Mémoire :
Chiffrement et fonctions de hachage

Defraiteur Maxence

Décembre 2022 - Janvier 2024

Sous la supervision de Volker Mayer

M2 Master MEEF

Introduction

Traditionnellement, la monnaie au format papier a besoin d'encres et de matériaux spéciaux pour éviter la contre-façon. Un problème majeur lié à l'ordinateur repose sur la copie et l'altération des données. Dans une société où les informations sont stockées et transmises électroniquement, il faut assurer la sécurité de l'information. Un outil fondamental utilisé dans la sécurité de l'information est la signature : utile pour le non-rejet, origine de l'authentification des données, identification, certificat d'authenticité. Cette signature doit tenter d'être unique. La cryptographie vise à assurer de manière sécurisée : la confidentialité, l'intégrité des données, l'authentification et le non-rejet de service via des opérations mathématiques. Les fonctions de hachage sont des fonctions répondant à ces enjeux. Elles permettent d'identifier rapidement si deux données sont identiques ou non.

J'ai choisi ce sujet de mémoire étant donné que le domaine de la cryptographie a toujours attisé ma curiosité. Je souhaitais connaître plus en profondeur les fondements et fonctionnements des algorithmes cachés dans notre vie de tous les jours.

Table des matières

I Partie théorique

1 Histoire sur l'origine du chiffrement

1.1 Généralités

La cryptographie est l'étude des techniques mathématiques en lien avec les communications secrètes. C'est un procédé qui est devenu commun dans la vie moderne, établissant une connection entre des appareils digitaux. On appelle cela le « handshaking[1] ». L'invention du *handshaking* est attribuée aux trois mathématiciens : Ron **R**ivest, Adi **S**hamir et Leonard **A**dleman. En 1977, ils développèrent l'algorithme **RSA** (cette procédure de cryptage leur fit gagner le Turing Award en 2002). On retrouve couramment la cryptographie dans le domaine des finances. Le cryptage est utilisé pour empêcher la compréhension des données par une personne tiers.

1.2 Etymologie

Le terme cryptographie vient du Grec « étude de l'écriture cachée ». Très souvent la cryptographie était utilisée pour sécuriser des messages. Dans ce cas, on peut distinguer deux types de textes : le texte brut et le texte crypté (de l'anglais *cipher*). Le code secret (cipher) est un algorithme. Par exemple, le chiffrement de César est un système de substitutions des lettres. Le chiffrement de César datant du Ier siècle av. J-C est un exemple de cryptage symétrique, de sorte que la même clé et code sont utilisés pour décrypter le message. Le code César est facilement cassable à la main ou avec un ordinateur en essayant toutes les possibilités de substitution. On appelle, ce type de décryptage « force brute ». Les codes et clés modernes rendent difficile la tâche de décryptage par force brute.

L'émergence de la cryptographie a permis de protéger des données sensibles dès l'Antiquité. À cette époque, les codes et clés étaient facilement trouvables. À travers les siècles, les codes sont devenus difficiles à casser grâce notamment aux calculs sur ordinateur. Les codes modernes sont presques impossibles à craquer. Ainsi, la cryptographie permet de transmettre des données en sécurité. Par ailleurs, les longs messages étaient vulnérables en utilisant la technique de décrytage par analyse fréquentielle. Le décryptage par analyse fréquentielle a été initié par le

1. Processus automatique pour établir l'entrée en communications de deux entités

mathématicien arabe al-Kindi au IX$^{\text{ème}}$ s. Cette technique employait la fréquence de chaque lettre de l'alphabet les plus souvent utilisées dans le langage particulier. Cette technique de décryptage cassait tous les codes de substitutions. Pour palier à l'analyse fréquentielle, il faut dissimuler le texte brut en utilisant un « code ». Pour améliorer le niveau de sécurité, on peut utiliser un polyalphabet secret où une lettre d'un texte brut peut être substituée à plusieurs lettres d'un texte secret. Cela supprime la possibilité des attaques par analyse fréquentielle. Ces types de codes ont été développés au XVI$^{\text{ème}}$s. mais le plus connu a été produit par la machine Enigma durant la seconde guerre mondiale. Cette machine a été utilisée par les services d'espionnage allemands entre 1923 et 1945. La machine Enigma possédait 158 962 555 217 paramétrages possibles, qui étaient changés tous les jours. Pour le cryptage symétrique, il fallait transmettre physiquement la clé rendant ainsi vulnérable la transmission du message crypté. Le cryptage asymétrique rentra en jeu dès l'émergence des ordinateurs dans les années 1960. Sur de grandes distances, les gens peuvent se transmettre des informations sans interagir entre eux physiquement. Cependant, avec Internet, le réseau est publique donc toutes les clés symétriques peuvent être interceptées. Dans les années 1970, IBM adopte le *US Federal Information Processing Standard* pour crypter les informations déclassées. En 1976, un changement majeur s'est produit lorsque Diffie et Hellman ont publié des nouvelles directions dans la cryptographie. Ils ont introduit la clé publique, étant une nouvelle méthode pour échanger les clés. Puis en 1978, Rivest, Shamir et Adleman ont découvert la première clé publique utile dans la cryptographie et la signature digitale. En 1991, la première signature digitale ISO/IED est réalisée.

1.3 Algortihmes et protocoles

1.4 Algorithmes de chiffrement faible

Les premiers algorithmes de chiffrement reposaient sur le remplacement des caractères par d'autres.

— ROT_{13} (rotation de 13 caractères, sans clé)

— Chiffre de César (décalage de 3 lettres dans l'alphabet sur la gauche)

— Chiffre de Vigenère (introduit la notion de clé)

1.5 Algorithmes de cryptographie symétrique (à clé secrète)

Principe : Une même clé pour chiffrer et déchiffrer un message. La transmission de la clé doit être faite de manière sûre.

— Chiffre de Vernam (le seul offrant une sécurité théorique absolue, à condition que la clé ait au moins la même longueur que le message à chiffrer, qu'elle

ne soit utilisée qu'une seule fois et qu'elle soit aléatoire). Gilbert Vernam a publié ce chiffre en 1926. On appelle cela un « masque jetable » et il n'y a pas besoin d'ordinateur. Claude Shannon prouva plus tard en 1949 que le chiffre de Vernam est parfaitement sûr.

1.6 Algorithmes de cryptographie asymétrique (à clé publique et privée)

Les algorithmes de cryptographie asymétrique résolvent le problème de l'échange de clés. Ce type d'algorithme a émergé dans les années 1970. La clé publlique permet le chiffrement et la clé privée permet le déchiffrement. La clé privée reste confidentielle. Voici quelques exemples de ce type d'algorithme :

— Protocole d'échange de clés Diffie-Hellman (1977)

— RSA pour Rivest, Shamir et Adleman (1977)

— DSA signifiant Digital Signature Algorithm (1993)

— Cryptographie sur les courbes elliptiques (1985) (Normalisation faite devant la loi)

— Algorithme de Schoof (compte les points de courbes elliptiques sur les corps finis)

— Algorithme de LUC (suites de Lucas)

1.7 Algorithme RSA

L'algorithme RSA a permis de développer le crytage asymétrique où l'envoyeur et le receveur utilisent 2 clés.

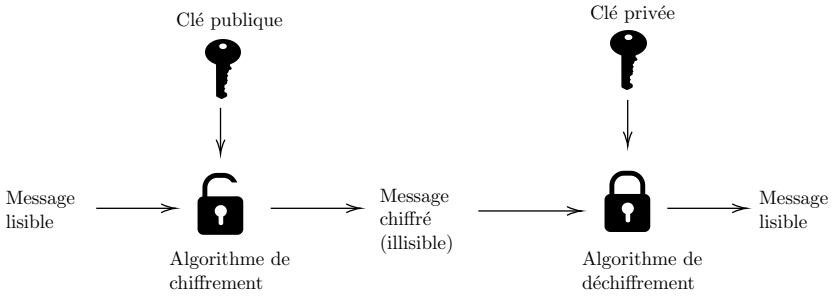


FIGURE 1 – Principe de l'agorithme RSA

Les inconvénients du chiffrement RSA et des autres algorithmes à clé publique résident dans la grande lenteur de processus par rapport aux algorithmes à clés secrètes. La plupart des algorithmes de cryptographie asymétrique sont vulnérables

à des attaques utilisant un calculateur quantique, à cause de l'algorithme de Shor. La cryptographie post-quantique tend à résoudre ce problème. [2]

2. Site de la NIST (National Institute of Standards and Technology), Cryptography in the Quantum Age

2 Fonctions

2.1 Les fonctions au sens large

Définition

Une fonction $f : X \to Y$ est dite **injective** si pour tout $y \in Y$, il existe au plus un $x \in X$ tel que $f(x) = y$. Dit autrement, si

$$\forall x, x' \in X, x \neq x' \Rightarrow f(x) \neq f(x')$$

Définition

Si $f : X \to Y$ est une bijection et g est une bijection de Y vers X : pour chaque $y \in Y$ définissent $g(y) = x$ où $x \in X$ et $f(x) = y$. Cette fonction g obtenue de la fonction f est appelée la *fonction inverse* et est notée $g = f^{-1}$.

Remarque

Si f est une bijection alors f^{-1} l'est aussi. En cryptographie, les bijections sont utilisées comme outil pour crypter des messages et inverser des transformations utilisées pour le décryptage. Si les transformations ne sont pas des bijections alors il n'est pas toujours possible de décrypter un message de façon unique.

Définition

Soit une fonction $f : X \to Y$ est appelée **fonction à sens unique** si $f(x)$ est « facile » à calculer pour tous les $x \in X$ mais pas « essentiellement » pour tous les $y \in \text{Im}(f)$ dont le calcul est infaisable pour trouver un $x \in X$ vérifiant $f(x) = y$.

Définition

Une fonction $f : X \to Y$ est appelée **fonction à sens unique à trappe** (TDF [a]), si elle est une fonction à sens unique ayant la propriété de donner une information supplémentaire : *l'information à trappe* rendant faisable le calcul pour trouver n'importe quel $y \in \text{Im}(f)$, d'un $x \in X$, où $f(x) = y$.

a. Trapdoor function

Exemple

Un nombre premier est un entier naturel strictement supérieur à 1 dont les seuls diviseurs sont 1 et lui-même. Prenons deux nombres premiers $p = 23189$ et $q = 43991$. Le nombre $n = pq$ vaut 1020107299 et X est l'ensemble $X = \{1, 2, \cdots, n-1\}$. Il est relativement « facile » [a] d'inverser cette fonction pour ces nombres. Par contre, s'il on prend des nombres ayant plus de 100 chiffres, le problème est rendu impossible sur un espace de temps fini pour factoriser n avec les ordinateurs actuels. Cela s'appelle le **problème de factorisation des entiers** et est la source d'une mutltitude de fonction à sens unique à trappe. Qui plus est, on ne connaît pas actuellement d'algorithme de factorisation en temps polynomiale.

a. Le terme « facile » désigne un temps de calcul raisonnable pour une machine moderne, de complexité algorithmique polynômiale.

2.2 Fonctions de hachage

2.2.1 Introduction

Les fonctions de hachage ont une utilité importante dans la cryptographie moderne. Elles jouent un rôle essentiel dans la sécurité des données, la finance, les crypto-monnaies et le téléchargement de fichier volumineux par exemple. Le principe d'une fonction de hachage est de renvoyer un *hash-code, hash-result, hash-value, hash, haché (en français)* à partir d'un message.

Les fonctions de hachage (abbrégées **FDH** dans la suite du document) doivent être déterministes, c'est-à-dire pour une même entrée, elles doivent toujours produire le même haché.

La fonction de hachage h désigne une chaîne de n-bits de longueur arbitraire vers une longueur fixée, dites n bits[3]. Un n-bit est un élément de l'ensemble $\{0,1\}^n = R$. Une FDH associe un texte de longueur arbitraire à un hash de longueur n-bit. D représente l'ensemble des textes possibles. Introduisons la fonction $h : D \to R$ avec $|D| > R$. h est une fonction de hachage. On appelle les images par h les *hash-value, hash, haché.*

Exemples : Les fonctions MD_4, MD_5, SHA_{256} sont des fonctions de hachages.

Comme les textes de D ont des longueurs arbitrairement grandes[4] alors que R est un ensemble fini. (de 2^n éléments). La fonction h n'est pas injective.

La fonction h renvoie une « image compacte ». En d'autres termes, l'un des intérêts du hachage est que l'image texte par h, soit le haché, est une représentation condensée du texte initial. Donc le haché est une chaîne de caractères réduite par rapport au texte initial mais comme on le verra par la suite, les hachés rassemblent suffisament d'informations pour retrouver le texte initial.

La non injectivité de cette fonction implique l'existence de collisions (deux hashs identiques avec des messages différents). Des collisions peuvent être testées par le paradoxe des anniversaires (expliqué ultérieurement dans la section 2.3). Les fonctions de hachage permettent d'obtenir des *hachés* servant de représentation d'image compacte. On les appelle aussi *imprint, digital fingerprint* ou *message digest* d'une entrée de caractères et ces *hachés* peuvent être utilisées comme si

3. un bit correspond à la plus petite unité d'information en informatique pouvant avoir deux valeurs : 0 ou 1.

4. Dans certains contextes, les textes ou bases de données initiaux sont de petites tailles comme pour les mots de passe par exemple. Néanmoins, les algorithmes des fonctions de hachage comprennent une étape de normalisation (dite « padding » signifiant remplissage). L'étape de padding consiste à ajouter des bits supplémentaires à l'entrée de manière à ce que sa longueur soit un multiple de n de la taille de bloc de la fonction de hachage. Cela est généralement nécessaire pour garantir que l'entrée puisse être traitée de manière cohérente par la fonction de hachage, quelle que soit sa longueur car $|D| > R$ par définition .

elles étaient uniquement identifiables avec ce nombre fini de caractère.

Les fonctions de hachage sont utilisées pour préserver l'intégralité des données avec les procédés de signatures digitales. En pratique, un message est d'abord haché puis sa *hash-value* est signée à la place du message initial.

Illustrons ce qu'il se passe en pratique par le schéma ci-dessous.

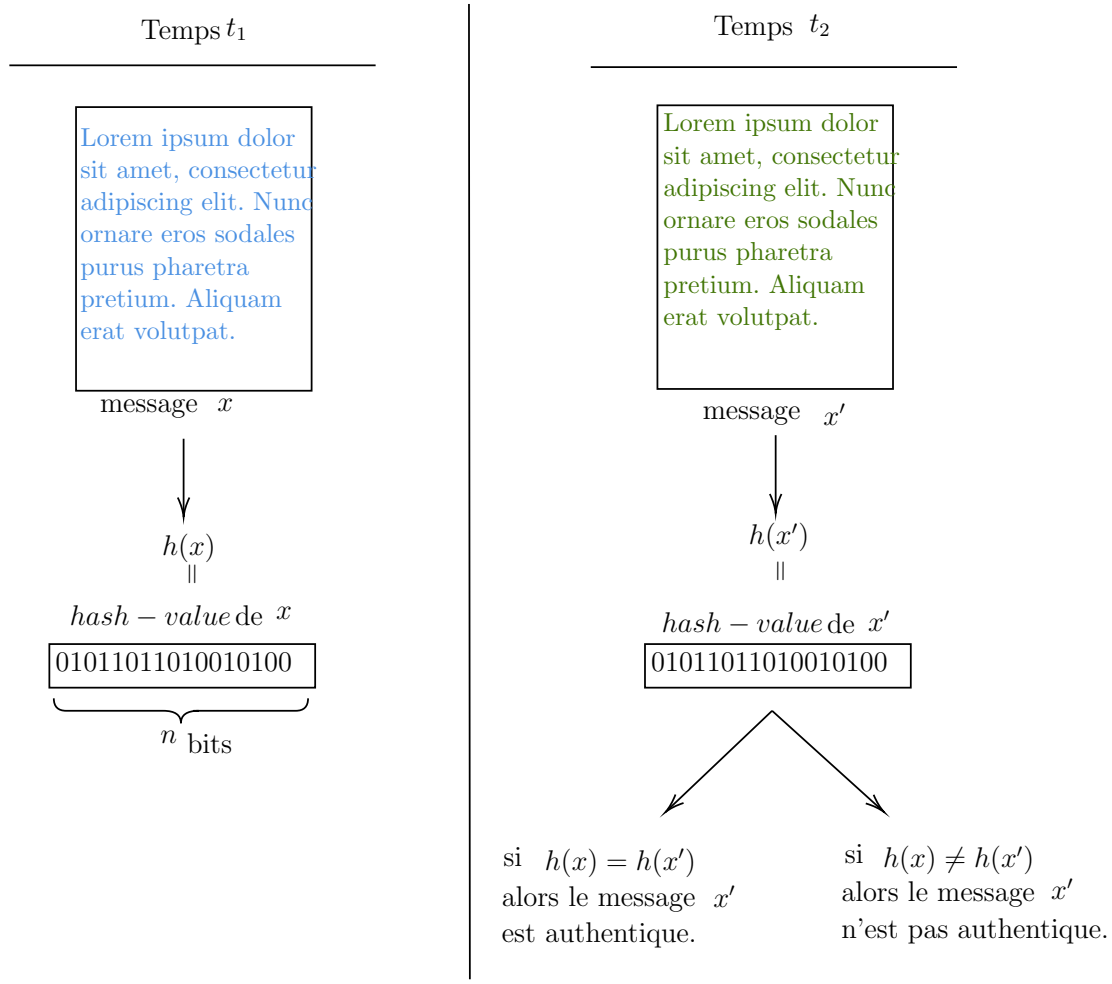


FIGURE 2 – Fonctionnement d'une fonction de hachage

Le **problème** qui se pose est la non injectivité des fonctions de hachage. Il faut ainsi trouver le n qui convient puis discuter de la taille des messages initiaux en fonction de la probabilité afin d'obtenir des collisions. On peut ainsi se demander dans quelle mesure les fonctions de hachage sont suffisament fortes pour palier au problème des collisions.

Lorsqu'une fonction de hachage produit un **hash identique**, cela ne signifie pas nécessairement que les données initiales sont identiques. Cela est dû au fait

que les fonctions de hachage sont conçues pour produire des hachés de taille fixe (n), quelle que soit la taille de l'entrée. (de longueur t). Ces collisions peuvent être intentionnelles ou accidentelles. Dans le cas des collisions intentionnelles, un attaquant peut chercher à trouver deux entrées différentes qui produisent le même haché afin de tromper le système. Dans le cas des collisions accidentelles, il s'agit simplement d'une coïncidence due à la nature probabiliste des fonctions de hachage. Bien que la probabilité de collision soit très faible, elle n'est pas nulle.

Lorsqu'une fonction de hachage renvoie des **hashs différents**, cela implique qu'il est fort probable que les textes soient différents. Une fonction de hachage est conçue pour avoir une faible probabilité que deux entrées produisent le même haché.

Exemple : Pour la fonction de hachage **SHA-256**, celle-ci produit un haché de 256 bits. Il existe donc 2^{256} car pour chaque bit, il y a deux choix possibles : 0 ou 1 (soit environ 10^{77} combinaisons possibles de hachés différents). Cette immense quantité de combinaisons rend pratiquement impossible la création de deux entrées différentes produisant le même haché.

2.2.2 Procédés mathématiques utilisés par les FDH

Les fonctions de hachage confrontent plusieurs théories mathématiques comme la théorie des groupes, la théorie des nombres, la théorie de l'information, la théorie de la complexité algorithmique et la théorie des probabilités.

Voici l'explication détaillée des concepts mathématiques sous-jacents aux FDH :

- **la théorie des groupes** : les fonctions de hachage utilisent souvent des opérations sur les groupes, telles que l'addition modulo un nombre premier, pour mélanger les bits d'un message avant de le hacher. Les groupes abéliens et non abéliens sont particulièrement utiles pour la conception de fonctions de hachage cryptographiques.
- **la théorie des nombres** : les fonctions de hachage utilisent souvent des propriétés des nombres premiers et des fonctions arithmétiques pour créer des hashs uniques. Par exemple, les fonctions de hachage cryptographiques utilisent souvent des nombres premiers aléatoires pour diviser le message en blocs et les hacher séparément.
- **la théorie de l'information** : les fonctions de hachage sont utilisées pour réduire la taille d'un message tout en préservant autant que possible l'information qu'il contient. Les fonctions de hachage sont conçues pour minimiser les collisions, c'est-à-dire les situations où deux messages différents produisent le même hash.
- **la théorie de la complexité algorithmique** : les fonctions de hachage sont conçues pour être efficaces à calculer, mais difficiles à inverser. Les fonctions de hachage sont souvent mesurées en termes de leur résistance aux attaques

de collision et aux attaques d'antécédents, qui sont toutes deux liées à la complexité des algorithmes nécessaires pour les casser.

— **la théorie des probabilités** : les fonctions de hachage sont sujettes à des collisions, c'est-à-dire des situations où deux messages différents produisent le même hash. Les fonctions de hachage sont conçues pour minimiser les collisions en utilisant des techniques telles que le mélange des bits du message, l'utilisation de nombres premiers aléatoires et la sélection d'une taille de hash appropriée.

Exemple de structure d'algorithme utilisé dans les fonctions de hachage :

— **Compression** : une FDH prend une entrée initiale qui peut être de taille variable et la comprime en une forme fixe. La compression est effectuée en plusieurs étapes : substitution, permutation, addition modulo 2.

— **Clé de hachage** : certaines FDH nécessitent une clé de hachage qui est un paramètre fixe, qui peut être utilisé pour personnaliser la FDH en fonction des besoins de l'utilisateur.

— **Bouclage** : la compression est effectuée plusieurs fois dans une boucle. Cela permet à la FDH de prendre en compte toutes les parties de l'entrée, quelque soit leur ordre.

— **Clé de hachage** : après la compression, la FDH peut appliquer une série d'opérations supplémentaires pour finaliser le haché. Cela peut inclure des opérations tel que la concaténation des bits, l'application des fonctions non linéaires et la réduction de la taille du haché.

2.2.3 Classification des familles

Une famille de ces fonctions de hachage est nommée **MACs** pour ***M**essage **A**uthentication **C**odes*, autorisant l'authentification des messages par des techniques symétriques.

Principe des algorithmes MACs :

— Entrée : un message et une clé tenue secrète.

— Sortie : une chaîne de caractère de n-bits.

Il est infaisable en pratique de produire le même output (haché) sans connaître la clé.

Exemples :

— **HMAC** (**H**ash-based **M**essage **A**uthentication **C**ode) : HMAC est une technique de calcul de code d'authentification de message qui utilise une fonction de hachage cryptographique (comme SHA-256, SHA-384 ou SHA-512) en combinaison avec une clé secrète partagée entre les parties qui communiquent. HMAC fournit à la fois l'authentification et l'intégrité des données.

— **Poly1305** : Poly1305 est une technique de calcul de code d'authentification de message qui utilise un algorithme de chiffrement à flux en combinaison avec une clé secrète partagée entre les parties qui communiquent. Poly1305 fournit à la fois l'authentification et l'intégrité des données.

— **GCM** (**G**alois/**C**ounter **M**ode) : GCM est une technique de chiffrement de blocs qui intègre également une fonction de calcul de code d'authentification de message. GCM utilise une fonction de hachage cryptographique (comme SHA-256 ou SHA-512) pour fournir l'authentification et l'intégrité des données. GCM est souvent utilisée pour les communications sécurisées via Internet (par exemple, dans les protocoles TLS/SSL).

À noter que les fonctions de hachage utilisées dans les MAC sont souvent les mêmes que celles utilisées dans les MDC (terme défini dans le paragraphe qui suit), mais les MAC utilisent ces fonctions de hachage d'une manière différente pour fournir l'authentification et l'intégrité des données.

Une autre famille est nommée **MDCs** pour ***M****odification* ***D****etection* ***C****ode*.

Exemples :

— **MD5** (**M**essage **D**igest 5) : MD5 est une FDH à sens unique et à résistance à la collision. Elle produit un hachage de 128 bits pour un message d'entrée donné. Bien que MD5 soit encore largement utilisée, elle est considérée comme peu sécurisée pour les applications critiques en raison de faiblesses découvertes dans sa conception.

— **SHA-1** (**S**ecure **H**ash **A**lgorithm 1) : SHA-1 est une fonction de hachage à sens unique et à résistance à la collision. Elle produit un hachage de 160 bits pour un message d'entrée donné. Tout comme MD5, SHA-1 est maintenant considérée comme peu sécurisée pour les applications critiques.

— **SHA-2** (**S**ecure **H**ash **A**lgorithm 2) : SHA-2 est une famille de fonctions de hachage cryptographiques à sens unique et à résistance à la collision. Elle comprend des fonctions de hachage telles que SHA-256, SHA-384 et SHA-512. Ces fonctions de hachage sont beaucoup plus sécurisées que MD5 et SHA-1 et sont couramment utilisées pour la signature numérique, la vérification d'intégrité des données, la génération de clés aléatoires, etc.

— **SHA-3** (**S**ecure **H**ash **A**lgorithm 3) : SHA-3 est une famille de fonctions de hachage cryptographiques à sens unique et à résistance à la collision. Elle est basée sur le nouveau standard de l'Institut National des Standards et de la Technologie (NIST) appelé Keccak. SHA-3 est conçue pour être plus résistante aux attaques cryptographiques que SHA-2.

Propriété

Une FDH h a au minimum les deux propriétés suivantes :

- **Compression** : h caractérise un input (un texte, une base de donnée) x arbitraire de longueur fini de bit vers un output $h(x)$ de longueur de bit n fixé.
- **Facile à calculer** : pour un h donné et un input, $h(x)$ est facile [a] à calculer.

a. Il n'y a pas de définition précise de ce que signifie qu'une fonction de hachage est « facile » à calculer, car cela dépend de nombreux facteurs tels que la puissance de calcul disponible, les attaques cryptographiques connues, la longueur de l'entrée, etc.

Cependant, on peut dire qu'une fonction de hachage est considérée comme « facile » à calculer si elle peut être évaluée en un temps raisonnable avec les ressources de calcul disponibles. Le temps raisonnable dépend de l'objectif de la fonction de hachage et du contexte dans lequel elle est utilisée. Par exemple, pour une application de vérification d'intégrité de fichiers simples, une fonction de hachage pouvant être évaluée en quelques millisecondes peut être considérée comme « facile », tandis que pour une application de sécurité critique comme la signature numérique, une fonction de hachage qui peut être brisée en quelques minutes par une attaque cryptographique puissante ne serait pas considérée comme « sûre ».

En général, les fonctions de hachage cryptographiques modernes sont conçues pour résister à de nombreuses attaques cryptographiques, telles que la cryptanalyse différentielle et la cryptanalyse par collision, et sont considérées comme sûres lorsqu'elles sont utilisées correctement. Cependant, de nouvelles attaques cryptographiques peuvent être découvertes à l'avenir, et les fonctions de hachage peuvent devenir plus vulnérables avec le temps si elles sont exposées à des ressources de calcul plus puissantes.

A propos des FDH sans clé, on peut ajouter des propriétés complémentaires :

Propriété

- **Résistance à l'antécédent** : pour la majorité de tous les outputs, il est difficile de calculer en temps polynomial un antécédent x' comme $h(x') = y$ pour un y donné. (y est l'output)
- **Résistance au second antécédent** : il est difficile de trouver x', pour un x donné, tel que $x \neq x' \Rightarrow h(x') = h(x)$.
- **Résistance au collision** : libre choix du x et x', il est difficile de trouver $x \neq x' \Rightarrow h(x) = h(x')$.

A propos des FDH avec clé, elles doivent facilement calculable, posséder les propriétés de compression et de résistance au collision.

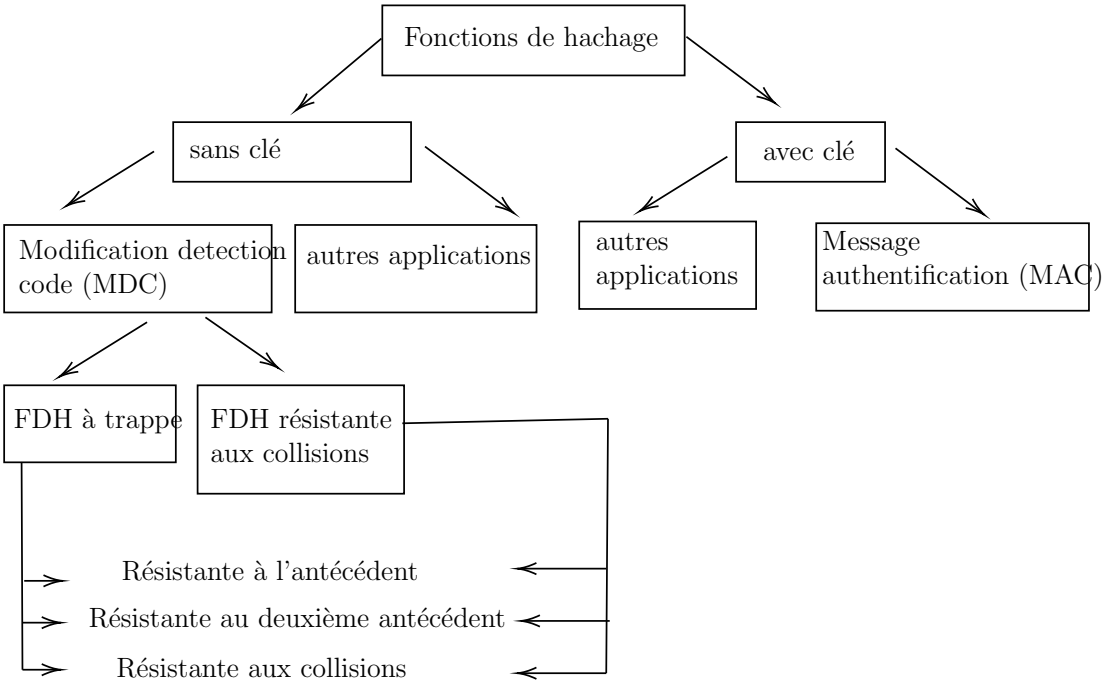


FIGURE 3 – Classification des fonctions de hachage

2.3 Attaque par le paradoxe des anniversaires

Que signifie l'attaque par le paradoxe des anniversaires et en quoi cela est-il un paradoxe ?

Une **attaque** est une tentative de compromettre la sécurité d'un système cryptographique en tentant de trouver une méthode pour déchiffrer un message ou pour découvrir la clé secrète utilisée pour chiffrer les données.

Dans le contexte des fonctions de hachage, l'attaque des anniversaires peut être utlisée pour modifier les communications entre deux personnes ou plus. L'attaque est possible grâce à la probabilité plus élevée de collisions avec des tentatives d'attaques aléatoires et un niveau fixe de permutations, comme dans le principe des tiroirs.

Le principe est de calculer la probabilité p pour que dans un groupe de n personnes deux au moins aient leur date d'anniversaire identique. (Cela correspond à une collision du haché.) Dans une année, il y a $N = 365$ jours [5].

Numérotons les personnes de 1 à n. Leurs dates d'anniversaires forment un n-uplet $(d_1, \cdots, d_n) \in \{1, \cdots, N\}^n$. Ainsi, il y a N^n possibilités. Parmi ces possibilités, seulement $N(N-1)(N-2)\cdots(N-n+1)$ ne contiennent pas deux dates identiques, c'est-à-dire toutes les dates sont différentes. [6]

La calculatrice ne renvoyant pas un résultat pour ce genre de nombre aussi grand, calculons cette probabilité grâce au code python ci-dessous pour $n = 30$.

5. En simpliant sans les années bissextiles.

6. Au rang 1, il y a N dates différentes, au rang 2, il n'y a plus que $N-1$ dates potentiellement différentes en enlevant d_1, au rang 3, on peut enlever deux dates : d_1, d_2, il reste $N-2$ dates pouvant être différentes. On procède ce raisonnement jusqu'au n-ième rang, où l'on a enlevé $n-1$ dates, il reste $N-n+1$ dates pouvant être différentes.

```
from random import *
from math import *

N = 1000000
succes = 0

# Calcul de la probabilité qu'au moins deux dates
   soient identiques.
for _ in range(N):
    annivs = [randrange(365) for __ in range(30)] # la
       valeur 30 correspond aux nombres de personnes
    if len(set(annivs)) != len(annivs):
        succes += 1
    p=succes/N
print(f"proba approchée qu'au moins deux dates sont
   identiques. {round((p)*100,3)} %")

# Calcul de la probabilité qu'au moins deux dates
   soient identiques :
print(f"proba approchée dates toutes différentes
   {round((1-p)*100,3)} %")
```

./programmes/anniv.py

Sur 10^7 essais, la probabilité vaut 70.612%.

Dressons un tableau de valeurs pour des valeurs remarquables n variant de 1 à 70 et observons ce qu'il se passe.

Nombre de personnes	**Probabilité que deux personnes soient nées le même jour**	**Probabilité que toutes les dates soient différentes**
1	0%	100%
2	0.27%	99.73%
3	0.82%	99.18%
5	2.699 %	97.301%
10	11.673%	88.327%
15	25.362%	74.638%
20	41.184%	58.816%
23	50.716 %	49.284%
25	56.957%	43.043%
30	70.596%	29.404%
40	89.136%	10.864%
50	97.027%	2.973%
70	99.917%	0.083%

On dit que c'est un paradoxe car en seulement 23 effectifs, on a une chance sur deux que deux personnes soient nées le même jour : ceci contredit l'intuition. (on penserait qu'il faudrait bien plus de personnes pour atteindre cette valeur de probabilité).

Remarque

Le calcul de la force et de la faiblesse des fonctions de hachage se base sur celui fait dans la résolution du paradoxe des anniversaires. Pour une empreinte égale à n, il faut $2^{\frac{n}{2}}$ essais pour trouver une collision au hasard.

Justifions rigoureusement la remarque précedente.

Nous allons utiliser le principe du paradoxe des anniversiares appliqué aux fonctions de hachage. Par définition une FDH h renvoie un hash de longueur n-bits où n est fixé. Comme le bit est l'unité en informatique prenant uniquement pour valeur 0 ou 1. Il y a 2^n combinaisons de hash différents. Cherchons à calculer la probabilité qu'il n'y ait pas de collision parmi k hachés. Dans un premier temps, on peut dénombrer le nombre de cas où les hashs sont tous différents :

$$2^n \cdot (2^n - 1)(2^n - 2) \times \cdots \times (2^n - (k-1))$$

Puis, le nombre de cas possibles correspond à : $\underbrace{(2^n \cdot 2^n \cdots 2^n)}_{\text{k fois}}$

En supposant qu'il y ait équiprobabilité : $p(k,n) = \dfrac{\text{nombre d'issues favorables}}{\text{nombre d'issues possibles}}$

$$\begin{aligned}
p(k,n) &= \frac{2^n \cdot (2^n - 1) \cdot (2^n - 2) \cdots (2^n - (k-1))}{2^n \cdot 2^n \cdots 2^n} \\
&= 1 \cdot \left(1 - \frac{1}{2^n}\right) \cdots \left(1 - \frac{k-1}{2^n}\right) \qquad \text{on suppose que } k \ll n \Rightarrow 1 - x \leq e^{-x} \\
&\leq 1 \times e^{-\frac{1}{2^n}} \times \cdots e^{-\frac{k-1}{2^n}} \\
&= \exp\left(-\frac{1 + \cdots + (k-1)}{2^n}\right) \\
&= \exp\left(-\frac{((k-1)+1) \cdot (k-1)}{2 \cdot 2^n}\right) \\
&= \exp\left(-\frac{-k(k-1)}{2 \cdot 2^n}\right) \\
&= \exp\left(-\frac{k^2}{2^{n+1}}\right)
\end{aligned} \tag{1}$$

Remarque

Dans la pratique, on suppose que $k \ll n$ car les valeurs de n deviennent sécurisée au-delà de n bits suffisament grands, k représente le nombre de haché en collision : les attaquants ne cherchent pas beaucoup de hachés qui soient identiques. En effet, le but est de trouver deux hachés identiques puis d'altérer la donnée initiale, ainsi $k = 2$ dans ce principe. Dans la suite du document, on discutera de quelle valeur de n pouvant convenir le mieux pour assurer la fiabilité des FDH.

D'après le paradoxe des anniversaires, nous avons obtenu deux séries de résultats dans le tableau **??** . Dans le deuxième colonne intitulée « probabilités que deux personnes soient nées le même jour », on remarque qu'à la ligne $n = 23$, on obtient une probabilité de collision de 50.716%.

Dans la suite des calculs, nous allons supposer qu'une FDH n'est pas suffisament sécurisée pour une probabilité supérieure à 0.5.

Cela revient à dire que $0.5 < p(k, n)$. Or, nous avions trouvé précédemment une majoration sur $p(k, n)$. Ainsi,

$$\begin{aligned}
0.5 < p(k,n) &\leq \exp(-\frac{k^2}{2^{n+1}}) \\
\Leftrightarrow \frac{1}{2} &\leq \exp(-\frac{k^2}{2^{n+1}}) \\
\Leftrightarrow \ln(\frac{1}{2}) &\leq -\frac{k^2}{2^{n+1}} \\
\Leftrightarrow 2^{n+1} \cdot \ln(2) &\geq k^2 \\
\Leftrightarrow \sqrt{2^{n+1} \cdot \ln(2)} &\geq k \quad \text{car } k \geq 0 \\
\Leftrightarrow \sqrt{2 \times \ln(2)} \cdot \sqrt{2^n} &\geq k \\
\Leftrightarrow \sqrt{2 \times \ln(2)} \cdot 2^{n/2} &\geq k \\
\approx 1,1 \cdot 2^{n/2} &\geq k
\end{aligned} \tag{2}$$

Par conséquent, pour un hash de taille n, il faut au moins $2^{n/2}$ essais pour trouver une collision.

Concrètement, pour des hashs de longueurs de 64 bits, une probabilité de collision supérieure à 0.5 sera trouvée en 1.1×2^{32} essais, soit moins de 5 milliards d'essais [7]. Ce nombre d'essais est largement faisable pour les machines modernes [8].

7. La valeur exacte est 4 724 464 025.6, il font donc précisément 4 724 464 026 essais.
8. TP jupyter de l'université Gustave Eiffel, Institut d'électronique et d'informatique gaspard monge.

La puissance de calcul d'un ordinateur dépend de plusieurs facteurs tels que la vitesse du processeur, la quantité de mémoire vive (RAM), la capacité de stockage, le type de carte graphique, la vitesse de l'interface d'E/S (entrée/sortie) et bien d'autres facteurs.

La mesure la plus courante pour évaluer la puissance de calcul d'un ordinateur est le nombre d'opérations à virgule flottante par seconde (**FLOPS**[9]). Cette mesure est utilisée pour évaluer la capacité de l'ordinateur à effectuer des calculs mathématiques complexes.

Le nombre de FLOPS d'un ordinateur dépend de la vitesse de son processeur, de la quantité et de la vitesse de sa mémoire RAM et de la qualité de sa carte graphique. Les ordinateurs modernes peuvent avoir une puissance de calcul allant de quelques gigaflops (milliards d'opérations à virgule flottante par seconde, c'est-à-dire de l'ordre de 10^9 FLOPS) à plusieurs téraflops (10^{12} opérations à virgule flottante par seconde) voire plus pour les supercalculateurs.

Il convient de noter que la puissance de calcul d'un ordinateur ne détermine pas nécessairement sa capacité à exécuter des tâches spécifiques. Par exemple, un ordinateur équipé d'un processeur plus lent mais avec une mémoire RAM plus rapide pourrait être plus performant pour certaines tâches que celui avec un processeur plus rapide et une mémoire RAM plus lente.

Les supercalculateurs[10] sont des ordinateurs conçus pour effectuer des calculs intensifs à grande échelle et ils sont donc beaucoup plus puissants que les ordinateurs de bureau ou les serveurs conventionnels.

Leur puissance de calcul est mesurée en petaflops (10^{15} opérations à virgule flottante par seconde). Les supercalculateurs modernes peuvent avoir une puissance de calcul allant de quelques petaflops à plusieurs exaflops.

En 2022, la barre de l'exaFLOPS (d'ordre de grandeur 10^{18}) a été dépassé par le superordinateur américain Frontier.[11]

Il convient de noter que la puissance de calcul des supercalculateurs est me-

9. Floating-Point Operations Per Second

10. Ces ordinateurs sont utilisés pour des applications qui nécessitent une grande puissance de calcul, telles que la simulation de phénomènes physiques, l'analyse de données complexes, la modélisation de systèmes météorologiques et climatiques, l'analyse de risques financiers, la conception de médicaments et d'autres applications scientifiques et industrielles. Les supercalculateurs sont généralement utilisés dans des environnements de recherche scientifique, de défense, d'industrie et de finance, où les exigences en matière de calcul intensif sont très élevées. Ils sont également utilisés dans les centres de données pour traiter de grandes quantités de données, notamment dans les domaines de l'intelligence artificielle et de l'apprentissage automatique.

11. D'après le site TOP500 (TOP500 est réalisé par Hans Meuer de l'université de Mannheim en Allemagne, Jack Dongarra de l'université du Tennessee à Knoxville, Erich Strohmaier et Horst Simon du National Energy Research Scientific Computing Center (NERSC) du Lawrence Berkeley National Laboratory (LBL)), « Frontier is the new No. 1 system in the TOP500. This HPE Cray EX system is the first US system with a peak performance exceeding one ExaFlop/s. »

surée en FLOPS et en petaflops pour évaluer leur capacité à effectuer des calculs intensifs à grande échelle. Cependant, la performance des supercalculateurs dépend également de leur architecture, de leur connectivité et de leur capacité à traiter efficacement de grandes quantités de données.

Prenons l'exemple de la fonction de hachage SHA_{256}, pour cette fonction de hachage donnant en sortie un hash de $n = 256$ bits. On a prouvé qu'il fallait $1.1 \times 2^{n/2}$ essais pour atteindre une collision. Il faut donc :

$$\begin{aligned} 1.1 \times 2^{256/2} &= 1.1 \times 2^{128} \\ &\approx 3.7 \cdot 10^{38} \text{ essais} \end{aligned}$$

Conclusion

Par conséquent, pour des attaquants utilisant des ordinateurs modernes, les puissances de calculs sont de l'ordre du téraflops (10^{12} FLOPS) et pour des attaquants utilisant des supercalculateurs, les puissances de sont de l'ordre de l'exaflops (10^{18} FLOPS). On peut remarquer que pour des FDH où $n = 64$ bits, les puissances de calculs sont de l'ordre du gigaflops (nécessitant $4 \cdot 10^9$ essais). Puis, pour les FDH à 128-bits, les puissances de calculs sont de l'ordre de 10 fois l'exaflops. Pour des ordinateurs modernes, la FDH reste suffisament sécurisée mais selon l'agence NIST [a] , elles ne le sont plus étant donnés qu'il existe des supercalculateurs capables de calculer en exaflops. Ainsi, on considère qu'à partir de 256-bits, les FDH sont suffisament sécurisées. (nécessitant $3.7 \cdot 10^{38}$ essais dépassant largement l'ordre de grandeur de l'exaflops). Exemple : La FDH $SHA - 256$ peut-être considérée comme une fonction fiable.

a. National Institute of Standards and Technology (Institut national des normes et de la technologie)

3 Applications des fonctions de hachage

On a pu voir dans les parties précédentes que les fonctions de hachages permettaient de compresser des données de grandes ou petites tailles. On peut être amener à penser que ces compressions de données peuvent jouer un rôle dans le cadre de conversion d'un fichier au format .mp4 vers .mp3. Cependant, la compression des fichiers au format MP3 ne repose pas sur une fonction de hachage mais sur un algorithme de compression audio avec pertes. L'algorithme de compression utilisé pour les fichiers MP3 est appelé **MPEG-1 Layer III** [12] . Cet algorithme utilise une technique de compression appelée « codage perceptuel ». Cela permet de supprimer certaines informations du signal audio qui ne sont pas perceptibles pour l'oreille humaine, tout en préservant les parties les plus importantes du signal.

3.1 Dans la bureautique

Les fonctions de hachages sont souvent utilisées pour sécuriser les mails en les rendant plus difficiles à falsifier ou à intercepter. Lorsqu'un courrier électronique est envoyé, il passe souvent par plusieurs serveurs avant d'arriver à sa destination finale. Chaque serveur est potentiellement vulnérable à des attaques de piratage ou de falsification, ce qui peut entraîner une altération du contenu du courrier électronique. Pour éviter cela, les fournisseurs de messagerie peuvent utiliser des fonctions de hachages pour créer une empreinte numérique (*digital fingerprint*) unique du contenu du courrier électronique. Cette empreinte est ensuite envoyée avec le courrier électronique et peut être vérifiée par le destinataire pour s'assurer que le contenu n'a pas été altéré pendant la transmission. On peut comparer la signature numérique dun e-mail à une signature manuscrite dans un courrier. Celle-ci est unique et sans équivoque. A noter, qu'une signature numérique ne chiffre pas le message mais apporte uniquement la preuve de son intégrité. Les contenus confidentiels doivent faire lobjet dun chiffrement supplémentaire. Plus généralement, ces fonctions permettent l'identification de fichiers. Elles peuvent être utilisées pour identifier de manière unique des fichiers ou des ensembles de données. Chaque fichier a une empreinte numérique unique qui peut être calculée à l'aide d'une fonction de hachage, ce qui permet de s'assurer que le fichier n'a pas été modifié ou remplacé.

12. Le format MP3 est un format propriétaire : dont les spécifications sont contrôlées par des intérêts privés.

3.2 En informatique

Les fonctions de hachage sont également utilisées pour stocker les mots de passe de manière sécurisée dans les bases de données des fournisseurs de messagerie, afin d'éviter les violations de données et les accès non autorisés aux comptes de messagerie. Au lieu de stocker les mots de passe en clair dans une base de données, les fournisseurs de services peuvent utiliser ces fonctions pour stocker des empreintes numériques des mots de passe. De cette façon, même si la base de données est compromise, les pirates informatiques ne peuvent pas facilement récupérer les mots de passe en clair.

Par ailleurs, les fonctions de hachage sont souvent utilisées pour garantir que des données n'ont pas été altérées ou corrompues. Par exemple, lorsqu'un fichier est téléchargé depuis un site web, sa somme de contrôle peut être calculée à l'aide d'une fonction de hachage et comparée à celle fournie par le site web pour s'assurer que le fichier n'a pas été modifié pendant le téléchargement.

Les fonctions de hachages sont aussi souvent utilisées dans les protocoles de cryptographie pour fournir des signatures numériques et des clés de chiffrement. Par exemple, une signature numérique peut être créée en utilisant une fonction de hachage pour créer une empreinte numérique du message, qui est ensuite chiffrée avec la clé privée de l'expéditeur.

Les fonctions de hachage peuvent être utilisées pour accélérer les opérations de recherche dans les grandes bases de données. Les valeurs de hachage peuvent être utilisées comme clés de recherche, ce qui permet de trouver rapidement les données correspondantes sans avoir à parcourir toute la base de données.

(Exemple : Les FDH sont présentes dans tous les langages de programmations modernes comme en Java, Python , C++)

Les fonctions de hachage sont des algorithmes qui permettent de transformer des données en une valeur de hachage unique et fixe. Cette valeur de hachage peut ensuite être utilisée pour vérifier l'intégrité des données, c'est-à-dire s'assurer qu'elles n'ont pas été modifiées ou altérées.

Dans le cadre de SSL[13], les fonctions de hachage sont utilisées pour garantir l'intégrité des données échangées entre le navigateur et le serveur web. Avant de chiffrer les données à l'aide de la clé de session partagée, le serveur web peut appliquer une fonction de hachage aux données. La valeur de hachage est ensuite envoyée au navigateur web, qui peut vérifier l'intégrité des données reçues en recalculant la valeur de hachage et en la comparant à celle envoyée par le serveur web.

13. SSL signifie Secure Sockets Layer est un protocole de sécurité qui permet de sécuriser les échanges de données entre un serveur web et un navigateur web. Il utilise des techniques de cryptographie pour garantir que les données échangées entre les deux parties ne peuvent être ni lues ni modifiées par des tiers malveillants.

Si la valeur de hachage calculée par le navigateur web correspond à celle envoyée par le serveur web, cela signifie que les données n'ont pas été altérées pendant le transfert et que leur intégrité est garantie.

En outre, le stockage temporaire des informations dans des caches est également chiffré à laide de hashs afin que les utilisateurs non autorisés ne puissent pas déterminer les sites internets visités et les données daccès de paiement utilisées lors de la consultation du cache.

De même, les fonctions de hachage publiques réputées fortes étant par nature à la disposition de tous, il est techniquement possible pour tout un chacun de calculer des empreintes. Aujourdhui, on trouve facilement sur internet des dictionnaires immenses dempreintes MD5 précalculées. Grâce à ces données, il est aisé de retrouver instantanément le mot de passe ayant été utilisé afin de générer ces empreintes. Afin de limiter ce risque, il est conseillé dutiliser des fonctions spécialisées appelées « fonction de dérivation de clé », telles que bscrypt [14] ou Argon2 [15] par exemple, qui sont conçues spécifiquement pour stocker des mots de passe.

14. bcrypt est une fonction de hachage créée par Niels Provos et David Mazières. Elle est basée sur l'algorithme de chiffrement Blowfish et a été présentée lors de USENIX en 1991. En plus de l'utilisation d'un sel pour se protéger des attaques par table arc-en-ciel (rainbow table), bcrypt est une fonction adaptative, c'est-à-dire que l'on peut augmenter le nombre d'itérations pour la rendre plus lente. Ainsi elle continue à être résistante aux attaques par force brute malgré l'augmentation de la puissance de calcul.

15. Argon2 est une fonction de dérivation de clé, gagnante de la Password Hashing Competition en juillet 2015. Elle a été conçue par Alex Biryukov, Daniel Dinu et Dmitry Khovratovich de l'Université de Luxembourg et publié sous licence Creative Commons CC0.

Une *rainbow table* (littéralement table arc-en-ciel) est une structure de données [16] pour retrouver un mot de passe à partir de son empreinte. La table arc-en-ciel ne dépend que de la fonction de hachage considérée. Cette table est constituée d'une grande quantité de paires de mots de passe. Pour obtenir chacune de ces paires, en partant d'un mot de passe, on calcule son empreinte. Une « fonction de réduction » (variant selon la position dans la table) permet de recréer un nouveau mot de passe à partir de cette empreinte. Après avoir effectué un nombre fixe d'itérations, on obtient un mot de passe qui forme le deuxième élément de la paire. La table est créée une fois pour toutes et permet la recherche d'un mot de passe, connaissant son empreinte par la fonction de hachage considérée.

L'algorithme développé par Oechslin optimise la détection des collisions et le parcours dans la table. Des solutions pour réduire la taille de la table ont également été proposées. Plusieurs tables peuvent être produites pour améliorer les chances de réussite.

Il existe dans le langage de programmation Python, une fonction hash() qui permet de hasher des caractères.

3.2.1 Flash Code

Les fonctions de hachage peuvent être utilisées pour générer un flash code (aussi appelé QR code), qui est un code-barres en deux dimensions qui peut être scanné à l'aide d'un smartphone ou d'un lecteur de code-barres pour accéder à des informations spécifiques.

Lorsqu'un flash code est généré, il est généralement associé à une URL ou à une autre forme de données. Pour garantir l'intégrité et la sécurité de ces données, un hachage peut être calculé à partir de ces données à l'aide d'une fonction de hachage. Le hachage est ensuite inclus dans le flash code pour permettre à l'application de vérifier que les données n'ont pas été altérées depuis leur création.

Lorsque le flash code est scanné, l'application qui le lit peut calculer le hachage des données et le comparer au hachage inclus dans le flash code. Si les deux hachages correspondent, cela indique que les données n'ont pas été altérées, et l'application peut alors afficher les informations associées au flash code. L'un des algorithmes de hachage les plus couramment utilisés dans la génération de codes QR est SHA-256. SHA-256 est considérée comme étant très résistante aux collisions et aux attaques par force brute [17], ce qui en fait un choix populaire pour la génération de codes QR sécurisés (Exemple : Les flashs-codes présent sur le pass vaccinal COVID-19). D'autres fonctions de hachage, telles que MD5 ou SHA-1, sont également utilisées pour la génération de codes QR, mais elles sont considérées comme moins sécurisées que SHA-256 et sont donc moins couramment utilisées.

16. Elle a été créée en 2003 par Philippe Oechslin de l'EPFL1.
17. Référence de la NIST

3.2.2 Dans la blockchain

SHA-256 est utilisé dans la technologie de la blockchain pour sécuriser les données et garantir l'intégrité de la chaîne de blocs.

Dans une blockchain, chaque bloc de données est associé à un hachage unique qui est généré à partir du contenu du bloc. Le hachage est une empreinte numérique de la donnée, qui peut être considérée comme une représentation unique et condensée du bloc de données. La fonction de hachage SHA-256 est utilisée pour générer ces empreintes numériques.

Le fonctionnement de la blockchain repose sur le fait que chaque bloc de données contient le hachage du bloc précédent, ce qui crée une chaîne de blocs interconnectés. Si une donnée est modifiée dans un bloc, cela entraînera un changement dans son hachage, qui se propagera à tous les blocs suivants dans la chaîne de blocs. Par conséquent, toute tentative de modification d'un bloc de données antérieur serait rapidement détectée et rejetée par le réseau, car elle ne correspondrait pas au hachage enregistré précédemment.

On peut citer la crypto-monnaie Bitcoin et aussi l'université de LILLE utilisant la blockchain pour partager de façon sûre les attestations numériques de réussite au diplôme. [18] [19]

4 Approfondissement avec la fonction de hachage SHA-256

4.1 Introduction sur la fonction de hachage SHA-256

SHA-256 (Secure Hash Algorithm 256 bits) est une fonction de hachage largement utilisée pour la sécurité des données et des communications. Elle produit une empreinte numérique unique pour chaque entrée de données, ce qui permet de vérifier l'intégrité et l'authenticité des données. La fonction SHA-256 prend en entrée des données de n'importe quelle taille et produit une empreinte numérique (hash) de 256 bits. Elle fait partie de la famille des algorithmes SHA-2 développée par la National Security Agency (NSA) et publiée par le National Institute of Standards and Technology (NIST) des États-Unis. Elle est utilisée dans les protocoles de chiffrement tels que TLS (Transport Layer Security) et SSL (Secure Sockets Layer) pour garantir la confidentialité et l'intégrité des données échangées entre un serveur et un client.

SHA-256 utilise un processus itératif pour transformer l'entrée de données en une sortie de 256 bits. Les étapes du processus incluent l'ajout de bits de remplis-

18. Livre Blanc du projet Dem-Attest-ULille
19. Attestation numérique de diplôme de Maxence Defraiteur

sage, le découpage en blocs, le traitement de chaque bloc en utilisant une série de fonctions de hachage, la concaténation des résultats et la production de la valeur finale de hachage. La sécurité de SHA-256 repose sur la difficulté de trouver deux entrées différentes qui produisent la même sortie de hachage (collision), ainsi que sur la résistance aux attaques par force brute et aux attaques par compromission partielle.

La longueur de 256 bits a été choisie pour fournir une sécurité suffisante contre les attaques cryptographiques connues à ce jour. Une sortie de hachage plus longue est généralement considérée comme plus sûre car elle augmente la probabilité que des entrées différentes produisent des sorties de hachage différentes.

SHA-256 est résistante à l'antécédent. Elle est conçue pour être résistante aux attaques par force brute, ce qui signifie qu'il est très difficile de trouver une entrée qui produit une sortie de hachage donnée sans passer par une recherche exhaustive.

Le fait que SHA-256 produise une sortie de 256 bits signifie qu'il y a 2^{256} (soit environ 10^{77}) résultats possibles. Cela rend pratiquement impossible pour un attaquant (en temps polynomial) de trouver deux entrées différentes qui produisent la même sortie de hachage (collision). En d'autres termes, la probabilité de collision est extrêmement faible, ce qui garantit une sécurité suffisante pour de nombreuses applications. Comme il y a 2^{256} combinaisons possibles pour les hachés crées, cela signifie que même une modification mineure dans l'entrée produira très probablement un haché différent.

4.2 Exemple d'exécution

```
import hashlib

# Créer un objet SHA-256
hash_object = hashlib.sha256()

# Ajouter la chaîne de caractères "pomme" à l'objet
   SHA-256
hash_object.update(b"pomme")

# Obtenir le hachage SHA-256 sous forme de chaîne de
   caractères binaire (bytes)
hash_value_bytes = hash_object.digest()

# Convertir chaque byte en une chaîne de 8 bits (un
   octet)
hash_value_bits = ''.join(format(byte, '08b') for byte
   in hash_value_bytes)

# Convertir le hachage binaire en hexadécimal pour une
   meilleure lisibilité
hash_value_hex = hash_value_bytes.hex()

# Afficher le hachage SHA-256 en bits
print("Le hachage en bits de SHA-256 de la chaîne
   'pomme' est :", hash_value_bits)

# Afficher le hachage SHA-256 en hexadécimal
print("Le hachage en hexadécimal SHA-256 de la chaîne
   'pomme' est :", hash_value_hex)
```

./programmes/exemple.sha.256.py

[20]

20. Documentation python de la fonction hashlib

Les résultats de l'exécution de ce code sont des chaînes de caractères :

— 10010001011010011011111100111110
01010000000111111110101000011001
011000010 10011001010110011010110
11010110010001101 011010100001011
01100011101010101000001000101011
11000010001101100000101001001101
10110000011010101110111001001100
11010101000001001100101101001111 (en bits : 256 caractères)

— 9169bf3e501fea19614cacd6d646b50b63aa822bc2360a4db06aee4cd504cb4f (en hexadécimal : 64 caractères pour faciliter la lecture)

Nous pouvons clarifier le lien entre les 64 caractères hexadécimaux et les 256 bits. La fonction de hachage SHA-256 produit une empreinte numérique de 256 bits, soit une séquence de 32 octets (1 octet = 8 bits). Chaque octet peut prendre une valeur hexadécimale de 00 à ff, ce qui donne une représentation hexadécimale de l'empreinte numérique de 64 caractères (32 x 2 = 64).

Le site internet sha256algorithm permet de visualiser en partie les calculs effectués par l'algorithme de la FDH SHA-256. On peut vérifier et l'exécuter avec le caractère « pomme ». Nous trouvons le même hash que la sortie du code compilé en Python précédemment.

Le processus de calcul de l'empreinte numérique peut être décrit en plusieurs étapes :

— **Prétraitement** : la chaîne de caractères d'entrée est transformée en une séquence de bits de longueur multiple de 512 bits.

— **Initialisation** : une série de constantes prédéfinies (appelées vecteurs d'initialisation) et un tableau de constantes de mots clés (appelés constantes de tour) sont définis.

— **Boucle de compression** : la chaîne de bits d'entrée est découpée en blocs de 512 bits et chaque bloc est traité par une série de transformations cryptographiques.

— **Finalisation** : une fois que tous les blocs ont été traités, une dernière transformation cryptographique est appliquée pour produire l'empreinte numérique finale.

II Partie expérimentale

5 Contexte de stage

Cette période de stage filée se déroule au collège Théodore Monod à Lesquin du mois de novembre 2023 à juillet 2024. Ce collège a été créé en 1983 et comptait en 2022 : 504 élèves, 266 garçons et 238 filles. C'est un collège public ne proposant pas d'internat. Les sections européennes en anglais, espagnol et allemand sont proposées dans ce collège. Le collège Théodore Monod est situé dans le centre-ville de Lesquin. C'est un établissement inscrit dans une agglomération moyennement urbanisée (7912 habitants pour 8 km^2 de superficie). Ce secteur géographique recense une part importante de milieux sociaux favorisés. [21] Dans la commune de Lesquin, la médiane du niveau de vie est de 23 044 euros et le taux de pauvreté est à 7 % . Pour un nombre de 5143 personnes sur la tranche de 15 à 64 ans, il y a 78 % de personnes actives et 8,2 % de personnes au chômage. [22] Entre 2017 et 2022, le taux de réussite du collège entre les 91 % et 97 % pour l'obtention du brevet. En 2022, on comptabilise un taux de réussite à 99 % . Le taux d'absentéisme est très faible dans ce collège. Durant ce stage, j'ai principalement suivi les cours de ma maîtresse de stage en Mathématiques pour les niveaux d'élèves de 6ème et 4ème. J'ai pu oberserver des séances en salle informatique :

- utilisation de Geogebra avec des 6ème (construction de droites perpendiculaires et parallèles)
- tableur excel avec des 4ème (programme de calcul, introduction douce à la notion de fonction)
- programme scratch avec des 4ème (programme de calcul)
- utilisation du lecteur PDF avec des 4ème (résolution d'énigmes de Noël : jeux logiques, systèmes d'équation du premier degré)
- séance PIX avec des 5ème (avec une autre professeure de Mathématiques)

Cette année, j'ai pu observer trois classes de 4ème. De manière générale, j'ai pu remarquer que le niveau des élèves de 4ème 2 et 3 était plus homogènement faible. Pour les élèves de 4 ème 5, j'ai trouvé que le niveau était plus hétérogène

21. https://annuaire-education.fr/etablissement/lesquin/college-theodore-monod/0593991T.html
22. Statistiques INSEE de la commune de Lesquin, en 2023

avec des profils très variés en terme de motivation de travail. En accord avec ma tutrice de stage, j'ai décidé d'effectuer mes deux activités avec les élèves de 4ème 5.

6 Tour d'horizon des manuels et des ressources en ligne

J'ai trouvé très peu de ressources, exercices dans les manuels de mathématiques de collège en rapport avec le **système binaire**. (Un seul exercice dans le manuel sésamaths de niveau 6ème, très peu contextualisé. [23]) Par contre, j'ai trouvé des éléments dans les manuels de technologies. En effet, le système binaire est désormais au programme au cycle 3 en Technologie. De même, on trouve facilement des éléments autour du système binaire dans les manuels de Sciences Numérique et Technologie dès la Seconde, Enseignement scientifique en classe de Première car le système binaire est un attendu explicite dans ces programmes. Par ailleurs, j'ai aussi trouvé des exercices de niveau collège sur des sites académiques dans le contexte de séance PIX. [24] et des ateliers proposées par l'IREM de l'université de Lille lors de journées académiques. [25] Enfin, j'ai trouvé une ressource sur le site de l'APMEP abordant les puissances de 2, le système binaire sous forme de cartes. [26] L'intitulé de l'activité est appelée *Tour de magie*. Deux concepts me déplaisent dans cette activité. Premièrement, mêler la notion de magie avec les mathématiques a un attrait *ludique* à l'activité mais cela ne permet pas de démysthifier le fait que les connaissances mathématiques n'ont rien de magiques, ne tombent pas de nul part et s'enchaînent logiquement. Selon Platon, la *mathesis* est la science qui s'enseigne par excellence, c'est un apprentissage dans l'enseignement qui n'a rien de dramatique, il est serein, on va du vrai au vrai. Deuxièmement, je trouve que la visualisation des puissances de 2 sur les cartes est un frein pour concevoir la notion abstraite de puissance. (surtout pour les *grandes* puissances)

Etant donné que les séances PIX sont encadrées par les professeurs de sciences, dont le professeur de Mathématiques. Nous avons pour mission de former au mieux aux outils informatiques, développer les compétences PIX des élèves et avoir une vision interdisciplinaire avec le programme de Technologie peut être un élément moteur pour donner du sens aux enseignements mathématiques.

23. https://manuel.sesamath.net/numerique/diapo.php?atome=60157&ordre=1
24. https://drne.region-academique-bourgogne-franche-comte.fr/wp-content/uploads/2020/04/Fiche-Pix-3.4-Mathématiques-bis.pdf
25. https://irem.univ-lille.fr/~site/spip.php?article409
26. https://www.apmep.fr/IMG/pdf/Tour_de_magie.pdf

7 Analyse a priori

7.1 Activité 1

7.1.1 Objectifs de l'activité

J'ai pu observer plus de cinq séances en salle informatique. A ma grande stupéfaction, j'ai été étonné des difficultés d'ordre générale rencontrées face à l'utilisation d'un ordinateur. Le niveau de capacité des outils numériques est encore plus hétérogène que la répartition des connaissances mathématiques chez les élèves. Mon sujet de mémoire portant sur les fonctions de hachage, je n'ai pas pu exploiter la possibilité de parler de fonctions, étant donné que c'est une notion étudiée en fin de cycle 4. Par contre, la notion de système de numération est introduite dès le cycle 1, formalisée et approfondie au cycle 3. Comme les fonctions de hachage prennent en paramètre des textes convertis en binaire, j'ai décidé de concevoir mes activités autour des systèmes de numération décimal et binaire. Afin de consolider les compétences informatiques des élèves, les objectifs de mes activités sont transversales. D'une part, les élèves appréhenderont une des connaissances fondamentales aux systèmes informatiques : **le binaire**. D'autre part, j'ai conçu mon support d'activité en faisant en sorte que la progression soit la moins linéaire possible : en ayant créé en quelque sorte une version simplifiée d'un terminal [27] (simulé sur une page internet, codé en javascript, css et html).

Du point de vue mathématique, cette activité a pour principal objectif de jongler entre base décimale et base binaire. Puis, de consolider majoritairement les connaissances mobilisables sur les puissances de nombres, puissances de dix et leurs préfixes, et enfin grandeur quotient.

Sur chacune des séances informatiques que j'ai pu observer, la séance ayant le mieux fonctionner sur le plan pédagogique a été celle des énigmes de Noël. C'est pourquoi j'ai décidé de créer mes activités autour de l'univers de la trilogie de film Matrix. Mon objectif est de plonger les élèves dans l'univers de Matrix (qu'ils ne connaissent probablement pas, date de sortie du premier film : 23 juin 1999 [28]). Ils incarneront un membre de l'équipage du vaisseau et devront résoudre un maximum de défis proposés par le capitaine de bord Morpheus.

27. outil incontournable quand on programme en informatique
28. https://fr.wikipedia.org/wiki/Matrix_(film)

7.1.2 Prérequis

Les élèves auront besoin de maîtriser l'écriture des nombres en base décimale (maîtriser la base 10, la valeur de chaque chiffre en fonction de sa position dans un nombre), effectuer des divisions euclidiennes, placer des nombres dans des tableaux de conversions, calculer avec les puissances d'un nombre (dont la puissance de 10 et de 2), connaître les techniques opératoires élémentaires en arithmétique. (addition et multiplication)

7.1.3 Liens avec les programmes et documents de recherche

Sur le plan mathématique, dans le domaine Nombres et calculs, à partir de la 4ème, les puissances de 10 et les puissances de base quelconque doivent être abordées.[29] En rapport avec le système de numération, *les élèves consolident le sens des nombres et confortent la maîtrise des procédures de calcul. [...] Les puissances sont introduites pour faciliter l'évaluation d'ordres de grandeurs (notamment avec d'autres disciplines).*[30] Comme j'aborde les tableaux de conversions avec les puissances et les préfixes, il était attendu que les élèves connaissent les préfixes de nano à giga et maîtrisent la notion de grandeur quotient.

Sur le plan numérique, on peut lire dans le Bulletin Officiel de 2020 du cycle 4 que *le monde contemporain a introduit à l'école les outils numériques qui donnent accès à une information proliférante dont le traitement constitue une compétence majeure. Le domaine 2 vise un usage éclairé de ces outils, à des fins de connaissances et pas seulement d'information, pour former des utilisateurs conscients de leurs potentialités mais aussi des risques qu'ils peuvent comporter et des responsabilités des utilisateurs.*[31] En vue de former des citoyens éclairés, la manipulation raisonnée des outils numériques me semble essentielle pour former au mieux les élèves au monde numérique. Dans cette même idée, on retrouve à la page suivante que *l'enseignement de l'informatique, dispensé en mathématiques et en technologie, permet d'approfondir l'usage des outils numériques et d'apprendre à progresser par essais et erreurs.*[32] Par ailleurs, *l'utilisation d'outils comme le tableur, la calculatrice, un logiciel de géométrie dynamique ou de programmation permet de gérer des données réelles ou expérimentales, de faire des représentations et des simulations, de programmer des objets techniques et d'inscrire l'activité mathématique dans les domaines 4 et 5 du socle.*[33]

29. page 4 des repères annuels de progression de cycle 4
30. page 131 du B.O. n°31
31. page 6 du B.O. n°31
32. page 7 du B.O. n°31
33. page 129 du B.O. n°31

7.1.4 Difficultés attendues par les élèves

La maîtrise des puissances de 10 puis des puissances de 2 freinera conceptuellement certains élèves. Déjà, pour les puissances de 10, beaucoup d'élèves de 3ème ont des difficultés à placer le bon nombre de zéros, en fonction de la notation de la puissance, alors pour les puissances de 2, comme ce sont des chiffres différents que l'on ajoute pour chaque puissance, c'est une difficulté supplémentaire. Par ailleurs, d'après Béatrice Drot-Delange, les élèves font beaucoup d'erreurs sur le système de numération positionnel. [34] En ce qui concerne le système binaire, les élèves peuvent confondre les différentes valeurs de poids des bits selon leur position. Il est indispensable d'écrire dans le bon ordre les valeurs des restes des divisions euclidiennes. La lecture oralisée des nombres binaires peut poser problèmes. Il ne faut pas lire les nombres binaires comme les nombres binaires. (Exemple $1001_{(2)}$ n'est pas mille-un mais un zéro zéro un)

7.2 Activité 2

7.2.1 Objectifs de l'activité

L'objectif est d'appréhender un système de **chiffrement symétrique**, différent des systèmes classiques du type chiffrement de Vigenère ou de César par exemple. J'ai choisi d'effectuer un chiffrement basé sur l'utilisation de l'opérateur `XOR` ($\oplus$). En générant une clé aléatoire grâce à l'opérateur de la même taille (en binaire). [35] Un autre point positif de ce chiffrement est qu'il est robuste à certaines attaques comme les attaques par l'étude fréquentielle des lettres. [36] Ce système de chiffrement est robuste à condition que les bits soient réellement générés aléatoirement. La clé doit être changée après chaque utilisation, on dit que c'est une clé jetable. La taille de la clé doit être de la même taille que le message chiffré. Le `XOR` est un des opérateurs utilisés par les fonctions de hachage pour renvoyer un hash.

Par intérêt pédagogique, je n'ai pas choisi d'encoder les lettres de l'alphabet avec un système d'encodage commun comme ASCII [37] ou UTF-8 [38] par exemple. Les lettres minuscules sont indexées par leur position dans l'alphabet de 0 à 25. J'ai ajouté aussi quelques symboles de ponctuations et les lettres accentuées : à,é ;

34. Enseigner l'informatique débranchée : analyse didactique d'activités, Béatrice Drot-Delange
35. Je suis conscient que la génération d'un véritable aléatoire sur un ordinateur est très difficile à obtenir. Dans le cadre de cette activité, je me contenterai de la fonction `random` de Scratch.
36. Ce type de chiffrement est utilisé pour établir un système de communication d'urgence sécurisé entre les dirigeants de pays. Il était utilisé notamment pendant la Guerre Froide avec le téléphone rouge.
37. American Standard Code for Information Interchange : Standard de communication entre systèmes informatiques
38. Unicode Transformation Format - 8bits

afin d'arriver à un nombre de caractères final à 32. (Cela permet d'encoder les caractères sur seulement 5 bits)

Les élèves apprendront à convertir à la main des lettres en binaire. Ils apprendront l'utilisation de l'opérateur `XOR` sur des exemples simples, à nouveau à la main. Ils devront placer dans le bon ordre des fonctions préprogrammées pour coder et décoder un mot. L'objectif final est de décrypter le message du capitaine avec une clé donnée. En question bonus, j'ai ajouté la formulation dans le bon ordre d'une fonction scratch permettant de convertir un nombre décimal en nombre binaire. Les élèves devront comprendre les boucles, affectations de variables, demande d'affectation de variables et le principe même du chiffrement pour réagancer les instructions dans le bon ordre.

7.2.2 Prérequis

Les élèves doivent avoir quelques notions d'algorithme et de programmation, avoir compris la conversion d'un nombre décimale vers un nombre binaire et réciproquement grâce à la première activité, avoir des notions de variable informatiques, connaître le déclenchement d'une action par un événement, connaître des séquences d'instructions, boucles et instructions conditionnelles. La notion de puissance, algorithme de la division euclidienne, décomposition d'un nombre binaire et décimale leur sera indispensable.

7.2.3 Liens avec les programmes et documents de recherche

Selon Béatrice Drot-Delange, *l'objectif est de faire découvrir des concepts et des méthodes spécifiques à la science informatique, de montrer en quoi elle se distingue des technologies de l'information et de la communication.* Dans ce document, on apprend que des activités en lien avec l'écriture binaire peuvent être proposées très tôt : dès l'âge de 7 ans. Le but étant de satisfaire, motiver, susciter un intérêt pour l'informatique. Laurent et Graffre montrent que les élèves, *après les séances, sont plus intéressés par l'informatique, se sentent significativement plus confiants en mathématiques, mais pas significativement plus intéressés par les mathématiques.* L'enjeu de mon activité est d'exacerber la motivation de faire des mathématiques, qui dans ce contexte : sont au service de la programmation, d'outils numériques.

Travailler sur le système binaire n'est pas hasard complet. En effet, *le système binaire semble être le B-A-BA de toute initiation à l'informatique. [...] Parmi les savoirs à enseigner dans un cours introductif en informatique figure en bonne place la représentation binaire de l'information. Dans les standards proposés par l'ACM, la représentation binaire doit être enseignée très tôt : dès le grade K2. (école primaire)*

Dans les repères annuels de progressions de cycle 4, dans la catégorie algorithmique et programmation, on peut lire que les élèves doivent être capables d'écrire, mettre au point et exécuter un programme. Les élèves doivent poursuivre l'initiation à la programmation, en créant des programmes, développpant des méthodes de programmation et revisitant les notions de variables. L'appréhension de la programmation au collège est réalisée via le logiciel Scratch. Il existe trois niveaux d'apprentissage :

1. mettre en ordre, compléter des blocs fournis par le professeur
2. écrire un programme en utilisant les séquences d'instructions, utilisation d'une variable, exécution d'un programme par des événements extérieurs
3. utilisation des boucles *répéter ... fois*, *répéter jusqu'à ...*, instructions conditionnelles, décomposition d'un problème complexe en sous-problème [39]

J'ai décidé de placé le niveau de mon activité entre le niveau 1 et le niveau 2. (Niveau 1 pour le sprite [40] chiffrement et niveau 2 pour le sprite décimale.binaire) A nouveau, la notion de puissance des nombres est étudiée dans le cadre de cette activité.

En 1989 à Paris, durant le premier colloque francophone didactique informatique, les chercheurs ont insisté sur le fait qu'il est *indispensable* de comprendre qu'à divers endroits de l'ordinateur (mémoire, écran, imprimante) l'information a une structure binaire. [41]

7.2.4 Difficultés attendues par les élèves

Au cycle 3, en abordant la structuration des nombres entiers : *c'est un saut majeur qui conduit l'élève à passer d'une suite de nombres à la compréhension plus fine de ce qu'est la numération décimale.* La compréhension du système binaire est un saut didactique supplémentaire. De même, pour être capable de l'implémenter algorithmiquement correctement, il faut comprendre la conversion de système décimale vers le système binaire. La notion de boucle qui s'arrête quand une condition est atteinte est un pas à atteindre. (notamment pour l'implémentation des divisions euclidiennes) L'utilisation des TICEs peut être un frein pour des élèves, comme l'utilisation de l'outil Scratch. Selon certaines sources, les TICEs se déclinent sous trois fonctions : médium (instrument d'acquisition de connaissances), matière enseignée (informatique comme discipline) et outil de production. La difficulté de la maîtrise des TICEs est d'être suffisament à l'aise dans ces trois champs. [42]

39. page 13 des repères annuels de progressions de cycle 4
40. Sur Scratch, un sprite (anciennement appelé ń lutin ż) est un objet effectuant des actions à l'écran (la scène). (https://fr.scratch-wiki.info/wiki/Sprite)
41. Enseigner l'informatique débranchée : analyse didactique d'activités, Béatrice Drot-Delange
42. Le nombre au cycle 3 apprentissages numériques, Eduscol, page 116

Les fonctions que j'ai implémenté dans Scratch n'ont pas vocation à être lues, elles seront « cachées » pour les élèves. Néanmoins, elles seront toujours présentes sur la planche de travail de Scratch. Si des élèves voient l'étendue des fonctions, cela peut les rendre confus. Ces fonctions n'ont pas vocation à être lues, ni comprises par les élèves. J'ai volontairement limiter un maximum la création de variables pour ne pas surcharger la liste des variables présente sur la fenêtre gauche de Scratch. La longueur de la clé du message à décrypter du capitaine est très longue (155 bits), cela peut freiner la compréhension du déchiffrement final chez les élèves.

7.2.5 Difficultés rencontrées par moi-même dans Scratch

Pour implémenter les fonctions utiles pour chiffrer et déchiffrer les messages. J'ai rencontré plusieurs problèmes liés à la conception même du logiciel Scratch. En effet, il n'existe pas de `return` au sein d'une fonction. Il y a un manque considérable de liberté sur les variables locales au sein d'une fonction. On ne peut pas placer une liste en paramètre d'une fonction. On ne peut pas facilement dupliquer des listes. Il n'y a pas de moyen simple pour implémenter un *dictionnaire*.

Cela m'a rendu assez dubitatif sur l'efficacité et la difficulté plus tard pour élèves de faire la transition du langage par Bloc de Scratch vers le langage Python.

8 Annexes

Accéder au support internet de mes activités, hébérgées sur GitHub Pages [43] [44] : https://maxencedefraiteur.github.io/MATHRIX/

8.1 Activité 1

8.1.1 Introduction

Bienvenue à toi, tu es dans la Mathrix : une dimension parallèle au réel dans l'univers informatique. Des pirates informatiques tentent de compromettre le monde en piratant les systèmes informatiques. Tu as été choisi par le capitaine Morpheus pour sauver la mathrix. A bord du vaisseau, tu devras prouver tes compétences faces aux attaques ennemies. Le capitaine et son équipage vont te tester pour évaluer tes connaissances en terme de communication dans le langage informatique.

Objectif : comprendre les systèmes de communication des langages informatiques, en relevant les différents défis du capitaine Morpheus

Le capitaine tient un journal de bord. Sa lecture journalière te sera profitable par la suite.

8.2 Journal de bord du capitaine

8.2.1 Jour 1 : Nombre décimal et nombre binaire

Dans la mathrix, j'ai appris à communiquer avec les autres résistants en leur envoyant des messages interprétés numériquement. Pour cela, les mots et les nombres sont écrits en système binaire. Les hommes utilisent principalement le système de numération en base 10, basé sur les 10 symboles (chiffres de 0 à 9). C'est le système décimal. Il existe d'autres systèmes de numération, comme le système binaire, en base 2 : utilisant uniquement des 0 et des 1.
Les appareils informatiques utilisent le système binaire pour communiquer en activant ou désactivant un courant électrique dans un transistor par exemple.

Voici des nombres décimaux : 16 ;324 ;11.
Voici des nombres binaires : 10 ; 10001 ;11 ;111.

43. Attention, le rendu du design du site n'est pas conçu pour être consulté sur des appareils mobiles du type téléphone ou tablette.
44. Je tiens à remercier Louis Aleaume pour toute son aide sur l'écriture du code source en Javascript.

Pour éviter de confondre 11 entre les différentes bases, on écrit la base entre parenthèse en indice du nombre. Exemple : $11_{(10)}$, $11_{(2)}$.

Question 1) Quels sont les deux systèmes de numération évoqués par le capitaine ?

8.2.2 Jour 2 : Décomposer un nombre décimal

Un nombre décimal peut être exprimé comme la somme des produits de chiffres, chacun multiplié par une puissance de 10 correspondante. Exemple :

$$\begin{aligned} 45357_{(10)} &= 4 \times 10000 + 5 \times 1000 + 3 \times 100 + 5 \times 10 + 7 \\ &= 4 \times 10^4 + 5 \times 10^3 + 3 \times 10^2 + 5 \times 10^1 + 7 \times 10^0 \end{aligned}$$

Question 2) Additionner les trois nombres composant ta date de naissance. Décomposer le résultat sous la forme de somme des produits de chiffres, chacun multiplié par une puissance de 10 correspondante.

Pour les nombres binaires, ils doivent être exprimé comme la somme des produits de chiffres, chaucun multiplié par des puissances de 2.

8.2.3 Jour 3 : Convertir un nombre décimal en nombre binaire

Méthode : Convertir un nombre décimal en nombre binaire. Effectuer successivement des divisions euclidiennes par 2. A l'étape suivante, on divise à nouveau le quotient par 2. Le reste valant 0 ou 1 donnera la décomposition du nombre en binaire. On s'arrête quand le quotient vaut 0. Attention : Pour écrire le résultat final, il faut prendre les restes de bas en haut.

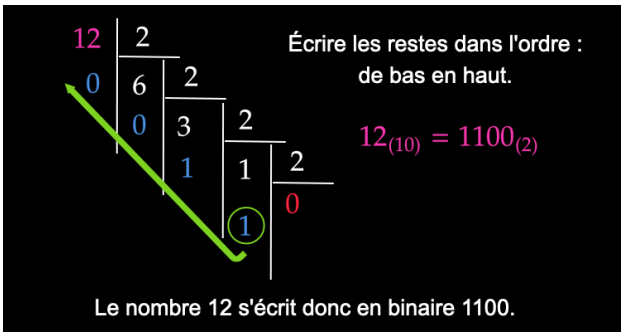


FIGURE 4 – Méthode : convertir un nombre décimal en nombre binaire

Cela signifie

$$\begin{aligned} 12 &= 1 \times 8 + 1 \times 4 + 0 \times 2 + 0 \times 1 \\ &= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 \end{aligned}$$

On peut le visualiser avec le tableau de conversion suivant :

En puissance	2^3	2^2	2^1	2^0	
En décimal	8	4	2	1	
En binaire	1	1	0	0	
Résultat	8+	4+	0+	0=	12

Question 3) Convertir les nombres décimaux $19_{(10)}$ $138_{(10)}$ en nombres binaires.

8.2.4 Jour 4 : Convertir un nombre binaire en nombre décimal

Pour convertir un nombre binaire en nombre décimal, on peut s'aider à nouveau du tableau de conversion avec les puissances de 2. Exemple : On veut convertir le nombre binaire $1001010_{(2)}$ en nombre décimal.

En puissance	2^6	2^5	2^4	2^3	2^2	2^1	2^0
En décimal	64	32	16	8	4	2	1
En binaire	1	0	0	1	0	1	0

$$\begin{aligned} 1001010_{(2)} &= 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ &= 1 \times 64 + 0 + 0 + 1 \times 8 + 0 + 1 \times 2 + 0 \\ &= 64 + 8 + 2 \\ &= 74_{(10)} \end{aligned}$$

Question 4) Le code d'authentification du vaisseau est écrit en binaire : $1100011_{(2)}$. Convertissez-le en nombre décimal.

8.2.5 Jour 5 : Affichage d'un octet

Un octet correspond à huit bits[45]. Pour afficher un octet, il faut allumer les bits correspondants. Les cercles avec des croix correspondent à des transistors (composant électronique), s'ils sont colorés : cela signifie qu'ils sont allumés sinon ils sont éteints. (Remarque : quand il y a plus de huit bits dans la représentation d'un nombre, il faut afficher un nouvel octet. L'octet est utilisé pour mesurer la capacité de stockage en mémoire ou sur un disque dur.

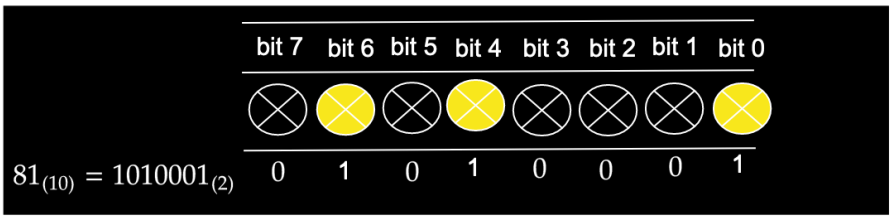


FIGURE 5 – Affichage de l'octet du nombre $81_{(10)}$

Question 5) Dessiner l'octet des nombres $35_{(10)}$ et $156_{(10)}$ en reproduisant le schéma avec les transistors allumés. (Il faut penser à convertir les nombres en système binaire d'abord).

8.2.6 Jour 6 : Grandeur quotient

Une grandeur quotient est le quotient de deux grandeurs.
Exemples : La vitesse et le débit sont des grandeurs quotients car elles s'obtiennent en faisant une division.

$$\text{Vitesse moyenne} = \frac{\text{distance}}{\text{durée}}$$

$$\text{Débit moyen} = \frac{\text{volume}}{\text{durée}} = \frac{\text{quantité d'information}}{\text{durée}}$$

Question 6) Des pirates informatiques veulent compromettre les systèmes du vaisseau. En sachant, qu'ils envoient un virus de 50 Mo en 1min40s.
Quelle est la vitesse de téléchargement de ce virus en Mo/s ?

Question 6 bis) Un membre de l'équipage affirme que l'on peut contrer cette attaque en envoyant un fichier à une vitesse de 1,5 Mo/s en moins de 35s.
A-t-il raison ? Justifiez.

45. BInary digiTS

8.3 Pense-bête du capitaine :

8.3.1 PB1 : Base

Une base, dans un système de numération positionnel, est le nombre de symboles (de chiffres) qui sont utilisés pour représenter les nombres.

8.3.2 PB2 : Puissances d'exposants positifs

Définition : Soit a un nombre relatif et n un entier avec $n \geq 2$. Le produit $\underbrace{a \times a \times \cdots \times a}_{n facteurs}$ de n facteurs égaux à a est une puissance de a et est noté a^n.

Exemples :

— $2^5 = 2 \times 2 \times 2 \times 2 \times 2$

— $(-4)^3 = (-4) \times (-4) \times (-4)$

Cas particuliers :	Exemples :
— $a^0 = 1$ (pour tout $a \neq 0$	— $5^0 = 1$
— $a^1 = a$	— $6^1 = 6$

8.3.3 PB3 : Puissances de dix et préfixes

Les préfixes sont utilisés pour simplifier le nom et l'écriture de mesures exprimées en puissances de dix.

Préfixe	giga	méga	kilo	unité	mili	micro	nano
Symbole	G	M	k		m	μ	n
10^n	10^9	10^6	10^3	$10^0 = 1$	10^{-3}	10^{-6}	10^{-9}

8.3.4 PB4 : Tableau de correspondance entre puissance de 10 et mesures en octets.

Nom	Symbole	Valeur
kilooctet	ko	10^3
mégaoctet	Mo	10^6
gigaoctet	Go	10^9
téraoctet	To	10^{12}

TABLE 1 – Tableau de correspondance entre puissance de 10 et mesures en octets.

8.3.5 PBC5 : Définition d'un nombre décimal

Définition : Un nombre décimal est un nombre qui s'écrit avec nombre fini de chiffres après la virgule en écriture décimale positionnelle. Les nombres décimaux sont les quotients d'entiers par les puissances de 10.

8.4 Manuel d'instructions inhérentes dans le vaisseau

Liste des commandes et explications :

— Introduction à l'activité : intro
— Consulter une page du journal de bord du capitaine : de jour1 à jour6.
— Consulter un pense-bête du capitaine : de pb1 à pb5
— Cacher une image : cacher
— Donner son prénom : prenom
— Arrêter la pluie de caractère : arreter
— Reprendre la pluie de caractère : reprendre
— Atténuer la pluie de caractère : attenuer couleur

8.5 Fiche élève : activité 1

Journal de bord du capitaine

- `jour1` : Système décimal et système binaire
- `jour2` : Décomposer un nombre décimal
- `jour3` : Convertir un nombre décimal en nombre binaire
- `jour4` : Convertir un nombre binaire en nombre décimal
- `jour5` : Afficher un octet
- `jour6` : Grandeur quotient

Manuel d'instruction

Liste des commandes fonctionnelles dans le vaisseau

- `intro` : explications introductives
- `prenom` : renseigner son prénom
- `jour1` : afficher la première page du journal de bord du capitaine
- `pb1` : afficher le premier pense-bête du capitaine
- `aide0` : afficher une aide
- `cacher` : cacher l'image affichée
- `arreter` : arrêter la pluie de caractères
- `reprendre` : reprendre la pluie de caractères

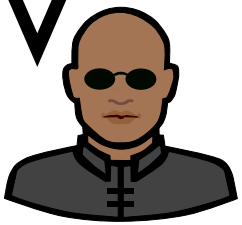
MATRIX

Conseils du capitaine

→ Tu peux consulter mon journal de bord dans le désordre sauf pour le `jour1` et `jour5`.

→ Commence par les commandes suivantes :

- `intro`
- `prenom`
- `jour1`



Pense-bêtes du capitaine

- `pb1` : Définition d'une base
- `pb2` : Puissances d'exposants positifs
- `pb3` : Puissances de dix et préfixes
- `pb4` : Tableau de correspondance entre puissance de 10 et mesures en octets
- `pb5` : Définition d'un nombre décimal
- `pb6` : Tableau de conversion de 12

MATHRIX

Nom : ..... **Prénom :** .....

CARNET DE BORD

Objectif : Répondre aux défis journaliers du capitaine.

(Chaque numéro de question correspond à un numéro de jour)

Q1) Les deux systèmes de numération évoqués par le capitaine sont :

Q2) Date de naissance (jj/mm/aaaa) :
Additionner jj + mm + aaaa = …
Décomposer le résultat précédent

Q3) Conversion de $19_{(10)}$:

Conversion de $138_{(10)}$

Q4) Conversion de $1100011_{(2)}$

Q5) Affichage de l'octet de $35_{(10)}$:

Affichage de l'octet de $156_{(10)}$:

Q6)

Q6 bis)

8.6 Activité 2

Accéder au support internet de mes activités, hébérgées sur GitHub Pages [46] : https://maxencedefraiteur.github.io/MATHRIX/

8.6.1 Introduction

Bienvenue à toi, tu es dans la Mathrix : une dimension parallèle au réel dans l'univers informatique. Le capitaine a disparu. Les membres de l'équipage ont réussi à établir une liaison avec le capitaine dans la mathrix. Malheureusement, les mots ont été codés. Ton objectif est d'apprendre le système de chiffrement utilisé par le capitaine pour décrypter le message qu'il a envoyé.

8.6.2 Jour 0 : Convertir un nombre entier en nombre décimal

Méthode : Convertir un nombre décimal en nombre binaire. Effectuer successivement des divisions euclidiennes par 2. A l'étape suivante, on divise à nouveau le quotient par 2. Le reste valant 0 ou 1 donnera la décomposition du nombre en binaire. On s'arrête quand le quotient vaut 1. Attention : Pour écrire le résultat final, il faut prendre les restes de bas en haut.

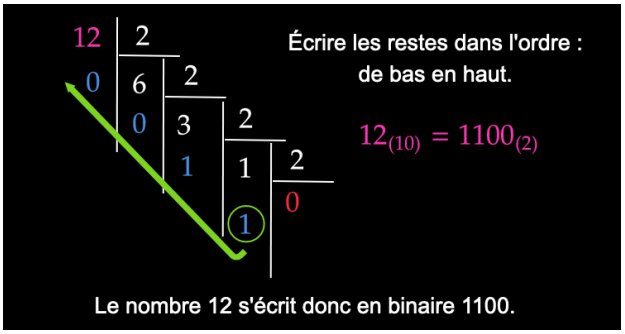


FIGURE 6 – Méthode : convertir un nombre décimal en nombre binaire

Cela signifie

$$\begin{aligned} 12 &= 1 \times 8 + 1 \times 4 + 0 \times 2 + 0 \times 1 \\ &= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 \end{aligned}$$

Question 0) En appliquant la méthode du capitaine, convertir le nombre décimal $9_{(10)}$ en binaire.

46. Attention, le rendu du design du site n'est pas conçu pour être consulté sur des appareils mobiles du type téléphone ou tablette.

8.6.3 Jour 1 : Encodage des lettres

Pour chiffrer les caractères, le capitaine a encodé chaque lettre minuscule, deux lettres accentuées et quelques symboles de ponctuation. La position dans l'alphabet correspondant à un nombre décimal a été convertie en nombre binaire.

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12
n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25
?	!	.		à	é							
26	27	28	29	30	31							

a	b	c	d	e	f	g	h
00000	00001	00010	00011	00100	00101	00110	00111
i	j	k	l	m	n	o	p
01000	01001	01010	01011	01100	01101	01110	01111
q	r	s	t	u	v	w	x
10000	10001	10010	10011	10100	10101	10110	10111
y	z	?	!	.		à	é
11000	11001	11010	11011	11100	11101	11110	11111

Question 1) a) A l'aide du tableau d'encodage du capitaine, encoder le mot *math* en nombre décimal.

Question 1) b) A l'aide du tableau d'encodage du capitaine, encoder le mot *math* en nombre binaire.

8.6.4 Jour 2 : Opérateur logique XOR

On peut effectuer des opérations sur les nombres binaires comme l'addition, la soustraction, la multiplication et la division. L'opérateur XOR (ou exclusif) est un autre opérateur logique, permettant de faire des calculs sur les bits.

$\oplus$	0	1
0	0	1
1	1	0

Ce tableau à double entrée signifie :

$$0 \oplus 0 = 0$$
$$0 \oplus 1 = 1$$
$$1 \oplus 0 = 1$$
$$1 \oplus 1 = 0$$

Question 2 a) Ecrire en binaire les lettres b et h.
Question 2 b) Calculer $b \oplus h$.

8.6.5 Jour 3 : Principe du chiffrement

Une clé est générée aléatoirement et est de la même taille que le mot visible. En parcourant les lettres du mot visible, pour chiffrer le mot, on calcule :
mot $\oplus$ clé = mot chiffré

Comme c'est un système de chiffrement symétrique, on peut déchiffrer le mot chiffré avec la même clé, en calculant : mot chiffré $\oplus$ clé = mot

8.6.6 Jour 4 : Implémenter le chiffrement sur Scratch

Le capitaine avait programmé le système de chiffrement dans Scratch. Des pirates ont corrompu le fichier. Aide les membres de l'équipage à replacer les blocs dans le bon ordre.

Question 3 a) Réorganiser les instructions du bloc rose « coder mot ».
Question 3 b) Réorganiser les instructions du bloc rose « décoder mot ».

8.6.7 Mission Finale : Décrypter le message envoyé par le capitaine

Le capitaine a envoyé un message crypté avec une clé correspondante. Dans le programme Scratch, copie-colle le message et la clé du sprite capitaine vers le sprite chiffrement. Enlève le bloc rose « clé.aléa » et ajoute un bloc d'instruction orange pour ajouter la clé du capitaine.

Question Finale) Quel est le message du capitaine décrypté ?

8.6.8 Jour Bonus : Programme Scratch : convertir un nombre décimal en nombre binaire

Un membre de l'équipage a crée un programme pour convertir un nombre décimal en nombre binaire. Des pirates ont corrompu le fichier. Aide les membres de l'équipage à replacer les blocs dans le bon ordre. Les instructions se trouvent dans le sprite « Déci.Binaire ».

Question Bonus a) Réorganiser les instructions des blocs dans le bon ordre.

Question Bonus b) Convertir le nombre 456789123 en binaire à l'aide du programme Scratch.

8.7 Manuel d'instructions inhérentes dans le vaisseau

Liste des commandes et explications :

— Introduction à l'activité : intro

— Consulter une page du journal de bord du capitaine : de jour1 à jour4.

— Consulter la mission finale : mission finale

— Consulter le jour bonus : jour bonus

— Atténuer la pluie de caractères : attenuer couleur

— Arrêter la pluie de caractère : arreter

— Reprendre la pluie de caractère : reprendre

8.8 Fiche élève : activité 2

Journal de bord du capitaine

- `jour0` : Convertir un nombre décimal en nombre binaire
- `jour1` : Encodage des lettres
- `jour2` : Opérateur logique XOR
- `jour3` : Principe du chiffrement
- `jour4` : Implémenter le chiffrement sur Scratch
- `mission finale` : Décrypter le message du capitaine
- `jour bonus` : Scratch : conversion(décimale, binaire)

Manuel d'instruction

Liste des commandes fonctionnelles dans le vaisseau

- `intro` : explications introductives
- `jour0` : afficher la première page du journal de bord du capitaine
- `cacher` : cacher l'image affichée
- `arreter` : arrêter la pluie de caractères
- `reprendre` : reprendre la pluie de caractères
- `attenuer couleur:` atténuer la pluie de caractères

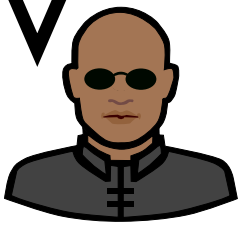
MATRIX

Conseils du capitaine

→ Tu dois consulter mon journal de bord dans l'ordre `jour1`,… `jour4`, `mission finale`, `jour bonus`.

→ Commence par les commandes suivantes :

- `intro`
- `jour1`
- `jour2` …



MATHRIX

Nom : ..... **Prénom :** .....

CARNET DE BORD

Objectif : Répondre aux défis journaliers du capitaine.

(Chaque numéro de question correspond à un numéro de jour)

Q0) Conversion de $9_{(10)}$:

Q1) a) Encodage de *math* en nombre décimal :

m	a	t	h

Q1) b) Encodage de *math* en nombre binaire :

m	a	t	h

Q2) a) Lettre *b* en binaire : Lettre *h* en binaire:

Q2) b) Calcul de : $b \oplus h =$

b					
h					
$b \oplus h$					

Q3) a) Réorganiser les instructions du bloc rose « coder mot ».
(Sprite : Chiffrement)

Q3) b) Réorganiser les instructions du bloc rose « décoder mot ».
(Sprite : Chiffrement)

Mission Finale)
(Sprite Chiffrement et Sprite Capitaine)

Message du capitaine décrypté :

Bonus a) Réorganiser les instructions des blocs dans le bon ordre.
(Sprite : Décimale.Binaire)

Bonus b) Conversion de $456789123_{(10)}$ (en utilisant le programme Scratch)

Fonctions Scratch

Sprite : Chiffrement

définir Coder Mot
mettre compteur.2 à 1
répéter longueur de mot fois
mettre ind3 à 1
répéter 32 fois
si lettre compteur.2 de mot = lettre ind3 de abcdefghijklmnopqrstuvwxyz:!. àé alors
mettre ind3s à ind3
ajouter 1 à ind3
mettre chaine binaire à élément ind3s de liste.valeur.binaire
XOR
mettre ind3 à 1
répéter 32 fois
si res xor = élément ind3 de liste.valeur.binaire alors
mettre mot chiffré à regrouper mot chiffré et lettre ind3 de abcdefghijklmnopqrstuvwxyz:!. àé
ajouter 1 à ind3
ajouter 1 à compteur.2
montrer la variable mot chiffré
cacher la variable mot déchiffré
mettre mot déchiffré à regrouper mot chiffré et mot déchiffré
montrer la variable mot
montrer la variable clef

définir Décoder Mot
Coder Mot
cacher la variable mot chiffré
montrer la variable mot déchiffré

Sprite : Chiffrement

Opérateur logique XOR (ou exclusif, en anglais : eXclusif OR)

Prend en paramètre deux éléments binaire : chainebinaire et cle.
Les éléments de chainebinaire sont parcourus et sont appelés resbina.
Les éléments de cle sont parcourus et sont appelés resbinb.

Applique l'opérateur XOR entre chaque resbina et resbinb.

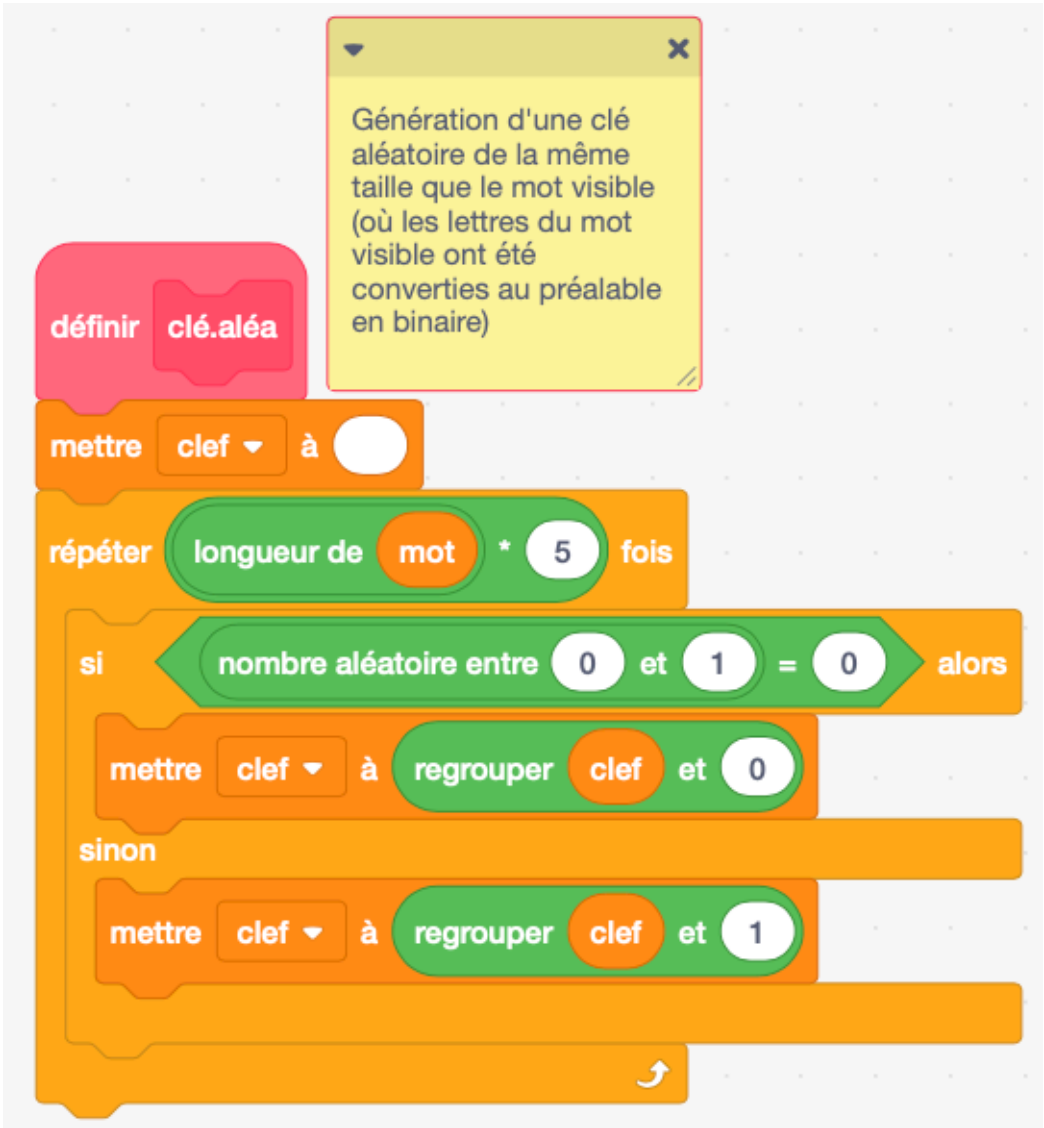
Exemple:

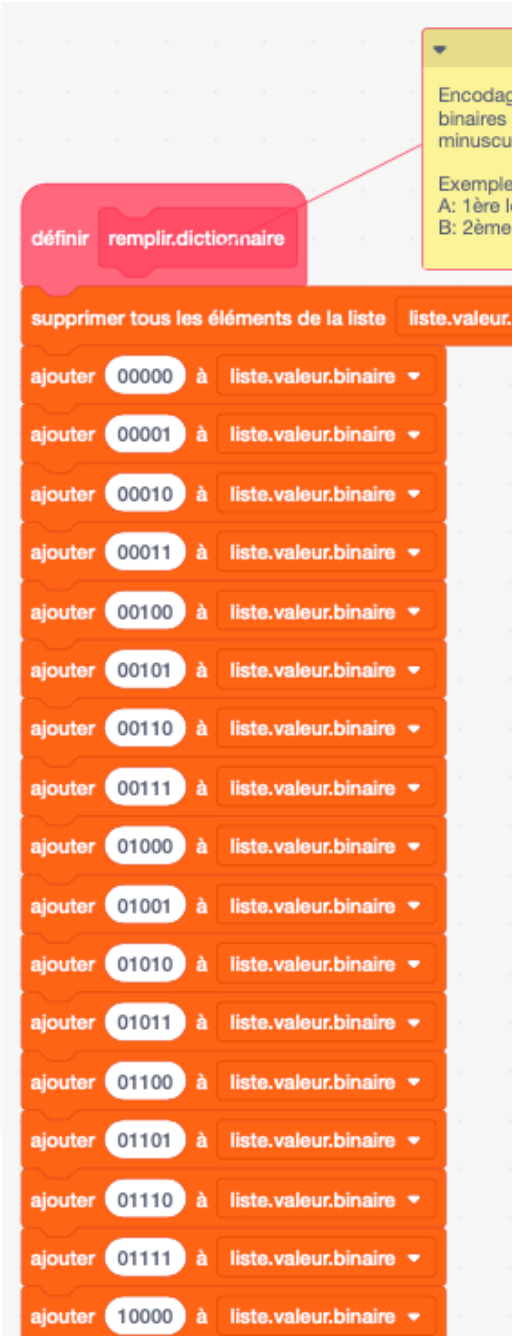
chainebinaire = 101
cle = 111

chainebinaire XOR cle = 1 XOR 1 || 0 XOR 1 || 1 XOR 1
chainebinaire XOR cle = 011

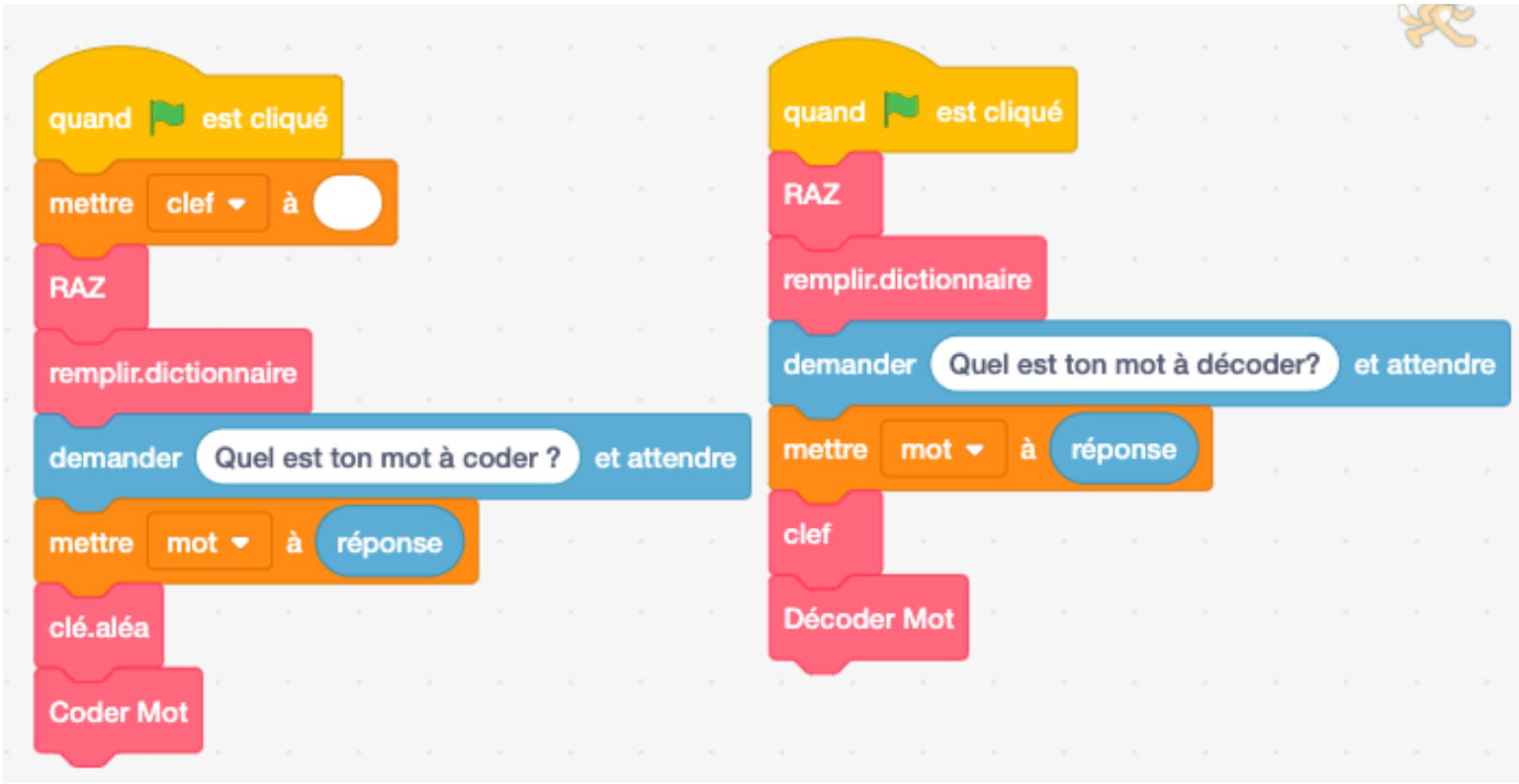
définir XOR
mettre compteur à 1
mettre i à 0
mettre res xor à
répéter longueur de chaine binaire fois
ajouter 1 à i
mettre resbina à lettre i de chaine binaire
mettre resbinb à lettre i de clef
si resbina = 0 et resbinb = 0 alors
mettre res xor à regrouper res xor et 0
sinon
si resbina = 0 et resbinb = 1 alors
mettre res xor à regrouper res xor et 1
sinon
si resbina = 1 et resbinb = 0 alors
mettre res xor à regrouper res xor et 1
sinon
mettre res xor à regrouper res xor et 0

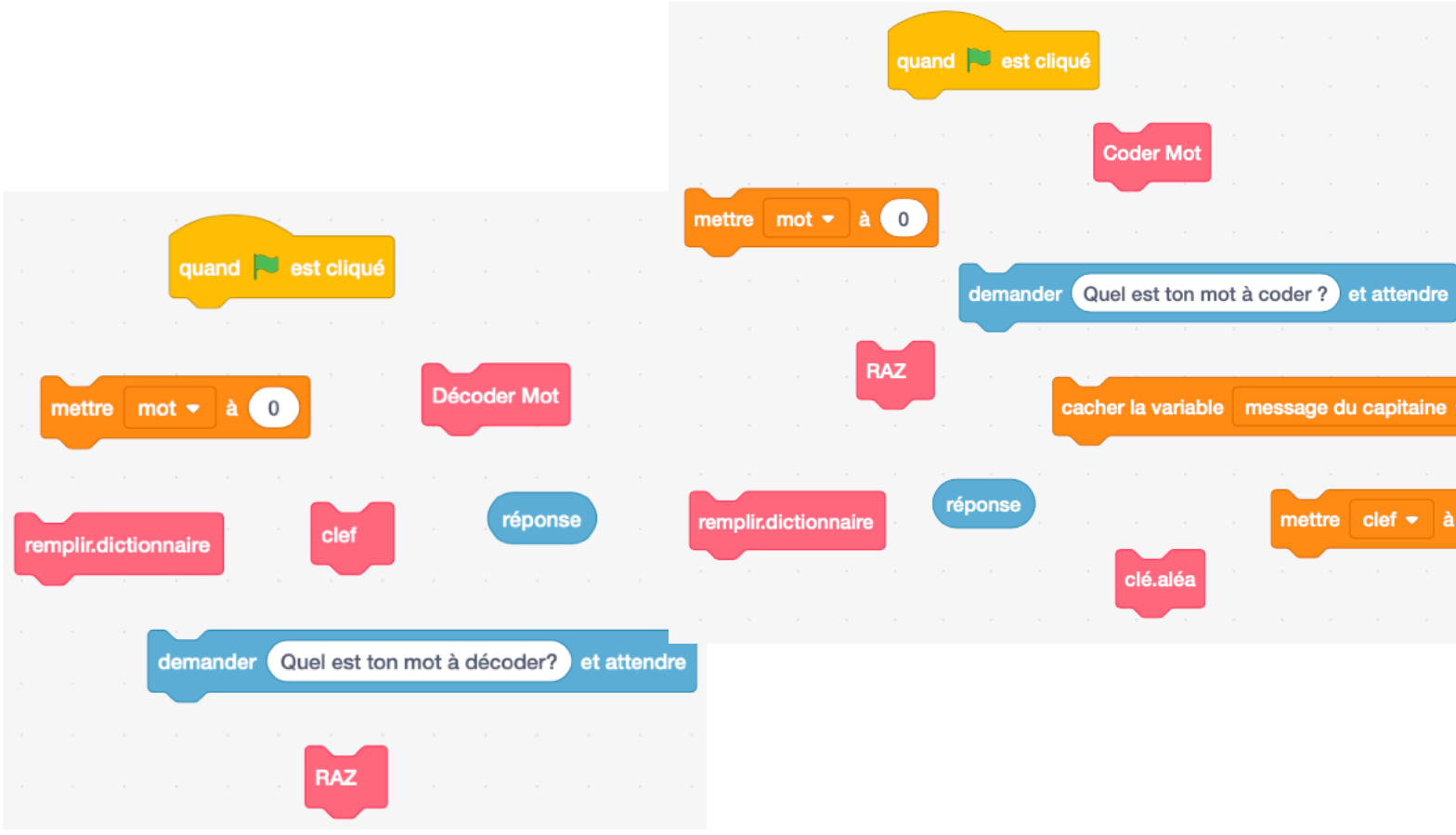
Sprite : Chiffrement



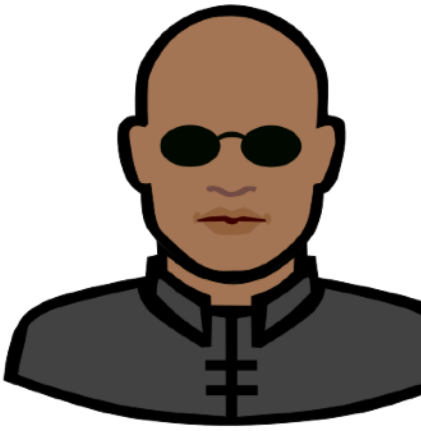


Sprite : Chiffrement

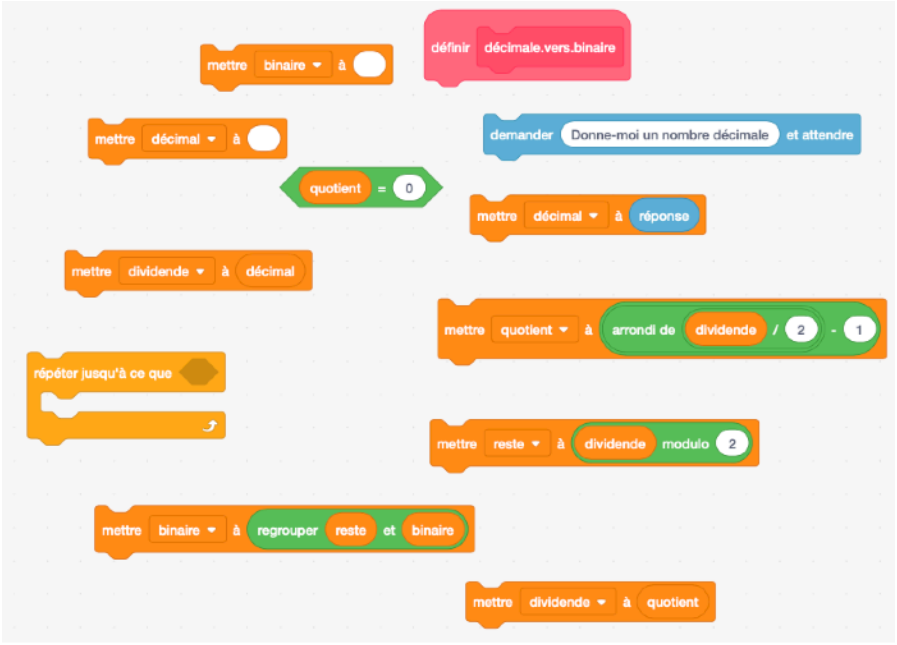




Sprite : Capitaine



Sprite : Décimale vers Binaire



définir décimale.vers.binaire
cacher
mettre binaire à
mettre décimal à
demander Donne-moi un nombre décimale et attendre
mettre décimal à réponse
mettre dividende à décimal
répéter jusqu'à ce que quotient = 0
mettre quotient à arrondi de dividende / 2 - 1
Effectue la division entière
mettre reste à dividende modulo 2
mettre dividende à quotient
mettre binaire à regrouper reste et binaire

9 Bibliographie

Références

[1] Marie-José Durand-Richard, Philippe Guillot, *Comment les mathématiques ont investi la cryptologie ?*, 15 mars 2017, Image des Maths

[2] Juan GOMEZ, *Mathématiciens, espions et pirates informatiques, Le Monde est Mathématique*, 2013

[3] Pierre-Antoine Guihéneuf, Alice Cleynen,*Le paradoxe des anniversaires*, Image des Maths CNRS,22 novembre 2022

[4] **Alfred J.Menezes, Paul C. van Oorschot et Scott A. Vanstone, *Handbook of applied criptography*, août 2001, CRC Press**

[5] Matt Parker, *The Maths Book*, publié par DK le 5 septembre 2019

[6] http://bruno.maitresdumonde.com/optinfo/ads-scie/2008.pdf, Epreuve commune de TIPE 2008, Les Fonctions de Hachage

[7] Repères annuels de progressions de cycle 4

[8] Programme du cycle 4, BO n°31 du 30 juillet 2020

[9] Béatrice Drot-Delange. Enseigner l'informatique débranchée : analyse didactique dactivités.

[10] Algorithme et programmation au cycle 4, Emmanuel Beffara, Malika More, Cécile Prouteau, Dominique Baroux-Raymond, Guillaume François-Leroux, Philippe Lac, Christophe Velut