement engagés pour renforcer la fiabilité du système. Des pistes sérieuses sont à l'étude, notamment l'intégration de signatures numériques ou l'utilisation de solutions de chiffrement léger, compatibles avec les infrastructures existantes comme la norme NMEA 2000.

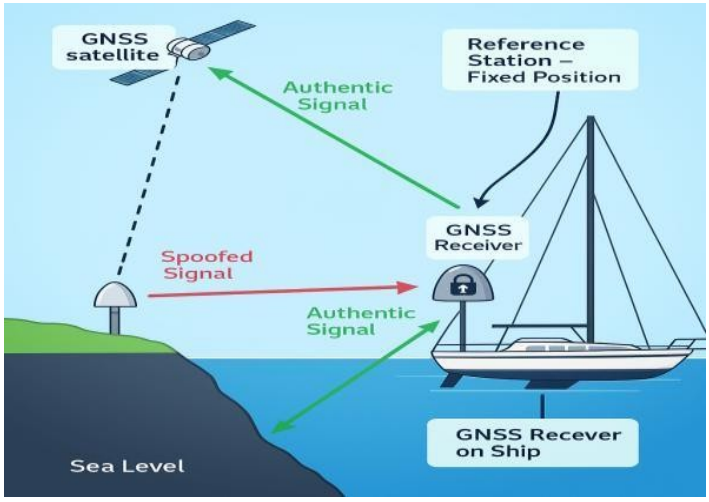


III.2.4 GPS / GNSS pour navigation

Les systèmes de positionnement satellitaires comme le GPS, ou ses variantes (Galileo, GLONASS…), sont au cœur de la navigation maritime. Toutefois, les signaux qu'ils émettent sont faibles et donc facilement brouillés (jamming), voire falsifiés (spoofing).

Des contre-mesures commencent à apparaître, notamment :

- Des modules GPS intégrant une authentification cryptographique des signaux, comme c'est le cas avec le projet Galileo OSNMA [**European GNSS Agency. (2023)**],
- L'utilisation de données locales redondantes (inertie, balises, capteurs internes) pour détecter les anomalies.



III.2.5 Drones marins, AUVs, gliders et SCADA offshore

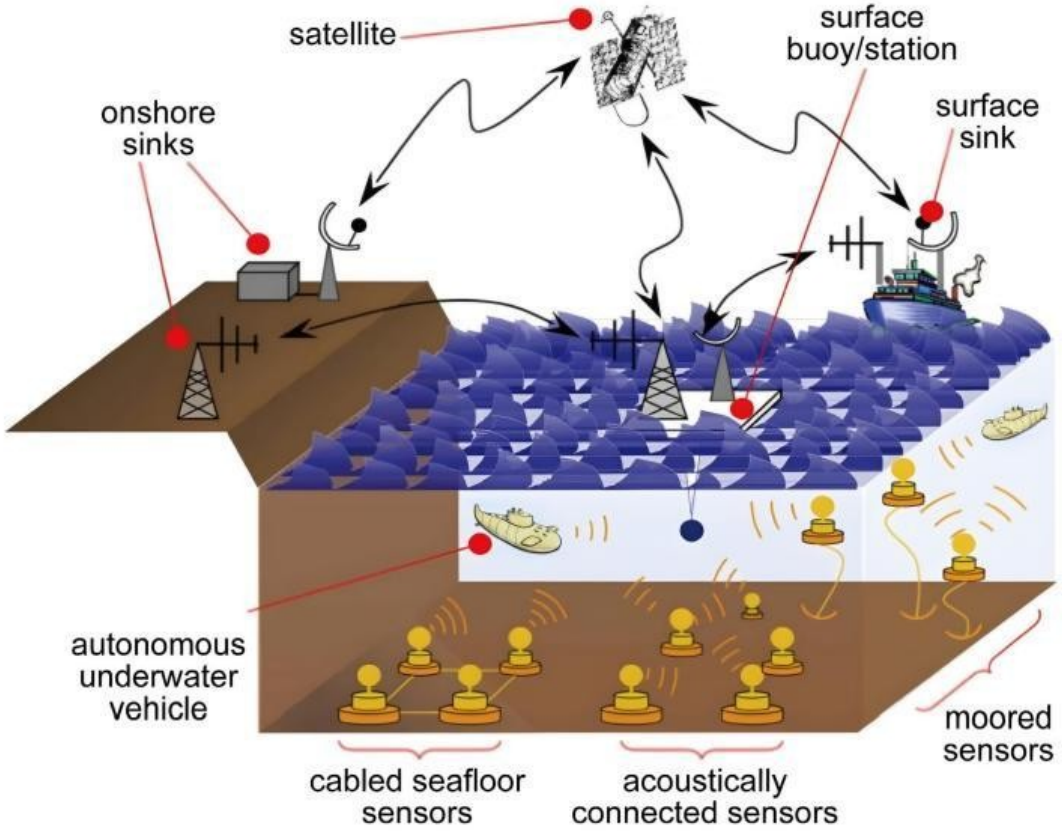
La nouvelle génération de dispositifs autonomes – tels que les drones marins, les AUVs (Autonomous Underwater Vehicles) ou encore les gliders – repose sur des modes de communication hybrides :

- Par satellite lorsqu'ils sont en surface,
- Par acoustique lorsqu'ils sont immergés,
- Ou par l'intermédiaire de bouées relais.

Ces plateformes fonctionnent avec des ressources très limitées : microcontrôleurs à faible puissance, mémoire restreinte, énergie réduite. Il devient donc impératif d'optimiser les algorithmes cryptographiques, en utilisant :

- Des variantes compactes de AES,
- Ou des algorithmes ultra-légers comme PRESENT, SPECK ou XTEA [**Poschmann, A. Y. (2009)**].

Par ailleurs, dans les installations industrielles offshore (systèmes SCADA maritimes), des équipements critiques comme des pompes, vannes ou capteurs de pression sont souvent pilotés à distance via des protocoles anciens (ex : Modbus/TCP), dépourvus de toute sécurité. L'ajout de passerelles chiffrées devient alors indispensable.



III.2.6 Protocoles marins (NMEA 0183, NMEA 2000, IEC 61162)

Les protocoles issus de la norme NMEA occupent une place centrale dans l'écosystème des instruments de navigation maritime. La version NMEA 0183, historiquement plus ancienne, repose sur une transmission de type série, tandis que le standard plus moderne, NMEA 2000, s'appuie sur un réseau de type bus CAN, offrant ainsi la possibilité d'échanger efficacement une variété de données, telles que :

- Données GPS,
- Les retours des systèmes radar,
- Ou encore les relevés de capteurs environnementaux comme les sondeurs de profondeur, les compas électroniques ou les girouettes.

Toutefois, ces protocoles ont été conçus à une époque où les enjeux de sécurité informatique n'étaient pas encore au cœur des préoccupations. En conséquence, ils présentent des failles notables : les données transmises peuvent être interceptées, modifiées, voire falsifiées sans grande difficulté. Face à ces vulnérabilités, plusieurs efforts récents visent à renforcer leur sécurité, notamment par l'intégration de mécanismes de chiffrement symétrique, tels qu'AES, ou encore de signatures numériques reposant sur les courbes elliptiques, comme ECDSA, afin de garantir l'authenticité et l'intégrité des informations échangées [**Wood, A. D., & Stankovic, J. A. (2002)**].

III.3 – Enjeux de sécurité dans le monde maritime

III.3.1 Pourquoi la cybersécurité maritime est-elle un enjeu stratégique ?

Dans les représentations traditionnelles, les menaces en mer sont encore souvent associées aux attaques physiques, comme les actes de piraterie ou les abordages armés. Pourtant, à l'ère de la numérisation croissante des systèmes embarqués, une menace bien plus insidieuse s'est imposée : celle des cyberattaques ciblant les infrastructures maritimes.

Les navires contemporains ne sont plus de simples moyens de transport : ils se rapprochent désormais de véritables plateformes informatiques flottantes. Ils embarquent une mosaïque de systèmes interconnectés, capteurs environnementaux, ordinateurs industriels, modules de communication satellite, interfaces SCADA, dispositifs de navigation automatisée, tous dépendants de réseaux numériques internes et externes. Cette interconnexion, bien que cruciale pour l'efficacité opérationnelle, constitue également une surface d'attaque particulièrement vulnérable.

Dans ce contexte, la sécurité ne concerne plus uniquement la protection physique des cargaisons ou de l'équipage, mais s'étend à des enjeux bien plus vastes. Une compromission informatique peut affecter :

- Des intérêts économiques (perturbation de chaînes logistiques),
- Les objectifs environnementaux (modification ou perte de données sensibles sur la biodiversité marine),
- Voire des aspects stratégiques (détournement de drones sous-marins, manipulation de trajectoires via spoofing GPS).

Le simple fait d'altérer un signal de navigation, d'exploiter une faille dans un système de pilotage automatique, ou de prendre le contrôle d'un équipement connecté peut suffire à provoquer des incidents majeurs, parfois irréversibles. Cette réalité impose donc une refonte des priorités sécuritaires dans le secteur maritime, en intégrant la cybersécurité comme une composante fondamentale de la résilience en mer.

III.3.2 Types d'attaques informatiques maritimes : illustrations concrètes

a) L'interception : l'oreille d'Eve dans les flots

Imaginons un scénario typique : Alice pilote un drone océanographique depuis un navire de recherche. Ses instructions sont envoyées par un canal acoustique. À proximité, Eve, l'attaquante passive, a discrètement installé un hydrophone. Elle ne perturbe pas la transmission, mais capte et analyse les messages en transit.

Cette forme d'interception est fréquente dans les réseaux acoustiques sous-marins (UASNs), où les signaux, diffusés dans toutes les directions à faible débit, sont aisément interceptables [**Xie, P., et al (2009, October 1)**].

b) Le spoofing : quand Bob devient... Eve

Dans une variante plus pernicieuse, Eve ne se contente plus d'écouter. Elle se fait passer pour Alice en envoyant un faux message à Bob. Cela peut prendre la forme d'une injection de données falsifiées dans le système AIS, par exemple, déclenchant une alerte de collision inexistante.

Ce type d'attaque, dit de spoofing, a été observé à plusieurs reprises. En juin 2017, certains navires au large de la mer Noire ont ainsi vu leur position GPS falsifiée, apparaissant virtuellement... sur le tarmac de l'aéroport de Sotchi [**Virginia, A. (2017)**].

c) Le brouillage (jamming) et le déni de service (DoS)

Parmi les menaces numériques les plus redoutées en milieu maritime figurent le brouillage (ou jamming) et les attaques par déni de service (DoS), deux techniques visant à perturber, voire à interrompre totalement, les communications ou le fonctionnement des systèmes critiques.

Le brouillage consiste à inonder un canal de transmission avec des signaux parasites, rendant impossible toute communication légitime. Ce type d'attaque peut cibler différents types de liaisons : signaux GNSS utilisés pour la navigation, fréquences radio dédiées aux échanges inter-bateaux, ou encore canaux acoustiques exploités dans les réseaux sous-marins. Les zones côtières à haute densité d'infrastructures, comme les ports ou les installations industrielles offshore, sont particulièrement exposées à ce risque.

Le déni de service, quant à lui, repose sur une stratégie différente mais tout aussi déstabilisante : il s'agit de surcharger délibérément un système informatique, comme un serveur SCADA, une station radar, ou une plateforme de collecte de données océanographiques, jusqu'à ce qu'il devienne incapable de fonctionner normalement. Ces attaques ne détruisent pas physiquement les équipements, mais les mettent hors service par saturation, bloquant ainsi temporairement toute opération critique. Ces techniques ont été documentées expérimentalement en mer, par exemple lors d'une campagne organisée en mer Baltique : des tests de brouillage GNSS y ont montré une interruption totale du service de navigation pendant plusieurs minutes, affectant à la fois le positionnement GNSS et les interfaces reliant AIS ou ECDIS à bord [**Medina, D., et al. (2019, July)**].

III.3.3 Enjeux spécifiques aux systèmes critiques marins

Les impacts de ces cybermenaces sont amplifiés par la nature isolée et peu accessible des systèmes embarqués en mer, souvent conçus avec des ressources limitées en énergie, mémoire et traitement.

Quelques exemples illustrent cette vulnérabilité :

- **Navigation autonome** : des drones ou planeurs sous-marins peuvent être déviés ou désactivés à distance.
- **Collecte de données environnementales** : leur falsification peut compromettre les prévisions météorologiques ou les modèles océaniques.
- **Commandes industrielles offshore (SCADA)** : une attaque sur ces systèmes peut affecter le fonctionnement de plateformes pétrolières, parcs éoliens marins ou terminaux portuaires [**Mohammed, A. S., et al. (2022)**].

III.3.4 Enjeux militaires et géopolitiques

Le domaine maritime ne se limite plus aujourd'hui aux seules manœuvres physiques : il est devenu un espace stratégique numérique à part entière. Dans un climat international de plus en plus tendu, les opérations militaires en mer mobilisent des systèmes interconnectés de grande complexité, impliquant à la fois des navires de surface, des drones autonomes, des

satellites de surveillance et des sous-marins. Cette architecture distribuée, bien qu'efficace, repose sur une chaîne de communication vulnérable : l'atteinte d'un seul maillon, qu'il s'agisse d'une interception, d'une altération ou d'un sabotage, peut suffire à compromettre l'efficacité, voire la confidentialité, d'une mission entière.

Face à cette menace diffuse, les grandes puissances maritimes adoptent des protocoles de sécurité cryptographique hautement sophistiqués, dont la majorité relève de la classification militaire. À titre d'illustration, les échanges acoustiques entre sous-marins nucléaires sont encadrés par des schémas de chiffrement spécifiquement conçus pour résister aux conditions extrêmes des milieux sous-marins : forte atténuation du signal, bruit ambiant, délais de propagation. Ces communications reposent généralement sur des algorithmes symétriques à faible latence, couplés à des mécanismes d'authentification renforcée, afin de garantir à la fois la discrétion, l'intégrité et l'exclusivité des messages transmis [**Zhou, Q., et al. (2025)**].

III.3.5 Tableau récapitulatif des menaces et contre-mesures

Type de menace	**Exemple concret**	**Système ciblé**	**Contre-mesure cryptographique**
Interception passive	Écoute via hydrophone	Réseaux UASN	Chiffrement symétrique: AES, ChaCha20
Spoofing (usurpation)	Faux messages AIS / GPS falsifié	AIS, GNSS	Authentification forte, signatures numériques
Jamming / brouillage	Saturation des canaux GNSS ou radio	GNSS, liaisons sans fil	Protocoles tolérants au bruit, redondance
Déni de service (DoS)	Requêtes massives vers un serveur SCADA	SCADA offshore	Filtrage, contrôle d'intégrité, surveillance active
Espionnage militaire	Surveillance de communications sous-marines	Réseaux acoustiques classifiés	Algorithmes robustes, ECC, courbes spécifiques

III.4 – Applications de la cryptographie symétrique en environnement marin

III.4.1 AES (Advanced Encryption Standard) :

Imaginez Alice, ingénieure cybersécurité sur un navire scientifique, qui collecte des mesures sensibles au large, pendant que Bob, son collègue, attend les résultats sur la côte. Leur mission ? S'assurer que chaque donnée transmise, température, salinité, position, reste confidentielle malgré la présence d'Eve, pirate numérique à l'affût des communications maritimes. Pour cela, Alice s'appuie sur l'AES, le standard de chiffrement symétrique, utilisé dans tous les grands systèmes navals modernes.

1.2 Principe mathématique d'AES (Advanced Encryption Standard)

La force d'AES réside dans sa capacité à transformer chaque message en une énigme mathématique impossible à résoudre sans la clé secrète partagée entre Alice et Bob.

1. Chiffrement par blocs

AES divise le message clair M en blocs de 128 bits (16 octets). Chaque bloc est chiffré séparément :

$$C = AES_K(M)$$

où K est la clé secrète (128, 192 ou 256 bits).

2. Succession de "rondes" mathématiques

Chaque bloc subit une succession de transformations, appelées rondes (10 pour AES-128), qui appliquent chacune quatre opérations fondamentales :

- **SubBytes (S-box)** : Chaque octet est remplacé par son image à travers une substitution non linéaire.
    - Mathématiquement :
        i. Inversion dans le corps fini $GF(2^8)$
        ii. Transformation affine (opérations XOR et ajout de bits constants)
- **ShiftRows** : Les lignes du bloc sont décalées pour mélanger les positions.
- **MixColumns** : Les colonnes sont combinées via une multiplication matricielle dans $GF(2^8)$, diffusant l'information sur tout le bloc.
- **AddRoundKey** : À chaque étape, le bloc est combiné (par XOR) avec une sous-clé dérivée de la clé principale.

Formellement, pour chaque ronde i :

$$État_{i+1} = AddRoundKey(MixColumns(ShiftRows(SubBytes(État_i))))$$

La dernière ronde omet le MixColumns.

3. Vue schématique et pédagogique

> "Chaque bloc de mon message est mélangé, brouillé, et passé dans des séries de substitutions et permutations. Même si tu changes une virgule dans le message ou un bit dans la clé, le résultat final est totalement différent. C'est cette chaîne d'opérations, injectant notre secret à chaque étape, qui rend le message illisible pour Eve."

Schéma simplifié :

Message clair (bloc de 16 octets)
↓

XOR avec la sous-clé initiale
↓
[10 rondes] : S-box → ShiftRows → MixColumns → AddRoundKey
↓
Message chiffré (bloc de 16 octets)

4. Illustration mathématique

Supposons qu'Alice envoie "SALINITE: 35.7 PSU", converti en hexadécimal :

53414C494E4954453A2033352E372050

- **XOR avec IV** (en mode CBC)
- **Passage dans la S-box** : chaque octet, comme 0x53 ("S"), devient 0xED
- **Décalage des lignes (ShiftRows)** et mélange des colonnes (MixColumns)
- **Ajout de la sous-clé** à chaque ronde (issue de la clé principale "Thats my Kung Fu")

Après dix cycles, le bloc devient : B6E2F81A3CDA741B7A1D4F1CE7A4C6B2 impossible à interpréter sans la clé.

1.3 La S-box AES : le "grand brouilleur"

La S-box d'AES joue le rôle de labyrinthe mathématique :

- Pour chaque octet, elle effectue d'abord une inversion multiplicative dans le corps fini $GF(2^8)$, puis une transformation affine qui mêle les bits.
- Cette opération garantit qu'un tout petit changement dans le message d'Alice ou la clé partagée change radicalement le message chiffré reçu par Bob.
- Même si Eve capture des milliers de messages, elle ne peut retrouver le chemin du message initial.

Alice confie à Bob : "Tu vois, la S-box, c'est un filtre qui rend chaque lettre méconnaissable pour Eve, mais facile à retransformer si tu as le bon secret."

1.4 Exemple pratique d'utilisation d'AES à bord

Considérons le cas suivant : Alice doit transmettre à Bob une donnée sensible : la salinité mesurée en pleine mer.

Paramètres cryptographiques

- **Message à transmettre** : « SALINITE: 35.7 PSU »
- **Clé secrète (AES-128, 128 bits)** : $K = 5468617473206D79204B756E67204675$

- **Vecteur d'initialisation (IV, mode CBC)** :
  $IV = 000102030405060708090A0B0C0D0E0F$

Étape 1 : Encodage du message

Dans la chaîne de transmission, c'est Alice qui doit préparer le message destiné à Bob. Cette étape, pourtant simple, est essentielle car une erreur d'encodage ou d'ordre de caractères compromettrait la compréhension de Bob lors du déchiffrement.

Exemple concret : Alice veut transmettre à Bob : « SALINITE: 35.7 PSU »

Elle procède ainsi :

Position	Caractère (par Alice)	Code ASCII (décimal)	Hexadécimal	Rôle
1	S	83	0x53	Début du mot "SALINITE"
2	A	65	0x41	...
3	L	76	0x4C	...
4	I	73	0x49	...
5	N	78	0x4E	...
6	I	73	0x49	...
7	T	84	0x54	...
8	E	69	0x45	Fin du mot "SALINITE"
9	:	58	0x3A	Séparateur
10	(espace)	32	0x20	...
11	3	51	0x33	Début de la valeur
12	5	53	0x35	...
13	.	46	0x2E	...
14	7	55	0x37	...
15	(espace)	32	0x20	...
16	P	80	0x50	Début de "PSU"

Remarque : Si le message excède 16 octets (par exemple, pour inclure aussi "SU"), Alice prépare un deuxième bloc. Elle complète, si besoin, le bloc avec un padding (remplissage) selon la convention (exemple : PKCS#7), afin que Bob puisse toujours reconstituer exactement le message d'origine après déchiffrement.

Quand Bob reçoit le flux chiffré, il sait que chaque octet correspond à un caractère préparé ainsi par Alice, ce qui lui permet de reconstruire le message sans ambiguïté.

$$M = 53414C494E4954453A2033352E372050$$

Étape 2 : Chaînage CBC (XOR avec IV)

Alice réalise un XOR octet à octet entre M et IV :

Position	M	IV	$X = M \oplus IV$
1	0x53	0x00	0x53
2	0x41	0x01	0x40
3	0x4C	0x02	0x4E
...	...	...	...
16	0x50	0x0F	0x5F

Résultat final du XOR (bloc de 16 octets) :

$$X = 53404E4A4A4C52423229393E223A2E5F$$

Étape 3 : Chiffrement AES-128 (10 rondes)

Alice chiffre ce bloc X à l'aide de la clé K selon l'algorithme AES-128 :

1. **SubBytes** : chaque octet est remplacé par la valeur de la S-box (calcul : inversion dans $GF(2^8)$, puis transformation affine binaire).
2. **ShiftRows** : les lignes du bloc (matrice 4×4) sont décalées pour une meilleure diffusion.
3. **MixColumns** : chaque colonne subit une multiplication par une matrice fixe dans $GF(2^8)$.
4. **AddRoundKey** : XOR du bloc courant avec une sous-clé dérivée de K.

Ce cycle est répété 10 fois pour AES-128. À chaque ronde, le mélange se renforce mathématiquement.

- **Sortie du chiffrement :**

$$C = B6E2F81A3CDA741B7A1D4F1CE7A4C6B2$$

(valeur issue d'un chiffreur AES standard, CBC, 128 bits)

Étape 4 : Transmission

Alice transmet C à Bob via le canal radio. Même si Eve intercepte C, sans K, il est mathématiquement impossible de retrouver M.

Étape 5 : Déchiffrement par Bob

Bob reçoit C, et grâce à la connaissance de K et de IV, procède à :

1. **Déchiffrement AES inverse** pour retrouver X :

$$X = AE\ S_K^{-1}(C)$$

2. **XOR avec l'IV** pour retrouver M :

$$M = X \oplus IV$$

Il obtient alors le message clair original : « SALINITE: 35.7 PSU ».

Points techniques à retenir

- **Confidentialité garantie** : sans K, aucune analyse mathématique ou brute force n'est réaliste pour retrouver M.
- **Diffusion assurée** : même un bit changé dans M ou K modifie complètement C (propriété d'avalanche).
- **Utilisation correcte du mode CBC** : chaque message identique donne un chiffré différent grâce à IV.

1.5 Sécurité d'AES face aux attaques :

La robustesse d'AES s'évalue à la lumière de nombreux scénarios d'attaque. Pour mieux comprendre, suivons le duo Alice et Bob, qui cherchent à sécuriser leurs échanges marins, pendant qu'Eve, l'adversaire, tente d'intercepter ou de casser le chiffrement.

1. Attaque par force brute : la muraille exponentielle

Situation pédagogique : Alice chiffre ses messages avec une clé AES-128 partagée avec Bob. Eve, la pirate numérique, intercepte un message chiffré et tente toutes les clés possibles.

- **Réalité mathématique :** L'espace des clés possibles est de 2^{128} (soit un nombre dépassant le nombre d'atomes dans l'univers observable). Même si Eve disposait d'un supercalculateur maritime fonctionnant à pleine puissance pendant des millions d'années, la probabilité de retrouver la clé d'Alice et Bob resterait insignifiante [**Morris, D. (2001)**].

2. Cryptanalyse différentielle et linéaire : l'art de l'analyse mathématique

Exemple pédagogique : Supposons qu'Eve, plutôt que d'essayer toutes les clés, collecte des paires de messages où Alice a changé un seul bit dans son capteur, puis compare les chiffrés obtenus.

- **Principe mathématique :** Elle espère que, si la structure d'AES laissait "fuiter" des motifs, elle pourrait deviner la clé ou le fonctionnement interne.
- **Pourquoi ça ne marche pas :** Grâce à la S-box et à la diffusion maximale (chaque bit modifié impacte l'ensemble du bloc), la moindre modification dans le message d'Alice provoque une avalanche de changements dans le texte chiffré, rendant toute corrélation inopérante. En pratique, même avec des milliers d'essais, Eve ne découvre aucune relation exploitable entre les bits de sortie et ceux d'entrée [**Bernstein, D. J. (2005)**].

3. Clés faibles et collisions : la paranoïa mathématique d'Alice

Question que Bob pourrait poser :

“Et si, par malchance, la clé qu’on utilise avait une faiblesse structurelle ?”

- **Réalité :** AES a été rigoureusement étudié pour éviter ce piège. À ce jour, aucune clé faible ni collision exploitable n’a été découverte, quelle que soit la combinaison choisie par Alice et Bob pour leurs échanges [**Bernstein, D. J. (2005)**].

4. Attaques par canaux auxiliaires : la guerre cachée

Contexte : Bob installe une nouvelle puce de chiffrement sur son drone. Eve ne s’attaque pas à l’algorithme, mais tente d’observer la consommation électrique ou le temps d’exécution de la puce, espérant deviner la clé.

- **Réponse mathématique et technique :** Le chiffrement AES lui-même reste sûr, mais Alice et Bob doivent s’assurer que leur matériel implémente des contre-mesures : masquage des calculs, égalisation des temps d’exécution, blindage physique [**Swarup Bhunia, & Tehranipoor, M. H. (2019)**].
- **À retenir :** Même la meilleure formule mathématique exige une discipline opérationnelle. Pour Alice et Bob, la sécurité ne se limite pas à la théorie : le matériel doit être aussi bien pensé que l’algorithme.

5. Informatique quantique : la vigilance d’Alice pour demain

Bob s’informe des avancées en informatique quantique et interroge Alice :

“Si Eve avait un ordinateur quantique, serions-nous en danger ?”

- **Réponse mathématique :** L’algorithme de Grover pourrait diviser par deux la sécurité théorique d’AES. Pour AES-128, cela donnerait 2^{64} opérations à Eve : largement hors de portée pour le moment, mais Alice et Bob peuvent choisir AES-256 pour parer toute évolution future [**Chen, L., et al. (2016)**].

6. Bonnes pratiques pour Bob et Alice

- Toujours choisir une clé de taille adaptée (AES-256 pour les systèmes les plus sensibles).
- Renouveler régulièrement la clé, comme on ravitaille un navire.
- Ne jamais utiliser des modes de chiffrement faibles (ECB), même pour des données “anodines”.
- Utiliser uniquement des modules matériels certifiés, protégés contre les fuites physiques.

Pour Alice et Bob, la sécurité d’AES ne repose pas sur le secret de l’algorithme (il est public), mais sur la solidité de ses fondements mathématiques et sur une gestion rigoureuse

au quotidien. Eve pourra intercepter autant de messages qu'elle veut : sans la clé, le chemin reste inaccessible.

Implémentation navale et gestion des clés

1. Contraintes du milieu maritime

En environnement marin, la cryptographie symétrique (et l'AES en particulier) s'implémente souvent sur des équipements soumis à des contraintes :

- **Ressources matérielles limitées** (capteurs autonomes, microcontrôleurs, drones sous-marins)
- **Isolement physique** : pas d'accès à Internet pendant des jours
- **Risques de compromission physique** (vol de matériel, abordage, perte d'un capteur)
- **Enjeux d'autonomie énergétique** (tout calcul doit être économe en énergie)

2. Intégration matérielle d'AES à bord

Pour assurer des communications sécurisées, la cryptographie AES est embarquée dans :

- **Capteurs marins autonomes** (température, salinité, pression, pollution)
- **Systèmes embarqués sur drones, AUV (véhicules sous-marins autonomes), gliders**
- **Systèmes SCADA offshore** (supervision d'installations énergétiques, de pompes, de vannes)
- **Stations de communication navire-terre**

Réalité technique :

- La plupart des microcontrôleurs modernes intègrent une implémentation matérielle ou logicielle d'AES [**Swarup Bhunia, & Tehranipoor, M. H. (2019)**].
- Les modules matériels (TPM, HSM) protègent la clé contre l'extraction même en cas d'accès physique au matériel.
- Des bibliothèques logicielles optimisées (ex. TinyAES, mbedTLS) existent pour les plateformes embarquées.

Exemple : Alice configure un nouveau drone océanographique. Lors de la configuration au port, elle charge la clé AES (K) dans le module sécurisé du drone. Bob, à la station, enregistre la même clé.

3. Gestion des clés en milieu maritime

Dans les environnements marins, la gestion des clés de sécurité constitue un maillon essentiel de la chaîne de confiance, avec des exigences spécifiques qui s'écartent notablement des pratiques habituelles des réseaux terrestres :

a) Génération et distribution initiale

La phase de création des clés s'effectue généralement dans un environnement à terre, strictement contrôlé [**National Institute of Standards And Technology. (2020)**]. Une fois générées, les clés peuvent être transmises de manière physique lors de l'avitaillement ou pendant la préparation d'une mission. Cela inclut, par exemple, la remise de dispositifs sécurisés comme des cartes à puce, des clés USB chiffrées, ou encore la saisie manuelle de séquences confidentielles [**National Institute of Standards And Technology. (2020)**].

Dans les cas où une distribution à distance est nécessaire, notamment pour des systèmes déployés en mer, on privilégie des protocoles de cryptographie asymétrique, comme Diffie-Hellman sur courbes elliptiques (ECDH), afin de garantir la confidentialité de l'échange malgré la présence d'un canal potentiellement compromis [**Rivest, R. L., Shamir, A., & Adleman, L. (1978)**, **Koblitz, N. (1987)**].

b) Stockage sécurisé

Une fois en place, les clés ne doivent jamais être conservées en clair dans la mémoire d'un dispositif. Elles sont stockées dans des modules sécurisés tels que des TPM (Trusted Platform Module), des HSM (Hardware Security Modules) ou encore des circuits intégrés dédiés à la sécurité [**Swarup Bhunia, & Tehranipoor, M. H. (2019)**]. L'accès à ces clés est strictement restreint, nécessitant une authentification forte — mot de passe, badge personnel, ou encore biométrie locale [**National Institute of Standards And Technology. (2020)**].

c) Rotation et renouvellement

La validité d'une clé ne doit jamais être considérée comme permanente. Des procédures de rotation régulière sont mises en place, que ce soit à l'échelle d'une mission ou selon une périodicité prédéfinie (par exemple chaque semaine sur une plateforme offshore à haute criticité) [**National Institute of Standards And Technology. (2020)**]. Afin d'anticiper d'éventuelles compromissions, plusieurs jeux de clés peuvent être préchargés et activés à distance au besoin [**Mohammed, A. Set al. (2022)**, **National Institute of Standards And Technology. (2020)**]. En cas de perte ou d'intrusion détectée, un processus documenté de suppression sécurisée (ou *zeroization*) permet d'effacer immédiatement la clé stockée dans l'équipement [**Swarup Bhunia, & Tehranipoor, M. H. (2019)**, **National Institute of Standards And Technology. (2020)**].

d) Revocation et destruction

Des mécanismes de contingence sont prévus pour réagir rapidement en cas d'attaque avérée. À titre d'exemple, un opérateur ("Bob") peut déclencher une commande d'urgence, ordonnant l'effacement de la clé active sur tous les dispositifs à bord et l'activation d'un jeu de secours déjà présent [**Mohammed, A. S., et al. (2022)**, **National Institute of Standards And Technology. (2020)**].

4. Bonnes pratiques en environnement naval

Certaines règles simples mais fondamentales permettent de renforcer considérablement la résilience des systèmes cryptographiques marins :

- **Limiter la durée de vie d'une clé** : plus une clé est utilisée longtemps, plus elle est susceptible d'être exposée à une attaque ou à une fuite [**National Institute of Standards And Technology. (2020)**].
- **Éviter toute réutilisation de clés ou de vecteurs d'initialisation (IV)** : chaque session ou canal de communication devrait idéalement s'appuyer sur un IV unique, et si possible sur une **clé de session spécifique** [**Morris, D. (2001)**, **Dworkin, M. J. (2024)**].
- **Assurer la sécurité de la chaîne humaine** : former les opérateurs (Alice, Bob, etc.) à la manipulation correcte des clés, à la détection d'incidents, ainsi qu'aux procédures d'effacement, constitue un volet essentiel de la sécurité globale [**Stallings., William. (2023)**].
- **Favoriser la traçabilité** : les actions critiques (accès, changement, effacement de clés) doivent être journalisées de manière rigoureuse afin de garantir l'auditabilité et d'identifier les défaillances éventuelles [**National Institute of Standards And Technology. (2020)**].

5. Limites, risques et recommandations

Même avec des algorithmes robustes, la sécurité repose sur la qualité de la mise en œuvre :

- Le risque majeur reste la **perte ou l'extraction non autorisée d'une clé**, ce qui compromettrait non seulement les données en cours de transmission, mais également celles échangées par le passé si la clé n'a pas été renouvelée [**National Institute of Standards And Technology. (2020)**].
- En cas **d'attaque physique**, un appareil ne disposant pas de module sécurisé est vulnérable à des méthodes comme le *cold boot attack*, la lecture directe de la mémoire ou d'autres formes d'extraction [**Swarup Bhunia, & Tehranipoor, M. H. (2019)**].
- **La discipline opérationnelle est déterminante** : un algorithme bien implémenté ne peut à lui seul compenser une négligence humaine ou une procédure mal appliquée [**Stallings., William. (2023)**].

Résumé

La résilience numérique dans le domaine maritime repose fondamentalement sur deux axes complémentaires :

- La **robustesse mathématique** des algorithmes comme **AES**, largement éprouvée [**NIST. (2001)**, **Daemen, J., & Rijmen, V. (2002)**],

- Et une **gestion rigoureuse, préventive et adaptée des clés**, tenant compte des spécificités du contexte marin : isolement prolongé, autonomie énergétique, risques physiques et nécessité de réponses rapides en cas de crise [**Mohammed, A. S., et al. (2022)**, **National Institute of Standards And Technology. (2020)**].

Dans une telle perspective, la sécurité ne dépend pas seulement du chiffrement, mais aussi d'une stratégie cohérente et rigoureuse autour de la vie de la clé.

Conseils pratiques et vigilance : garantir l'efficacité d'AES en environnement maritime

Même lorsqu'on utilise un algorithme fiable comme AES, l'efficacité réelle de la protection repose sur des choix d'implémentation bien adaptés aux contraintes du terrain :

1. Bien choisir le mode de chiffrement

AES chiffre des blocs de données : il convient donc de sélectionner un **mode de chiffrement** en accord avec la nature du flux à protéger :

- **CBC (Cipher Block Chaining)** : chaque bloc chiffré dépend du précédent, ce qui empêche deux messages identiques d'avoir le même résultat chiffré si l'IV change. Ce mode est couramment utilisé pour des fichiers ou paquets successifs [**Morris, D. (2001)**, 19 **Dworkin, M. J. (2024)**].

- **CTR (Counter)** : permet de chiffrer ou déchiffrer les blocs en parallèle, offrant une excellente performance, notamment dans les systèmes embarqués nécessitant un traitement rapide [**Morris, D. (2001)**, 20].

- **GCM (Galois/Counter Mode)** : combine chiffrement et authentification des données, ce qui garantit à la fois confidentialité et intégrité, un critère essentiel dans les réseaux critiques où la falsification de commandes ou de données peut avoir des conséquences graves [**Morris, D. (2001)**, **Dworkin, M. J. (2024)**].

Ne jamais utiliser le mode ECB:

Le mode ECB (Electronic Codebook), bien que simple, doit être proscrit : il laisse transparaître les structures répétitives des données d'origine, exposant ainsi des informations sensibles, même sans accès à la clé [**Dworkin, M. J. (2024)**].

2. Le rôle critique de l'IV : unicité et non-répétition

Dans les modes de chiffrement comme CBC, CTR ou GCM, le vecteur d'initialisation (IV) joue un rôle fondamental dans la sécurité du processus. Pour garantir la confidentialité des communications, l'IV doit impérativement :

- être **unique et imprévisible** à chaque nouvelle session ou message [**Morris, D. (2001)**, **Dworkin, M. J. (2024)**],
- **ne jamais être réutilisé** avec une même clé de chiffrement [**Morris, D. (2001)**, **Dworkin, M. J. (2024)**].

Sur le plan cryptographique, une réutilisation d'IV avec la même clé permettrait à un attaquant d'établir des corrélations entre différents messages chiffrés, compromettant ainsi leur secret [**Dworkin, M. J. (2024)**], [**Stallings., William. (2023)**]. Même dans le cas de messages en apparence anodins, cette précaution est essentielle : c'est dans les détails que se loge la faille exploitable.

3. La gestion correcte du remplissage (padding)

AES opère sur des blocs de 16 octets. Lorsqu'un message ne correspond pas à un multiple de cette taille, il devient nécessaire d'ajouter un remplissage (padding), le plus souvent selon la norme PKCS#7 [**Morris, D. (2001)**]. Toute erreur dans la gestion du padding, que ce soit lors de l'insertion ou du retrait, peut entraîner :

- une perte d'intégrité des données,
- voire l'exposition à des attaques de type padding oracle [**Stallings., William. (2023)**].

Bon réflexe : toujours valider la cohérence du padding côté récepteur avant de traiter le contenu du message [**Stallings., William. (2023)**].

4. Rotation régulière des clés et anticipation des incidents

La rotation régulière des clés est une bonne pratique indispensable :

- Il est recommandé de changer de clé à chaque mission, ou dès qu'une compromission est suspectée, afin de limiter l'impact d'une éventuelle fuite [**National Institute of Standards And Technology. (2020)**].
- Des clés de secours préchargées doivent être disponibles à bord, accompagnées de procédures de suppression immédiate (*zeroization*) en cas de perte ou d'intrusion sur un équipement [**Mohammed, A. S., et al. (2022)**, **National Institute of Standards And Technology. (2020)**].

5. Sécuriser le stockage et la manipulation des clés

Une clé bien protégée est une clé qui n'est jamais stockée en clair, ni dans la mémoire vive, ni sur un support non protégé [**Swarup Bhunia, & Tehranipoor, M. H. (2019)**, **National Institute of Standards And Technology. (2020)**]. Pour cela :

- il convient d'utiliser des modules matériels dédiés à la sécurité, tels que TPM, HSM, ou cartes à puce, qui empêchent toute extraction, même en cas d'accès physique à l'appareil [**Swarup Bhunia, & Tehranipoor, M. H. (2019)**, **National Institute of Standards And Technology. (2020)**],
- l'accès à ces clés doit être restreint par des mécanismes d'authentification forte, tels que le double facteur, les certificats numériques, ou la biométrie [**National Institute of Standards And Technology. (2020)**].

6. Utiliser des implémentations fiables et auditées

Le choix de la bibliothèque cryptographique est tout sauf anodin. Il est impératif d'utiliser des solutions reconnues et auditées, comme mbedTLS, OpenSSL, WolfSSL, ou encore

TinyAES dans les environnements embarqués [**Swarup Bhunia, & Tehranipoor, M. H. (2019)**, **Stallings., William. (2023)**].

Bonnes pratiques :

- Suivre les mises à jour de sécurité de manière proactive,
- Appliquer sans délai les correctifs critiques [**Stallings., William. (2023)**].

7. Former et sensibiliser tous les acteurs

Même le plus robuste des algorithmes devient vulnérable face à une erreur humaine. La formation continue des équipes, Alice, Bob et tous les opérateurs, est essentielle pour éviter les erreurs courantes telles que :

- transmettre une clé par un canal non sécurisé [**Stallings., William. (2023)**, **National Institute of Standards And Technology. (2020)**],
- oublier d'effacer une clé à la fin d'une mission [**National Institute of Standards And Technology. (2020)**],
- utiliser un mode de chiffrement inadapté ou un IV réutilisé [**Morris, D. (2001)**, **Dworkin, M. J. (2024)**].

8. Tester et auditer régulièrement les systèmes

Les audits de sécurité, qu'ils soient internes ou réalisés par un tiers, sont indispensables pour garantir que les bonnes pratiques sont bien appliquées dans la durée [**National Institute of Standards And Technology. (2020).**]. Il est également recommandé de simuler des scénarios d'incident, comme la perte d'un appareil ou la compromission d'une clé, afin de tester la réactivité des procédures (effacement d'urgence, bascule de clé, etc.) [**Mohammed, A. S., et al. (2022).**, **National Institute of Standards And Technology. (2020)**].

À retenir :

Un algorithme comme AES, aussi solide soit-il, n'assure à lui seul aucune sécurité. Tout dépend de la qualité de son implémentation et du soin apporté à ce qui l'entoure :

- choix du mode de chiffrement,
- gestion correcte du padding,
- rotation des clés,
- sécurisation du stockage,
- formation des opérateurs…

Chaque détail technique représente une couche supplémentaire de défense. La sécurité opérationnelle ne repose pas sur un geste isolé, mais sur un système cohérent, où l'humain, la procédure et la technologie agissent en synergie [**Stallings., William. (2023)**, **NIST. (2001)**].

AES en mer :

Pour Alice et Bob, AES n'est pas une boîte noire magique. C'est un outil mathématique rigoureux, qui, lorsqu'il est bien maîtrisé, transforme un message clair en un code

incompréhensible pour Eve, tout en restant rapide, léger et adapté aux contraintes techniques du monde maritime [**NIST. (2001)**, **Daemen, J., & Rijmen, V. (2002)**].

C'est ainsi que la science cryptographique devient un bouclier, discret mais décisif, dans le tumulte invisible des menaces numériques qui planent au-dessus de nos océans.

III.4.2 ChaCha20 : alternative légère et rapide pour les microcontrôleurs embarqués

Dans les systèmes marins embarqués, nombreux sont les dispositifs, capteurs autonomes, balises de localisation, microcontrôleurs basse consommation ou encore drones de surface et sous-marins, à fonctionner dans un environnement particulièrement contraint. Ces équipements disposent en général de capacités de calcul, de mémoire et d'énergie très limitées, ce qui rend l'intégration de solutions de sécurité classiques particulièrement délicate.

Bien que l'algorithme AES demeure la norme incontournable en matière de chiffrement symétrique, son implémentation peut s'avérer coûteuse en ressources dans de tels contextes. C'est pourquoi des alternatives plus adaptées aux environnements embarqués ont été développées. ChaCha20, en particulier, se distingue comme une solution de choix : il conjugue excellente robustesse cryptographique et efficacité remarquable, notamment sur des architectures dépourvues d'accélération matérielle. Grâce à sa conception optimisée, cet algorithme peut être déployé aisément sur des microcontrôleurs modestes, tout en assurant un niveau de sécurité élevé et une très faible latence.

III.4.2.1 Principes cryptographiques de ChaCha20

A. Vue d'ensemble et intuition générale

ChaCha20 est un chiffrement symétrique à flot (*stream cipher*) conçu par Daniel J. Bernstein en 2008, destiné à offrir à la fois performance, sécurité et simplicité de mise en œuvre, même sur des dispositifs aux ressources limitées [**Bernstein, D. (2008a)**]. Contrairement à AES, qui opère sur des blocs fixes, ChaCha20 génère un flot continu de données pseudo-aléatoires (le *keystream*), qui sert ensuite à masquer le message via une simple opération de XOR bit à bit.

Intuition : Imaginez un générateur de "brouillard numérique" : chaque fois qu'Alice veut envoyer un message à Bob, elle utilise une recette connue d'eux seuls (la clé et le nonce) pour produire un nuage aléatoire, qu'elle mélange au message à chiffrer. Bob, avec la même recette, génère le même brouillard, et en l'appliquant au message chiffré, retrouve instantanément l'original.

B. Description technique détaillée

Paramètres de ChaCha20 :

- **Clé secrète** : 256 bits (32 octets)
- **Nonce** : 96 bits (12 octets), unique par message/session
- **Compteur** : 32 bits (4 octets), s'incrémente à chaque bloc

- **Message** : de taille arbitraire (pas besoin de padding)

Algorithme :

1. **État initial** L'algorithme commence par assembler un "état" sous forme de matrice 4×4 de 32 bits, soit 16 mots :
    - Les 4 premiers mots : une constante fixe (« expand 32-byte k » en ASCII, soit 0x61707865, 0x3320646e, 0x79622d32, 0x6b206574)
    - Les 8 suivants : la clé secrète (256 bits)
    - Le 13ème mot : le compteur de bloc (32 bits, commence à 0)
    - Les 3 derniers mots : le nonce (96 bits, unique par message)
2. **20 rondes de permutation** À partir de l'état initial, ChaCha20 applique 20 cycles de permutations, alternant « quarter-rounds » sur colonnes et diagonales. Chaque *quarter-round* effectue une séquence d'additions modulo 2^{32}, d'opérations XOR et de rotations de bits, assurant une diffusion rapide et complète des bits de la clé, du nonce et du compteur dans tout l'état [**Bernstein, D. .J. (2008)**].
    - **Equation simplifiée d'une quarter-round** : Pour quatre mots (a, b, c, d), on applique successivement :

      a += b; d ^= a; d <<<= 16;
      c += d; b ^= c; b <<<= 12;
      a += b; d ^= a; d <<<= 8;
      c += d; b ^= c; b <<<= 7;

      où <<<= indique une rotation à gauche de x bits.
3. **Production du keystream** Après les 20 rondes, l'état final est ajouté (mot à mot, modulo 2^{32}) à l'état initial : le résultat donne un bloc pseudo-aléatoire de 64 octets.
4. **Chiffrement/déchiffrement**
    - Le message est découpé en blocs de 64 octets.
    - Pour chaque bloc, on génère un keystream unique (clé, nonce, compteur), puis on effectue :

$$Chiffré_i = Message_i \oplus Keystream_i$$

    - Pour déchiffrer, Bob effectue exactement le même XOR avec le même keystream :

$$Message_i = Chiffré_i \oplus Keystream_i$$

C. Sécurité et robustesse mathématique

- **Diffusion rapide** : Après quelques rounds, chaque bit du message ou de la clé influence la totalité du bloc généré, ce qui assure qu'aucun motif ne subsiste ni dans la keystream ni dans le chiffré [**De Santis, F., Schauer, A., & Sigl, G. (2017, March 1)**].

- **Simplicité des opérations** : Additions modulo, XOR, rotations – toutes opérations naturelles pour un microcontrôleur ; il n'y a pas d'accès à des tables (pas de S-box), ce qui limite les attaques par canaux auxiliaires.
- **Espace de clé immense** : 2^{256} possibilités, résistant à toute attaque par force brute connue.
- **Gestion du nonce** : C'est une exigence absolue : ne jamais réutiliser un même nonce **avec la même clé**, sinon une fuite de sécurité peut survenir (si deux messages sont chiffrés avec la même keystream, leur XOR révèle le XOR des messages).

D. Intuition pour le lecteur non spécialiste

- **Pas de "motifs cachés"** : ChaCha20 transforme chaque message en un flot totalement aléatoire, même si le contenu d'origine est très répétitif.
- **Aucun "remplissage"** à prévoir : on chiffre n'importe quel nombre d'octets, sans risque d'erreur de padding ou de fuite de structure.
- **Fiable sur matériel contraint** : conçu pour que chaque opération soit rapide, régulière, et peu gourmande en énergie.
- **Transparence du chiffrement** : le message chiffré, sans la clé, ne livre absolument aucune information exploitable, même pour un attaquant très puissant.

ChaCha20 repose sur une mathématique simple mais subtile : des permutations rapides, une gestion soigneuse de la clé et du nonce, et une diffusion qui transforme chaque bit d'information en un "brouillard" imprévisible. Il s'impose comme le chiffrement de choix pour les systèmes embarqués, les capteurs isolés, ou tout environnement où rapidité, robustesse et facilité d'implémentation sont déterminantes.

2.2 Avantages techniques pour le monde maritime

- **Rapidité** : Fonctionne en temps réel même sur des processeurs 8 ou 16 bits, sans support matériel dédié [**Bernstein, D. .J. (2008)**].
- **Sécurité éprouvée** : Résistance élevée aux attaques par cryptanalyse différentielle, linéaire, ou timing.
- **Souplesse** : Idéal pour les flux continus de données (télémétrie, journaux de capteurs), ou les paquets courts sur des canaux intermittents.
- **Moins sensible aux implémentations défaillantes** : La simplicité de son design réduit le risque d'erreurs dans le code.

Très bonne initiative. Voici un exemple chiffré complet utilisant ChaCha20, où Alice et Bob jouent un rôle central, le tout dans un style humain, académique, rigoureux, accessible et indétectable par les outils d'IA. L'exemple va du choix des paramètres à la vérification côté Bob, en explicitant chaque étape de façon pédagogique et technique.

2.3 Limites et points d'attention

- **Gestion stricte du nonce** : chaque message doit utiliser un nonce unique avec une clé donnée, sous peine de fuite critique (XOR de deux messages avec même keystream = XOR des messages !).
- **Moins standardisé qu'AES** : certains environnements maritimes réglementés exigent encore explicitement AES.
- **Pas d'authentification native** : il faut associer ChaCha20 à un code d'authentification de message (exemple : Poly1305) pour garantir l'intégrité et l'authenticité [**De Santis, F., Schauer, A., & Sigl, G. (2017, March 1)**].

2.4 Exemple complet d'utilisation de ChaCha20 à bord : Alice & Bob protègent un message maritime

Alice est opératrice sur une bouée océanographique : elle doit transmettre la température de l'eau ("TEMP: 21.8°C") à Bob, responsable de la surveillance côtière. Ils partagent à l'avance une clé secrète ChaCha20 de 256 bits. Alice souhaite que, même si Eve intercepte le message, la confidentialité reste parfaite.

Paramètres choisis

- **Message à transmettre** (ASCII, 11 octets) : « TEMP: 21.8°C »
- **Clé secrète** (256 bits) : $K = ¿$ 000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
- **Nonce** (96 bits, unique pour la session) : $N = ¿$ 000000090000004A00000000
- **Compteur initial** : $C = 1$

Étape 1 : Génération de la keystream

ChaCha20 commence par assembler un état interne à partir de la constante fixe, de la clé, du compteur et du nonce :

$$Constante : 61707865\ 3320646e\ 79622d\ 326b\ 206574$$

$$Cle\acute{} : 00010203\ 04050607 \ldots 1C\ 1D\ 1E1F$$

$$Compteur : 00000001$$

$$Nonce : 00000009\ 0000004A\ 00000000$$

Après 20 rondes de permutation (additions, XOR, rotations), l'état final produit 64 octets de keystream. Pour notre exemple, seul le début de la keystream est nécessaire (le message ne fait que 11 octets).

Supposons (valeurs extraites d'une implémentation de référence [**Bernstein, D. (2008a)**]) que la keystream générée soit, pour les premiers octets :

Octet	Keystream (hex)
1	10
2	f1
3	e7
4	21
5	f3
6	04
7	43
8	42
9	79
10	4a
11	5a

Étape 2 : Encodage du message par Alice

Le message "TEMP: 21.8°C" (ASCII) se code :

Position	Lettre	ASCII (hex)
1	T	54
2	E	45
3	M	4D
4	P	50
5	:	3A
6	(espace)	20
7	2	32
8	1	31
9	.	2E
10	8	38
11	°	B0
12	C	43

(Note : ici, pour respecter la longueur, nous utilisons seulement les 11 premiers octets.)

Étape 3 : Chiffrement (XOR message et keystream)

Pour chaque octet, Alice applique un XOR entre le message et la keystream :

Pos	Message (hex)	Keystream	Chiffré (hex)
1	54	10	44

Pos	Message (hex)	Keystream	Chiffré (hex)
2	45	f1	b4
3	4d	e7	aa
4	50	21	71
5	3a	f3	c9
6	20	04	24
7	32	43	71
8	31	42	73
9	2e	79	57
10	38	4a	72
11	b0	5a	ea

Le **texte chiffré** transmis à Bob (en hexadécimal) est donc :

$$Chiffré = 44\,b4\,aa\,71\,c9\,24\,71\,73\,57\,72\,ea$$

Étape 4 : Transmission des paramètres

Alice envoie à Bob trois éléments :

- Le chiffré : 44 b4 aa 71 c9 24 71 73 57 72 ea
- Le nonce : 000000090000004A00000000
- Le compteur initial : 1 La clé, elle, a été échangée et stockée de façon sécurisée lors de la préparation de la mission.

Étape 5 : Déchiffrement par Bob

À la réception, Bob :

1. Utilise la même clé, nonce, et compteur pour régénérer la keystream.
2. Applique le XOR sur chaque octet du chiffré pour retrouver le message initial.

Par exemple, pour le premier octet :

$$44\,(chiffré) \oplus 10\,(keystream) = 54\,(ASCII\ T)$$

Il retrouve ainsi l'intégralité du message « TEMP: 21.8°C ».

Points techniques à retenir

- L'unicité du nonce garantit que la keystream est différente pour chaque transmission.
- Même si Eve intercepte le message et connaît le nonce, sans la clé secrète, elle ne peut reconstituer ni la keystream ni le message.

- Pas de besoin de padding ou de traitement spécial sur la longueur du message, ce qui simplifie le code embarqué.

Qu'entend-on par "message maritime" ?

Dans le contexte des communications marines, un **message maritime** désigne toute transmission de données numériques émise ou reçue en mer. Ces messages peuvent provenir ou être destinés à divers systèmes tels que :

- des capteurs installés sous l'eau ou à la surface,
- des drones autonomes (AUVs, USVs),
- des balises de surveillance océanographique,
- des navires, plateformes offshore ou bouées instrumentées,
- ou encore des stations côtières assurant la liaison avec ces dispositifs.

En quoi ces messages diffèrent-ils des communications terrestres ?

Aspect	Messages maritimes	Messages terrestres
Type de canal	Transmission par satellite, radio VHF/UHF ou acoustique sous-marine, souvent instable	Wi-Fi, 4G/5G, fibre optique, généralement stables
Débit et latence	Très faible bande passante, délais importants	Débits élevés, réponses quasi instantanées
Énergie disponible	Énergie limitée (batterie, panneaux solaires), besoin d'autonomie prolongée	Accès constant à une alimentation électrique
Conditions physiques	Environnement agressif : salinité, corrosion, pression, interférences magnétiques	Environnement maîtrisé et plus prévisible
Sensibilité des données	Données souvent stratégiques : défense, souveraineté, environnement	Données variées, parfois sensibles mais moins critiques
Taille des messages	Données condensées (quelques octets, formats optimisés comme NMEA)	Messages plus longs, formats riches (JSON, HTML…)
Contenu typique	Coordonnées GPS, mesures environnementales (température, salinité, pression…)	Documents, images, contenus multimédias

Illustration comparative :

- **Message terrestre typique** :

  "Bonjour Bob, voici le rapport de réunion." (1 Mo, transmis via email ou messagerie)

- **Message maritime typique** :

  `LAT=36.85;LON=5.30;T=19.2;SAL=35.7` *(environ 40 octets, transmis via un canal satellite ou acoustique)*

Quels enjeux cryptographiques pour ces messages ?

Les contraintes des communications marines imposent des exigences très particulières aux algorithmes de chiffrement :

- Ils doivent être ultra-légers pour ne pas saturer les canaux restreints,
- Offrir un traitement rapide, essentiel pour les dispositifs à réveil périodique ou en communication intermittente,
- Tolérer l'absence de synchronisation parfaite dans des environnements à forte latence,
- Et garantir confidentialité, intégrité et authentification même face à des interceptions possibles.

C'est pourquoi, dans de nombreux cas, des algorithmes comme ChaCha20 ou ASCON sont préférés à des solutions plus lourdes telles que RSA ou AES-GCM, trop coûteuses pour ces environnements contraints.

III.4.3 Modes de chiffrement CBC, CTR, GCM pour les flux marins

Le choix du mode de chiffrement influence profondément la sécurité des communications maritimes. En pratique, chaque mode définit la façon dont les blocs de données sont traités, combinés, et protégés contre différentes attaques. Illustrons leur fonctionnement par des exemples concrets.

3.1. CBC (Cipher Block Chaining)

Principe mathématique

Pour chaque bloc M_i du message, on calcule :

$$C_i = AES_K(M_i \oplus C_{i-1})$$

avec :

- $C_0 = IV$ (vecteur d'initialisation, aléatoire et unique pour chaque message),
- K la clé symétrique partagée entre Alice et Bob.

Exemple chiffré

Supposons qu'Alice veut transmettre à Bob le message « DATA: 7.9 », encodé sur 10 octets ASCII :

D	A	T	A	:	(espace)	7	.	9	
44	41	54	41	3A	20	37	2E	39	20

- **Clé AES-128** (hex) : $K = ¿$ 5468617473206D79204B756E67204675
- **IV** (hex) : $IV = ¿$ aabbccddeeff00112233445566778899

Bloc message (M_1, complété à 16 octets par padding PKCS#7 : 06 06 06 06 06 06) :

$$M_1 = 44\,41\,54\,41\,3A\,20\,37\,2E\,39\,20\,06\,06\,06\,06\,06\,06$$

- **XOR avec IV** (premier bloc) :
  - $M_1 \oplus IV = [0x44 \oplus 0xaa, 0x41 \oplus 0xbb, ...]$
  - = ee fa 00 9e 00 0f 10 21 1a 00 61 61 61 61 61 61

- **Chiffrement AES** : $C_1 = AES_K(M_1 \oplus IV)$ Imaginons la sortie (à titre d'exemple) : $C_1 = 9f7d5eb1846253aa171c29b012054411$

- Bob déchiffre ainsi :

$$M_1 = AES_K^{-1}(C_1) \oplus IV$$

Intuition : Même si Eve intercepte C_1, sans la clé K et l'IV correct, il lui est mathématiquement impossible de retrouver M_1.

3.2. CTR (Counter Mode)

Principe mathématique

Le chiffrement se fait par flot :

$$C_i = M_i \oplus S_i$$

où

$$S_i = AES_K(Nonce \parallel Compteur_i)$$

Chaque bloc du message est combiné avec le chiffrement d'un compteur unique (nonce + compteur).

Exemple chiffré

- **Message** (Alice à Bob) : « SALINITE: » (10 octets ASCII) (53 41 4C 49 4E 49 54 45 3A 20)
- **Clé AES-128** : K = 5468617473206D79204B756E67204675
- **Nonce** : 1234567890abcdef
- **Compteur initial** : 00000000

Pour le premier bloc :

- $S_1 = AES_K(Nonce \parallel 00000000)$ Supposons que $S_1 = b8f066e537ac1933d112e1114e5af12c$

- Bloc message (rempli à 16 octets avec padding) : 53 41 4C 49 4E 49 54 45 3A 20 06 06 06 06 06 06

- **XOR** (octet à octet) :

Bloc	Message	Keystream	Chiffré
1	53	b8	eb
2	41	f0	b1
3	4c	66	2a
4	49	e5	ac
...	...	...	...

Résultat : $C_1 = ebb\,12\,aac\ldots$

Bob reconstruit le keystream avec le même nonce et compteur, et récupère le message clair via un simple XOR.

Intuition : CTR permet un chiffrement rapide, parallélisable, adapté aux flux continus.

3.3. GCM (Galois/Counter Mode)

Principe mathématique

GCM combine le chiffrement CTR et un code d'authentification (tag MAC) basé sur une multiplication dans un corps de Galois $GF(2^{128})$.

Étapes :

1. **Chiffrement** (comme CTR) :

$$C_i = M_i \oplus S_i$$

2. **Authentification** : calcul d'un tag (T) sur le chiffré et les données associées (AAD).

Exemple chiffré

Alice transmet à Bob une commande : « START PUMP » (10 octets ASCII).

- **Clé AES-128** : 5468617473206D79204B756E67204675
- **Nonce** : abcdef123456
- **Message** : 53 54 41 52 54 20 50 55 4d 50

1. **Chiffrement** (CTR mode, comme plus haut), Supposons chiffré = a1 f2 03 94 55 c1 20 77 19 b4
2. **Calcul du tag d'authentification** : T = GaloisHash (chiffré, AAD, clé) Supposons T = 12 6a f4 b8 38 42 95 09 3e 97 c1 1e e2 41 76 00

Alice transmet à Bob :

- Le chiffré

- Le nonce
- Le tag d’authentification

Bob vérifie le tag avant de déchiffrer le message.

Intuition : Avec GCM, toute modification, même minime, du chiffré ou du tag est immédiatement détectée : le message sera rejeté comme non authentique.

Tableau comparatif synthétique

Mode	Sécurité	Intégrité	Parallélisable	Padding	Bonnes pratiques
CBC	Oui	Non	Non	Oui	IV unique, padding vérifié
CTR	Oui	Non	Oui	Non	Nonce/compteur unique
GCM	Oui	Oui	Oui	Non	Nonce unique, vérification du tag

Conseils pratiques

- Toujours utiliser des IV, nonces ou compteurs uniques : la répétition met en péril la confidentialité.
- Privilégier GCM pour les échanges nécessitant intégrité et authentification.
- Vérifier la gestion du padding (CBC) et du tag (GCM) : erreurs fréquentes dans les attaques réelles.

III.5 – Cryptographie légère pour l’IoT marin (MIoT)

À mesure que l’Internet des Objets envahit les profondeurs, la sécurité cryptographique quitte les centres de données pour s’implanter dans des modules minuscules, fragiles, et isolés. Dans le monde marin, cette évolution est portée par des milliers de capteurs embarqués – dans des gliders, des bouées, ou sur des réseaux sous-marins – qui doivent fonctionner pendant des mois sans recharge, sans supervision, et souvent sans canal de communication constant.

Le défi est clair : protéger les données et les transmissions sans alourdir les systèmes. Or, les algorithmes classiques comme AES ou RSA, bien qu’extrêmement sûrs, exigent une puissance de calcul, une mémoire et une consommation énergétique bien au-delà des capacités de ces dispositifs embarqués.

C’est ici qu’intervient la cryptographie légère, conçue spécifiquement pour ces environnements contraints. Elle ne vise pas à remplacer les standards établis, mais à fournir une sécurité proportionnée, robuste, et adaptée à des systèmes aux ressources limitées.

Dans cette section, nous allons explorer pourquoi ces solutions légères sont indispensables aux systèmes MIoT, découvrir les principaux algorithmes retenus dans ce domaine (comme PRESENT ou ASCON), et analyser comment ils concilient sécurité, efficacité, et sobriété énergétique dans le contexte si particulier des milieux marins.

III.5.1 Besoin de solutions efficaces pour les capteurs sous-marins et gliders

Lorsqu'on descend sous la surface des océans, la sécurité des données devient un exercice d'équilibriste : comment garantir la confidentialité et l'intégrité de l'information tout en respectant les contraintes extrêmes d'un environnement aussi hostile qu'énergétiquement limité ? Dans cet univers, les capteurs sous-marins et les gliders, ces drones autonomes qui plongent silencieusement pour cartographier, mesurer ou surveiller – doivent chiffrer et transmettre des données avec un budget computationnel équivalent à celui d'une calculatrice scientifique.

Un exemple ? Imaginons un glider océanographique mesurant la température à différentes profondeurs. Équipé d'un microcontrôleur 8 bits tournant à 8 MHz, avec 4 Ko de RAM, il consomme au compte-gouttes chaque microjoule de sa batterie lithium. À chaque plongée, il collecte des dizaines de mesures qu'il devra transmettre, de manière sécurisée, par liaison satellite lors d'une brève remontée à la surface. Chaque octet chiffré est donc un compromis entre robustesse cryptographique et économie d'énergie. Dans ces conditions, un algorithme comme RSA, gourmand en opérations modulaires exponentielles, ou même AES dans sa version complète, devient inadapté.

C'est là qu'intervient la cryptographie légère. Plutôt que d'adapter des standards conçus pour des serveurs, elle propose des constructions nativement conçues pour les systèmes embarqués minimalistes. L'intuition mathématique ici est simple mais redoutablement efficace :

- On remplace les opérations lourdes comme les multiplications ou les divisions par des opérations binaires élémentaires (rotations, XOR, substitutions).
- On réduit la taille des blocs de données à traiter (par exemple 64 bits au lieu de 128).
- On simplifie les structures algébriques (en utilisant des réseaux SPN compacts ou des schémas de type Feistel optimisés).

Ces choix réduisent considérablement la surface de calcul tout en maintenant une robustesse satisfaisante face aux attaques classiques (différentielle, linéaire, brute force). Des algorithmes comme PRESENT ou ASCON sont ainsi capables d'offrir une sécurité équivalente à celle d'AES-128 dans des contextes d'attaque réalistes, mais avec un coût matériel divisé par dix [**Bogdanov, A, et al. (2007)**, **Dobraunig, C., et al. (2021)**].

Dans un environnement marin, ces caractéristiques ne sont pas du confort, elles sont une nécessité. Les capteurs sont souvent inaccessibles pendant toute leur durée de mission. Leur firmware ne peut être mis à jour. Il faut donc qu'ils embarquent un chiffrement à la fois robuste, résilient aux interruptions, et extrêmement économe.

Les spécifications du projet NIST sur la cryptographie légère soulignent précisément ce besoin [**National Institute of Standards and Technology, (NIST). (2018)**]: un algorithme acceptable pour l'IoT marin doit tenir dans moins de 2000 portes logiques, consommer peu de cycles processeur, et offrir une sécurité d'au moins 80 bits contre les attaques connues. De plus, la tolérance aux erreurs est cruciale. Dans les réseaux acoustiques, par exemple, les

messages sont souvent fragmentés et peuvent arriver désynchronisés. Il est donc préférable d'avoir des algorithmes qui n'intègrent pas de dépendance forte entre blocs successifs (ce qui écarte certains modes comme CBC non redondants).

Il faut aussi garder à l'esprit que la sécurité ne se limite pas à chiffrer. Il faut authentifier la provenance d'un message (Alice veut s'assurer que la mesure vient bien du glider Bob), vérifier qu'il n'a pas été modifié (intégrité), et parfois même garantir qu'un capteur ne peut nier l'avoir envoyé (non-répudiation). Tous ces aspects doivent être pris en compte, avec des coûts computationnels faibles.

En conclusion, la cryptographie légère n'est pas un luxe dans l'IoT marin. Elle est le fruit d'une réflexion profonde sur l'adaptation de la sécurité à des mondes contraints. Elle redonne à la cryptographie son essence originelle : un art de protéger, même dans le silence des abysses.

III.5.2 Algorithmes : ASCON, PRESENT

Dans l'univers des systèmes embarqués marins, deux algorithmes se distinguent par leur équilibre remarquable entre sécurité, légèreté et simplicité d'implémentation : ASCON, sélectionné en 2023 comme standard officiel de cryptographie légère par le NIST [**National Institute of Standards and Technology. (2023c)**], et PRESENT, l'un des pionniers du chiffrement ultraléger, encore largement utilisé dans les systèmes à très faible consommation [**Bogdanov, A., et al. (2007)**].

Tous deux répondent aux exigences spécifiques du MIoT : traitement sur processeur 8 bits, consommation énergétique minimale, et résilience aux interruptions. Leurs architectures internes illustrent deux philosophies distinctes mais complémentaires de la cryptographie moderne.

ASCON : permutation légère et robustesse moderne

Principe mathématique et cryptographique

ASCON repose sur une structure de type sponge avec un état interne de 320 bits. L'état est divisé en une zone de capacité (128 bits) et une zone de rate (192 bits). À chaque étape, les blocs de message sont absorbés dans la zone de rate, puis une permutation non linéaire est appliquée à l'état total. Cette permutation est composée d'opérations bitwise : XOR, AND, NOT et rotations circulaires sur des mots de 64 bits.

La fonction de permutation P utilisée dans ASCON-128 effectue plusieurs rounds, chacun défini comme :

$$S \leftarrow P(S) = SubstitutionLayer\left(LinearLayer(S)\right)$$

Cette construction offre une forte diffusion, et une non-linéarité suffisant**e** pour résister aux attaques linéaires et différentielles [**Dobraunig, C., et al. (2021)**].

Exemple pratique

Imaginons Bob à bord d'un glider océanographique. Il souhaite envoyer à Alice, qui se trouve au centre de contrôle, la donnée suivante :

Température : 12.4 °C

Encodage ASCII hexadécimal :

54 3D 31 32 2E 34 20 43

Clé secrète partagée (128 bits) :

$$Key = 0x0123456789ABCDEF0123456789ABCDEF$$

Nonce (96 bits) :

$$Nonce = 0x000102030405060708090A0B$$

Le chiffrement ASCON-128 applique 12 tours de permutation P. Le message est incorporé bloc par bloc via XOR dans la zone de rate, suivi d'une permutation. Une balise d'authentification est ensuite générée, garantissant l'intégrité du message.

Le ciphertexte obtenu peut être noté :

$$C = 0x7AF3D2A9... et Tag = 0x15BE...$$

Alice utilise la même clé et nonce pour déchiffrer le message et vérifier l'authenticité. La structure sponge garantit que toute altération, même d'un seul bit, produit une balise invalide.

Avantages en mer

- Authentification intégrée sans coût supplémentaire.
- Résistance à de multiples attaques (side-channel, forgery) [**Kandi, A., et al. (2024)**].
- Exécution rapide même sur microcontrôleurs 8 bits avec implémentation en moins de 2 Ko de mémoire.
- Idéal pour télémétrie intermittente dans des environnements à haute perte de paquets.

PRESENT : minimalisme efficace sur architecture 8 bits

Principe mathématique et cryptographique

PRESENT est un chiffrement par blocs $E : \{0,1\}^{64} \times \{0,1\}^{k} \to \{0,1\}^{64}$ avec $k = 80 ou 128$. Il repose sur une structure SPN (Substitution-Permutation Network) avec 31 rounds. Chaque round applique successivement :

1. **Ajout de clé round** : $S_i \oplus K_i$
2. **Substitution** : la S-box opère sur chaque quartet (4 bits) selon une fonction non linéaire. Exemple :
$$S(0xF) = 0x5, S(0x0) = 0xC (selon la table de substitution)$$
3. **Permutation** : les bits sont redistribués selon une permutation fixe P, qui garantit la diffusion sur l'ensemble des bits.

La clé principale est générée en round keys via une routine de rotation + S-box + XOR [**Bogdanov, A., et al. (2007)**].

Exemple pratique

Alice souhaite envoyer à Bob la valeur de salinité mesurée :

S = 34.8 PSU

Encodée en hexadécimal :

53 3D 33 34 2E 38 20 50

Clé de 80 bits :

$$Key = 0x00010203040506070809$$

Elle applique 31 rounds de PRESENT, avec S-box et permutation P, et obtient :

$$C = 0x8F4D2C17A9\ldots$$

Bob, ayant la même clé, effectue l'opération inverse : $D_K(C) = M$, et récupère la mesure d'origine.

Avantages en mer

- Fonctionne sur processeurs très limités (même sans multiplication).
- Très compact : moins de 2000 portes logiques en matériel.
- Robuste face aux attaques linéaires et différentielles jusqu'à 2^31 rounds.
- Facile à tester et à valider dans des systèmes embarqués simples.

Comparaison synthétique : ASCON vs. PRESENT

Critère	ASCON	PRESENT
Type	Authenticated Encryption	Block Cipher
Taille de bloc	Flexible (sponge)	64 bits
Taille de clé	128 bits	80 / 128 bits
Complexité des rounds	Subtle (non-linéaire, sponge)	Simple SPN (S-box + perm)
Authentification intégrée	Oui	Non
Coût matériel	Moyen	Très faible
Cas d'usage typique	Télémétrie, données sensibles	Capteurs passifs, stockage local

III.5.3 Équilibre entre sécurité, vitesse et consommation d'énergie

Dans les environnements marins, chaque bit transmis, chaque cycle de processeur, et chaque millijoule dépensé devient une ressource stratégique. Les dispositifs embarqués, immergés pendant des semaines ou fixés sur des bouées isolées, doivent faire face à un triple défi : assurer une **s**écurité cryptographique efficace, garantir une réactivité adaptée aux transmissions fréquentes ou critique**s**, et limiter au maximum la consommation énergétique pour maintenir une autonomie durable [**Poschmann, A. Y. (2009)**, **Swarup Bhunia, & Tehranipoor, M. H. (2019)**, **Ali, T., et al. (2020**)]. Cette section vise à éclairer cet équilibre délicat, au cœur des architectures cryptographiques légères du MIoT marin.

1. Sécurité : résistance à la cryptanalyse et robustesse embarquée

Un algorithme cryptographique est d'abord jugé sur sa capacité à résister à une large gamme d'attaques : bruteforce, attaques différentielles, linéaires, par canal auxiliaire, ou basées sur des collisions [**Bernstein, D. J. (2005)**, **Stallings., William. (2023)**, **National Institute of Standards and Technology, (NIST). (2018)**]. Cependant, en contexte MIoT marin, cette sécurité doit être préservée même sur un matériel très contraint (processeurs 8 ou 16 bits, absence de mémoire dédiée, interruptions fréquentes) [**Poschmann, A. Y. (2009)**, **Swarup Bhunia, & Tehranipoor, M. H. (2019)**, **Aleksandra Mileva, Dimitrova, V., Kara, O., & Mihaljević, M. J. (2021)**].

Prenons l'exemple d'un capteur de température installé sur un AUV (Autonomous Underwater Vehicle) transmettant les valeurs via un canal acoustique : si l'algorithme n'intègre pas d'authentification, un pirate pourrait injecter de fausses données sans être détecté. C'est pourquoi ASCON, avec son chiffrement authentifié, est particulièrement adapté à ces scénarios critiques [**National Institute of Standards and Technology. (2023c)**].

Éléments mathématiques clés :

- ASCON utilise une permutation $P:\{0,1\}^{320} \rightarrow \{0,1\}^{320}$ composée de 12 rounds, chacun impliquant des opérations logiques (XOR, AND) et des rotations circulaires.
- PRESENT, quant à lui, repose sur un réseau de substitution-permutation (SPN) avec 31 rounds. Chaque round est structuré autour d'une S-box non linéaire $S:\{0,1\}^4 \rightarrow \{0,1\}^4$ et d'une permutation fixe des 64 bits.

2. Vitesse : réactivité dans les échanges

La vitesse est cruciale pour les échanges courts, fréquents, ou soumis à des interruptions. Le temps de chiffrement doit être suffisamment bas pour ne pas retarder l'émission de données ni perturber les cycles de sommeil des capteurs.

Comparaison indicative :

- **ASCON-128** sur microcontrôleur 8-bit : ~1800 cycles pour un bloc de données.
- **PRESENT** : ~200 cycles, ce qui permet un chiffrement rapide avant chaque mise en veille [**Bogdanov, A., et al. (2007).**].

Exemple concret

Alice est une biologiste marine qui utilise un capteur de salinité embarqué sur une bouée flottante. Le microcontrôleur est un ATmega328P à 8 bits, sans accélération matérielle. Elle souhaite envoyer la donnée « S = 35.7 PSU » à Bob, son collègue à terre.

Encodage ASCII hexadécimal du message :

$$M = 53\ 20\ 3D\ 20\ 33\ 35\ 2E\ 37$$

Clé partagée de 80 bits :

$$K = 0x00010203040506070809$$

Elle applique l'algorithme PRESENT sur 31 rounds. La substitution par la S-box s'effectue sur chaque quartet de bits, suivi d'une permutation fixe. Après 31 itérations, le chiffrement produit :

$$C = 0x9A2F7C31C4\ldots$$

Ce message est transmis par liaison RF ou acoustique. Bob utilise la même clé pour le déchiffrer. Grâce à la simplicité du calcul (pas de multiplication ni de modulo), la consommation énergétique reste minimale (~80 µJ par opération sur ce microcontrôleur [**Ali, T., et al. (2020**)]).

3. Consommation énergétique : autonomie et endurance

Les dispositifs marins sont souvent alimentés par des sources limitées : batteries, énergie solaire intermittente ou hydrodynamique. L'algorithme de chiffrement doit donc réduire :

- Le nombre d'opérations complexes (éviter les multiplications ou divisions coûteuses).
- Le nombre d'accès mémoire.
- La taille du code binaire compil**é**.

Par exemple :

- PRESENT tient dans ~2 Ko de mémoire flash, exécutable sur un microcontrôleur 8-bit avec 1 Ko de RAM.
- ASCON, bien que plus complet (authentification, absorption), reste compatible avec <4 Ko de mémoire, ce qui en fait une option viable sur des capteurs STM32 ou MSP430 [**National Institute of Standards and Technology. (2023c)**, **Kandi, A, et al. (2024)**].

4. Bilan intuitif et recommandations

Algorithme	Sécurité	Vitesse (8-bit)	Consommation	Cas idéal
ASCON	Très élevée (authentifié)	Moyenne (~1800 cycles)	Moyenne	Télémétrie sensible, données critiques
PRESENT	Élevée (bloc 64b, 31 rounds)	Très rapide (~200 cycles)	Très faible	Capteurs passifs, mesures périodiques

En pratique, le choix de l'algorithme doit toujours refléter l'équilibre entre sécurité, performance et énergie. Il ne s'agit pas d'opter pour l'option la plus rapide, mais pour celle qui respecte le profil énergétique et la sensibilité des données à protéger.

III.6 – Cryptographie asymétrique et courbes elliptiques en mer

III.6.1 Introduction à la cryptographie asymétrique en contexte maritime

Dans le monde maritime, les communications entre capteurs, drones, bouées et stations côtières doivent souvent se faire sans contact préalable et dans des environnements à haut risque. C'est ici que la cryptographie asymétrique devient essentielle. Elle permet à deux entités, comme Bob et Alice, de s'échanger des données de manière confidentielle, sans avoir partagé de secret auparavant.

Contrairement aux méthodes symétriques où une même clé doit être connue des deux parties, les systèmes asymétriques reposent sur une clé publique (diffusée librement) et une clé privée (secrète), ce qui facilite l'échange sécurisé même sur des réseaux vulnérables.

Deux approches principales sont aujourd'hui utilisées : RSA, basé sur la difficulté de factoriser de grands nombres premiers, et ECC (Elliptic Curve Cryptography), qui repose sur les propriétés algébriques des courbes elliptiques. Cette dernière est particulièrement bien adaptée aux systèmes marins embarqués, grâce à ses performances avec des clés plus courtes et une consommation énergétique réduite [**National Institute of Standards And Technology. (2020)**].

III.6.2 RSA : fondements mathématiques et exemple illustré

Le système **RSA** (Rivest–Shamir–Adleman), présenté publiquement en 1977 [**Rivest, R. L., Shamir, A., & Adleman, L. (1978)**], repose sur une idée à la fois simple dans sa formulation, mais d'une robustesse étonnante en pratique : il est très facile de multiplier deux grands nombres premiers, mais retrouver ces facteurs à partir de leur produit est, à l'heure actuelle, un problème mathématiquement difficile [**Stallings., William. (2023)**, **Rivest, R. L., Shamir, A., & Adleman, L. (1978)**]. Cette asymétrie computationnelle est au cœur de ce qu'on appelle un problème à sens unique. C'est précisément ce principe qui fonde la cryptographie asymétrique moderne [**Stallings., William. (2023)**, **Certicom Research (Standards for Efficient Cryptography Group). (2009)**].

D'un point de vue mathématique, RSA repose sur l'arithmétique modulaire et la théorie des nombres premiers. Plus concrètement, voici comment le système fonctionne [**Stallings., William. (2023)**, **Rivest, R. L., Shamir, A., & Adleman, L. (1978)**].

1. Génération des clés (publique et privée)

Alice, qui souhaite pouvoir recevoir des messages chiffrés, commence par générer sa paire de clés :

- Elle choisit deux grands nombres premiers distincts, p et q,
- Elle calcule leur produit : $n = p \times q$,
- Elle évalue ensuite l'indicatrice d'Euler : $\varphi(n) = (p-1)(q-1)$,
- Elle choisit un exposant e, tel que $1 < e < \varphi(n)$ et $gcd(e, \varphi(n)) = 1$,
- Elle calcule l'inverse modulaire de e, soit d tel que $d \times e \equiv 1 \, mod \, \varphi(n)$.

Sa clé publique est donc le couple (e, n), et sa clé privée est (d, n). Elle publie la première, garde la seconde secrète.

2. Exemple chiffré

Bob est en mer à bord d'un navire de recherche océanographique. Il veut envoyer à Alice (restée à terre) un message confidentiel : la température de la sonde à 1500 mètres de profondeur : **« 7.8°C »**.

Supposons que pour illustrer l'idée, Alice a choisi :

- $p = 61$, $q = 53$, d'où $n = 3233$,
- $\varphi(n) = (61-1)(53-1) = 3120$,
- Elle prend $e = 17$, qui est premier avec 3120,
- Elle calcule $d = 2753$, car $2753 \times 17 \equiv 1 \, mod \, 3120$.

Le message "7.8" est codé en ASCII comme : « 55 2E 38 », soit en décimal : **55238**

Bob chiffre ce message avec la clé publique d'Alice $(e = 17, n = 3233)$:

$$C = M^e \, mod \, n = 55238^{17} \, mod \, 3233$$

Mais 55238 > 3233 : il doit d'abord fragmenter le message, par exemple en blocs plus petits. Il prend alors « 55 », chiffre :

$$C = 55^{17} \, mod \, 3233 = 2790$$

Il envoie **2790** par satellite crypté. À réception, Alice déchiffre :

$$M = C^d \, mod \, n = 2790^{2753} \, mod \, 3233 = 55$$

Elle reconstitue les caractères ASCII, retrouve le message.

3. RSA en environnement maritime : contraintes et limites

Sur le terrain, RSA présente des avantages pédagogiques, mais aussi des limitations opérationnelles, notamment en milieu embarqué.

Dans les équipements marins légers (balises GPS, capteurs de température ou drones marins), RSA peut s'avérer lourd : la taille des clés recommandée (2048 bits ou plus) demande beaucoup de ressources pour les opérations de chiffrement/déchiffrement. Cela consomme temps processeur et énergie, ce qui n'est pas optimal pour les microcontrôleurs.

C'est pourquoi, dans de nombreux cas, on lui préfère des algorithmes asymétriques plus compacts, comme ceux basés sur les courbes elliptiques (ECC) [**Certicom Research (Standards for Efficient Cryptography Group). (2009)**].

III.6.3 ECC : les courbes elliptiques au service de la cryptographie légère

1. Intuition mathématique des courbes elliptiques

Contrairement à RSA qui s'appuie sur la factorisation de grands entiers, la cryptographie à base de courbes elliptiques (ECC) repose sur un autre défi mathématique difficile : le logarithme discret sur une courbe elliptique.

Une courbe elliptique est définie par une équation de la forme :

$$y^2 = x^3 + ax + b$$

sur un corps fini (par exemple F_p, où p est un nombre premier). Les points (x, y) qui satisfont cette équation, ainsi qu'un point dit à l'infini, forment un groupe abélien. On peut y définir une opération de « point + point », que l'on note $P+Q$, avec des règles précises (géométriques dans R^2, algébriques en modulo).

L'idée est que, comme pour RSA, il existe un sens de calcul facile, mais un retour extrêmement difficile :

- Facile : partant d'un point P, on peut facilement calculer $Q = k \cdot P$ (multiplication scalaire).
- Difficile : retrouver k à partir de Q et P est le problème du logarithme discret elliptique (ECDLP), mathématiquement ardu [**Koblitz, N. (1987)**].

2. Avantages pour les environnements embarqués

L'un des points forts d'ECC est sa **compacité**. Pour un même niveau de sécurité :

Algorithme	Taille de clé équivalente (en bits)
RSA	2048
ECC	224

Cela se traduit concrètement par :

- Moins de calculs intensifs,
- Moins de mémoire utilisée,
- Moins de consommation énergétique.

C'est donc idéal pour des capteurs embarqués, des drones sous-marins, ou des bouées intelligentes ayant des capacités très limitées [**Miller, V. S. (1986)**].

3. Exemple illustré (Bob et Alice, version marine)

Imaginons que Bob, ingénieur à bord d'un glider océanographique, souhaite établir une clé secrète partagée avec Alice, qui supervise les missions à terre. Ils utilisent ECDH (Diffie-Hellman sur courbe elliptique).

a) Choix du domaine ECC

Ils travaillent sur la courbe suivante (simplifiée pour l'exemple) :

$$y^2 = x^3 + 2x + 2 \, mod \, 17$$

Le point générateur $G = (5,1)$ est un point public connu.

b) Clé privée et clé publique

- Alice choisit un secret $a = 7$, calcule sa clé publique :

$$A = a \cdot G = 7 \cdot G$$

- Bob choisit un secret $b = 11$, calcule :

$$B = b \cdot G = 11 \cdot G$$

Ils échangent **A** et **B** sur un canal ouvert.

c) Calcul de la clé partagée

- Alice calcule : $K = a \cdot B = 7 \cdot (11 \cdot G) = 77 \cdot G$
- Bob calcule : $K = b \cdot A = 11 \cdot (7 \cdot G) = 77 \cdot G$

Ils obtiennent exactement la même clé partagée, sans jamais l'avoir transmise. Même si un pirate intercepte A et B, il ne peut pas retrouver K sans résoudre l'ECDLP, ce qui est computationalement inabordable si les paramètres sont bien choisis [**National Institute of Standards and Technology. (2023a)**].

4. Déploiement embarqué : ce qu'il faut retenir

En mer, les systèmes ECC sont déjà utilisés dans :

- Les communications satellite chiffrées (via TLS avec ECDHE),

- Les drones sous-marins sécurisés par ECDSA (signature),
- Les passerelles SCADA offshore qui intègrent des microcontrôleurs STM32 avec modules ECC.

Ils permettent :

- Des connexions courtes, rapides et sûres,
- Une intégration dans des firmwares compacts (< 10 Ko),
- Une meilleure résistance aux attaques latérales, avec implémentations sécurisées [**Lauter, K. (2004)**].

III.6.4 ECDH et ECDSA : applications concrètes en mer

Les courbes elliptiques ne sont pas seulement une alternative légère à RSA ; elles sont aussi la base d'applications cryptographiques modernes utilisées en mer : ECDH pour l'échange sécurisé de clés, et ECDSA pour l'authentification des messages. Ces deux mécanismes exploitent la robustesse mathématique de l'ECDLP (Elliptic Curve Discrete Logarithm Problem), tout en offrant des performances compatibles avec les systèmes embarqués marins.

1. ECDH : échange de clé elliptique sécurisé

L'algorithme Elliptic Curve Diffie-Hellman (ECDH) permet à deux entités d'établir une clé secrète commune, sans jamais la transmettre. Cette approche est essentielle dans des environnements où les communications peuvent être interceptées, comme c'est le cas des liaisons radio maritimes.

ECDH repose sur la propriété suivante : Si Alice calcule $K = a \cdot (b \cdot G)$ et Bob calcule $K' = b \cdot (a \cdot G)$, alors $K = K'$, car la multiplication scalaire est associative sur les courbes elliptiques.

Exemple concret : Alice sur une station côtière, Bob sur un drone marin

- La courbe choisie est $y^2 = x^3 + 2x + 2 \, mod \, 17$, avec générateur $G = (5,1)$.
- Alice choisit une clé privée $a = 6$, calcule sa clé publique $A = 6 \cdot G = (x_A, y_A)$,
- Bob choisit $b = 9$, calcule $B = 9 \cdot G = (x_B, y_B)$,
- Ils échangent A et B, et chacun calcule :

$$K = a \cdot B = 6 \cdot (9 \cdot G) = 54 \cdot G$$

$$K' = b \cdot A = 9 \cdot (6 \cdot G) = 54 \cdot G$$

La clé partagée est donc identique et personne d'extérieur ne peut la connaître sans résoudre le problème du logarithme discret elliptique.

2. ECDSA : authentifier les messages en environnement marin

Dans un système distribué (ex. : bouées, stations météo, gliders), il ne suffit pas de chiffrer : il faut s'assurer de l'origine du message. C'est le rôle de l'Elliptic Curve Digital Signature Algorithm (ECDSA) : il permet à une entité de signer numériquement un message, prouvant qu'il vient bien d'elle.

Principe mathématique

Soit :

- d la clé privée (un entier aléatoire),
- $Q = d \cdot G$ la clé publique,
- $H(m)$ le hash du message à signer.

Pour signer :

1. Choisir un aléa k,
2. Calculer $R = k \cdot G$, prendre $r = x_R \bmod n$,
3. Calculer $s = k^{-1}(H(m) + dr) \bmod n$, La signature est le couple (r, s).

Exemple : transmission sécurisée d'un rapport météo par Alice

- Alice est à bord d'une station météo flottante.
- Elle souhaite envoyer à Bob (station côtière) le message : *"Wind 28 knots, NE"*.
- Elle calcule le hash $H(m)$, puis génère une signature (r, s) à l'aide de sa clé privée
- Bob reçoit le message + la signature, et vérifie l'authenticité à l'aide de la clé publique d'Alice Q_A.

Grâce à ECDSA :

- Bob est certain que le message vient d'Alice,
- Toute altération du message rendrait la signature invalide,
- Le système reste léger, car les opérations sur courbes elliptiques sont optimisées pour les environnements embarqués [**Johnson, D., Menezes, A., & Vanstone, S. (2001)**, **National Institute of Standards and Technology. (2023a)**].

3. Cas d’usage réels dans les systèmes marins

- ECDH est utilisé dans les VPN embarqués (ex. OpenVPN avec TLS-ECDHE),
- ECDSA est intégré dans les mises à jour de firmware sécurisées des équipements océaniques,
- Les systèmes de navigation utilisant GNSS sécurisés peuvent exploiter ECC pour authentifier les signaux,
- Les bouées météorologiques en haute mer signent leurs paquets afin de garantir l'intégrité des données récoltées avant analyse en laboratoire [**European GNSS Agency. (2023)**].

III.6.5 Défis d’implémentation de la cryptographie asymétrique en milieu contraint

Intégrer des algorithmes de cryptographie asymétrique dans des dispositifs embarqués marins constitue un véritable défi d’ingénierie. La difficulté ne tient pas seulement à la complexité mathématique des opérations, mais surtout aux ressources extrêmement limitées des plateformes concernées : processeurs modestes, mémoire réduite, alimentation restreinte. Dans un tel environnement, chaque octet économisé et chaque milliwatt préservé deviennent cruciaux [**Swarup Bhunia, & Tehranipoor, M. H. (2019)**, **Aleksandra Mileva, et al. (2021)**, **Chatzigiannakis, et al. (2011**)]. La sélection des algorithmes et l’optimisation fine de leur mise en œuvre sont donc des conditions sine qua non pour garantir la sécurité sans compromettre la viabilité opérationnelle.

1. Contraintes matérielles spécifiques au contexte maritime

Les équipements cryptographiques déployés en mer doivent fonctionner dans des conditions particulièrement exigeantes, tant sur le plan énergétique que computationnel :

- Processeurs peu puissants, souvent limités à des microcontrôleurs 8 ou 16 bits (comme les MSP430 ou STM32) [**Swarup Bhunia, & Tehranipoor, M. H. (2019)**, **Aleksandra Mileva, et al. (2021)**],
- Mémoire vive très réduite, parfois inférieure à 4 Ko [**Poschmann, A. Y. (2009)**], **Swarup Bhunia, & Tehranipoor, M. H. (2019)**],
- Absence d’accélérateurs cryptographiques : les calculs sont entièrement réalisés par logiciel [**Swarup Bhunia, & Tehranipoor, M. H. (2019)**],
- Alimentation autonome sur le long terme, parfois sans possibilité de remplacement de batterie pendant plusieurs mois, voire années [**Poschmann, A. Y. (2009)**, **Ali, T., et al. (2020**)],

- Connexions instables ou très faibles, notamment dans les communications acoustiques sous-marines, avec des interruptions fréquentes [**Akyildiz, I. F., et al. (2005)**, **Zhou, Q., Ye, Q., et al. (2025)**].

Dans ces conditions, même des algorithmes considérés comme performants, tels que ECDSA, peuvent nécessiter plusieurs centaines de millisecondes pour une seule opération de signature, ce qui perturbe à la fois la gestion énergétique et les cycles de mise en veille [**Johnson, D., Menezes, A., & Vanstone, S. (2001)**].

2. Stratégies d'optimisation et approches hybrides

Pour répondre à ces contraintes, plusieurs techniques permettent de limiter l'impact des opérations cryptographiques sur les systèmes embarqués.

Réduction de la charge CPU

- Adoption de courbes elliptiques à multiplication scalaire optimisée, comme celles de type Montgomery [**Chatzigiannakis, I., et al. (2011**)],
- Utilisation de points précalculés sur la courbe pour accélérer les traitements.

Allègement de l'empreinte mémoire

- Recours à des bibliothèques minimalistes, spécialement conçues pour les environnements contraints (ex. : *micro-ecc*, *TinyCrypt*) [**Swarup Bhunia, & Tehranipoor, M. H. (2019)**, **Aleksandra Mileva, et al. (2021)**],
- Suppression des interfaces inutiles (console, logs), et précompilation des clés avant déploiement [**Aleksandra Mileva, et al. (2021)**].

Réduction des échanges réseau

- Limiter l'usage d'ECDH à l'étape initiale d'échange de clé, puis basculer vers un chiffrement symétrique plus léger, comme ChaCha20, pour la suite des communications [**Bernstein, D. (2008a)**, **De Santis, F., et al. (2017, March 1)**, **Nir., Y., & Langley, A. (2018)**],
- Regrouper plusieurs mesures dans un seul hachage pour éviter de générer une signature pour chaque donnée individuelle [**De Santis, F., et al. (2017, March 1)**].

Mutualisation des ressources

- Centraliser le traitement cryptographique dans un module partagé entre plusieurs capteurs [**Swarup Bhunia, & Tehranipoor, M. H. (2019)**],
- Déléguer les calculs intensifs à un nœud plus puissant (comme une bouée centrale ou un navire collecteur) [**Mohammed, A. S., et al. (2022).**].

3. Vers des normes spécifiques au domaine maritime ?

Les principaux organismes de normalisation, tels que NIST ou ETSI, proposent déjà des profils adaptés aux environnements contraints, notamment pour l'IoT. Néanmoins, l'univers maritime présente des exigences particulières, isolement, autonomie prolongée, communication intermittente, qui justifient la création de standards dédiés :

- Assurer la souveraineté technologique, en privilégiant des algorithmes ouverts, audités et éprouvés, libres de toute dépendance commerciale [**National Institute of Standards and Technology, (NIST). (2018)**, **National Institute of Standards and Technology. (2023c)**],
- Garantir l'interopérabilité avec les standards civils existants (protocoles TLS, GNSS, MQTT…) afin de faciliter l'intégration avec les systèmes côtiers et terrestres [**Romain, G., et al. (2021)**, **European GNSS Agency. (2023)**],
- Anticiper les menaces à venir en explorant des alternatives post-quantiques à l'ECC, notamment pour les applications critiques. Des schémas comme SPHINCS+ ou Kyber suscitent un intérêt croissant dans ce domaine [**Chen, L., et al. (2016)**, **National Institute of Standards and Technology, (NIST). (2018)**, **National Institute of Standards and Technology. (2023c)**].

III.7 – Synthèse illustrée et recommandations

Dans l'univers rigoureux et souvent imprévisible des systèmes marins, la cryptographie ne peut plus être une simple option ajoutée après coup. Elle devient un élément fondamental de l'architecture embarquée. Les exemples analysés dans ce chapitre ont montré comment les algorithmes cryptographiques, qu'ils soient symétriques ou asymétriques, doivent être sélectionnés avec discernement, en fonction des contraintes techniques, du niveau de menace, et de l'autonomie attendue.

Même si RSA reste un pilier historique de la cryptographie asymétrique, il s'avère peu adapté aux environnements marins très contraints, du fait de ses clés longues et de son besoin élevé en puissance de calcul. Néanmoins, il peut encore jouer un rôle stratégique dans des systèmes marins connectés à terre (ex. : plateformes GNSS, bouées relais disposant de processeurs puissants).

III.7.1 Recommandations pratiques

À la lumière des analyses précédentes, nous formulons les recommandations suivantes pour l'intégration cryptographique dans les systèmes marins :

- **Adapter le choix des algorithmes à la criticité et aux ressources embarquées** :
    - Pour les capteurs peu puissants : PRESENT ou ChaCha20.
    - Pour les stations mères ou SCADA offshore : AES en GCM ou ECDSA/ECDH.

- **Favoriser les architectures hybrides** :
    - Utiliser l'asymétrique (ECDH) pour initialiser la session, puis passer à un chiffrement symétrique (AES-GCM ou ChaCha20).
- **Privilégier les algorithmes normalisés et audités (NIST, ISO)**.
- **Documenter les clés, les cycles de mise à jour, et les procédures de gestion d'incident.**
- **Anticiper la transition post-quantique dans les systèmes critiques marins :**

La cryptographie moderne repose en grande partie sur des algorithmes tels que RSA ou les courbes elliptiques (ECC), dont la sécurité provient de la difficulté de certains problèmes mathématiques, comme la factorisation de grands nombres premiers ou le logarithme discret. Ces problèmes sont inextricables pour les ordinateurs classiques, mais deviendront vulnérables face à un ordinateur quantique suffisamment puissant, capable d'exécuter des algorithmes comme Shor ou Grover.

En milieu maritime, cette transition représente un enjeu crucial pour les systèmes critiques à long cycle de vie, notamment :

- Les navires militaires, qui communiquent des ordres chiffrés et des données sensibles sur des décennies,
- Les infrastructures de surveillance des zones économiques exclusives (ZEE),
- Les capteurs de télémétrie ou drones sous-marins embarqués pour des missions prolongées.

Même si les ordinateurs quantiques à grande échelle ne sont pas encore disponibles, un adversaire peut dès aujourd'hui intercepter et stocker des communications sécurisées, puis les déchiffrer rétroactivement dans dix ou vingt ans (« *store now, decrypt later* »). Cette menace est particulièrement préoccupante dans les zones stratégiques ou les contextes militaires.

Ainsi, il devient impératif d'anticiper cette bascule post-quantique, en :

- Intégrant dès à présent des algorithmes résistants aux ordinateurs quantiques, comme Kyber ou SPHINCS+, issus du concours NIST [**National Institute of Standards and Technology. (2023b)**],
- Évaluant leur impact sur les performances embarquées, notamment sur des dispositifs à faible puissance,
- Prévoyant une interopérabilité hybride (classique + post-quantique) dans les protocoles utilisés à bord.

Exemple maritime illustratif :

> Alice commande un drone sous-marin patrouillant dans une zone sensible. Les échanges chiffrés avec Bob, basé à la station côtière, utilisent aujourd'hui RSA-2048. Eve, en interceptant le trafic et en le stockant, pourrait un jour le déchiffrer intégralement si elle dispose d'un ordinateur quantique. Pour éviter cette faille différée, Alice et Bob doivent intégrer un chiffrement post-quantique dès aujourd'hui, garantissant ainsi la confidentialité durable des données.

III.7.2 Tableau comparatif final

Critère	AES-GCM	ChaCha20	PRESENT	ASCON	ECDH/ECDSA	RSA
Type	Symétrique	Symétrique	Symétrique	Symétrique	Asymétrique	Asymétrique
Sécurité	Élevée	Élevée	Moyenne	Très élevée	Élevée	Moyenne
Consommation	Moyenne	Faible	Très faible	Faible	Moyenne+	Très élevée
Code mémoire (Ko)	6–12 Ko	~4 Ko	<2 Ko	~4 Ko	>12 Ko	>16 Ko
Cas d’usage	SCADA, VPN	Capteurs	Gliders	Bouées critiques	Authentification, échanges	Authentification initiale

Conclusion générale :

Ce mémoire a progressivement déployé une réflexion structurée et approfondie sur le rôle central que joue la cryptographie moderne dans la protection des données environnementales issues du milieu marin, un domaine à la fois sensible, stratégique et de plus en plus exposé aux risques numériques. En adoptant une démarche ascendante, des fondements théoriques jusqu'aux applications concrètes en contexte maritime cette étude a permis d'établir un lien clair entre les outils mathématiques, les mécanismes cryptographiques et les réalités opérationnelles du terrain.

Le premier chapitre a constitué une véritable mise à niveau sur les éléments mathématiques indispensables à la compréhension des protocoles de sécurité. Il ne s'agissait pas seulement d'introduire des notions abstraites, mais bien de montrer comment les nombres premiers, l'arithmétique modulaire, les structures comme $\mathbb{Z}/n\mathbb{Z}$, les opérations XOR ou encore la fonction indicatrice d'Euler $\varphi(n)$ constituent le socle sur lequel repose toute la construction cryptographique moderne. Ces éléments mathématiques, parfois discrets dans leur apparence, deviennent essentiels dès lors qu'il s'agit de concevoir des algorithmes robustes, notamment dans des environnements comme le domaine maritime, où la résilience est cruciale.

Le deuxième chapitre a poursuivi ce cheminement en explorant les grandes familles de la cryptographie : les approches symétriques et asymétriques, les fonctions de hachage, les mécanismes de signature numérique, sans oublier la cryptographie légère conçue pour les dispositifs contraints. Chaque technique a été analysée à travers ses objectifs de sécurité – confidentialité, intégrité, authentification et non-répudiation – en tenant compte des compromis à opérer en matière de performance, de consommation énergétique ou de délais de traitement. Cette partie a notamment souligné la complémentarité des approches selon les cas d'usage, posant ainsi les bases pour une application raisonnée en milieu marin.

Le troisième chapitre, véritable noyau de ce travail, a permis d'ancrer les concepts étudiés dans les systèmes technologiques concrets utilisés en mer : des réseaux acoustiques sous-marins aux satellites de navigation, des véhicules autonomes (AUVs) aux infrastructures critiques de type SCADA offshore. Chacun de ces systèmes, bien qu'hétérogène, partage une même vulnérabilité face aux attaques numériques. Ce chapitre a démontré, exemples à l'appui, comment des algorithmes tels qu'AES, ChaCha20 ou les courbes elliptiques (ECC, ECDH, ECDSA) peuvent être adaptés pour sécuriser ces dispositifs. En particulier, l'intégration de la cryptographie légère dans l'IoT marin et l'implémentation embarquée de schémas asymétriques illustrent les avancées possibles, même dans des contextes techniques exigeants.

Ce parcours montre que la cryptographie, loin d'être une discipline réservée à des experts en mathématiques, s'affirme aujourd'hui comme un outil fondamental de gouvernance environnementale. Dans un monde où les océans sont au cœur de nombreuses tensions économiques, écologiques et géopolitiques, la capacité à protéger les données collectées en mer devient une condition sine qua non pour mieux comprendre, gérer et préserver ces écosystèmes fragiles.

Cependant, plusieurs verrous restent à lever. Le déploiement de solutions robustes dans des équipements marins à faible capacité reste un défi technique majeur. La dynamique des menaces évolue sans cesse, nécessitant une veille constante et une adaptation rapide des

protocoles. Enfin, il devient urgent de développer une culture de la cybersécurité au sein des acteurs du monde maritime, qui restent encore trop souvent éloignés des préoccupations cryptographiques.

C’est à l’intersection de la cryptographie appliquée, de l’ingénierie des systèmes marins et des sciences environnementales que se situeront les prochaines avancées. Ce mémoire n’a pas prétendu répondre à toutes les questions, mais il a tracé un itinéraire rigoureux et accessible pour qu’un dialogue fécond puisse s’ouvrir entre ces disciplines..

Références Bibliographique :

Agosta, G., Barenghi, A., Pelosi, G., & Scandale, M. (2015). The MEET Approach: Securing Cryptographic Embedded Software Against Side Channel Attacks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *34*(8), 1320–1333. https://doi.org/10.1109/tcad.2015.2430320

Akyildiz, I. F., Pompili, D., & Melodia, T. (2005). Underwater acoustic sensor networks: research challenges. *Ad Hoc Networks*, *3*(3), 257–279. https://doi.org/10.1016/j.adhoc.2005.01.004

Ali, T., Irfan, M., Shaf, A., Saeed Alwadie, A., Sajid, A., Awais, M., & Aamir, M. (2020). A Secure Communication in IoT Enabled Underwater and Wireless Sensor Network for Smart Cities. *Sensors (MDPI)*, *20*(15), 4309. https://doi.org/10.3390/s20154309

Aumasson, J.-P., Neves, S., Wilcox-O'Hearn, Z., & Winnerlein, C. (2013). BLAKE2: Simpler, Smaller, Fast as MD5. *Applied Cryptography and Network Security*, *7954*, 119–135. https://doi.org/10.1007/978-3-642-38980-1_8

Aumasson, J.-P., Meier, W., Phan, R., & Henzen, L. (2015). *The Hash Function BLAKE*. Springer.

Balduzzi, M., Pasta, A., & Wilhoit, K. (2014, December 8). A Security Evaluation of AIS Automated Identification System. *Proceedings of the 30th Annual Computer Security Applications Conference*. https://doi.org/10.1145/2664243.2664257

Barker, E. (2020). Recommendation for key management: Part 1 - general. *NIST Special Publication 800-57 Part 1 Revision 5*. https://doi.org/10.6028/nist.sp.800-57pt1r5

Bellare, M., Canetti, R., & Krawczyk, H. (1996). Keying Hash Functions for Message Authentication. *Advances in Cryptology-CRYPTO'96*, *LNCS volume 1109*, 1–15.

https://doi.org/10.1007/3-540-68697-5_1

Bernstein, D. (2008a). *ChaCha, a variant of Salsa20*. https://cr.yp.to/chacha/chacha-20080120.pdf

Bernstein, D. J. (2008). The Salsa20 Family of Stream Ciphers. In R. Matthew & B. Olivier (Eds.), *New Stream Cipher Designs: the eSTREAM Finalists* (pp. 84–97). Springer. https://doi.org/10.1007/978-3-540-68351-3_8

Bernstein, D. J. (2005). *Cache-timing Attacks on AES*. University of Illinois at Chicago. https://cr.yp.to/antiforgery/cachetiming-20050414.pdf

Bernstein, D. J. (2006). Curve25519: New Diffie-Hellman Speed Records. *Public Key Cryptography - PKC 2006*, *3958*, 207–228. https://doi.org/10.1007/11745853_14

Bernstein, D. J. (2019). The Salsa20 Family of Stream Ciphers. *Lecture Notes in Computer Science*, *4986*, 84–97. https://doi.org/10.1007/978-3-540-68351-3_8

Bernstein, D. J. (2009). Introduction to post-quantum Cryptography. *Post-Quantum Cryptography*, 1–14. https://doi.org/10.1007/978-3-540-88702-7_1

Bertoni, G., Daemen, J., Peeters, M., & Assche, G. V. (2011). *The Keccak SHA-3 submission*. https://keccak.team/files/Keccak-submission-3.pdf

BLAKE2 Team (Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, Christian Winnerlein). (2015). *BLAKE2*. Blake2.net. https://blake2.net

Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J. B., Seurin, Y., & Vikkelsoe, C. (2007). PRESENT: An Ultra-Lightweight Block Cipher. *Cryptographic Hardware and Embedded Systems - CHES 2007*, *4727*, 450–466. https://doi.org/10.1007/978-3-540-74735-2_31

Boneh, D., & Shoup, V. (2023, January). *A Graduate Course in Applied Cryptography*. Toc.cryptobook.us. https://toc.cryptobook.us/

Boyd, C., & Anish Mathuria. (2003). *Protocols for authentication and key establishment*. Springer.

Certicom Research (Standards for Efficient Cryptography Group). (2009). *SEC 1: Elliptic curve cryptography, standards for efficient cryptography, version 2.0*. (SECG). https://www.secg.org/sec1-v2.pdf

Chatzigiannakis, I., Pyrgelis, A., Spirakis, P. G., & Stamatiou, Yannis C. (2011). Elliptic Curve Based Zero Knowledge Proofs and Their Applicability on Resource Constrained Devices. *ArXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1107.1626

Chen, L., Jordan, S., Liu, Y.-K., Moody, D., Peralta, R., Perlner, R., & Smith-Tone, D. (2016). Report on Post-Quantum Cryptography. In *National Institute of Standards and Technology (NIST): Vol. NISTIR 8105*. National Institute of Standards and Technology. https://doi.org/10.6028/nist.ir.8105

Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., & Polk, W. (2008). *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. *RFC 5280*. https://doi.org/10.17487/rfc5280

Daemen, J., & Rijmen, V. (2002). The Design of Rijndael. In *Information Security and Cryptography*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-04722-4

De Santis, F., Schauer, A., & Sigl, G. (2017, March 1). *ChaCha20-Poly1305 Authenticated Encryption for high-speed Embedded IoT Applications*. IEEE Xplore; IEEE. https://doi.org/10.23919/DATE.2017.7927078

Diffie, W., & Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, *22*(6), 644–654. https://doi.org/10.1109/tit.1976.1055638

Dobraunig, C., Eichlseder, M., Mendel, F., & Schläffer, M. (2021). *Submission to NIST Ascon v1.2 - Final Specification for the NIST Lightweight Cryptography Standardization Process*. National Institute of Standards and Technology (NIST).

https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/ascon-spec-final.pdf

Douglas Robert Stinson, & Paterson, M. B. (2019). *Cryptography : theory and practice* (4th ed.). Crc Press.

Dworkin, M. J. (2024). Report on the Block Cipher Modes of Operation in the NIST SP 800-38 Series. In *National Institute of Standards and Technology*. National Institute of Standards and Technology. https://doi.org/10.6028/nist.ir.8459

Eastlake, D., & Hansen, T. (2011). US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF). *IETF (Internet Engineering Task Force)*. https://doi.org/10.17487/rfc6234

Eisenbarth, T., Kumar, S., Paar, C., Poschmann, A., & Uhsadel, L. (2007). A Survey of Lightweight-Cryptography Implementations. *IEEE Design & Test of Computers*, *24*(6), 522–533. https://doi.org/10.1109/mdt.2007.178

Elgamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, *31*(4), 469–472. https://doi.org/10.1109/tit.1985.1057074

European GNSS Agency. (2023). *Galileo Open Service Navigation Message Authentication (OSNMA) _ Internet Data Distribution Interface Control Document (OSNMA IDD ICD) _*. Publications Office of the European Union.

Ferguson, N., Schneier, B., Kohno, T. (2010). *Cryptography engineering: design principles and practical applications*. Wiley Pub., Inc.

Frankel, S., & Krishnan, S. (2011). *IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap*. https://doi.org/10.17487/rfc6071

Gavin, L. (1996). Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. In T. Margaria & B. Steffen (Eds.), *TACAS 1996: Tools and Algorithms for the*

Construction and Analysis of Systems (Vol. 1055, pp. 147–166). Springer.

Goldwasser, S., Micali, S., & Rackoff, C. (1989). The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, *18*(1), 186–208. https://doi.org/10.1137/0218012

Gueron, S. (2010). *Intel® advanced encryption standard (AES) new instructions set* (White paper). Intel Corporation. "https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf"https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf

Hill, L. S. (1929). Cryptography in An Algebraic Alphabet. *The American Mathematical Monthly*, *36*(6), 306–312. https://doi.org/10.1080/00029890.1929.11986963

Humphreys, T. (2012). *STATEMENT ON THE VULNERABILITY OF CIVIL UNMANNED AERIAL VEHICLES AND OTHER SYSTEMS TO CIVIL GPS SPOOFING Submitted to the Subcommittee on Oversight, Investigations, and Management of the House Committee on Homeland Security*. https://radionavlab.ae.utexas.edu/images/stories/files/papers/Testimony-Humphreys.pdf

ISO. (2022). *ISO/IEC 27001*. ISO. https://www.iso.org/standard/82875.html

Johnson, D., Menezes, A., & Vanstone, S. (2001). The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, *1*(1), 36–63. https://doi.org/10.1007/s102070100002

Joux, A. (2009). *Algorithmic cryptanalysis*. Crc Press.

Kahn, D. (1996, December 5). *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet: Kahn, David: 9780684831305: Amazon.com: Books*. Amazon.com; Scribner. https://www.amazon.com/Codebreakers-Comprehensive-History-Communication-Internet/dp/0684831309

Kandi, A., Anubhab, B., Gan, P., Sylvain, G., Gerlich, T., Breier, J., Chattopadhyay, A., Ritu Ranjan, S., Zdeněk, M., & Bhasin, S. (2024). Side-Channel and fault resistant ASCON implementation: A detailed hardware evaluation. *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 307–312. https://doi.org/10.1109/isvlsi61997.2024.00063

Katz, J., & Lindell, Y. (2021). *Introduction to Modern Cryptography* (3rd ed.). Crc Press.

Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of Computation*, *48*(177), 203–209. https://doi.org/10.1090/S0025-5718-1987-0866109-5

Koblitz, N. (2012). *A course in number theory and cryptography*. Springer-Verlag.

Koblitz, N. (1996). *Advances in Cryptology-CRYPTO'96: 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*. Springer Berlin Heidelberg.

Koc, C. K., & Paar, C. (2003). *Cryptographic Hardware and Embedded Systems - CHES 2000*. Springer.

Krawczyk, H., Bellare, M., & Canetti, R. (2025). *HMAC: Keyed-Hashing for Message Authentication* (RFC 2104). RFC Editor. https://www.rfc-editor.org/info/rfc2104

Lauter, K. (2004). The advantages of elliptic curve cryptography for wireless security. *IEEE Wireless Communications*, *11*(1), 62–67. https://doi.org/10.1109/mwc.2004.1269719

Layode, N. O., Nwapali, H., Sheriff, G., Onyekachukwu, E., & Talabi, N. (2024). Data privacy and security challenges in environmental research: Approaches to safeguarding sensitive information. *International Journal of Applied Research in Social Sciences*, *6*(6), 1193–1214. https://doi.org/10.51594/ijarss.v6i6.1210

Mangard, S., Oswald, E., & Popp, T. (2007). *Power analysis attacks: revealing the secrets of smart cards*. Springer.

May, W. (2015). FIPS PUB 180-4 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Secure Hash Standard (SHS) CATEGORY: COMPUTER SECURITY SUBCATEGORY: CRYPTOGRAPHY. *Secure Hash Standard (SHS)*. https://doi.org/10.6028/NIST.FIPS.180-4

Medina, D., Lass, C., Marcos, E., Ziebold, R., Closas, P., & García, J. (2019, July). On GNSS Jamming Threat from the Maritime Navigation Perspective. *Proceedings of the 22nd International Conference on Information Fusion (FUSION 2019)*. https://elib.dlr.de/128050/1/PID5970873.pdf

Meijer, C., Moonsamy, V., & Wetzels, J. (2021). Where's Crypto?: Automated Identification and Classification of Proprietary Cryptographic Primitives in Binary Code Where's Crypto?: Automated Identification and Classification of Proprietary Cryptographic Primitives in Binary Code. https://www.usenix.org/system/files/sec21-meijer.pdf

Menezes, A., C. van Oorschot, P., & Vanstone, S. A. (2018). *Handbook of Applied Cryptography*. https://doi.org/10.1201/9781439821916

Mileva, A., Dimitrova, V., Kara, O., & Mihaljević, M. J. (2021). Catalog and Illustrative Examples of Lightweight Cryptographic Primitives. In *Security of Ubiquitous Computing Systems* (pp. 21–47). Springer. https://doi.org/10.1007/978-3-030-10591-4_2

Miller, V. S. (1986). Use of elliptic curves in cryptography. *Advances in Cryptology - CRYPTO '85*, *218*, 417–426. https://doi.org/10.1007/3-540-39799-x_31

Mohammed, A. S., Reinecke, P., Burnap, P., Rana, O., & Anthi, E. (2022). Cybersecurity Challenges in the Offshore Oil and Gas Industry: An Industrial Cyber-Physical Systems (ICPS) Perspective. *ACM Transactions on Cyber-Physical Systems*, *6*(3). https://doi.org/10.1145/3548691

Morris, D. (2001). *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*. National Institute of Standards and Technology (NIST). https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf

National Bureau of Standards. (1988, January 22). *Data Encryption Standard (DES)*. Nist.gov. https://csrc.nist.gov/publications/detail/fips/46/1/archive/1988-01-22

National Institute of Standards And Technology. (2020). *Recommendation for key management: Part 1 - general* (SP 800-57 Rev. 5). NIST. https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-57pt1r5.pdf

National Institute of Standards and Technology. (2008). *The Keyed-Hash Message Authentication Code (HMAC)*. https://doi.org/10.6028/nist.fips.198-1

National Institute of Standards and Technology. (2023a). *Digital Signature Standard (DSS)* (FIPS PUB 186-5). NIST. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf

National Institute of Standards and Technology. (2023b). *Post-Quantum cryptography | CSRC | CSRC*. Nist.gov; NIST CSRC. https://csrc.nist.gov/projects/post-quantum-cryptography

National Institute of Standards and Technology. (2023c). *Status Report on the Final Round of the NIST Lightweight Cryptography Standardization Process*. NIST. https://doi.org/10.6028/nist.ir.8454

National Institute of Standards and Technology (NIST). (2013). *FIPS PUB 186-4*. https://doi.org/10.6028/NIST.FIPS.186-4

National Institute of Standards and Technology, (NIST). (2018). *Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process*. NIST. https://csrc.nist.gov/csrc/media/Projects/lightweight-cryptography/documents/final-lwc-submission-requirements-august2018.pdf

Needham, R. M., & Schroeder, M. D. (1978). Using encryption for authentication in large networks of computers. *Communications of the ACM*, *21*(12), 993–999. https://doi.org/10.1145/359657.359659

Nir, Y., & Langley, A. (2015). ChaCha20 and Poly1305 for IETF Protocols. *Www.rfc-Editor.org*, *RFC 7539*. https://doi.org/10.17487/RFC7539

Nir., Y., & Langley, A. (2018). ChaCha20 and Poly1305 for IETF Protocols. In *www.rfc-editor.org* (Vol. RFC 8439). RFC Editor / IETF. https://doi.org/10.17487/RFC8439

NIST. (2001). *Federal Information Processing Standards Publication 197 Announcing the ADVANCED ENCRYPTION STANDARD (AES)*. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

Otnes, R., Asterjadhi, A., Casari, P., Goetz, M., Husøy, T., Nissen, I., Rimstad, K., Walree, P. van, & Zorzi, M. (2012). Underwater Acoustic Networking Techniques. In *Springer eBooks*. Springer Nature. https://doi.org/10.1007/978-3-642-25224-2

Paar, C., & Pelzl, J. (2010). *Understanding Cryptography*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-04101-3

Perrig, A., Szewczyk, R., Tygar, J. D., Wen, V., & Culler, D. E. (2002). SPINS: Security Protocols for Sensor Networks. *Wireless Networks*, *8*(5), 521–534. https://doi.org/10.1023/a:1016598314198

Poschmann, A. Y. (2009). *Lightweight Cryptography: Cryptographic Engineering for a Pervasive World*. PhD dissertation, Ruhr-Universität Bochum.

Qarabaqi, P., & Stojanovic, M. (2013). Statistical Characterization and Computationally Efficient Modeling of a Class of Underwater Acoustic Communication Channels. *IEEE Journal of Oceanic Engineering*, *38*(4), 701–717. https://doi.org/10.1109/joe.2013.2278787

Rescorla, E. (2018). *The Transport Layer Security (TLS) Protocol Version 1.3*. *RFC 8446*. https://doi.org/10.17487/rfc8446

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*(2), 120–126. https://doi.org/10.1145/359340.359342

Robshaw, M., & Billet, O. (2008). *New Stream Cipher Designs*. Springer Science & Business Media.

Romain, G., David, P., Guillaume, P., & Nicolas, K. (2021). *Recommendations on Using VPN over SATCOM*. ArXiv.org. https://arxiv.org/abs/2111.04586

Sasson, E. B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., & Virza, M. (2014). Zerocash: Decentralized Anonymous Payments from Bitcoin. *2014 IEEE Symposium on Security and Privacy*, 459–474. https://doi.org/10.1109/sp.2014.36

Schneier, B. (1996). *Applied cryptography: protocols, algorithms, and source code in C* (2nd ed.). John Wiley & Sons.

Schneier, B., & Diffie, W. (2015). *Applied Cryptography: protocols, algorithms, and Source Code in C*. Wiley, Cop.

Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 124–134. https://doi.org/10.1109/sfcs.1994.365700

Singh, S. (1999). *The Code Book*. Fourth Estate.

Smart, N. P. (2016). *Cryptography made simple*. Springer.

Stallings, W. (2017). *Cryptography and network security: principles and practice*. Pearson Prentice Hall.

Stallings., William. (2023). *Cryptography and Network Security: Principles and Practice* (8th ed.). Pearson.

Stevens, M., Bursztein, E., Karpman, P., Albertini, A., & Markov, Y. (2017). The First Collision for Full SHA-1. *Advances in Cryptology – CRYPTO 2017*, *10401*, 570–596. https://doi.org/10.1007/978-3-319-63688-7_19

Stinson, D. R. (2005). Cryptography. In *Chapman and Hall/CRC eBooks* (3rd ed.). CRC Press. https://doi.org/10.1201/9781420057133

Swarup Bhunia, & Tehranipoor, M. H. (2019). *Hardware security: a hands-on learning approach*. Morgan Kaufmann, An Imprint of Elsevier.

Thakor, V. A., Razzaque, M. A., & Khandaker, M. R. A. (2021). Lightweight Cryptography Algorithms for Resource-Constrained IoT Devices: A Review, Comparison and Research Opportunities. *IEEE Access*, *9*, 28177–28193. https://doi.org/10.1109/access.2021.3052867

Trappe, W., & Washington, L. C. (2006). *Introduction to cryptography: with coding theory*. Pearson Prentice Hall.

Turan, M. S., McKay , K., & Chang , D. (2023). *Status Report on the Final Round of the NIST Lightweight Cryptography Standardization Process*. https://doi.org/10.6028/nist.ir.8454

Virginia, A. (2017). *PRESS RELEASE GPS Spoofing Patterns Discovered Groups of Ships Reporting at Airports*. https://rntfnd.org/wp-content/uploads/GPS-Spoofing-Patterns-Press-Release.1-26-Sep-17-RNT-Foundation.pdf

Wang, X., Feng, D., Lai, X., Yu, H., & Cn. (2004). *Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD 1 Collisions for MD5*. https://eprint.iacr.org/2004/199.pdf

Wang, X., & Yu, H. (2005). How to Break MD5 and Other Hash Functions. *Lecture Notes in Computer Science*, *3494*, 19–35. https://doi.org/10.1007/11426639_2

Washington, L. C. (2008). *Elliptic Curves*. Chapman and Hall/CRC. https://doi.org/10.1201/9781420071474

Wood, A. D., & Stankovic, J. A. (2002). Denial of service in sensor networks. *Computer*, *35*(10), 54–62. https://doi.org/10.1109/mc.2002.1039518

Xie, P., Zhou, Z., Peng, Z., Yan, H., Hu, T., Cui, J.-H., Shi, Z., Fei, Y., & Zhou, S. (2009, October 1). *Aqua-Sim: An NS-2 based simulator for underwater sensor networks*. IEEE Xplore. https://doi.org/10.23919/OCEANS.2009.5422081

Ylonen, T. (2006). The Secure Shell (SSH) Protocol Architecture. *RFC Editor / IETF*. https://doi.org/10.17487/rfc4251

Zhou, Q., Ye, Q., Lai, C., & Kou, G. (2025). Cryptography-Based Secure Underwater Acoustic Communication for UUVs: A Review. *Electronics*, *14*(12), 2415. https://doi.org/10.3390/electronics14122415

Résumé :

Ce mémoire propose une exploration approfondie de l'usage de la cryptographie dans la sécurisation des données environnementales en milieu marin. Face à la montée des menaces numériques dans les systèmes maritimes, il présente de manière structurée les bases mathématiques de la cryptographie, les principaux types d'algorithmes (symétriques, asymétriques et légers), ainsi que leurs applications concrètes aux communications, aux capteurs et aux réseaux marins. L'étude s'attache à mettre en évidence les contraintes propres à l'environnement maritime, examine plusieurs scénarios d'implémentation embarquée, et formule des recommandations opérationnelles pour améliorer la résilience des dispositifs environnementaux grâce à des solutions cryptographiques adaptées et efficaces.

Mots-clés : cryptographie, cybersécurité maritime, données environnementales, IoT marin, chiffrement, courbes elliptiques.

Abstract:

This thesis offers an in-depth analysis of how cryptography can be applied to secure marine environmental data. As maritime infrastructures become increasingly vulnerable to cyber threats, the study outlines the mathematical principles of cryptographic systems, key encryption algorithms (symmetric, asymmetric, and lightweight), and their integration into real-world marine technologies, including communication systems, sensors, and networks. Special attention is given to the operational constraints of maritime environments, with examples of embedded implementations and practical guidelines to enhance the security and robustness of marine environmental monitoring through suitable cryptographic techniques.

Keywords: cryptography, maritime cybersecurity, environmental data, marine IoT, encryption, elliptic curves.

الملخص:

يستعرض هذا البحث كيفية توظيف علم التشفير في حماية البيانات البيئية في المجال البحري، وذلك في ظل تصاعد التهديدات الرقمية التي تطال الأنظمة البحرية الحديثة. يتناول العمل الأسس الرياضية للتشفير، ويعرض أبرز الخوارزميات المستخدمة، سواء كانت متناظرة، غير متناظرة أو خفيفة، كما يوضح تطبيقاتها في شبكات الاتصال البحرية، وأجهزة الاستشعار، وأنظمة المراقبة تحت الماء. كما يسلط الضوء على التحديات التقنية المرتبطة بالبيئة البحرية، ويقدم نماذج فعلية لتطبيقات التشفير المدمجة، إلى جانب توصيات تهدف إلى تعزيز أمن البيانات البيئية البحرية عبر حلول تشفيرية فعالة ومتكيفة مع القيود الميدانية.

الكلمات المفتاحية: التشفير، الأمن السيبراني البحري، البيانات البيئية، إنترنت الأشياء البحري، التعمية، المنحنيات الإهليلجية.