# Sécurité des données stockées sur molécules d'ADN

Chloé Berton

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

# THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE
PAYS DE LA LOIRE – IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 648
*Sciences pour l'Ingénieur et le Numérique*
Spécialité : *Informatique*

Par
**Chloé BERTON**

## Sécurité des données stockées sur molécules d'ADN

**Thèse présentée et soutenue à IMT Atlantique, Plouzané, le 1er octobre 2024**
**Unité de recherche : Unite INSERM 1101 Latim, IMT Atlantique**
**Thèse N° : 2024IMTA0431**

### Rapporteurs avant soutenance :

| | |
|---|---|
| Pascal BARBRY | Directeur de recherche, Université Côte d'Azur |
| Maryline LAURENT | Professeure, Telecom Sud Paris |

### Composition du Jury :

| | | |
|---|---|---|
| Président : | Yannick RONDELEZ | Directeur de recherche, CNRS/ESPCI Paris |
| Examinateurs : | Emmanuelle GENIN | Directrice de recherche, Inserm Brest |
| | Maryline LAURENT | Professeure, Telecom Sud Paris |
| | Pascal BARBRY | Directeur de recherche, Université Côte d'Azur |
| Dir. de thèse : | Gouenou COATRIEUX | Professeur, IMT Atlantique Brest |
| Co-dir. de thèse : | Dominique LAVENIER | Directeur de recherche, CNRS Rennes |

### Invité :

| | |
|---|---|
| Reda BELLAFQIRA | Maître de conférences, IMT Atlantique Brest |

# Remerciements

Cette thèse a constitué pour moi une aventure de 4 ans, jour pour jour, qui m'a permis de découvrir de nombreux aspects de la recherche scientifique et de ce sujet passionnant qu'est le stockage de données dans l'ADN. Plus fondamentalement, cette thèse m'a aussi beaucoup appris sur moi-même, mes forces et mes faiblesses, et j'ai grandi avec elle. Mais j'ai aussi conscience que persévérer lors des moments difficiles n'a été possible que grâce à mes proches. Leur soutien et leur présence m'est inestimable. J'espère n'oublier personne dans ces remerciements.

Tout d'abord, cette thèse n'existerait pas sans mes deux directeurs de thèse, Gouenou Coatrieux et Dominique Lavenier, que je remercie vivement pour leur investissement et leur confiance. En particulier, je remercie Gouenou pour son encadrement au jour le jour et tous ses conseils. Je tiens aussi à remercier Reda Bellafqira pour sa pédagogie et sa patience lors de nos discussions. J'aimerais aussi remercier Sahar Haddad, qui en 2022 et 2023 a été une interlocutrice bienveillante et pédagogue pour me guider dans mon doctorat.

J'adresse également tous mes remerciements à Pascal Barbry et Maryline Laurent pour avoir accepté d'être les rapporteurs de cette longue thèse, ainsi que Yannick Rondelez d'avoir été le président de mon jury, et Emmanuelle Genin d'avoir été examinatrice lors de celui-ci.

Lors de cette thèse, je me suis formée à un domaine très éloigné du mien, la biologie. Je remercie donc Emeline Roux et Julien Leblanc pour le temps qu'ils ont accordé à mes questions. En particulier, je tiens à remercier Julien pour toute son aide et sa pédagogie lors d'observations de manipulations en laboratoire, de longues réunions de vulgarisation et d'échanges par mail, qui m'ont permis de progresser au contact d'un expert. Je tiens aussi à remercier Olivier Boulle et Dominique pour le partage de leurs expertises en bio-informatique, ainsi que les membres du Labex Cominlabs DnarXiv, notamment Elsa Dupraz et Belaid Hamoum pour tous nos échanges enrichissants, mais aussi Aref, Jacques et Yann.

Merci également à toutes les personnes avec qui j'ai pu travailler au cours des 4 dernières années : Jean et Hélène que j'ai eu le plaisir d'encadrer en stages de fin d'études, mes collègues passés Maxime, Lucie, Anass, Ambre, Raphaël, Prasanth, Mounia, Camélia, Yan, Maëlle, Théo, Tamara, ainsi que l'équipe CyberHealth actuelle : Mohammed, Mehdi, Marc, Chala, Nesrine, Kassem et Hichem, qui je l'espère continuera à cultiver la convivialité et l'entraide. Merci pour nos échanges, les soirées à la plage, les sessions surf et les randonnées sur le sentier côtier. Je remercie aussi les membres du département DSD de l'IMT Atlantique, en particulier Corinne pour son professionnalisme et sa bonne humeur.

Sur une note plus personnelle, je remercie celles et ceux que j'ai rencontrés lors du Master Cryptis, notamment Lucie, Moerava, Johanna, Morgane et Clément, un master qui a été pour moi une belle découverte de la cryptographie et une aventure humaine inoubliable. En particulier, j'aimerais remercier Lucie et Moerava pour leur soutien infaillible comme pour le nombre incroyable de fois où elles me font rire (et parfois danser ?). J'aimerais aussi remercier ceux qui m'ont fait me sentir chez moi à Brest, les archers d'Iroise, notamment Florence, Christian et l'équipe féminine, et ceux qui sont devenus des amis :

Romain, Amandine, Amina, Charaf, Hubert, Julien.

Enfin, j'aimerais remercier ma famille. En particulier, je remercie mon frère Matthieu qui a toujours été un phare, mon camarade dans les galères et les bonheurs tout au long de nos vies. Merci d'avoir été là pour moi. Je remercie ma mère et mon beau-père pour leurs nombreux accueils lors de ma thèse, les soirées au coin du poêle m'ont réchauffé le cœur. En particulier, je remercie ma mère, la personne la plus forte et résiliente que je connaisse, de m'avoir soutenue sans relâche et transmis sa détermination, notamment lors de nombreux échanges téléphoniques, et d'un mythique semi-marathon. Ta présence m'a portée. Je remercie aussi mon père, merci d'être cette force tranquille dans ma vie, d'être venu partager du temps brestois avec moi, et de croire en moi. Enfin, j'aimerais remercier Amel et Noam, avec qui j'aime tant discuter, et ma grand-mère Louisette, pour ses appels rituels du dimanche soir qui ont rythmé mes semaines.

# Table of Contents

**Introduction** **9**

**1 Stockage de données dans l'ADN et sécurité** **13**

1.1 Stockage de données sur molécules d'ADN . . . 14

1.1.1 Qu'est-ce que l'ADN ? . . . 14

1.1.2 Chaîne de stockage ADN : vue d'ensemble . . . 15

1.1.3 Écriture des données dans des molécules d'ADN . . . 16

1.1.3.1 Encodage : contraintes, erreurs, indexation . . . 16

1.1.3.2 Synthèse . . . 18

1.1.4 Stockage . . . 21

1.1.4.1 Stockage *in vitro* . . . 21

1.1.4.2 Stockage *in vivo* . . . 22

1.1.5 Récupération des données à partir des molécules d'ADN . . . 23

1.1.5.1 Sélection de molécules d'ADN et préparation à la lecture . . . 23

1.1.5.2 Séquençage . . . 25

1.1.5.3 Traitement des données lues et décodage . . . 27

1.1.6 Simulation de la chaîne de stockage . . . 28

1.2 Analyse de la menace . . . 29

1.2.1 Définition d'une analyse de menace et des besoins en sécurité . . . 29

1.2.1.1 Normes et standards de sécurité . . . 30

1.2.1.2 Objectifs de sécurité . . . 32

1.2.2 Étude du contexte : caractérisation du Système d'Information . . . 33

1.2.2.1 Les scénarios avec flux de données . . . 33

1.2.2.2 Scénario étudié . . . 35

1.2.3 Identification des menaces . . . 38

1.2.3.1 Évènements redoutés et profils d'attaquants . . . 38

1.2.3.2 Motivations d'un attaquant et stockage de données sur molécules d'ADN 39

1.3 Étude des scénarios de menace : analyse STRIDE de la chaîne par étapes . . . 41

1.3.1 Les menaces du modèle STRIDE . . . 41

1.3.2 Menace à toutes les étapes . . . 43

1.3.3 Encodage . . . 44

1.3.4 Synthèse . . . 44

1.3.5 Préparation au stockage longue durée . . . 45

1.3.6 Archivage . . . 45

1.3.7 Séquençage d'ADN . . . 46

1.3.8 Décodage numérique . . . 48
1.3.9 Vue d'ensemble des menaces . . . 49
1.4 Traitement du risque . . . 49
1.4.1 Traçabilité . . . 49
1.4.2 Intégrité . . . 51
1.4.3 Disponibilité . . . 52
1.4.4 Confidentialité . . . 53
1.4.4.1 Management du Système d'Information . . . 53
1.4.4.2 Chiffrement . . . 54
1.4.4.3 Stéganographie ADN . . . 54
1.4.4.4 Cryptographie ADN . . . 56
1.5 Conclusion . . . 59

**2 Encodage avec contraintes de données dans l'ADN 63**
2.1 État de l'art des méthodes d'encodage avec contraintes . . . 64
2.1.1 Contraintes biologiques . . . 64
2.1.2 État de l'art . . . 65
2.1.2.1 Méthodes avec une contrainte . . . 65
2.1.2.2 Méthodes avec gestion d'une taille d'homopolymère et du taux de GC . . . 66
2.1.2.3 Seule méthode concurrente considérant les motifs interdits en plus des homopolymères et du taux de GC . . . 69
2.1.3 Contributions . . . 71
2.2 Chiffrement . . . 71
2.3 DSWE . . . 74
2.3.1 Dynamic Encoding avec dictionnaires préservant la distribution uniforme . . . 74
2.3.1.1 Méthode d'encodage . . . 75
2.3.1.2 Calcul du taux d'information . . . 79
2.3.2 Encodage à fenêtre glissante SWE . . . 80
2.3.3 Dynamic Sliding Window Encoding . . . 82
2.3.3.1 Méthode d'encodage . . . 82
2.3.3.2 Méthode de décodage . . . 84
2.3.3.3 Performances théoriques de DSWE . . . 86
2.4 Alternatives à DSWE . . . 87
2.4.1 Gestion des homopolymères par la fenêtre glissante . . . 87
2.4.1.1 Méthode d'encodage . . . 87
2.4.1.2 Performances théoriques de $SWE^{(4)}$ . . . 88
2.4.2 Fenêtre glissante binaire . . . 89
2.4.2.1 Méthode d'encodage . . . 89
2.4.2.2 Performances théoriques . . . 90
2.5 Codes correcteurs d'erreurs . . . 91
2.5.1 Codes LDPC . . . 91

2.5.1.1 LDPC binaire . . . 91
2.5.1.2 LDPC quaternaire . . . 92
2.5.2 Codes GRS concaténés . . . 93
2.5.2.1 Codes de Reed-Solomon . . . 94
2.6 Résultats expérimentaux . . . 96
2.6.1 Paramètres . . . 96
2.6.2 Résultats de DSWE . . . 97
2.6.2.1 Taux d'information . . . 97
2.6.2.2 Taux de GC . . . 99
2.6.2.3 Complexité algorithmique . . . 101
2.6.3 Performances comparées . . . 101
2.6.3.1 Comparaison aux méthodes satisfaisant 1 à 2 contraintes biologiques . . . 101
2.6.3.2 Comparaison à mCGR . . . 104
2.6.4 Alternatives à DSWE . . . 108
2.6.5 Codes correcteurs et encodage dynamique . . . 110
2.6.5.1 Codes LDPC et encodage dynamique . . . 111
2.6.5.2 Création d'une structure de codes concaténés binaires : 2 codes de Reed-Solomon . . . 113
2.6.6 Compression et encodage DSWE . . . 116
2.7 Conclusion . . . 118

**3 Une chaîne de stockage sécurisé de données dans l'ADN, du numérique au moléculaire 121**
3.1 Définition des opérateurs biologiques . . . 122
3.1.1 Classe 1 - Modifications des extrémités de l'ADN . . . 123
3.1.2 Classe 2 - Coupes d'ADN . . . 126
3.1.3 Classe 3 - Ligations . . . 129
3.1.4 Classe 4 - Modifications de bases . . . 132
3.1.5 Classe 5 - Amplification . . . 134
3.1.6 Classe 6 - Séparation de molécules spécifiques . . . 137
3.1.7 Définition d'une fonction avancée de rotation biologique . . . 146
3.1.8 Définition d'une fonction avancée de permutation de deux blocs de deux molécules différentes . . . 149
3.2 Solution de sécurité . . . 153
3.2.1 Chaîne de stockage de données dans l'ADN . . . 153
3.2.2 Encodage numérique . . . 155
3.2.2.1 Fonctions numériques inverses des fonctions biologiques . . . 155
3.2.2.2 Processus de chiffrement des données . . . 156
3.2.3 Déchiffrement biomoléculaire . . . 159
3.2.4 Complexité et analyse de sécurité . . . 161
3.3 Résultats expérimentaux . . . 164
3.3.1 Impact de DNACipher sur le processus de consensus . . . 164

3.3.1.1 Paramètres de tests . . . 164
3.3.1.2 Consensus sur des données chiffrées par rotations . . . 165
3.3.1.3 Consensus sur les séquences chiffrées par rotations et permutations . . . 166
3.3.1.4 Clustering et consensus . . . 168
3.3.1.5 Perspectives . . . 169
3.3.2 Preuve de concept biologique - Fonction de rotation . . . 170
3.3.2.1 Objectif et étapes du protocole . . . 171
3.3.2.2 Résultats expérimentaux . . . 178
3.3.3 Discussion . . . 190
3.3.3.1 Améliorations possibles . . . 190
3.3.3.2 Anciennes versions du protocole . . . 192
3.4 Conclusion . . . 194

**Conclusion** **197**

**Bibliography** **201**

**A Fonction de rotation biologique - Algorithme** **213**

**B Fonction de permutation biologique - Algorithme** **215**

**C Liste d'extrémités saillantes de taille 4 créées par plusieurs enzymes** **217**

**D Paramètres ITHOS de choix d'amorces** **219**

# INTRODUCTION GÉNÉRALE

Depuis plusieurs années, la production mondiale de données numériques connaît une croissance exponentielle. De 33 zetta-octets (*i.e.* $10^{21}$ octets) de données en 2018, elle devrait atteindre 181 zetta-octets [1] en 2025. Il devient donc essentiel de relever le défi du stockage de toutes ces données car les technologies électroniques actuelles, telles que la mémoire flash et les disques durs, atteignent leurs limites [2]. Celles-ci incluent les coûts énergétiques et environnementaux, la durabilité, et la densité des supports de stockage. En effet, la majorité des données produites dans le monde est stockée dans des centres de données ou "data centers", volumineux et consommateurs d'énergie. Ils pourraient couvrir un millième de la surface émergée de la planète d'ici 2040 [3], contre un millionième actuellement. De plus, les data centers nécessitent une importante quantité d'énergie pour fonctionner : aux États-Unis, ils représentaient $1,8\%$ de la consommation totale d'électricité [4] en 2018. D'autre part, la durée de vie des supports de stockage actuels est de cinq à vingt ans maximum. Au delà, les données doivent être copiées sur de nouveaux supports, ce qui nécessite de l'énergie et des nouveaux matériaux. Enfin, la capacité des supports actuels de stockage est limitée : un disque dur stocke au maximum quelques teraoctets (*i.e.* $10^{12}$ octets) de données. Ainsi, le stockage sur ce type de supports nécessite un remplacement régulier et entraîne de forts coûts énergétiques et écologiques, alors même que la majorité des données archivées ne seront jamais consultées. Le développement d'une solution de stockage dense, durable et économique est donc un enjeu majeur.

Pour y répondre, le stockage de données sur des molécules d'acide désoxyribonucléique (ADN) s'est récemment révélé très prometteur [5]. En effet, cette technologie pourrait être un million de fois plus dense que les disques durs, stockant de l'ordre de $10^{18}$ octets par gramme d'ADN [6], pour une consommation d'énergie quasi nulle (les molécules peuvent être stockées à température ambiante sans maintenance), et une durée de vie 100 fois plus longue. La durabilité de l'ADN a été prouvée à de nombreuses reprises, il est notamment possible d'analyser de l'ADN naturellement préservé pendant des centaines de milliers d'années [7] [8][9]. Le choix de l'ADN comme support de stockage d'information présente aussi d'autres avantages. En effet, l'ADN est le support de l'information du vivant. Savoir le lire et l'écrire est donc utile pour de nombreuses autres applications que le stockage de données. Ainsi, dans le domaine médical, le séquençage du génome humain permet de détecter les altérations moléculaires chez les patients atteints de cancer du poumon, et ainsi de guider leur traitement [10]. En conséquence, de nombreux acteurs investissent dans le développement de technologies permettant de lire et créer de l'ADN [11] et il n'y a pas d'obsolescence pour ces technologies. Enfin, la biologie moléculaire, et donc l'ensemble des techniques de manipulation de l'ADN, est un domaine actuel et pertinent. En témoigne le prix Nobel de chimie 2020 récompensant la technique d'édition génomique CRISPR-Cas9. Cet outil moléculaire permet de couper des brins d'ADN à des endroits précis, et ainsi corriger un gène.

En pratique, stocker des données dans l'ADN consiste à transformer des données numériques en molécule d'ADN synthétiques. Pour cela, le processus de stockage comprend trois étapes principales : i)

les données numériques sont converties en séquences ADN numériques (conversion de binaire en bases ADN) ; ii) les séquences sont synthétisées en molécules d'ADN ; iii) les molécules d'ADN sont stockées à l'intérieur de conteneurs spécifiques. Pour lire les données, les brins d'ADN sont lus par séquençage, générant des séquences ADN numériques qui sont décodées pour retrouver l'information originale.

Le concept d'utilisation de l'ADN pour le stockage de données n'est pas nouveau. Il remonte aux années 1960, avec l'idée de mémoire génétique [12]. Toutefois, la première démonstration expérimentale de stockage de données dans l'ADN date de 1988 [13], avec l'encodage d'une image de 35 bits. Jusqu'au début des années 2010, plusieurs autres preuves de concept [14] [15] [16] montrent la faisabilité de stocker des petites quantités de données dans l'ADN, *i.e.* jusqu'à $10^3$ octets. Ces études ont toutes un point commun : les données sont stockées dans des cellules vivantes, ce qui permet de faciliter le clonage de l'ADN. Cependant, le stockage de données dans l'ADN in-vivo nécessite de stocker des cellules vivantes de grande taille et ne pourrait donc pas présenter une alternative viable au stockage de données à grande échelle. D'où l'importance des résultats de deux études de 2012 [17] et 2013 [18], prouvant que l'on peut stocker respectivement 650 kilo-octets (*i.e.* $10^3$ octets) et 630 kilo-octets dans des molécules d'ADN et les conserver *in-vitro* à long terme. Dans ce but, les molécules d'ADN sont isolées de l'oxygène et de l'humidité [18] et encapsulées [19] [20] afin de pouvoir être conservées à température ambiante, protégées de toute dégradation [21]. Depuis, plusieurs études ont dépassé ces valeurs, stockant jusqu'à 200 mega-octets (*i.e.* $10^6$ octets) de données [22]. En pratique, plusieurs points doivent être améliorés pour que le stockage de données sur ADN soit viable à grande échelle. Tout d'abord, les coûts de synthèse actuels ne permettent de stocker que de petits volumes de données, avec un prix d'environ 1$ pour 10 bases ADN. D'autre part, la vitesse d'écriture est actuellement très lente, nécessitant plusieurs minutes pour chaque base d'ADN synthétisée [23]. Le rapide développement des technologies de biologie moléculaire permet néanmoins d'espérer une accélération des processus et une forte baisse des coûts de production.

## Problématique de la sécurité du support de stockage

Avec le développement du stockage de données dans l'ADN, se pose la question de la sécurité de ces données. Actuellement, le stockage de grands volumes d'information se fait généralement dans des data centers. Ceux-ci sont régulièrement la cible de cyberattaques [24], rendant vulnérables les données sensibles de millions d'utilisateurs. L'ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information) recense une forte hausse des compromissions touchant les ESN (Entreprises de Services Numériques) permettant d'externaliser des services, qui incluent les services de Cloud Computing et l'utilisation de data centers distants. En effet, 14 évènements [25] touchant des ESN ont eu lieu en 2022, contre 4 en 2020. Les attaques contre les data centers, causées par exemple par un rançongiciel, ont un coût financier immédiat : l'entreprise de conseil Gartner [26] évalue celui-ci à 5600 dollars par minute de panne d'un data center. Dans son rapport [27] sur le coût d'une fuite de données de 2023, IBM étudie 553 entreprises impactées dans 16 pays différents, et estime le coût moyen d'une fuite de données à $4,45$ millions de dollars. Il est également rapporté que les coûts liés aux violations de données ont augmenté dans plusieurs secteurs. Le secteur le plus impacté par les fuites de données est celui de la santé, avec une hausse des coûts de $53,3\%$ entre 2020 et 2023. Au delà du coût financier, la fuite de ces données porte atteinte à la vie privée des citoyens (via leurs données à caractère personnel), ainsi qu'à leur accès aux soins dans le cas des données médicales. En France, celles-ci doivent être conservées au minimum 20 ans après la dernière

consultation d'un patient, selon les recommandations du Conseil National de l'Ordre des Médecins. Il est donc nécessaire de disposer de solutions de stockage longue durée sécurisées.

Sécuriser le stockage de données est donc une question centrale car les vulnérabilités de ces systèmes peuvent avoir des conséquences financières et humaines importantes. Pour garantir un haut niveau de sécurité et assurer la compatibilité avec les processus de la chaîne de stockage, il est essentiel de développer les solutions de sécurité en parallèle avec la technologie de stockage dans l'ADN. Celle-ci étant encore émergente, sans méthode standard ou norme pour chaque étape de la chaîne, une telle opportunité existe. De plus, la question de la sécurité est ouverte, car les menaces spécifiques à ce support de stockage ne sont pas toutes identifiées, et il n'existe pas de solution de sécurité des données stockées dans l'ADN qui prend en compte les spécificité de ce support.

C'est dans ce contexte que s'inscrivent les travaux de recherche menés au cours de ce doctorat. Cette thèse est structurée comme suit.

Le Chapitre 1 fournit des définitions générales sur les domaines principaux que nous avons abordés afin de positionner les problèmes sur lesquels nous nous sommes concentrés. Ainsi, nous définissons l'ADN et présentons toutes les étapes de la chaîne de stockage de données dans l'ADN. Nous introduisons les contraintes et les avancées actuelles pour chaque procédé biologique nécessaire au stockage, à l'archivage et à la récupération des données stockées dans l'ADN. Afin d'évaluer les risques de sécurité qui pèsent sur la chaîne de stockage, nous définissons une analyse de tels risques, ainsi que les législations et normes mises en place pour garantir un niveau de sécurité adapté. Nous effectuons ensuite une analyse des risques de sécurité pesant sur cette chaîne, qui constitue la première contribution de cette thèse. Elle permet de caractériser le système d'informations lié au stockage de données dans l'ADN et d'identifier un certain nombre de vulnérabilités qui lui sont associées. Plus précisément, nous étudions les scénarios de menace pesant sur la chaîne de stockage selon la méthode STRIDE. Elle classe les types de menaces selon les propriétés de sécurité affectées, notamment la confidentialité, la disponibilité, l'intégrité des données et l'authentification des utilisateurs du système. Dans la dernière partie de ce chapitre, nous décrivons les mécanismes actuellement mis en place dans la littérature pour traiter les risques et mettons en évidence les lacunes que ces solutions présentent. Nous concluons sur la nécessité de développer des solutions de sécurité adaptées à la technologie du stockage dans l'ADN et aux spécificités du support biologique.

Dans ce contexte, nous avons cherché à mettre en place une solution de sécurité assurant la confidentialité des données dans toute la chaîne de stockage et qui profite de la durabilité de l'ADN. Ainsi, nous avons orienté notre recherche vers une solution de chiffrement basée sur des manipulations biologiques, et non sur des chiffrements numériques pouvant devenir fragiles ou obsolètes à court terme. Afin de pouvoir manipuler l'ADN, il est nécessaire de maîtriser la structure des séquences. C'est pourquoi nous avons développé un algorithme d'encodage de données binaires chiffrées en séquences ADN, que nous présentons dans le Chapitre 2. Cet algorithme encode des séquences conformes aux exigences des technologies biologiques de stockage dans l'ADN, et permet d'interdire l'encodage de certaines séquences de bases de l'ADN, appelées motifs interdits. Cette dernière propriété nous est indispensable pour pouvoir manipuler l'ADN avec des opérateurs biologiques, ce que nous proposerons avec notre solution de sécurité biomoléculaire dans le Chapitre 3.

Dans le Chapitre 2, nous détaillons tout d'abord les contraintes structurelles sur les séquences ADN, et faisons un état de l'art des méthodes d'encodage de données dans l'ADN avec contraintes. Cela nous

permet de préciser l'originalité de notre algorithme Dynamic Sliding Window Encoding (DSWE), qui code des flux de données chiffrées en séquences ADN en prenant en compte ces contraintes. Il repose pour cela sur un encodage dynamique et un encodage avec fenêtre glissante. Les performances de notre méthode d'encodage sont testées expérimentalement avec des données chiffrées et nous montrons avec un canal bruité simulant les étapes biologiques d'une chaîne de stockage qu'il est possible de récupérer à partir de molécules d'ADN des données encodées avec notre DSWE. Nous comparons aussi les performances de notre encodage aux méthodes de l'état de l'art, notamment en termes de densité d'information et de nombre de contraintes prises en compte. De plus, nous montrons qu'il est possible d'utiliser des codes correcteurs avec notre encodage, afin de corriger d'éventuelles erreurs causées par les procédés biologiques de la chaîne de stockage.

Dans le Chapitre 3, nous présentons une chaîne complète de stockage sécurisé de données dans l'ADN, du numérique au moléculaire. Plus précisément, nous proposons un nouveau protocole permettant de modifier la structure d'une molécule ADN encodée avec DSWE à l'aide d'opérateurs biologiques. Dans notre scénario, ces manipulations biologiques sont obligatoires pour pouvoir déchiffrer l'ADN et récupérer l'information stockée. Dans la première partie de ce chapitre, les différents opérateurs biologiques élémentaires à disposition sont identifiés. Certains d'entre eux nécessitent d'interdire des séquences de bases ADN lors de l'encodage, d'où l'utilisation de notre algorithme DSWE. A partir des opérateurs biologiques élémentaires, nous avons développé deux fonctions biologiques plus avancées, une rotation et une permutation. Celles-ci permettent respectivement d'échanger les positions de deux blocs de données adjacents dans une molécule d'ADN, et de permuter des blocs de données entre deux molécules différentes. Notre solution de sécurité est présentée dans la deuxième partie du Chapitre 3, avec la chaîne complète de stockage incluant le chiffrement numérique et le déchiffrement biologique des molécules d'ADN. Notre chiffrement numérique, DNACipher, s'appuie sur plusieurs fonctions de rotation et de permutation appliquées à des copies d'une même séquence ADN. Les molécules d'ADN obtenues sont synthétisées et mélangées dans la même solution. Pour retrouver l'information, il faut obligatoirement déchiffrer avec notre algorithme biomoléculaire DNADecipher chaque version de la molécule d'ADN avant le séquençage. Dans une troisième partie, nous présentons les résultats expérimentaux obtenus sur la base de simulations afin de tester la robustesse de cet algorithme de chiffrement biologique ainsi que sur des manipulations biologiques. Ces dernières constituent une preuve de concept montrant qu'une fonction de rotation sur des molécules d'ADN est possible. Enfin, nous concluons par des perspectives pour améliorer cette fonction biologique.

Chapitre 1

# Stockage de données dans l'ADN et sécurité

Ces dernières années, les technologies liées au stockage de données dans l'ADN ont évolué, permettant de stocker plusieurs centaines de megaoctets d'information, et ce dans une perspective de stockage longue durée. Ce chiffre tend à augmenter, du fait de l'intérêt de nombreux acteurs, comme le prouve la création de la DNA Data Storage Alliance, regroupant des entreprises telles que Microsoft, Twist Bioscience et plusieurs organisations académiques. Une des ambitions de ces acteurs est de réduire le coût du stockage à quelques dollars pour un teraoctet de données d'ici 2030 [11]. Cela passe entre autres par une optimisation des étapes d'écriture (synthèse) et de lecture (séquençage) de l'ADN. Comme évoqué en introduction, ces technologies sont encore en développement. Il n'existe pas de méthode standard et prendre en compte la sécurité des données et les menaces pesant sur ce stockage est un enjeu crucial. Ces aspects ne sont par ailleurs pas pris en compte dans leur globalité. En effet, il faut assurer la sécurité de toute la chaîne de stockage, ce qui inclut les données comme les dispositifs de traitement numérique et biologique de l'information.

Classiquement, il convient de s'intéresser à des propriétés ou exigences de sécurité [28] de disponibilité, d'intégrité, de confidentialité et de traçabilité des données. Ces dernières doivent être : utilisables à la demande, exactes et complètes, accessibles seulement aux entités autorisées, et leur évolution doit être traçable, etc. Cependant, et comme nous le verrons, ces objectifs de sécurité sont à revisiter en raison des spécificités du stockage de données sur ADN. En effet, de nouvelles menaces qui n'existent pas pour le stockage numérique sont à prendre en compte. La sécurité des données stockées dans l'ADN est donc encore une question ouverte.

Ce premier chapitre s'articule en trois parties. Tout d'abord, nous commençons par revenir plus en détail sur la définition de l'ADN, sa structure, ainsi que toutes les étapes de la chaîne permettant de stocker des données numériques dans des molécules d'ADN. Dans un second temps, nous analysons les risques de sécurité ou plutôt la menace associée à cette chaîne, prenant en compte des scénarios où différents acteurs se partagent ses fonctionnalités. Enfin, nous décrivons les mécanismes actuellement mis en place dans la littérature pour traiter ces risques et concluons sur les lacunes présentes dans ces solutions et la nécessité de mettre en place des solutions de sécurité adaptées ; solutions que nous développons dans les autres chapitres de ce manuscrit.

## 1.1 Stockage de données sur molécules d'ADN

Dans cette section, nous revenons sur la définition de l'ADN, sa structure et son rôle dans le vivant, avant de présenter les étapes constituant une chaîne de stockage de données dans des molécules d'ADN.

### 1.1.1 Qu'est-ce que l'ADN ?

**Rôle de l'ADN** L'ADN (acide désoxyribonucléique) est le support de l'information génétique de tous les êtres vivants. Il est notamment porteur des gènes, qui indiquent entre autres à chaque cellule son rôle dans un organisme. Cette information génétique se transmet de génération en génération grâce au phénomène de réplication de l'ADN. Lire et manipuler l'ensemble des gènes d'un organisme, c'est-à-dire son génome, est primordial dans des domaines tels que la médecine pour, par exemple, aider à établir un diagnostic et personnaliser un traitement à un patient de la manière la plus précise possible.

**Définition de l'ADN** La molécule d'ADN, ou brin double d'ADN, est formée par l'association de deux brins simples antiparallèles enroulés l'un autour de l'autre pour former une double hélice, comme schématiquement présenté en Figure 1.1. Celle-ci présente un diamètre d'environ 2 nm, *i.e.* $2 \times 10^{-9}$ mètres.

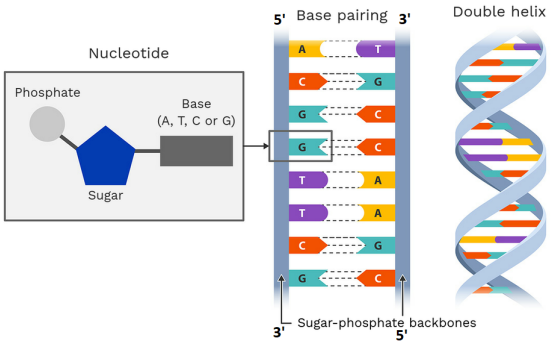



FIGURE 1.1 – Représentation schématique des structures d'un nucléotide et d'une molécule d'ADN. A gauche, la structure d'un nucléotide est représentée avec l'association d'un sucre (désoxyribose), d'un groupe phosphate et d'une base nucléique (A,C,G, ou T). Au centre, la formation de paires A-T et G-C permet de lier deux brins simples d'ADN. Les paires A-T sont liées par 2 liaisons hydrogènes en pointillés, tandis que les paires G-C en ont 3. A droite, la molécule d'ADN est schématiquement représentée sous sa forme de double hélice. *Copyright 2023, with permission from Labster ApS*

Un brin d'ADN est une suite de molécules, les nucléotides. Ces derniers sont constitués notamment d'une base nucléique parmi l'adénine (A), la cytosine (C), la guanine (G) ou le thymine (T). Par la suite, nous utiliserons indifféremment les termes base et nucléotide pour faire référence aux quatre molécules possibles : A, C, G, et T. Les molécules d'ADN peuvent être constituées de plusieurs centaines de milliers de paires de nucléotides, voire plusieurs millions. Par exemple, le plus grand chromosome humain est constitué de deux brins de 220 millions de nucléotides environ, tandis que le génome d'un être humain compte environ 3 milliards de paires de bases distribuées sur 23 paires de chromosomes.

Comme illustré en Figure 1.1, ceux-ci sont constitués d'une base nucléique, d'un sucre et d'un à trois groupes phosphates. Pour former un brin simple d'ADN, deux nucléotides sont liés grâce à une liaison forte, ou liaison covalente, qui associe le sucre d'un nucléotide au groupe phosphate d'un autre. Cette

association joue un rôle important dans la synthèse, ou création, d'ADN pour le stockage. En effet, les nucléotides d'un brin sont assemblés avec une orientation. Un brin est toujours synthétisé de son extrémité $5'$ vers son extrémité $3'$, la convention veut donc que l'on le lise dans ce sens. L'extrémité $5'$ ou $5'-P$ d'un brin d'ADN désigne l'extrémité terminant par un groupe phosphate d'un nucléotide, tandis que son extrémité $3'$ ou $3'-OH$ se termine par un sucre. Ainsi, lors de la synthèse d'un brin d'ADN, les nucléotides sont ajoutés un à un à l'extrémité 3' du brin en construction. Lorsque l'on décrit la séquence d'un brin d'ADN, on le fait donc dans le sens $5'$ vers $3'$. Une molécule d'ADN est elle formée deux brins simples complémentaires tous deux lus dans le sens $5'$ vers $3'$. Les deux brins ont une orientation antiparallèle : comme présenté dans le schéma du milieu de la Figure 1.1, le brin de gauche est lu de haut en bas, et le brin de droite de bas en haut.

Pour constituer un brin double d'ADN, deux brins simples se lient en formant des liaisons faibles entre leurs nucléotoides ou bases, selon les couples A-T et G-C, appelés les paires Watson-Crick. Ces liaisons sont plus fragiles que les liaisons entre deux nucléotides du même brin. A nouveau, comme illustré en Figure 1.1, la paire A-T est liée par 2 liaisons hydrogènes représentées en pointillés, tandis que la paire G-C est liée par 3 liaisons hydrogènes. Ces liaisons se font et défont selon des changements de température, des opérations d'hybridation ou de dénaturation. Par exemple, au-delà de 94 à $98°C$, les deux brins d'une molécule d'ADN se détachent. Au contraire, baisser la température sous un seuil compris entre 48 et $72°C$ favorise l'établissement de liaisons faibles et donc l'hybridation de brins complémentaires. La température d'hybridation se calcule selon la proportion de bases A-T et G-C dans les brins ADN ciblés. Nous le verrons dans les sections à venir, ces deux opérations seront très utiles dans plusieurs processus du stockage de données dans l'ADN.

### 1.1.2 Chaîne de stockage ADN : vue d'ensemble

Nous donnons en Figure 1.2 la vue globale d'une chaîne de stockage de données dans des molécules d'ADN avec, en vert, les étapes biologiques et, en bleu, les étapes numériques. Cette chaîne peut aussi être divisée en trois processus : l'écriture de données sur ADN, le stockage, et la récupération des données à partir des molécules.

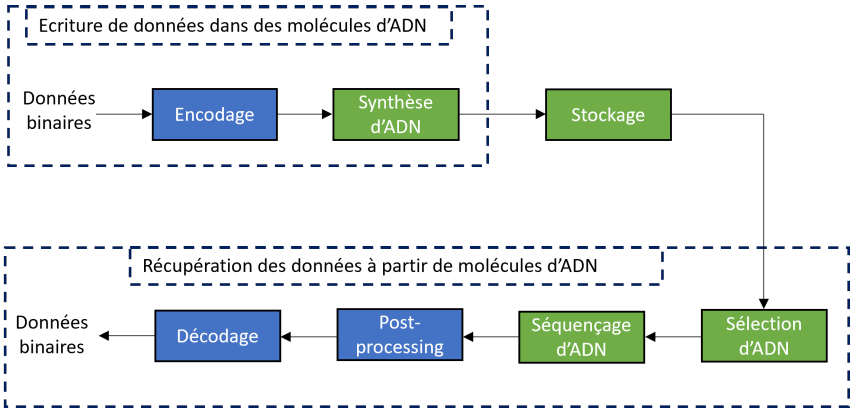



FIGURE 1.2 – Vue d'ensemble d'une chaîne de stockage de données sur molécules d'ADN avec, en vert, les étapes biologiques et, en bleu, les étapes numériques.

Le processus d'écriture de données numériques dans des molécules d'ADN est divisé en deux étapes. La première est numérique. Elle consiste en l'organisation et la conversion de données binaires en un encodage en base 4 associé aux 4 bases ADN ou nucléotides : A, C, G ou T pour obtenir une séquence ADN. La seconde est biologique, c'est la synthèse de la séquence d'ADN. Lors de celle-ci, les nucléotides sont ajouté un à un pour former un brin d'ADN synthétique. L'ADN synthétique peut être obtenu sous forme de brins simples ou de brins doubles (*i.e.* deux brins simples complémentaires formant une double hélice).

Le processus de stockage a pour but de conserver *in vitro* ou *in vivo* les molécules d'ADN qui résultent de la synthèse à l'abri de tout facteur environnemental nuisible, c'est-à-dire l'eau, l'oxygène, les rayons UV. En effet, lors de la synthèse, l'ADN est contenu dans une solution liquide qui permet la manipulation des molécules. Lors du stockage, l'ADN doit être isolé des facteurs nuisibles décrits précédemment.

Le processus de récupération des données se fait en 4 étapes. La première est la sélection d'ADN. Pour récupérer les données, il est nécessaire de sélectionner les brins d'ADN d'intérêt. Pour cela, l'ADN ciblé est généralement amplifié, et parfois isolé. Le but de l'amplification est de disposer de suffisamment de copies de chaque brin d'ADN pour faciliter le séquençage de l'ADN et pouvoir retrouver l'information stockée en minimisant les erreurs. Nous reviendrons sur ce point plus en détails. Un séquenceur à ADN lit ensuite chaque copie des molécules d'ADN et produit des séquences équivalentes numériques, qu'il stocke dans des fichiers. Notons que ces séquences comportent des erreurs liées aux différentes étapes de la chaine : synthèse, stockage, séquençage, etc. Les erreurs de chaque étape de la chaîne ne sont pas nécessairement de la même nature. L'étape suivante fait usage d'algorithmes issus de la bio-informatique pour traiter ces fichiers et déterminer des "séquences consensus". Ce "post-processing" (ou post-traitement) a pour but de réduire les erreurs car les technologies de synthèse et de séquençage ne sont pas parfaites. Enfin, les séquences d'ADN originales sont décodées en binaire. Parfois, des codes correcteurs d'erreurs sont utilisés pour minimiser encore plus les erreurs de lecture. Nous décrivons plus en détail ces processus dans les sections suivantes.

### 1.1.3 Écriture des données dans des molécules d'ADN

#### 1.1.3.1 Encodage : contraintes, erreurs, indexation

Le procédé d'encodage est la première étape de la chaîne de stockage. Il permet de formater les données pour pouvoir les écrire en molécules d'ADN et correctement les récupérer. Il se compose généralement de trois étapes : l'indexation des données, l'utilisation d'un code correcteur d'erreurs et la conversion de données binaires en nucléotides.

Afin d'encoder les données en séquences ADN, un fichier numérique est tout d'abord divisé en séquences binaires de taille fixe. Cette taille est limitée à quelques centaines de bases en raison des technologies de synthèse chimique et enzymatique actuelles [29], que nous décrirons dans la section suivante. Chaque séquence doit être indexée pour indiquer sa position dans son fichier numérique. Une première solution consiste à encadrer chaque séquence de suites de 18 à 30 bases appelées amorces. Dans ce scénario, chaque séquence a son unique couple d'amorces [30], placées en début et en fin de séquence. Cette méthode d'indexation limite la quantité de brins d'ADN qui peut être présente dans une même solution, car chaque brin différent devra avoir un couple d'amorces unique. Par ailleurs, nous le verrons par la

suite, ces couples d'amorces ne doivent pas être trop similaires entre eux, pour pouvoir être récupérés séparément. Une autre solution consiste à diviser un fichier en séquences binaires de même taille, à allouer un seul couple d'amorces à toutes les séquences [31] [22], puis à ajouter quelques bits au début de chaque séquence pour coder un index précisant la position de la séquence dans le fichier [32] [17]. En conséquence, toutes les données d'un fichier peuvent être récupérées avec le couple d'amorces du fichier.

**Encodage de binaire en ADN et contraintes sur la structure des séquences ADN** Les séquences binaires sont encodées avec un code en base 4 associé aux 4 nucléotides : A, C, G et T. L'encodage classique associe simplement deux bits de données à un nucléotide, par exemple $A = \{00\}$, $C = \{01\}$, $G = \{10\}$ et $T = \{11\}$. Toutefois, certaines séquences ADN sont difficiles voire impossibles à écrire ou à lire. C'est le cas lorsqu'elles contiennent de trop longues répétitions du même nucléotide, appelées homopolymères [33]. Les homopolymères ne peuvent pas dépasser une certaine taille, celle-ci variant de 3 à 5 bases actuellement selon les technologies utilisées [33] et leurs avancées. Une autre contrainte est le G-C content, ou taux de GC. C'est le taux de nucléotides G et C dans une séquence, qui doit être proche de 50%. Il est souhaitable de respecter cette contrainte dans chaque sous-partie d'une séquence, afin de ne pas avoir de longue sous-séquence sans G ou C. Une proportion très inégale de bases A-T et G-C provoque des pertes [33] lors du séquençage et des erreurs lors de l'amplification par PCR [34]. Un brin d'ADN peut former des structures secondaires lorsqu'une partie de sa séquence est complémentaire à une autre. Le brin peut se replier, prenant alors une forme d'épingle à cheveux ("hairpin") et peut poser problème lors de la synthèse ou de la formation d'ADN double brin. En outre, si l'ADN est stocké *in vivo*, *i.e.* dans une cellule vivante, certains motifs ou sous-séquences sont à éviter afin que la séquence ADN ne soit pas utilisée par l'organisme hôte comme information génétique. En effet, certains motifs, appelés les séquences d'initiation de transcription, indiquent à des enzymes qu'il faut transcrire de l'ADN en ARN. L'ARN est ensuite traduit en une séquence formant une protéine. Des motifs conduisant à la production d'ARN ou de protéines sont donc interdits.

Les solutions récentes d'encodage cherchent souvent à prendre en compte plusieurs de ces contraintes sur la structure de l'ADN. Le taux d'information encodée est alors inférieur à celui de l'encodage classique, *i.e.* 2 bits par base (BPB), mais permet de garantir la bonne écriture et lecture des données stockées. Un compromis est à faire en terme de densité d'information stockée et de respect des contraintes structurelles. Certaines approches prennent en compte une taille maximale d'homopolymère ainsi qu'un contrôle du taux de GC [35] [36], tandis que d'autres ajoutent à leur méthode d'encodage la possibilité de limiter l'apparition de structures secondaires [37] ou plus généralement les motifs interdits [38]. Nous reviendrons sur les solutions existantes d'encodage sous contraintes dans le chapitre 2, pour positionner une solution que nous proposons.

**Types d'erreurs et codes correcteurs** Prendre en compte ces contraintes permet de minimiser les erreurs présentes dans l'ADN encodé. Toutefois, les étapes de séquençage, de stockage et de synthèse restent sujettes à des erreurs de substitution, de délétion et d'insertion de nucléotides. Pour y remédier, un code correcteur d'erreurs peut être appliqué à chaque séquence avant sa synthèse en molécule d'ADN. Une redondance est introduite dans une séquence afin de la rendre plus résistante aux erreurs. Une telle protection peut être appliquée avant la conversion des données en base 4 [39] [22] [40] ou après [41] [42].

Ces codes correcteurs sont généralement basés sur des codes très connus tels que les turbo-codes [43] ou les codes LDPC [44]. La structure de turbo-code repose sur deux codes correcteurs successifs [32] [45] et permet de corriger un grand nombre d'erreurs. Les codes LDPC [40] sont des codes avec une capacité de correction presque optimale, utilisant une matrice de parité clairsemée. Ils permettent de corriger principalement des erreurs de substitution, mais aussi quelques effacements [46] [47]. Des codes correcteurs d'erreurs ont également été créés pour répondre aux spécificités des séquences d'ADN. Nous pouvons citer les codes de distance de Lee asymétriques [41] ou encore un code concaténé [42]. Ce dernier est composé d'un premier code convolutif et d'un second code LDPC en base 4. Le code convolutif permet de resynchroniser la séquence ADN, *i.e.* d'éliminer les erreurs de délétion et insertion, ainsi que de corriger une partie des substitutions. Quand à lui, le décodeur LDPC en base 4 corrige les substitutions restantes. Les codes de distance de Lee asymétriques [41] sont aussi des codes en base 4. Le décodage d'un message repose sur le calcul de la distance entre le message et le mot du code le plus proche. La distance dépend des modèles d'erreurs de la technologie de séquençage nanopore.

#### 1.1.3.2 Synthèse

La synthèse [29] est l'étape de passage du numérique au biologique. La synthèse de l'ADN est le procédé de création de brins simples d'ADN synthétiques par ajouts successifs de nucléotides. Plus clairement, une machine lit les séquences numériques de nucléotides et crée les fragments d'ADN correspondants. L'ADN synthétique créé se trouve en solution dans un tube à essai ou une plaque avec de nombreux puits, contenant généralement quelques microlitres de solution. Les deux méthodes principales sont la synthèse enzymatique ("Enzymatic DNA Synthesis") et la synthèse chimique par phosphoramidite. Elles sont basées sur des cycles qui permettent d'ajouter les nucléotides un par un à un brin existant. Les deux méthodes permettent donc de générer des simples brins d'ADN. Si l'on a besoin d'ADN double brins, il faut synthétiser les brins complémentaires séparément, puis les hybrider. Comme nous le verrons en fin de section, il convient aussi de s'interroger sur la validité d'une séquence avant toute synthèse.

**Synthèse chimique** La méthode standard actuelle est la synthèse chimique par phosphoramidites. Elle a été utilisée pour nombre d'expérimentations de stockage de données dans l'ADN [17] [18][32], dont celle encodant le volume de données le plus important à ce jour [22]. Afin d'initialiser la synthèse chimique, le premier nucléotide à écrire sur un brin d'ADN est greffé sur un support solide. Ce nucléotide est protégé d'un groupement protecteur, un ensemble d'atomes qui empêche une molécule de réagir, afin qu'aucun autre nucléotide ne s'y hybride. Cette synthèse se fait en parallèle avec plusieurs exemplaires d'un même brin initialisé, de quelques centaines pour une synthèse en colonne à des dizaines de milliers pour une synthèse en matrice ("microarray"). Pour ajouter un nucléotide phosphoramidite au brin existant, un cycle de 4 étapes est effectué. Le groupement protecteur est tout d'abord retiré de l'extrémité du brin d'ADN en utilisant un acide, généralement l'acide trichloroacétique, puis le nucléotide que l'on souhaite ajouter au brin est introduit dans la solution et est lié au brin sous l'action d'un autre acide, généralement de la famille des tétrazoles. Notons que ce nucléotide vient avec un groupement protecteur. Lors de ce processus, certains nucléotides ne parviennent pas à s'attacher aux brins. Dans ce cas, les brins d'ADN sont protégés jusqu'à la fin de la synthèse : aucun nouveau nucléotide ne sera ajouté. Cela permet de ne pas obtenir de brins d'ADN avec des erreurs de délétion dans la séquence. Pour finir le cycle, la solution

est nettoyée des molécules indésirables et des résidus. A la fin d'un cycle, chaque brin d'ADN a été allongé d'un nouveau nucléotide, ou d'une protection qui empêche toute nouvelle liaison. A la fin du processus, le brin d'ADN ainsi créé sera séparé du support par un clivage chimique.

Cette méthode peut être parallélisée en contrôlant les brins auxquelles sont ajoutées les différentes bases. A différentes positions du support solide, des brins différents sont créés en parallèle, cette méthode est appelée "array-based synthesis" [29].

Ce processus nécessite en général d'utiliser des solvants chimiques polluants. Toutefois, son efficacité en termes de couplage de nucléotides (plus de 99% par cycle [48]) et son faible taux d'erreur (moins d'une erreur pour 3000 nucléotides en moyenne) en font actuellement la méthode de synthèse d'ADN standard pour le stockage de données sur ADN. Afin de maintenir un niveau élevé de qualité, les fabricants limitent la taille des brins d'ADN synthétisés à 200 bases maximum. D'autres limitations font de la synthèse chimique le goulot d'étranglement du stockage de données sur molécules d'ADN. En effet, cette technologie coûte à ce jour environ 10 euros pour 100 nucléotides synthétisés, et synthétise un nucléotide en 100 à 200 secondes [49].

**Synthèse enzymatique** La synthèse enzymatique [50] est une alternative récente à la synthèse chimique. Elle ne nécessite pas l'utilisation de solvants polluants et repose sur l'action d'une enzyme particulière dans une simple solution aqueuse. Une enzyme est une substance organique produite par les cellules vivantes. Elle agit comme catalyseur dans les changements chimiques. L'enzyme utilisée pour la synthèse est l'ADN polymérase. Cette enzyme a plusieurs fonctions : elle intervient aussi dans la réparation de l'ADN, et peut synthétiser le brin complémentaire d'un brin simple, par exemple, pour amplifier un brin d'ADN (*i.e.* en faire plusieurs copies). L'ADN polymérase parcourt un brin simple dans le sens $5'$ vers $3'$. Elle ajoute des nucléotides à l'extrémité $3'$ d'un brin simple d'ADN en une action d'élongation de ce brin. La synthèse enzymatique est dite "de novo", car elle est effectuée *in vitro* sans "template" (ou modèle), contrairement à l'assemblage et à la modification de séquences ADN dans le vivant [29]. Pour la synthèse enzymatique, l'ADN polymérase particulière utilisée, *i.e.* la TdT ("Terminal deoxynucleotidyl Transferase"), ne requiert aucun modèle pour lier des nucléotides en un brin d'ADN.

Plus précisément, le processus de synthèse fonctionne comme suit. Tel que représenté en Figure 1.3, une suite de quelques nucléotides est collée sur un support solide. Cette suite est l'iDNA ("initiator DNA"), sur lequel est construit le brin d'ADN. En fin de synthèse, le brin d'ADN créé sera séparé de l'iDNA. Un cycle d'ajout de nucléotide fonctionne en deux temps, l'élongation du brin existant et la déprotection du brin. Premièrement, l'enzyme TdT vient se coller à l'iDNA et ajoute un nucléotide à la séquence. Le nucléotide a un embout, un terminateur réversible, qui empêche un autre nucléotide de venir se lier à lui. Dans un second temps, un réactif acide est ajouté à la solution pour déprotégé le nucléotide ajouté, c'est-à-dire supprimer son embout représenté en rouge sur la Figure 1.3. Le cycle se termine par un lavage, permettant de se débarrasser de tous les déchets de la solution, notamment l'embout. Un nouveau cycle d'ajout d'un nucléotide peut alors être initié. A l'issue des $n$ cycles d'ajout de nucléotide, l' enzyme est utilisée pour cliver le brin d'ADN de son support solide. Ainsi, le brin n'est composé que des nucléotides ajoutés un à un, sans l'iDNA.

A titre d'exemple de système d'une telle synthèse, nous pouvons citer l'imprimante SYNTAX [51] de DNA Script qui effectue la synthèse enzymatique en quelques heures, nécessitant plusieurs minutes

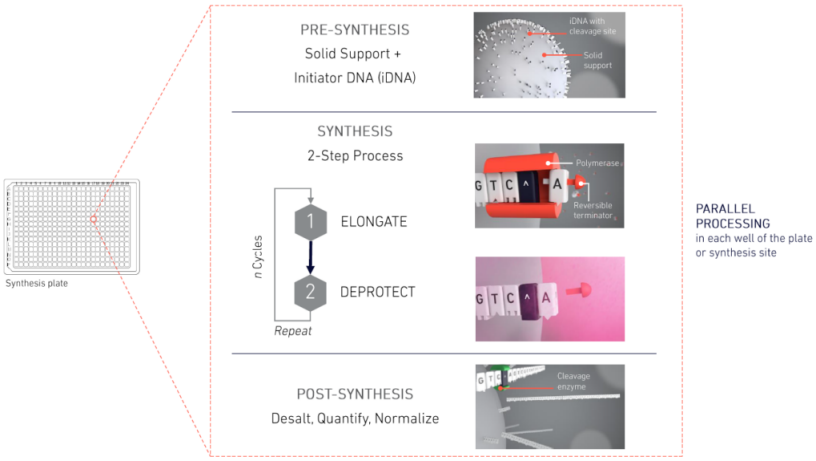



FIGURE 1.3 – Cycle de synthèse enzymatique. L'étape de pré-synthèse consiste à attacher quelques nucléotides ("initiator DNA") à un support solide. Puis, chaque cycle d'ajout de nucléotide repose sur 2 étapes : l'élongation avec l'ajout d'un nucléotide protégé à la séquence, et la déprotection de ce nucléotide. Enfin, à l'issue de $n$ cycles d'ajouts de nucléotides, l'ADN est séparé de son support solide. *Copyright 2022, reproduced with authorization from DNA Script*

par base ADN, avec un taux d'erreur moyen de $0,4\%$ pour des brins de 60 nucléotides. Notons que la synthèse enzymatique est actuellement limitée à quelques centaines de bases par brin pour conserver de faibles taux d'erreurs. Toutefois, elle peut en théorie générer des brins d'ADN bien plus longs, de plusieurs milliers de bases.

**Assemblage de courts brins simples d'ADN** Comme nous venons de le voir, l'ADN créé par synthèse enzymatique ou chimique est sous forme de courts brins simples. Or, les molécules d'ADN peuvent être infiniment plus grandes. Il existe aussi des technologies de lecture d'ADN "long-read" qui permettent de lire des brins de plusieurs centaines de milliers de bases [52] [53]. L'intérêt d'avoir de longues molécules est d'une part d'augmenter la densité d'information stockée, et d'autre part d'optimiser la taille des amorces utilisés pour l'indexation de l'information [54]. D'autre part, l'ADN double brin se conserve mieux [49] car il est plus stable que des brins simples qui peuvent notamment se casser, s'hybrider à d'autres brins similaires, ou se replier sur eux-mêmes. Enfin, certains opérateurs biologiques ne peuvent modifier que de l'ADN double brin, tels que les enzymes de restrictions qui coupent un brin double à une position précise.

Aujourd'hui, pour obtenir de longues séquences ADN, il faut passer par une étape de création de courts brins simples d'ADN synthétique, que l'on assemble ensuite en une longue molécule. Cette méthode est utilisée par de nombreux fabriquants d'ADN synthétique pour proposer à leurs clients la vente de longues molécules d'ADN. Par exemple, l'entreprise IDT (Integrated DNA Technologies) propose la création de gBlocks de 150 à 3000 paires de bases [55]. Les méthodes de ligation ne sont pas données par les fabricants, toutefois nous pouvons supposer qu'un assemblage de type PCA ("Polymerase Cycling Assembly") [56] est utilisé. Celle-ci consiste à créer des petits brins simples d'ADN, où chacun a une extrémité complémentaire avec un autre. Ils sont ensuite assemblés en un long brin double d'ADN grâce à l'enzyme ADN polymérase.

**Test de la validité d'une séquence** Avant de synthétiser des séquences ADN, un fabricant d'ADN vérifie qu'elles sont viables, afin de s'assurer que la synthèse se fera correctement. Chaque entreprise dispose de son propre logiciel de tests avec ses contraintes structurelles. Par exemple, l'entreprise Thermofisher dispose d'une application test, GeneArt Instant Designer [57], permettant d'entrer des séquences numériques de nucléotides (200-3000 bases), et qui indique si la séquence pourra être correctement synthétisée. Le cas échéant, elle indique quels nucléotides et quelles caractéristiques posent problème, et propose des modifications. Les principales contraintes sont le taux de nucléotides G et C ("G-C content" ou taux de G-C), qui doit se situer entre 20 et 80 % et les motifs spécifiques tels que les structures secondaires et les répétitions de longues séquences à éviter. Parmi ces motifs nous pouvons noter les longs homopolymères ainsi que l'absence de bases A et T dans une zone de plus de 25 bases (ou de 20 bases pour C et G). Cette étape de vérification peut aussi permettre d'évaluer si des fragments d'ADN pourront être assemblés. Pour cela, ils doivent notamment avoir un taux de GC proche. Une fois une séquence approuvée par le logiciel, elle peut être synthétisée.

### 1.1.4 Stockage

Pour être conservées à long terme, les molécules d'ADN synthétisées doivent être stockées dans des conditions particulières. De fait, la lumière (les rayons UV en particulier), les hautes températures, l'eau et l'oxygène détériorent l'ADN [58]. Plusieurs méthodes de stockage existent. Elles peuvent être regroupées dans deux catégories : le stockage *in vitro* et le stockage *in vivo*, *i.e.* dans des organismes vivants.

#### 1.1.4.1 Stockage *in vitro*

Afin de conserver l'ADN *in vitro*, les molécules sont isolées de l'humidité et de l'oxygène par encapsulation [19]. Par exemple, l'entreprise Imagene utilise des capsules en acier inoxydable, appelées DNAShells, avec un intérieur en verre pour stocker de l'ADN. L'ADN est desséché et confiné sous atmosphère inerte, sans humidité ni oxygène, dans des capsules fermées hermétiquement, scellées et gravées au laser. Le seul paramètre qui pourrait affecter les brins d'ADN stockés est l'élévation de la température. Pour éviter toute dégradation, les capsules doivent être conservées à température ambiante. Pour récupérer les fragments, elles sont perforées avec un désencapsulateur et les fragments sont réhydratés. Les capsules sont donc à usage unique. Elles peuvent contenir jusqu'à $0,8$ g d'ADN chacune, ce qui correspond théoriquement à un maximum de $10^{18}$ octets de données, pour une capsule de 18 milimètres de hauteur et $7,5$ mms de diamètre. Lors d'une preuve de concept [19], de telles capsules DNAShells ont été soumises à un processus de vieillissement accéléré par augmentation de la température, processus qui correspond à l'alteration causée par 100 ans à $25°C$. Chacune des capsules contenait $10^{-6}$ g d'ADN desseché, ce qui équivaut à 1 téra-octet de données. Aucune dégradation signification n'a été observée, prouvant que des données stockées sur ADN dans ces capsules peuvent être récupérées après plusieurs dizaines d'années de stockage à température ambiante. Le coût estimé du stockage [19] est de 4 à 5 euros par échantillon d'ADN encapsulé.

Une autre méthode de stockage *in vitro* repose sur l'encapsulation chimique d'ADN par des nanobilles de silice [20], représentées en Figure 1.4. L'ADN est enrobé d'une fine couche imperméable de silice de quelques nanomètres d'épaisseur. Ces billes sont placées dans des puits sur une plaque. Afin de prouver

que la solution est viable, des molécules d'ADN contenant 83 kB de données ont été stockées [32] dans les particules de silice et conservées à $70°C$ pendant une semaine, afin de simuler un vieillissement accéléré. Celles-ci ont pu être récupérées et lues, prouvant que les données peuvent être conservées *in silica* à température ambiante plusieurs milliers d'années. Ce stockage imite la protection des acides nucléiques dans les fossiles anciens, une comparaison [32] suggère même que le taux de dégradation de l'ADN stocké *in silica* est comparable à celui de l'ADN dans des os fossiles. Une telle stabilité permettrait d'espérer une grande longévité pour l'ADN stocké *in silica*, sachant qu'il est possible de séquencer des échantillons d'ADN d'os vieux de 300 000 ans [32]. Afin de récupérer les molécules d'ADN, la couche de silice est retirée à l'aide d'un solvant qui n'endommage pas l'ADN, qui est ensuite immergé dans une solution afin de pouvoir le manipuler, comme pour la méthode précédente.

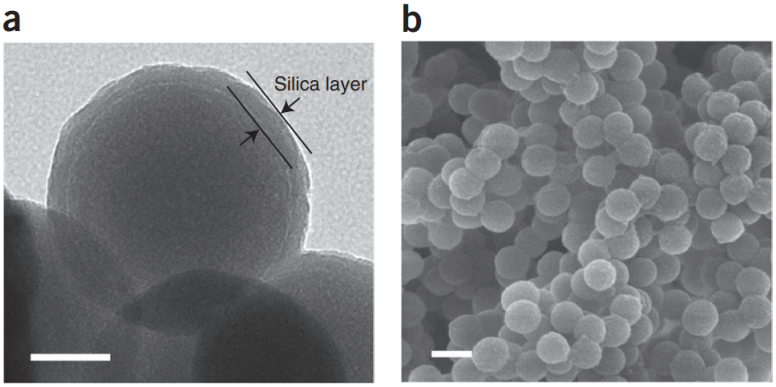



FIGURE 1.4 – Stockage d'ADN *in vitro* par encapsulation dans des nanoparticules de silice. La couche protectrice est d'épaisseur 12 nms (nanomètres) et isole des dégradations environnementales. a) Exemple d'une bille de silice contenant des molécules d'ADN de 113 paires de bases. Barre d'échelle de taille 50 nms. b) Ensemble de nanoparticules, avec barre d'échelle de 200 nms. *Copyright 2013, with permission from Elsevier [20]*

#### 1.1.4.2 Stockage *in vivo*

Le stockage d'ADN *in vivo* [59] repose lui sur des organismes unicellulaires, tels que des bactéries. Généralement, l'ADN est d'abord cloné dans des plasmides puis transféré dans des bactéries. L'intérêt de ce stockage est qu'il permet de profiter du phénomène naturel de réplication de l'information. L'ADN est conservé dans les bactéries ainsi que dans leurs descendantes. Toutefois, cette méthode peut poser des problèmes en cas de mutation des plasmides dans la bactérie. Ce phénomène modifie alors des nucléotides des séquences stockées. Certains motifs peuvent aussi être interprétés par l'organisme, la séquence ADN est alors utilisée comme information génétique et peut entraîner la production d'ARN ou de protéines, parfois dangereuses pour l'organisme. D'autre part, conserver l'ADN dans des bactéries peut présenter un problème de mise à l'échelle, car un tel stockage est volumineux. C'est pourquoi tous les projets récents de stockage avec plus de $10^5$ octets stockent l'ADN *in vitro*.

### 1.1.5 Récupération des données à partir des molécules d'ADN

Le processus de récupération des données commence par la sélection de molécules d'ADN ciblées. Celles-ci, appelées librairie de séquençage, sont ensuite préparées avant d'être lues. Les résultats du séquençage sont traités avec des algorithmes bio-informatiques puis décodés.

#### 1.1.5.1 Sélection de molécules d'ADN et préparation à la lecture

Cette étape permet de récupérer une partie ou l'ensemble des données stockées sur les molécules d'ADN. Lors de ce processus, les fragments d'ADN ciblés sont sélectionnés et préparés à la lecture.

Si l'on souhaite séquencer toute la librairie, l'ADN peut être fragmenté en molécules plus petites [60], car certaines méthodes de séquençage telles que le séquençage Illumina [29] ne lisent pas les longues molécules. La fragmentation par sonication permet de casser l'ADN à une taille spécifique en faisant varier la puissance des des ultra-sons et la durée d'exposition. Cette méthode est aussi appelée le cisaillement ultrasonique. Une autre solution, la fragmentation enzymatique, permet de couper l'ADN avec des enzymes appelés enzymes de restriction au niveau de certains motifs, *i.e.* séquences d'ADN précises de 4 à 10 nucléotides.

**Sélection d'ADN** Si quelques fragments en particulier sont à lire, on peut utiliser une approche de type "random access" [22] (accès direct). Sélectionner des fragments est avantageux car le séquençage est coûteux et source d'erreurs. Pour ce faire, nous décrivons deux méthodes qui utilisent les amorces présentes aux extrémités des molécules ciblées. Rappelons que ces amorces servent usuellement à indexer les données d'une fichier ou un fichier lui-même (*cf.* Section 1.1.3.1).

Une première méthode est la capture par hybridation [61]. Ce procédé repose sur l'utilisation de sondes : des brins d'ADN complémentaires aux brins d'ADN visés. Ici, les sondes sont les brins complémentaires aux amorces de l'ADN visé. Les sondes sont disposées sur un support solide, permettant aux brins d'ADN visés de s'hybrider avec elles. Après l'hybridation, les brins non ciblés sont éliminés par lavage.

La deuxième méthode de sélection d'ADN est la PCR [34], ou "Polymerase Chain Reaction". Celle-ci permet d'obtenir un nombre exponentiel de copies de chaque brin double d'ADN encadré des mêmes amorces. Cela permet de s'assurer que l'on dispose d'assez de matériel génétique pour retrouver l'information stockée dans les molécules d'ADN ciblées, et que celles-ci deviennent majoritaires dans une solution. En effet, les opérations biologiques de lecture et d'écriture de données dans l'ADN ont souvent des taux d'erreurs élevés. Une étape d'amplification est donc souvent nécessaire avant la lecture de l'ADN.

La chaîne de stockage considérée dans nos travaux s'appuie sur la PCR. Son fonctionnement est illustré en Figure 1.5, où l'objectif est d'amplifier la séquence d'intérêt en bleu de l'ADN double brin. La PCR est la répétition d'un même cycle qui double le nombre de molécules encadrées par les amorces. Un cycle de PCR consiste en 3 étapes :

— La dénaturation, où la température de la solution est augmentée jusqu'à un seuil d'environ 94 à $98°C$ pour séparer les brins doubles en brins simples, *i.e.* en brisant les liaisons faibles entre brins simples.

— L'hybridation ou alignement consiste à baisser la température en introduisant des amorces (représentées en violet) dans la solution qui vont s'hybrider à l'extrémité du brin simple qui leur est complémentaire.

— L'élongation ou extension, lors de laquelle la température est augmentée à $72°C$ environ pour activer les enzymes ADN polymérase ajoutées à la solution. Ainsi, l'enzyme se place sur un brin simple d'ADN au niveau de l'amorce et parcourt le brin simple en créant son brin complémentaire à l'aide de nucléotides flottants dans la solution.

Ainsi, à la fin d'un cycle de PCR, chaque fragment d'ADN encadré par les amorces a été copié. Le taux d'erreur est en général très faible : d'environ une erreur toutes les 10000 bases [62]. Au bout de $n$ cycles de PCR ($n \in [20, 35]$ en général), on obtient environ $2^n$ copies de ce fragment, comme présenté en Figure 1.5.

Notons que les amorces doivent respecter un grand nombre de contraintes structurelles [63] pour assurer le bon fonctionnement de la PCR. Notamment, elles ne doivent pas s'hybrider à une partie de la séquence qu'elles encadrent, ni l'une à l'autre. Il faut aussi prendre en compte le scénario où plusieurs couples d'amorces sont utilisés pour des fragments d'ADN différents dans une même solution. Dans ce cas, tous les couples d'amorces doivent être suffisamment différents les uns des autres afin de ne pas entraîner d'amplification des mauvaises données et donc de problème à la lecture des données. Afin d'utiliser des couples d'amorces adaptées, il est possible de s'aider de programmes de design d'amorces tel que Primer3 [64].

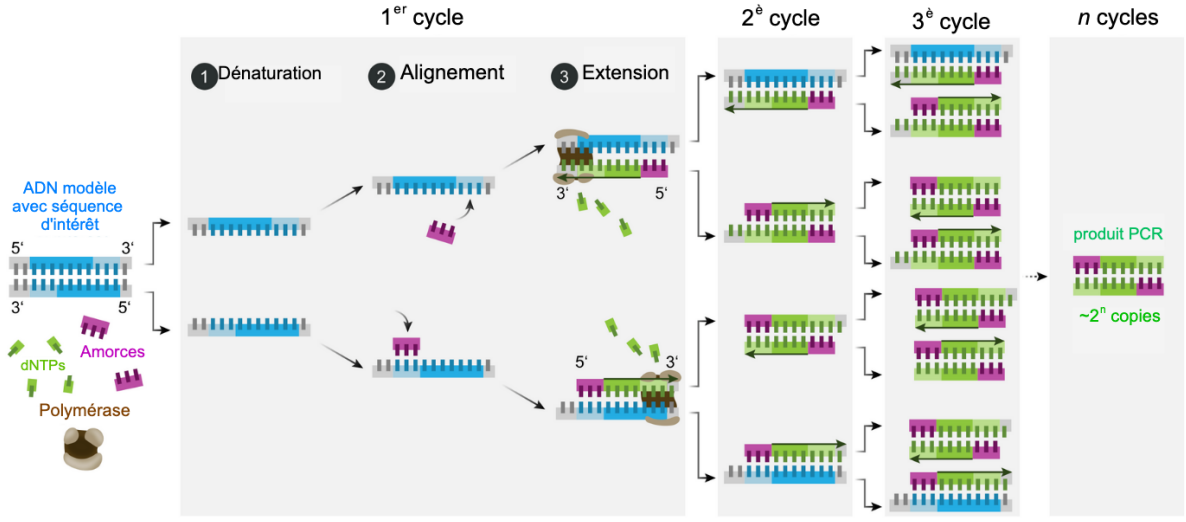



FIGURE 1.5 – Représentation schématique de l'amplification d'ADN par PCR ("Polymerase Chain Reaction"). Chaque cycle permet de doubler la quantité d'ADN cible, et en particulier la séquence d'intérêt représentée en bleu foncé. La PCR effectue des cycles de 3 étapes : la dénaturation, l'hybridation ou alignement et l'élongation ou extension. *Source : Wikimedia Commons CC BY-SA 4.0.*

**Préparation au séquençage** Afin de finaliser la préparation de la librairie de séquençage, des étapes supplémentaires en laboratoire sont généralement nécessaires. Un exemple de préparation de librairie

dans le cas du séquençage nanopore est présenté en Figure 1.6. Tout d'abord, une étape de réparation des dégradations que peut avoir subi l'ADN est essentielle. On peut notamment citer les entailles dans les brins, *i.e.* absence d'un nucléotide. Si un brin simple a une entaille, il sera séparé en plusieurs morceaux et ne pourra pas être lu entièrement. Des solutions commerciales fournies par des entreprises telles que New England Biolabs sont utilisées : Elles contiennent des enzymes qui parcourent l'ADN double brin et réparent les dégradations. D'autre part, un séquençage peut regrouper de l'ADN provenant de plusieurs solutions. Dans ce cas, il est important d'équilibrer les quantités d'ADN provenant de chaque solution. Pour cela, des dosages permettent d'évaluer la concentration d'ADN dans chaque solution, et donc les volumes permettant d'obtenir des proportions d'ADN équilibrées. Pour identifier les molécules d'ADN provenant de chaque solution, des "barcodes" spécifiques sont liés aux extrémités de l'ADN. Les extrémités de l'ADN doivent alors être préparées : une base A est ajoutée à chaque extrémité 3' des molécules [65] et un groupe phosphate (représenté par un P en Figure 1.6) est ajouté à chaque extrémité 5'. Ainsi, les barcodes avec un T à leur extrémité 3' se lient aux extrémités des molécules d'ADN qui se terminent par un A, la base complémentaire à T, grâce à la liaison entre le groupe phosphate de l'ADN et le sucre du barcode. Les molécules d'ADN des différentes solutions ont toutes des barcodes, elles peuvent être mélangées dans une unique solution. Enfin, dans le cas du séquençage nanopore et comme présenté en Figure 1.6, des adaptateurs de séquençage sont liés à l'ADN. Ils permettent aux brins d'ADN de bien être lus.

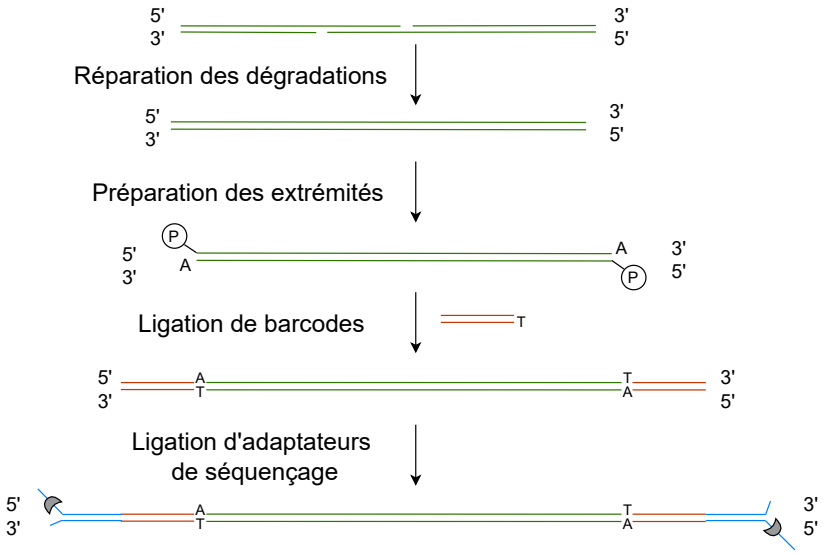



FIGURE 1.6 – Exemple de préparation de librairie de séquençage pour le séquençage nanopore : Réparation des entailles, préparation des extrémités, ligation de barcodes et d'adaptateurs.

#### 1.1.5.2 Séquençage

Le séquençage est la lecture des fragments d'ADN qui composent la librairie de séquençage, permettant de revenir au domaine numérique. Il y a deux types de méthodes de séquençage :

**Séquençage par synthèse** Cette méthode de synthèse a été développée en 1975 par Sanger [66]. La méthode de Sanger permet de séquencer quelques centaines de fragments d'ADN (jusqu'à 1000 bases) mais avec une très haute fidélité ( $99,99\%$ par base) [67]. Elle est donc essentiellement utilisée pour des séquençages à très petite échelle. De son côté, la méthode Illumina permet de séquencer de façon massivement parallèle, *i.e.* à haut débit, jusqu'à 1 téraoctet de données partant de brins de 300 bases maximum [29]. Le séquençage Illumina présente aussi de faibles taux d'erreurs, avec une fiabilité de $99,9\%$. La méthode de séquençage par synthèse repose sur la création (synthèse) du brin complémentaire du brin à lire et sur la détection de chaque ajout de nucléotide. Ainsi, les molécules d'ADN sont chauffées pour obtenir des brins simples. Des sondes complémentaires aux extrémités des brins d'ADN sont attachées sur une "flow cell", une lame de verre. Les brins simples d'ADN s'hybrident aux sondes et sont donc fixés sur la "flow cell". Par la suite, une enzyme ADN polymérase est utilisée pour synthétiser le brin complémentaire à chaque brin, nucléotide par nucléotide. Chaque type de nucléotide (A, C, G ou T) est marqué différemment, par exemple pour la méthode Illumina [68] avec des fluorophores (colorants fluorescents). A chaque incorporation de nouveau nucléotide au brin, une photographie est prise afin d'identifier le nucléotide. Une réaction chimique est appliqué pour enlever sa fluorescence et passer au nucléotide suivant. A la fin du processus, l'analyse des photographies permet de reconstituer la séquence ADN du brin.

**Séquençage nanopore** Contrairement aux méthodes précédentes limitées à quelques centaines de bases, cette méthode de séquençage [52], développée depuis 1999, permet de lire de très longs brins. Elle fait usage de nanopores : des trous d'un diamètre de l'ordre du nanomètre construits dans un résidu de silicium ou un pore de protéine cellulaire transmembranaire. Son principe est le suivant : Lorsqu'un nucléotide traverse un nanopore plongé dans un liquide conducteur, le courant qui traverse le nanopore change. Les variations de courant sont enregistrées au fur et à mesure que le brin simple d'ADN passe dans le nanopore [52]. Chaque k-mer, sous-séquence de $k$ bases, passant par le nanopore est associé à un signal électrique spécifique. Ainsi, la lecture se fait en temps réel grâce à un algorithme de basecalling [69]. Ce dernier utilise des réseaux de neurones entraînés sur des données de séquençage pour associer correctement une variation de courant à un k-mer de bases ADN. Les logiciels Guppy [69], et plus récemment Dorado [70], développés par Oxford Nanopore Technologies et intégrés à leurs séquenceurs sont majoritairement utilisés, d'autant qu'ils sont régulièrement mis à jour lorsque les motifs d'erreurs sont affinés. Un avantage du séquenceur nanopore est sa vitesse de lecture, plusieurs centaines de bases d'un brin d'ADN passent dans un nanopore chaque seconde. Ce nombre est actuellement limité à 420 bases par seconde pour ne pas entraîner trop de difficultés d'interprétation des variations du courant. En effet, le séquençage nanopore a des taux d'erreur par séquence assez élevés, de 5 à 10% selon des tests de 2016 [71] et 2021 [52]. Considérant le fort taux d'erreur de cette méthode, les fournisseurs conseillent de faire fonctionner le système 48 ou 72 heures afin d'obtenir le plus de "reads" (ou lectures) possibles d'un même brin, et maximiser la probabilité de retrouver les séquences ADN exemptes d'erreurs. Il est prouvé qu'avec suffisamment de copies et des algorithmes performants, la précision augmente au delà de $99,9\%$ [71]. Notons que le séquençage nanopore a récemment montré de bien meilleures performances grâce notamment aux nouvelles générations de modèles nanopore et de flowcell, garantissant une précision de plus de 99% par lecture [72]. Un nouvel algorithme de basecalling (Bonito) est en cours de développement

par ONT. Sa version beta démontre déjà de bien meilleures performances que l'algorithme actuel [73]. La moitié des erreurs est imputée [74] aux homopolymères (répétition de la même base, voir Section 1.1.3.1) [33]. Ceux-ci sont donc typiquement limités à une longueur maximale de 5 bases environ [74] ; longueur évoluant avec l'amélioration des technologies, et notamment des logiciels de basecalling. Un exemple de séquenceur nanopore est le MinION [53] de Oxford Nanopore Technologies (ONT), présenté en Figure 1.7. Ce séquenceur portable (455 grammes) lit des séquences ADN jusqu'à 4 MB (Méga bases) de longueur, et a un rendement maximal de 50 giga-bases de données. Afin d'optimiser encore le débit en sortie du séquenceur, ONT a développé le PromethION [53], séquenceur à grande échelle qui peut lire jusqu'à 6 téra-bases de données par jour. La troisième génération de séquençage, et notamment le séquençage nanopore, permet donc de lire de très longs brins d'ADN, en temps réel, pour un coût inférieur à celui du séquençage Illumina [71]. Toutefois, son taux d'erreur important nécessite la prise en compte de contraintes telles que les longs homopolymères afin de garantir la récupération des données.

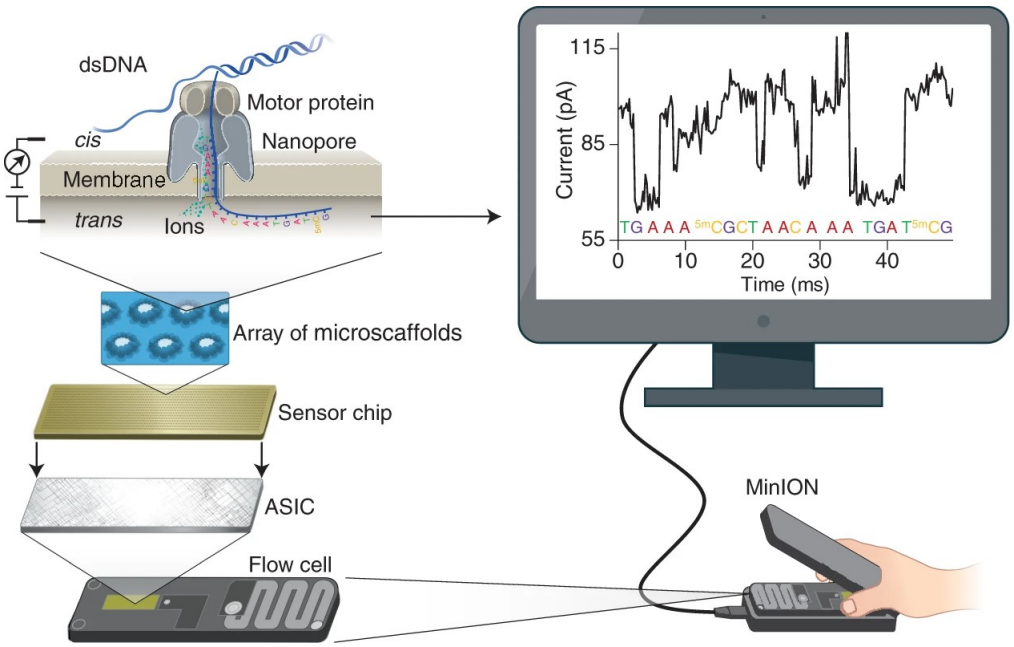



FIGURE 1.7 – Séquençage nanopore avec le dispositif MinION [53]. Une "flow cell" du MinION contient 512 canaux avec 4 nanopores chacun. Sur une puce de détection, chaque canal est associé à une électrode et mesuré individuellement. Un courant traverse le nanopore parce qu'une tension constante est appliquée à travers la membrane. Un simple brin d'ADN est acheminé à travers le nanopore, faisant varier le courant. Les variations de courant sont interprétées en direct avec un logiciel de basecalling, directement dans le MinION ou sur un ordinateur branché à celui-ci. *Copyright 2021 with permission from Clearance Center [52].*

#### 1.1.5.3 Traitement des données lues et décodage

En sortie du séquençage, on dispose donc d'un ensemble de lectures des copies d'un brin d'ADN. Des algorithmes de consensus bio-informatiques sont alors utilisés pour reconstruire une séquence consensus. De tels algorithmes s'appuient généralement sur des mécanismes d'alignement de séquences et de vote majoritaire pour trouver un consensus [75] [76] [77], et sont développés pour retrouver des séquences d'ADN génomique. Par la suite, nous considérerons l'algorithme Constrained Consensus Sequence Algorithm [78]

(CCSA), adapté au stockage de données dans l'ADN. Il considère plusieurs caractéristiques distinctes des séquences ADN synthétisées, qui diffèrent des séquences issues d'organismes vivants : les homopolymères sont de taille limitée, chaque séquence est encadrée d'amorces, et la longueur des séquences est connue. CCSA prend en entrée $n$ reads $D_1, ..., D_n$ d'un même brin d'ADN, la taille attendue $t$ de ce brin, et les amorces $For$ (pour Forward) et $Rev$ (pour Reverse) encadrant chaque séquence. L'algorithme construit un graphe de chevauchement des k-mers, *i.e.* des sous-séquences de $k$ bases. Chaque noeud de ce graphe dirigé correspond à un k-mer qui apparaît un minimum de fois dans les $n$ séquences. Comme les amorces en début et en fin de la séquence sont connues, elles permettent de définir les premiers et derniers k-mers. Une arête est créée entre deux noeuds seulement si il y a un chevauchement d'au moins $d$ bases entre les deux noeuds. Pour simplifier l'arbre des possibilités, un algorithme de Viterbi est utilisé. Celui-ci élimine les arêtes avec les plus faibles probabilités. La complexité du graphe dirigé passe ainsi de exponentielle à linéaire, et permet de déterminer une séquence consensus possible, c'est-à-dire un chemin dans le graphe qui part d'un k-mer situé au début des reads, et qui va vers un k-mer situé en fin des reads. Idéalement, ce chemin doit être de la taille attendue $t$. Avec suffisamment de reads en entrée de l'algorithme CCSA, on peut raisonnablement s'attendre à ce que la séquence consensus soit exempte d'erreurs. Ce n'est cependant pas toujours le cas. Une séquence consensus peut toujours contenir des erreurs, et être de taille légèrement différente à celle attendue. L'utilisation d'un code correcteur apparaît alors utile. Cependant les codes ne sont pas non plus parfaits et sont coûteux en termes de redondance d'information.

Si plusieurs molécules avec des amorces différentes ont été séquencées, il faut procéder à une étape de démultiplexage. En effet, comme nous l'avons vu dans la préparation au séquençage, lorsque des échantillons de plusieurs sources sont séquencés, ils sont encadrés de barcodes uniques. Ceux-ci sont utiles car en sortie du séquençage, tous les reads sont mélangés. Pour réassigner chaque read à sa solution de départ, les reads sont triés selon leurs barcodes. Les reads avec des barcodes non identifiables sont supprimés. Il est aussi possible d'utiliser des algorithmes de clustering pour regrouper les séquences par similarité, c'est-à-dire avec des contenus proches.

### 1.1.6 Simulation de la chaîne de stockage

Les fonctionnalités biologiques de la chaîne de stockage peuvent être simulées par un modèle de canal comme pour les communications numériques. Différents modèles ont été développés. Ils reposent généralement sur la détermination d'un profil d'erreurs à partir de séquençages de données expérimentales. Les probabilités d'erreurs peuvent être fixes, considérées comme indépendantes et uniformément distribuées [46] [79], ou bien dépendre du k-mer (sous-séquence de $k$ bases) séquencé et de sa position dans une séquence [80] [81] [42]. Ces méthodes s'appuient généralement sur le séquençage de données génomiques pour construire un modèle d'erreurs. Du fait du projet DnarXiv dans lequel s'inscrit nos travaux, nous avons utilisé le simulateur décrit par Hamoum *et al.* [42], qui ne s'appuie pas seulement sur des données génomiques pour déterminer les probabilités d'erreurs, mais aussi sur des données expérimentales. Celui-ci simule les procédés de synthèse, de stockage, et de séquençage implémentés avec les technologies suivantes :

— Synthèse : La méthode chimique standard (phosphoramidite) est prise en compte.

— Stockage : Les molécules d'ADN sont stockées *in vitro*, isolées par encapsulation de tout facteur

environnemental nuisible tel que l'humidité ou l'oxygène.

— Séquençage : La technologie de séquençage nanopore est utilisée avec le modèle nanopore R9.4 du séquenceur MinION [53] et le logiciel de basecalling Guppy v5.0.7 en mode "super-accurate".

Dans son principe, le simulateur prend en entrée une séquence ADN encadrée par des amorces et produit plusieurs copies de cette séquence qu'il stocke dans un fichier fastQ. Bien entendu, ces copies contiennent généralement des erreurs de substitution, d'insertion, et de délétion de manière semblable à celles causées par la chaîne de stockage précédemment décrite.

Afin d'obtenir les profils d'erreurs des k-mers (sous-séquences de $k$ bases), 9 jeux de données ont été stockés sur molécules d'ADN et récupérés. Le modèle d'erreurs obtenu repose sur un canal avec mémoire, il peut être considéré comme une chaîne de Markov. Considérant l'ensemble des évènements pouvant affecter une base ADN $B$ lors de son passage dans le canal : insertion, délétion, substitution ou "match" (bonne correspondance), la probabilité d'un évènement sur la base $B_i$ à la position $i$ de la séquence ADN codée $S = (B_1, ..., B_m)$ dépend du k-mer précédant cette base, et de l'évènement survenu à $B_{i-1}$. Pour résumer, les bases $B_{i-k+1}, ..., B_{i-1}$ et l'évènement de $B_{i-1}$ influent sur l'évènement affectant $B_i$. Les profils d'erreurs identifiés avec les données expérimentales permettent donc d'obtenir une simulation proche des évènements réels.

Pour conclure, cette première partie de chapitre a permis de définir l'ADN et les grands principes et étapes d'une chaîne de stockage des données dans de telles molécules, avec les technologies et algorithmes qui permettent son bon fonctionnement. Nous avons également pu voir que le codage des données doit tenir compte de différentes contraintes chimiques et biologiques, comme les longs homopolymères qui sont à éviter pour une bonne récupération des données, et que la lecture des données est entachée d'erreurs. Ces contraintes seront examinées plus en détail dans le chapitre 2, où nous proposons une solution d'encodage adaptée, qui nous sera utile pour détailler dans le chapitre 3 une méthode de sécurité au niveau biologique. Dans la suite de ce chapitre, nous analysons les besoins en sécurité de cette chaîne de stockage à travers une analyse de risques, une démarche qui, à notre connaissance, n'a pas encore été entreprise.

## 1.2 Analyse de la menace

La chaîne de stockage de données sur molécules d'ADN telle que décrite dans la section précédente implique de nombreux acteurs et des données qui peuvent être sensibles. Afin de savoir comment protéger ces données, nous analysons la menace ou encore les risques de sécurité associés à cette chaîne. Pour cela, nous définissons tout d'abord ce qu'est une analyse de la menace, quelles législations et normes qui imposent de sécuriser les données sont déjà mises en place pour garantir un certain niveau de sécurité, avant d'évaluer les risques liés à la chaîne de stockage dans des molécules d'ADN.

### 1.2.1 Définition d'une analyse de menace et des besoins en sécurité

Une chaîne de stockage de données sur ADN reste un Système d'Information, c'est-à-dire un ensemble organisé de ressources qui permet de collecter, stocker, traiter et distribuer de l'information. Pour sécuriser un tel système, une démarche classique consiste à identifier les éléments du système qui nécessitent d'être sécurisés, spécifier la nature des menaces, déterminer leur impact sur le système, et enfin élaborer des

politiques, procédures et dispositifs techniques pour protéger le système. Dans ce contexte, les normes et législations fournissent des directives claires aux entreprises pour leurs pratiques de sécurité. D'autre part, les analyses de menaces permettent d'anticiper les vulnérabilités du système et d'y remédier avec des mesures de réduction des risques.

#### 1.2.1.1 Normes et standards de sécurité

**Normes et législations** Les prestataires effectuant l'archivage de données sont soumis à de nombreuses règles déontologiques et législatives qui les obligent à assurer la sécurité des données, celles de leurs clients comme les leurs. Un objectif important est de maintenir la confiance des utilisateurs envers le système. Ces normes et législations sont très nombreuses et dépendent du contexte applicatif. Il est difficile d'être exhaustif mais nous pouvons par exemple citer le RGPD (Règlement Général sur la Protection des Données) qui s'applique à toute organisation traitant des données personnelles en Union Européenne. Il harmonise les lois sur la sécurité des données personnelles sur ce territoire. En particulier, le RGPD impose que les données des citoyens soient traitées selon les principes suivants :

- Sécurité - Elle concerne l'intégrité, la propriété d'exactitude et de complétude des données, qui ne doivent pas pouvoir être illégitimement modifiées ; et la confidentialité : les données ne sont accessibles qu'aux entités autorisées.
- Finalité - Les données sont collectées pour un but bien déterminé et légitime et ne sont pas traitées ultérieurement de façon incompatible avec cet objectif initial.
- Pertinence - Seules les données strictement nécessaires à la réalisation de l'objectif poursuivi doivent être collectées.
- Durée limitée de conservation - Les données ne doivent être conservées sous une forme identifiante que le temps nécessaire à la réalisation de l'objectif poursuivi et doivent être par la suite détruites, anonymisées ou archivées dans le respect des obligations légales.
- Droit des personnes - Selon ce principe, un citoyen conserve la maîtrise des données qui le concernent. Ainsi, il a le droit d'accéder aux données, de les rectifier et de s'opposer à leur utilisation.

La CNIL (Commission Nationale de l'Informatique et des Libertés) est le régulateur des données personnelles en France. Elle est habilitée à contrôler, sanctionner et mettre en demeure les organismes qui ne respectent pas les dispositions du RGPD. Le montant de telles sanctions peut s'élever à 20 millions d'euros ou pour une entreprise à 4% de son chiffre d'affaire annuel. En 2021, la CNIL a sanctionné 17 projets pour une somme totale de 21410600 euros [82].

**Stockage et règles bioéthiques** Lorsqu'un Système d'Information inclut des ressources telles que des molécules d'ADN, des standards et règles de sécurité particuliers s'appliquent. En effet, les outils et techniques utilisés sont proches voir identiques à ceux que l'on utilise pour manipuler l'ADN des êtres vivants. C'est le cas par exemple de la technique Crispr-Cas9 (ciseaux moléculaires) qui permet la modification du génome humain et donc qui pose des problèmes éthiques. C'est pourquoi le Conseil Constitutionnel a modifié en 2021 l'article 16-4 du Code Civil afin de favoriser une recherche responsable en lien avec la médecine génomique. En particulier, la modification d'un embryon humain par adjonction de cellules provenant d'autres espèces et l'édition génomique dans le but de modifier la descendance sont

interdites. Les manipulations possibles de l'ADN sont donc encadrées. C'est pourquoi des entreprises spécialisées dans la biologie de synthèse, telles que Twist Bioscience ou Thermo Fisher Scientific disposent d'un code de conduite professionnelle et d'éthique qui inclut une section Bioéthique. Celle-ci engage la responsabilité de l'entreprise à produire de façon sûre et efficace, en respect des normes biotechnologiques. Cela inclut de bonnes pratiques en laboratoire, et le contrôle des manipulations génétiques afin de ne pas créer de mutations dangereuses pour la santé ou l'environnement. Elles se sont aussi dotées d'un comité bioéthique qui veille au respect de pratiques transparentes et éthiques au sein de l'entreprise.

Concernant le stockage de données dans l'ADN, l'Office parlementaire d'évaluation des choix scientifiques et technologiques a produit une Note Scientifique en 2021 pour informer le Sénat [83]. La consultation d'experts a permis de déterminer que le stockage *in vitro* "ne semble pas poser de problématiques éthiques puisque les molécules synthétisées ne sont pas introduites dans des cellules et donc susceptibles d'être interprétées". Ce n'est pas le cas du stockage d'ADN *in vivo*, qui pourrait lui poser des problèmes au niveau bioéthique. En effet, si les molécules sont interprétées, il y a un risque de créer un virus, comme présenté dans la Section 1.1.2. C'est pourquoi il est bien moins utilisé.

Pour revenir au domaine numérique, ThermoFisher dispose aussi d'un responsable de la sécurité de l'information dont l'équipe est chargée de diriger la stratégie, la politique, les normes, l'architecture et les processus en matière de sécurité de l'information à l'échelle de l'entreprise. En 2019, le programme de cybersécurité de l'entreprise a obtenu un certificat ISO/IEC 27001 [84], norme standard pour les systèmes de management de la sécurité de l'information.

**Standards de sécurité et analyses de sécurité** Les normes et standards de sécurité pour les systèmes d'informations (SI) sont nombreux et en constante évolution depuis la prise de conscience de la menace cyber [85] au niveau mondial.

Ils concernent tout aussi bien la marche à suivre pour sécuriser un système d'information que les mécanismes de sécurité comme les signatures numériques [86] ou les certificats d'échange de clés cryptographique (*e.g* ; ec. [87]). Afin de garantir la sécurité d'un système d'informations, les recommandations et bonnes pratiques mises à disposition par l'ANSSI (Agence Nationale de Sécurité des Systèmes d'Information) peuvent aussi être suivies. Les agences gouvernementales telles que l'ANSSI participent activement à la standardisation de la sécurisation des SI (systèmes d'information) dans leurs pays. Une norme très connue pour le management de la sécurité des systèmes d'information, qui peut amener par ailleurs à une certification, est la norme ISO/IEC 27001 [84]. Elle est basée sur l'amélioration continue de la protection des données selon le modèle PDCA (Plan Do Check Act) et explique donc comment planifier, mettre en œuvre, vérifier et mettre à jour la sécurité d'un SI. L'analyse de risques est un élément clé dans la planification et la vérification de la sécurisation d'un SI. Elle a pour objectif d'identifier les menaces qui pèsent sur ce système et les besoins et objectifs de sécurité à atteindre pour y répondre. Les méthodes sont là aussi nombreuses. Nous pouvons citer par exemple les méthodes STRIDE de Microsoft, MEHARI (Méthode harmonisée d'analyse des risques) du Clusif et EBIOS (Expression des besoins et identification des objectifs de sécurité) Risk Manager de l'ANSSI ou encore la norme ISO/CEI 27005, qui est complémentaire à la norme ISO 27001, et qui traite en particulier la gestion des risques selon le modèle PDCA.

De telles méthodes d'appréciation et de traitement des risques sont constituées de plusieurs étapes,

permettant de lister, d'évaluer, de prioriser les risques et de mettre en place des contre-mesures. Une vue d'ensemble du processus d'analyse de risques est donnée en Figure 1.8. Celui-ci est constitué des quatre étapes suivantes :

— Étude du contexte : établissement du scénario global permettant de caractériser le Système d'Information. Cela passe par l'identification des acteurs, actifs et flux de données.

— Identification des menaces : Identification des sources de risque potentielles, c'est-à-dire des possibles attaquants et de leurs objectifs.

— Évaluation des risques : Évaluation des causes, probabilité et impact d'un risque afin de définir sa gravité. Le risque peut avoir un impact sur différentes propriétés de sécurité des données et des ressources. Il est possible de donner un score de risque à celui-ci, afin d'y associer des contre-mesures appropriées.

— Management et traitement des risques : Mise en place d'outils de protection afin de prévenir ou minimiser les risques identifiés.

En particulier, les trois dernières étapes sont à mettre à jour régulièrement, pour prendre en compte l'apparition de potentielles nouvelles menaces.

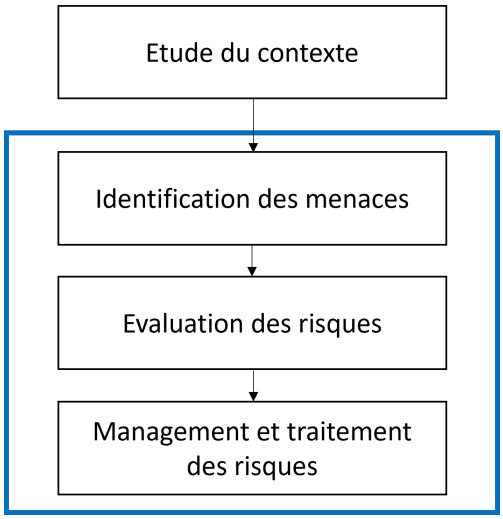



FIGURE 1.8 – Vue d'ensemble du processus d'analyse de risques

Notre analyse de menaces repose sur le modèle STRIDE, permettant d'identifier les menaces de sécurité dans un Système d'Information. Nous avons choisi STRIDE pour sa facilité d'utilisation, de compréhension, et le large éventail de menaces envisagées. En effet, chaque lettre du mot STRIDE correspond à une menace, elle-même associée à une violation d'un objectif de sécurité que l'on souhaite atteindre dans le système. Ces objectifs sont décrits dans la section suivante. Nous caractériserons ensuite le Système d'Informations associé au stockage dans l'ADN avec ses acteurs et flux de données.

#### 1.2.1.2 Objectifs de sécurité

Les objectifs, ou besoins, de sécurité sont les propriétés de sécurité à garantir pour une composante du système, définie par exemple par EBIOS comme "un service, une fonction support, une étape dans un

projet et toute information ou savoir-faire associé" important pour l'organisation dans l'accomplissement de la mission du système.

Les objectifs de sécurité les plus classiques [28] sont les suivants :

— Disponibilité : Propriété d'une composante du système d'être accessible et utilisable à la demande par une entité dans les conditions d'utilisation prévues.

— Intégrité : Propriété d'exactitude et de complétude d'une composante. Une modification illégitime d'une composante doit pouvoir être détectée.

— Confidentialité : Propriété d'une composantes du système à n'être accessible qu'aux entités autorisées.

— Preuve [88] : Propriété d'une composante du système permettant de retrouver, avec une confiance suffisante, les circonstances dans lesquelles cette composante évolue. Cette propriété englobe notamment la traçabilité des actions menées, de la trace de l'authentification des utilisateurs, des autorisations données et l'imputabilité du responsable de l'action effectuée.

Nous allons donc étudier ces aspects d'analyse de risques dans le cadre d'une chaîne de stockage de données sur molécules d'ADN, notre Système d'Information, en commençant ce processus par l'étude du contexte.

### 1.2.2 Étude du contexte : caractérisation du Système d'Information

La première étape d'une analyse de menaces, présentée en Figure 1.8, est l'étude de contexte qui pour nous est la chaîne de stockage dans sa globalité. Elle comprend ses fonctionnalités potentiellement gérées ou fournies par des entités ou des prestataires différents. Nous avons identifié plusieurs scénarios de déploiement de la chaîne de stockage, où les interactions entre les acteurs, les actifs et les données varient ; scénarios que nous décrivons ci-après. Par la suite, nous détaillerons le résultat de notre analyse de risques prenant en compte le plus réaliste d'entre eux.

#### 1.2.2.1 Les scénarios avec flux de données

Nous avons choisi de nous placer dans des scénarios de prestation de Service pour des Clients. En particulier, les acteurs du système sont :

— les Clients qui souhaitent stocker les données de leurs utilisateurs,

— les Prestataires qui fournissent leurs Services. Plusieurs Employés du Prestataire effectuent les Services comme le stockage et la récupération de données.

Comme présenté en Figure 1.9, les acteurs du Système d'Information peuvent interagir avec la chaîne de stockage des données ADN de plusieurs façons. Dans tous les cas d'usage, le Client (en bleu) fait appel à un ou plusieurs Prestataires (en violet) pour effectuer trois Services :

— la préparation au stockage

— le stockage

— la récupération de données stockées

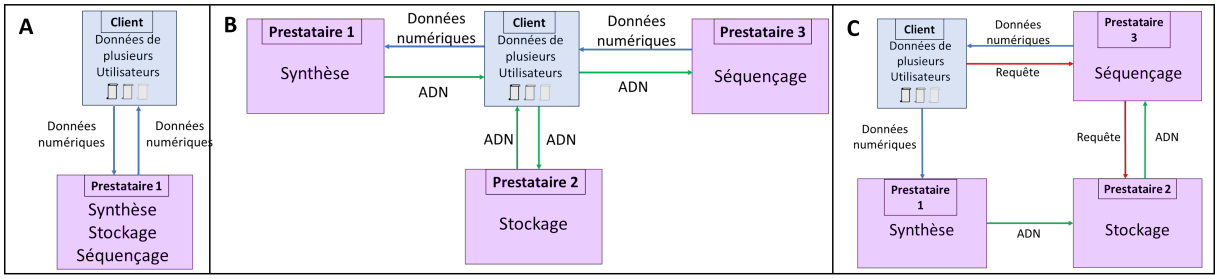



FIGURE 1.9 – Cas d'usage pour le stockage de données d'un Client dans l'ADN par un ou plusieurs Prestataires. A gauche, le Client s'adresse à un unique Prestataire. Au milieu, le Client échange pour chaque Service avec des Prestataires différents. A droite, le Client fait appel à 3 Prestataires qui communiquent et se transmettent les données entre eux.

Quel que soit le scénario, des données numériques (flèches bleues) et des données biologiques (flèches vertes) sont échangées entre les différents acteurs. Les données numériques peuvent être : les données binaires d'un utilisateur, les séquences ADN indexées avant synthèse, le résultat d'un séquençage, l'index des données d'un Utilisateur, etc. Les données biologiques sont des solutions, des capsules de stockage contenant l'ADN, etc.

Le premier cas d'usage est présenté en Figure 1.9A : un Prestataire effectue tous les Services. Il récupère les données numériques, les synthétise et les stocke. Lorsque le Client a besoin d'accéder à certaines données, il envoie une requête à ce Prestataire. Dans ce scénario, le Prestataire gère à la fois l'indexation numérique des données des utilisateurs du Client (encodage de chaque fichier en de multiples séquences ADN indexées) et l'indexation biologique des séquences ADN (ajout d'amorces aux différents fichiers des utilisateurs). Sachant qu'il archive l'ADN, il a donc toutes les informations disponibles pour accéder à des données précises.

Dans un second scénario donné en Figure 1.9B, le Client fait appel à 3 Prestataires, un pour chaque Service. Après chaque Service, les données transitent par le Client qui les confie à un nouveau Prestataire. Les Prestataires ne communiquent donc jamais directement entre eux, le Client donne des informations à un Prestataire seulement lorsqu'il a besoin de ses services. Ce scénario présente toutefois plusieurs inconvénients pour le Client. Tout d'abord, il doit s'assurer que les services fournis par les différents Prestataires sont compatibles. Par exemple, le mode de stockage doit être compatible avec la méthode de préservation de l'ADN. D'autre part, le Prestataire effectuant la récupération des données a besoin des informations d'indexation numérique et biologique, c'est au Client de les gérer dans ce scénario. Enfin, ce scénario implique des échanges de données biologiques entre les Prestataires et le Client. Ce dernier doit donc avoir un personnel formé aux procédures de sécurité et aux réglementations propres au transport de telles données.

Le dernier scénario est un compromis entre les deux solutions précédentes. Il permet au Client de déléguer les échanges de données. En effet, dans le scénario présenté en Figure 1.9C, les Prestataires sont en communication et ne nécessitent pas l'intervention du Client. Un premier Prestataire effectue l'encodage de données dans de l'ADN, puis transmet le support de stockage au second Prestataire pour archivage. Il transmet aussi les informations d'indexation, soit au Client soit au Prestataire faisant l'archivage. Enfin, lorsque le 3ème Prestataire reçoit une demande du Client, il récupère le matériel génétique auprès du second Prestataire et décode les données pour les transmettre au Client. C'est ce scénario que nous

étudions dans l'analyse de risques à suivre.

### 1.2.2.2 Scénario étudié

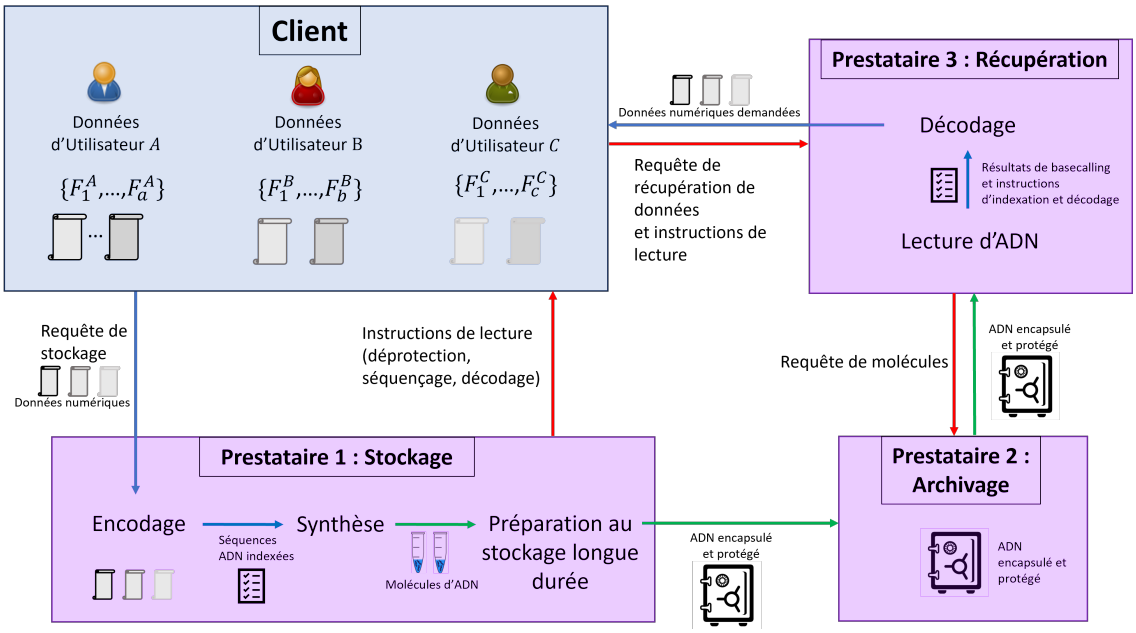



FIGURE 1.10 – Diagramme de flux de données pour le scénario choisi. Le stockage de données des utilisateurs d'un Client dans l'ADN passe par plusieurs Prestataires de services.

Le scénario choisi est présenté en détails en Figure 1.10, où sont précisés les flux de données entre acteurs du système. Lorsqu'un Client souhaite stocker des données, il les confie au Prestataire 1, qui effectue l'encodage, la synthèse et l'organisation des molécules dans un support physique. Le Prestataire 1 donne les instructions de lecture au Client (flèche rouge), telles que les paramètres d'encodage numérique, l'indexation numérique et biologique (amorces), les instructions de déprotection de l'ADN des supports physiques, etc. D'autre part, il donne le support de stockage au Prestataire 2, qui est chargé du service d'archivage. Lorsque le Client souhaite récupérer des données, il fait une requête au Prestataire 3 (flèche rouge) et lui transmet les informations nécessaires (indexation numérique et biologique des données). Le Prestataire 3 récupère les supports physiques correspondant aux données auprès du Prestataire 2. Le Prestataire 3 récupère alors les données via des étapes de séquençage et de décodage numérique, avant de les transmettre au Client. Ce système permet un certain cloisonnement entre les Prestataires. En effet, le fournisseur d'ADN synthétique ne conserve pas l'ADN qu'il synthétise. Le Prestataire qui conserve l'ADN n'a pas les informations d'indexation dans l'ADN, seulement les informations d'organisation du support physique. Enfin, le Prestataire 3 chargé du séquençage et de la récupération de données n'obtient les informations de lecture que selon le besoin du Client.

Dans ce qui suit, nous présentons en détail chacun des trois Services avec les acteurs, actifs et flux de données impliqués. Cela nous sera utile pour traiter des risques au sein des entreprises prestataires de services considérant, par exemple, le niveau d'expertise nécessaire pour un acteur pour mener une attaque.

**Service 1 : Préparation au stockage de données sur molécules d'ADN** Comme présenté en Figure 1.11, le Prestataire reçoit des données numériques, et celles-ci sont stockées de façon organisée dans des molécules d'ADN, elles-mêmes encapsulées dans un dispositif de stockage physique.

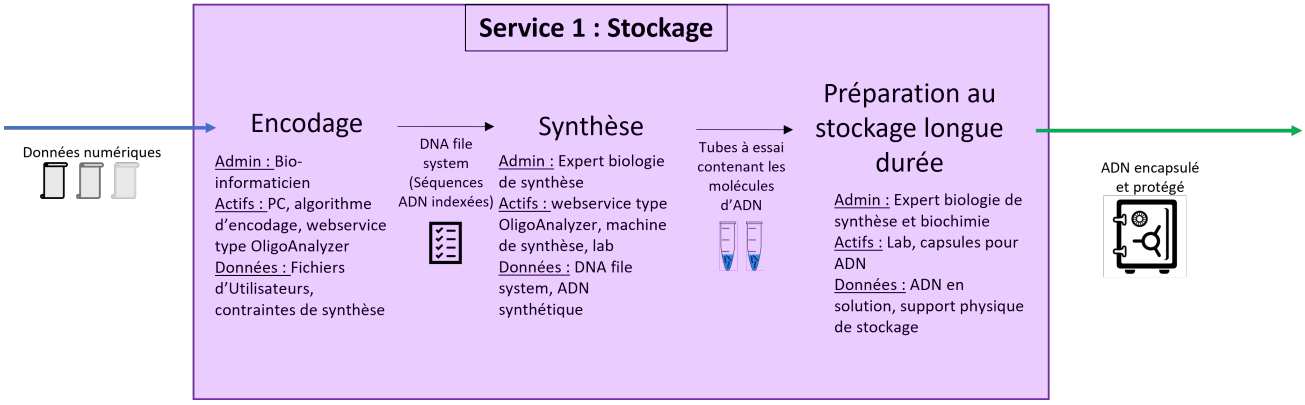



FIGURE 1.11 – Service 1 : Préparation au stockage de données dans des molécules d'ADN

Ce Service repose sur 3 étapes permettant l'encodage, la synthèse des molécules d'ADN et la préparation à leur archivage. Chaque étape a ses propres actifs, manipule et génère des données, et est menée par un Employé avec l'expertise nécessaire.

La première étape a pour objectif l'organisation et l'encodage des données numériques, dans le but d'obtenir un DNA file system, *i.e.* système de fichiers avec séquences ADN indexées et encadrées d'amorces. Cette indexation consiste à faire, par exemple, le lien entre les utilisateurs du Client et leurs fichiers avec les amorces des molécules d'ADN associées. Les fichiers considérés comme des flux binaires sont convertis en bases ADN. Cette conversion tient compte d'un ensemble de contraintes structurelles pour garantir l'efficacité de la synthèse. La vérification de la viabilité des séquences ADN à produire repose sur un algorithme privé ou public tel que OligoAnalyzer d'IDT ou GeneArt Assistant de ThermoFisher. L'Employé est donc un bio-informaticien, qui utilise plusieurs algorithmes d'encodage et de vérification, des logiciels en local ou des webservices. En sortie, l'Employé dispose d'un système de fichiers contenant les séquences ADN indexées prêtes à être synthétisées, ainsi que le système d'indexation de l'information. Lors de ce procédé, il a donc eu accès à toutes les données transmise par le Client : les fichiers des utilisateurs et les informations d'indexation.

La deuxième étape est la synthèse des séquences ADN. Elle prend en entrée un système de fichiers contenant des séquences ADN, et se charge de créer de l'ADN synthétique selon ces séquences. L'Employé est un expert en biologie de synthèse avec une maîtrise des technologies de manipulation de l'ADN. Il connaît le fonctionnement du synthétiseur ADN, des procédés de conservation de l'ADN. L'Employé effectue un séquençage de vérification sur un échantillon pour vérifier la qualité de la synthèse. Il a donc accès à toutes les données confiées par le Client. En sortie, l'ADN synthétique se trouve dans un tube à essai ou des plaques avec de nombreux puits, avec typiquement quelques $\mu L$ de solution par puits ou tube.

La troisième étape est la préparation au stockage longue durée. Pour cela, les molécules peuvent être encapsulées dans des capsules chimiques ou métalliques (stockage *in vitro*) ou transférées dans des

plasmides (stockage *in vivo*), *cf.* Section 1.1.4. En ce qui nous concerne, nous partons du principe que l'ADN sera conservé *in vitro*, car c'est la méthode la plus largement utilisée et posant le moins de questions éthiques, *cf.* Section 1.2.1.1. L'Employé, expert en biochimie et biologie de synthèse, reçoit les molécules d'ADN réfrigérées en solution, dans des tubes ou des puits. Tout d'abord, les fragments sont desséchés et isolés de l'oxygène et de l'humidité, puis encapsulés. Puis, l'Employé organise le stockage dans plusieurs capsules, avec un stockage équimolaire, *i.e.* une quantité équivalente de chaque molécule d'ADN différente. Ainsi les données stockées dans l'ADN sont non seulement indexées biologiquement mais aussi organisées physiquement, dans plusieurs capsules et espaces physiques de stockage. Une fois ce Service effectué, l'ADN encapsulé et protégé est envoyé au Prestataire 2, et les instructions de lecture (système de fichiers ADN, instructions de déprotection de l'ADN, etc) sont envoyées au Client.

**Service 2 : Archivage des données conservées dans des molécules d'ADN** Le Service 2 présenté en Figure 1.12 est l'archivage des données. Il a pour but la conservation des supports physiques contenant les molécules d'ADN pour une très longue durée, plusieurs décennies. De tels supports permettent généralement la conservation à température ambiante, il n'y a donc pas d'exigence particulière, excepté un milieu stable et sécurisé. Lorsque le Prestataire 2 reçoit une requête de la part du Prestataire 3, il lui transmet les supports de stockage demandés. Un Employé est ici en charge de répondre aux requêtes pour stocker ou récupérer des capsules, et de la maintenance de l'espace de stockage. L'Employé n'a pas besoin d'être un expert en biologie moléculaire, il manipule des supports de stockage mais pas d'ADN.

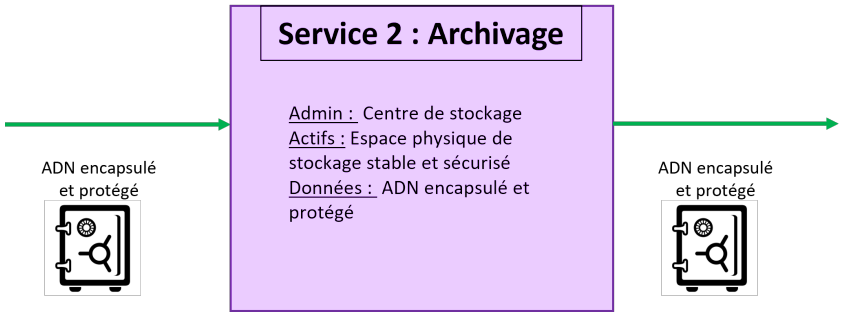



FIGURE 1.12 – Service 2 : Archivage des molécules d'ADN

**Service 3 : Récupération des données stockées dans l'ADN** Le Service 3 est la récupération des données stockées dans des molécules ADN. Un Client envoie une requête et des informations d'indexation au Prestataire, afin de récupérer les données de certain de ses utilisateurs. Le Prestataire contacte alors le Fournisseur du Service d'Archivage d'ADN pour récupérer les supports physiques aux index donnés.

La première étape est la lecture des molécules d'ADN ciblées. L'Employé reçoit les molécules d'ADN dans leur support de stockage, et utilise les informations fournies par le Client pour désencapsuler l'ADN, le préparer et le lire. Il est donc expert en biologie moléculaire. Pour effectuer ces tâches, l'Employé dispose des informations d'indexation et de séquençage (amorces, paramètres de PCR et de lecture). Cette étape permet d'obtenir des fichiers numériques contenant les lectures de chaque molécule d'ADN séquencée.

La deuxième étape est bio-informatique, c'est le décodage numérique des données lues par séquençage. L'Employé reçoit les résultats du basecalling, et utilise les données d'indexation (amorces et index) et de

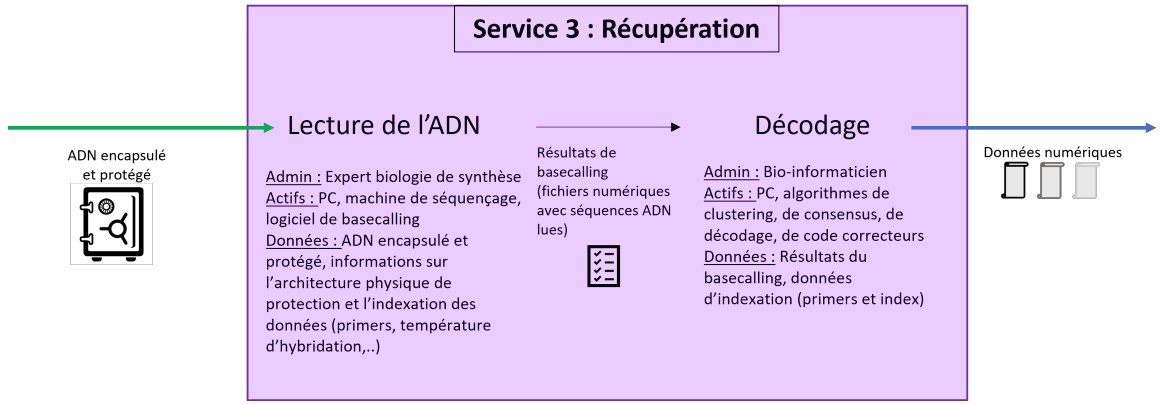



FIGURE 1.13 – Service 3 : Récupération de données stockées dans des molécules d'ADN

décodage fournies par le Client. Ainsi, il retrouve les séquences ADN lues et peut décoder les versions binaires des fichiers des utilisateurs. A cette étape, il peut éventuellement utiliser des algorithmes de consensus et des codes correcteurs. Les actifs sont ici des algorithmes bio-informatiques de traitement de séquences ADN, de données binaires, de correction d'erreurs, etc. En sortie, l'Employé a fournit au Client les fichiers numériques auxquels il souhaitait accéder.

Nous avons donc défini le scénario d'usage que nous envisageons, avec un cloisonnement entre 3 Prestataires qui effectuent la préparation au stockage, l'archivage, et la récupération des données. Pour cela, le Client transmet les données à stocker à un Prestataire, qui les prépare et les envoie à un autre Prestataire pour archivage. Le Client dispose des instructions de récupération des données stockées. Lorsqu'il souhaite avoir accès à ses données, il les transmet à un troisième Prestataire. Les Prestataires se transmettent donc les données à stocker selon les besoins du Client, qui dispose lui des instructions de lecture et d'indexation des données. C'est sur la base de ce scénario de stockage des données d'un Client dans l'ADN à l'aide de plusieurs Prestataires que nous allons identifier les menaces en matière de sécurité.

### 1.2.3 Identification des menaces

Les menaces qui pèsent sur notre système peuvent se manifester sous forme accidentelle ou malveillante, et elles sont souvent le résultat des actions de deux types d'attaquants : les profils passifs et actifs. Ces attaquants peuvent adopter différents modes d'action pour compromettre la sécurité de notre système. Après avoir défini les différentes natures des menaces et les profils d'attaquants possibles dans la première partie, nous passons en revue de manière détaillée les sources de risque concrètes associées à notre scénario de stockage de données, ainsi que leurs motivations et leurs objectifs.

#### 1.2.3.1 Évènements redoutés et profils d'attaquants

Un événement redouté est associé à une composante du système et porte atteinte à un objectif de sécurité de cette composante. Par exemple, l'indisponibilité d'un logiciel d'encodage porte atteinte à l'objectif de sécurité "Disponibilité" du service de stockage de données dans l'ADN. Les évènements redoutés sont de plusieurs types :

— Les accidents qui peuvent résulter de dysfonctionnements ou de négligences. Un exemple est la destruction de matériel par un phénomène naturel, tel un tremblement de terre.

— Les erreurs qui surviennent par exemple lors de transmission d'information ou de manipulation.

— Les malveillances de tiers ou de personnels de l'entreprise. Elles sont nombreuses ; il peut s'agir d'attaques logicielles ou logiques(hacking, malware), de sabotage physique, de vol de matériels, menaces internes, tout comme de l'envoi de mails volontairement non chiffrés par un employé malveillant pour causer une fuite de données.

Dans le cas d'attaques malveillantes, on peut considérer deux profils d'attaquants :

— Passif. Un attaquant passif ne fait qu'observer les canaux de transmission et espionner des messages. Il n'interfère pas dans le fonctionnement du système.

— Actif. Un attaquant actif va non seulement observer le système mais aussi interagir avec lui. Un tel attaquant peut mener une grande variété d'attaques : modifications, interceptions, interruptions, générations de nouveaux messages et données. La menace est donc d'autant plus grave.

On peut également discriminer trois principaux modes d'actions pour un attaquant :

— La voie informatique : un attaquant passif écoute un canal non sécurisé, tandis qu'un attaquant actif détourne ou parasite le trafic, utilise des logiciels malveillants pour saboter un système.

— La voie cognitive : un attaquant influence un utilisateur du Système d'Information, par le biais de manipulations, de pressions. Le modèle MICE (Money, Ideology/Interest, Coercion, Ego) mis au point par les services de renseignement britanniques permet d'identifier les moyens de pression sur un individu, avec des leviers tels que l'argent, les convictions politiques ou religieuses, la coercition ou le chantage, et l'ego.

— La voie physique : un attaquant mène des actions concrètes sur du matériel ou une infrastructure, telles que le vol de données physiques, le sabotage de matériel ou encore la manipulation physique d'un serveur piège. Il commet ainsi des délits d'effraction ou de destruction.

Un Système d'Informations est donc menacé par plusieurs évènements, qu'ils soient accidentels ou volontaires. Dans ce dernier cas, un attaquant peut agir de façon passive ou active sur le système, empruntant plusieurs voies : cognitive, physique ou informatique. Dans la section suivante, nous avons pris en compte ces aspects pour définir les motivations possibles d'un attaquant dans le cadre de notre Système d'Informations, pour le stockage de données dans l'ADN.

#### 1.2.3.2 Motivations d'un attaquant et stockage de données sur molécules d'ADN

On peut différencier différentes sources de risque selon leur objectif, la motivation de l'attaquant et leur pertinence. Celle-ci est évaluée en fonction de la probabilité de concrétisation du risque et de son impact. Un attaquant ne devient une menace que lorsqu'il a une motivation, on dit alors que c'est une source de risque [89].

Nous donnons dans le Tableau 1.1 des exemples de sources de risque que nous avons identifié dans le cadre du stockage de données sur molécules d'ADN, en nous inspirant de la classification de la méthode EBIOS [89]. Une première source de risque est liée à la concurrence entre les entreprises développant

des machines de synthèse, de séquençage, des supports de stockage. En effet, le domaine est en plein développement et en cours de normalisation. Cela ouvre la porte à des pratiques déloyales entre entreprises concurrentes pour obtenir un avantage stratégique. On parle ici d'espionnage industriel afin d'obtenir des informations sur les technologies R & D, voire d'attaques d'entrave au fonctionnement ou de sabotage. Une autre source de risque se réfère aux états qui pourraient avoir les mêmes objectifs dans un but de souveraineté de leur recherche nationale. Les différences entre ces deux sources sont liées aux moyens techniques et humains et ainsi à l'expertise alors disponible. Une autre source de risque possible est l'activisme. Le stockage de données sur ADN soulevant des questions de bioéthique (voir Section 1.2.1.1), des groupes extrémistes pourraient s'attaquer à des espaces d'archivage. La synthèse chimique, étant un processus très polluant, peut être une cause d'inquiétude pour certains ou un argument justifiant des actions malveillantes (cyberattaque, destruction physique). Une source de risque peut toujours être interne. Elle peut provenir d'un employé ou futur ex-employé avec des buts très variés comme le sabotage par esprit de vengeance ou à des fins lucratives. Une forme de sabotage est de porter atteinte à l'intégrité des données stockées, empêchant leur relecture. Prenons l'exemple des dossiers médicaux : ces altérations peuvent empêcher une relecture des données, ou introduire des erreurs ayant pour conséquences des confusions entre les dossiers de patients. Ces données de par leur caractère privé constituent des cibles privilégiées pour la revente (*e.g.* données hospitalières de VIP) ou pour le chantage (*e.g.* divulgation de pathologie grave). L'ADN comme support d'information est aussi particulièrement utile pour les données sensibles de petite taille auquel l'accès est rarement requis. Cela en fait un support de choix pour des cibles d'importance telles que les clefs cryptographiques ou les porte-monnaie de cryptomonnaie. A noter que ces attaques profitent des connaissances de l'employé et de ses accès normalement autorisés. Elles peuvent en conséquences être très efficaces avec des conséquences lourdes. Une autre source de risque qui a un sens selon nous est ce que l'on pourrait désigner comme le cyber-bioterrorisme. Selon INTERPOL [90], le bioterrorisme désigne la dissémination délibérée d'agents biologiques ou de toxines en vue de nuire ou provoquer le décès d'êtres vivants, dans l'intention d'intimider un gouvernement ou une population civile ou de les contraindre à servir des objectifs politiques ou sociaux. Ici, un de ses objectifs est par exemple l'altération de la composition de séquences ADN à synthétiser à des fins de bioterrorisme avec la création de virus. Cette attaque repose alors sur la corruption de fichiers ou l'altération d'algorithmes, d'où la notion de cyber-bioterrorisme. Enfin, une dernière source de risque est le Client lui-même, qui peut envoyer des données ou des instructions erronées. Le but du Client est alors de saboter le travail des Prestataires. Par la suite, nous faisons l'hypothèse que le Client est honnête ; nous ne prenons pas en compte cette menace.

La définition des sources de risque est généralement suivie d'une analyse des menaces à y associer, que nous effectuons dans la section suivante avec la méthode STRIDE. Les menaces sont les actions par lesquelles les sources de risque peuvent nuire à un élément du Système d'Informations. Nous nous intéressons en particulier aux menaces intentionnelles et ciblées.

D'autres modèles d'analyses de risques telles que MEHARI [91] incluent typiquement une étape d'évaluation du risque afin de déterminer sa gravité. Pour cela, deux indicateurs permettent la quantification du risque. Premièrement, la potentialité traduit la capacité que l'évènement se produise. Elle dépend de paramètres tels que la la probabilité et les conditions de survenance, la motivation et l'impunité de l'attaquant, etc. Le deuxième indicateur est l'impact du risque, c'est-à-dire l'importance des conséquences

TABLE 1.1 – Sources de risque (SR) pour le stockage de données dans l'ADN. Chaque SR a son objectif visé, ainsi qu'une évaluation de son degré de motivation et des ressources disponibles. En fonction de ces paramètres, une estimation de la pertinence de la menace est donnée.

| Source de risque | Objectif visé | Motivation | Ressources | Pertinence |
|---|---|---|---|---|
| Concurrent | Obtenir un avantage stratégique | +++ | +++ | ++++ |
| Service étatique | Atteindre la souveraineté de leur recherche internationale | +++ | ++++ | ++++ |
| Activiste | Saboter toute utilisation d'ADN comme support d'information | ++ | ++ | +++ |
| (Ex-) Employé | Saboter par vengeance, Gagner de l'argent | +++ | ++ | +++ |
| Cyber-bioterroriste | Altérer la composition des molécules à des fins de bioterrorisme | +++ | + | ++ |
| Client | Saboter le stockage | + | ++ | + |

de l'évènement. Il est mesuré par l'expansibilité des conséquences, le niveau de préparation aux situations de crises, etc. Cette quantification permet d'évaluer la gravité du risque, de déterminer son urgence, et de mettre en place des mesures adaptées de réduction des risques.

Après avoir défini les sources de risque et leurs objectifs, nous pouvons maintenant y associer les menaces qui pèsent sur la chaîne de stockage avec la méthode STRIDE. Une menace est une expression du risque que présentent les sources de risque pour la sécurité du système.

## 1.3 Étude des scénarios de menace : analyse STRIDE de la chaîne par étapes

Nous avons conduit cette analyse sur la base de la méthode STRIDE [92], du fait de sa simplicité. Nous présentons ci-après les différents types de menaces que celle-ci considère avant de lister les scénarios de menaces (risques) à chaque étape de la chaîne de stockage. De telles menaces sont les expressions du risque que présentent les sources de risque, présentées dans la section précédente. Il est important de noter que nous nous sommes intéressés aux menaces spécifiques associées à la chaîne de stockage de données sur molécules d'ADN et non aux systèmes d'information des prestataires de services, pour lesquels les menaces en cybersécurité sont déjà bien documentées (*e.g.* attaques réseau, attaques DDOS ("Distributed Denial of Service"), les rançongiciels et logiciels malveillants, etc) , et avec lesquelles les entreprises du domaine sont normalement en conformité (voir Section 1.2.1.1). Nous partons aussi du principe que les données sont transmises de façon sécurisée, par un canal sûr, entre le Client et chaque Prestataire. Pour cela, les données sont chiffrées et des échanges de clefs de chiffrement sont effectués pour transmettre les données de façon sécurisée et privée entre deux acteurs.

### 1.3.1 Les menaces du modèle STRIDE

Comme présenté dans le Tableau 1.2, chaque lettre de STRIDE correspond à un type de menace, et porte donc sur une propriété de sécurité spécifique.

TABLE 1.2 – Définition de la méthode STRIDE : Chaque lettre de STRIDE est associée à une menace, qui concerne une propriété de sécurité.

| | **Menace** | **Propriété de sécurité concernée** |
|---|---|---|
| S | Spoofing (Usurpation d'identité) | Authentification |
| T | Tampering (Altération) | Intégrité |
| R | Répudiation | Traçabilité (Non-répudiation) |
| I | Divulgation d'Information | Confidentialité |
| D | Déni de service | Disponibilité |
| E | Élévation de privilège | Traçabilité (Autorisation) |

— Le Spoofing correspond à l'usurpation d'identité et permet à un attaquant l'accès à des informations confidentielles en se faisant passer pour une autre entité. Cette menace est souvent causée par une attaque par voie cognitive, en manipulant des utilisateurs peu méfiants pour obtenir des informations telles que des identifiants de connexion et des mots de passe. Ce type d'attaque affecte donc l'objectif de sécurité d'authentification.

— Le Tampering ou Altération comprend différents types d'altérations, pouvant concerner toutes les données, les paramètres et les processus du système. Que celles-ci soient accidentelles ou malveillantes, elles pèsent sur l'intégrité des données.

— La Répudiation, c'est-à-dire le fait qu'une entité nie faire quelque chose dans le système. Les attaques par répudiation tirent parti de l'absence de contrôles ou de surveillance et modifient le système. C'est alors la propriété de traçabilité qui est concernée.

— La divulgation d'Informations correspond à toute transmission d'information provenant du système à une entité qui n'est pas autorisée à la recevoir. Cette menace impacte la confidentialité, qu'elle soit accidentelle ou bien malveillante. Dans le dernier cas, il peut s'agir d'espionnage comme une écoute simple par un attaquant passif, ou l'exploitation de faiblesses par un attaquant actif, avec notamment des analyses (*e.g.* des protocoles de communication, des logiciels, etc) utilisés permettant de connaître le fonctionnement interne du système.

— Le Déni de service consiste en l'épuisement des ressources nécessaires pour fournir un service, agissant ainsi sur la disponibilité. Ces attaques empêchent les utilisateurs légitimes d'utiliser le système.

— L'Élévation de privilège est l'obtention par une entité d'un niveau supérieur d'autorisation, auquel elle n'a normalement pas accès. Cela lui permet de réaliser librement des actions qui lui seraient inaccessibles autrement. Par exemple, un simple utilisateur réussit à obtenir des privilèges d'administrateur, via un logiciel malveillant ou la voie cognitive. La propriété de sécurité atteinte est la traçabilité, et en particulier l'autorisation.

### 1.3.2 Menace à toutes les étapes

Parmi toutes les menaces qui portent sur la chaîne de stockage, deux d'entre elles concernent toutes les étapes de cette dernière dans notre scénario d'usage.

**Divulgation d'information** L'objectif est la fuite de données principalement pour nuire aux utilisateurs ou au Client, voire simplement faire des bénéfices. Elle peut s'exprimer sous de nombreuses formes. De manière générale, cette attaque concerne toutes les étapes de la chaîne, de la synthèse au décodage en passant par l'archivage, sans oublier le Client lui-même. Du côté du codage et de la synthèse, une attaque consiste pour un Employé du Prestataire à mêler les données d'un utilisateur à celles d'un autre, en changeant les index numériques ou les amorces biologiques structurant les données. Une autre attaque consiste à faire une copie des données numériques, avant l'encodage ou avant la synthèse. Pour cela, un attaquant peut être un service étatique très motivé à récupérer les données qui déploie des ressources importantes pour faire pression sur l'Employé. Au niveau de l'archivage, un Employé peut faire une copie des molécules d'ADN conservées en zone de stockage, ou même voler un support de stockage. Notons que contrairement à des problématiques de sécurité de systèmes d'information, les données ne sont pas stockées en ligne. La seule vulnérabilité de l'archivage est ici l'accès physique au lieu d'entrepôt. Toutefois, étant donnée la concentration des supports de stockage de l'ordre de $10^{18}$ octets par gramme d'ADN [6], l'attaquant peut causer des dégâts catastrophiques et donc des pertes conséquentes d'argent pour le Prestataire. Au décodage, les informations récupérées peuvent être transmises à un tiers illégalement. Enfin, un employé du Client peut lui aussi tenter de faire fuiter des données, en jouant par exemple sur les droits des utilisateurs sur les données envoyées pour stockage.

**Élévation de privilège** L'objectif principal est de causer des dommages dans le système de l'intérieur pour nuire aux acteurs du système. Un attaquant peut élever ses privilèges de différentes manières, par exemple en influençant un Employé autorisé (attaque par voie cognitive), ou bien en profitant d'une faiblesse dans les contrôles d'accès. Les activités malveillantes qu'un attaquant peut alors effectuer dans le système sont diverses, et elles peuvent concerner toutes les étapes de la chaîne de stockage. Avec un accès aux machines (PC, synthétiseur ADN, de séquençage, etc), un attaquant peut installer des logiciels malveillants, compromettre des données numériques ou modifier des configurations telles que les paramètres d'encodage ou de synthèse. Un accès physique aux laboratoires ou zones de stockage permet à l'attaquant de compromettre les données biologiques aux étapes de synthèse, d'archivage, et de séquençage. Grâce à son haut niveau d'autorisation dans le système, l'attaquant peut empêcher la traçabilité de ses actions malveillantes. Lors de la récupération des données stockées dans l'ADN, l'Employé reçoit des requêtes de la part du Prestataire qui va lire l'ADN. Il peut profiter de son accès et confier des données non attendues, dans un but de sabotage. En particulier, lorsque l'Employé répond à plusieurs requêtes de Prestataires et leur donne une fraction des copies d'une molécule d'ADN chacun, l'origine de ces copies n'est pas traçable.

### 1.3.3 Encodage

Dans notre scénario, l'encodage est effectué par un Employé du Prestataire qui fait le stockage des données numériques du Client dans de l'ADN. L'Employé reçoit les données du Client ainsi que des contraintes structurelles sur les séquences ADN. Il organise les données numériques pour les préparer à la synthèse, vérifie leur viabilité et les transmet à un autre Employé.

**Altération** L'objectif est le sabotage de l'encodage, qui peut être effectué par l'Employé lui-même, dans le but de nuire au Client ou à l'image de marque et le relation de confiance du Prestataire avec ses Clients. L'Employé peut altérer l'encodage (paramètres d'encodage, d'indexation, ajout de séquences ADN parasites), similairement à une attaque par injection de fautes. Il profite de ses accès privilégiés. Comme l'ADN est un stockage prévu pour le long terme et les données ne seront lues qu'en cas de besoin, cette attaque peut être détectée très tardivement.

### 1.3.4 Synthèse

A cette étape, un Employé du Prestataire chargé du stockage dans l'ADN effectue la synthèse. Il reçoit les fichiers numériques contenant les séquences ADN ainsi que les amorces à utiliser, vérifie la viabilité des séquences, et utilise le synthétiseur ADN. En sortie, il obtient les molécules d'ADN souhaitées en solution.

**Altération** L'Employé peut ajouter des molécules d'ADN imprévues ou altérer la composition des molécules à des fins de bioterrorisme. Dans ce cas, les molécules d'ADN contiennent des motifs conduisant à la production de virus. Le virus peut alors être activé dans le cas de stockage *in vivo*, ou lors de certaines manipulations pour récupérer les données. L'attaquant profite du fait que la synthèse d'ADN produit une quantité variable de molécules. Ainsi, l'ajout de quelques molécules d'ADN dans une solution peut passer inaperçu. Les conséquences ne seront ressenties que lors de la tentative de récupération des données stockées. Pour saboter la récupération des données, l'Employé peut aussi ajouter des molécules d'ADN aléatoires associées à des amorces qui existent déjà. Ainsi, il sera plus difficile voire impossible de retrouver les données mélangées à des leurres. Notons que cette altération peut être effectuée à plusieurs étapes de la chaîne de stockage. Enfin, un Employé vengeur peut profiter du fait que l'ADN n'est pas encore encapsulé, devant donc être conservé dans un congélateur. Il peut couper l'alimentation électrique, laisser la porte ouverte, sortir l'ADN du congélateur, etc, pour le détruire globalement ou partiellement.

**Répudiation** L'attaquant profite ici de l'incertitude des étapes biologiques pour modifier l'ADN de quelques bases par rapport aux séquences qu'il est sensé synthétiser. Les erreurs de séquençage sont fréquentes, il peut donc nier avoir modifié les données. Ce sabotage est donc une entreprise à long terme, elle fait perdre les données sensibles à l'utilisateur, et celui-ci ne s'en rende compte que lorsqu'il souhaite y accéder.

**Divulgation d'information - Espionnage** Une source de risque est un service étatique ou un concurrent qui vole des données stratégiques en espionnant le processus de synthèse. Pour cela, il effectue une

analyse de modèle, qui permet d'associer un modèle de fonctionnement (bruit, courant) à des données. Par exemple, une attaque possible [93] est le placement d'un smartphone à proximité d'un synthétiseur ADN, associant un ensemble de caractéristiques acoustiques de la machine à la synthèse d'un nucléotide. Cela est possible grâce à un algorithme de Machine Learning préalablement entraîné sur une machine identique. Cette attaque passive menace la confidentialité des données avant même qu'elles soient stockées dans l'ADN.

**Déni de service** Un attaquant extérieur peut utiliser la lenteur du synthétiseur ADN (plusieurs secondes par nucléotides [93]) pour effectuer une version biologique de l'attaque informatique classique Ddos ("Distributed denial of service"). Le principe est d'envoyer un grand nombre de requête de synthèse à la machine, jusqu'au dépassement des limites du système et à la mise hors service. Cela entrave le bon fonctionnement de la chaîne de stockage, et ainsi porte atteinte à l'image de marque du Prestataire.

### 1.3.5 Préparation au stockage longue durée

Le troisième Employé du Prestataire de stockage prépare les molécules d'ADN pour le stockage longue durée. Il reçoit l'ADN en solution et fait l'encapsulation des molécules et l'organisation physique des supports de stockage. L'Employé transmet ensuite ce support au Prestataire qui le conserve en sécurité.

**Altération** L'objectif est de porter atteinte à l'intégrité des données biologiques. Pour cela, comme à l'étape de synthèse, l'Employé peut ajouter de l'ADN dans un but de bioterrorisme, ou bien laisser volontairement l'ADN vulnérable à des facteurs extérieurs (eau, oxygène, température). Cette atteinte à l'intégrité de l'ADN ne sera découverte qu'à la lecture, ayant ainsi des conséquences à long terme.

**Répudiation** L'objectif d'un attaquant est ici de nier la responsabilité de ses actions. Pour cela, l'Employé peut prélever un échantillon d'ADN puis effectuer une amplification par PCR (*cf.* Figure 1.5) pour compenser ce prélèvement. Ainsi, la quantité d'ADN à stocker n'est pas impactée, et l'attaquant peut nier tout vol. De plus, le contenu exact d'un support ne peut être connu qu'avec un séquençage, opération qui nécessite plusieurs étapes biologiques avec un rendement variable. Un attaquant peut donc nier avoir prélevé de l'ADN en soulignant l'aléa apporté par les étapes biologiques.

### 1.3.6 Archivage

Lors de cette étape de notre scénario, les fragments d'ADN sont stockés dans un espace physique dédié. L'Employé reçoit les supports de stockage, fait l'archivage en lieu sûr, et répond aux requêtes venant de Prestataires qui font la récupération de données.

**Altération** Tout d'abord, un attaquant peut modifier les conditions de stockage ambiantes qui affectent l'ADN telles que : l'humidité, la lumière, et la température dans l'espace de stockage. Ces altérations peuvent être faites sans accès direct aux molécules. Une attaque efficace [19] sur les capsules de stockage DNA Shells (*cf.* Section 1.1.4) est l'augmentation brutale de la température du lieu de stockage. Cette attaque n'a pas pour but d'être discrète, mais d'avoir un effet de destruction rapide sur l'ADN. Au

contraire, une attaque qui n'est remarquée que lors de la récupération des données est le sabotage biologique ou chimique. Si l'attaquant accède directement aux molécules, il peut effectuer des modifications chimiques ou biologiques : insertion d'acide, d'enzymes, etc. Le but est alors de modifier la structure de l'ADN pour empêcher la récupération des données. D'autres attaques consistent à ajouter de grandes quantités d'ADN dans une solution pour noyer et rendre illisibles les données stockant de l'information. En particulier, un attaquant peut ajouter de l'ADN leurre avec les mêmes amorces que l'ADN stockant de l'information. Ainsi, il altère le contenu d'une solution et fait obstacle au bon séquençage de l'ADN stockant de l'information. Ces sabotages ont un effet à long terme car ils ne sont détectés que lorsque l'utilisateur demande l'accès à ses données. L'Employé en charge de l'archivage peut aussi intervertir certains supports de stockage de différents clients, ou bien altérer les informations d'indexation. Enfin, des altérations peuvent se produire naturellement dans les molécules d'ADN. En effet, elles peuvent muter dans le temps, entraînant des changements de base ou des délétions. En particulier, si celles-sont conservées *in vivo*, certaines séquences se traduisent en protéines potentiellement dangereuses. Si ces changements sont trop nombreux, ils nuisent à la disponibilité des données. Le risque provient alors d'un mauvais choix de conservation d'ADN dans le temps.

**Répudiation** Comme à l'étape de synthèse, l'Employé peut extraire l'ADN de son support et prélever un petit volume d'ADN. Pour compenser ce prélèvement, il peut effectuer une amplification par PCR. Ainsi, il n'a pas impacté la quantité d'ADN stockée et peut nier tout vol. De plus, même si le contenu d'une solution n'est pas exactement celui attendu, cela peut être expliqué par les rendements variables des étapes biologiques. Cette attaque nécessite du temps et des moyens afin d'extraire l'ADN de son support physique et de le manipuler.

**Divulgation d'information** Pour assurer leur confidentialité, les données stockées dans des molécules d'ADN sont généralement chiffrées. Toutefois, le stockage dans l'ADN étant une technologie destinée à l'archivage à long terme, il se peut que l'algorithme de chiffrement choisi devienne obsolète (sécurité cassée, taille de clef trop petite, etc). Cela présente alors une faille de sécurité, puisqu'il n'y a pas besoin de connaître la clef pour déchiffrer les données.

**Déni de service** L'objectif du déni de service est la perturbation de l'accès aux services du système. Un attaquant peut causer des dégâts importants en volant un petit support de stockage d'ADN, étant donnée la concentration de l'ADN. Cette attaque se fait par la voie physique, par intrusion dans les lieux du stockage, par exemple par effraction. L'attaquant compromet immédiatement la disponibilité des données.

Une autre menace est le vol des informations d'indexation et de séquençage (amorces, paramètres de PCR et de lecture), ce qui cause une interruption de service car les données stockées dans l'ADN ne sont plus récupérables.

### 1.3.7 Séquençage d'ADN

L'Employé du Prestataire de récupération des données utilise les instructions données par le Client et fait une requête au Prestataire en charge de l'archivage. En raison de la grande densité de l'ADN, même

si l'Employé ne reçoit que quelques grammes d'ADN, les données de multiples utilisateurs sont présentes. Une fois les supports de stockage récupérés, il prépare l'ADN et le séquence, permettant de passer du domaine biologique au domaine numérique. Après séquençage, il transmet les fichiers contenant toutes les séquences ADN lues à l'Employé chargé du décodage numérique.

**Altération** Comme à l'étape d'archivage, l'Employé peut altérer les données en ajoutant beaucoup de molécules d'ADN à la solution contenant l'ADN stockant l'information. Si les molécules ont les mêmes amorces, elles peuvent dégrader la qualité du séquençage et ajouter des erreurs dans les séquences lues, menaçant ainsi l'intégrité des données.

D'autre part, comme lors de la synthèse, l'ADN est vulnérable aux facteurs environnementaux après désencapsulation. Un Employé peut donc profiter de cette étape pour dégrader volontairement l'ADN réfrigéré en modifiant sa température. Cela pourrait aussi être un simple accident.

**Répudiation** L'Employé peut modifier des fichiers numériques contenant les lectures de l'ADN, afin d'empêcher la récupération des données, et nier l'avoir fait. Il peut se cacher derrière le fait que cette modification est imputable aux imperfections des processus biologiques impliqués. De plus, comme l'ADN lu est détruit, il est impossible de séquencer deux fois le même échantillon pour prouver la responsabilité de l'Employé. C'est un enjeu.

Un Employé corrompu peut voler et redistribuer une partie de l'ADN qu'il a reçu et nié l'avoir fait. En effet, une PCR est systématiquement faite avant le séquençage pour multiplier exponentiellement la quantité d'ADN. En conséquence, prélever une petite partie de l'ADN n'influe pas sur la qualité du séquençage. Et, comme le processus de séquençage détruit l'ADN, l'analyse des molécules d'ADN utilisées est difficile. La traçabilité des données est donc menacée en plus de leur confidentialité. Cette attaque requiert un niveau d'expertise avancé en biologie, l'Employé effectue un dosage pour vérifier que la densité d'ADN est suffisante pour retrouver l'information et subtilise le superflu. Notons que l'Employé peut aussi conserver une copie des fichiers contenant les résultats du séquençage.

**Divulgation d'information** Une vulnérabilité est le fait que les données de plusieurs utilisateurs sont séquencées en même temps. En conséquence, certaines séquences d'un utilisateur peuvent se retrouver sur le fichier d'un autre utilisateur. Cette erreur d'indexation est appelée "index hopping" (ou saut d'index), elle cause une fuite de données accidentelle. Un attaquant pourrait se servir de ce phénomène en faisant une requête de récupération de données à un moment stratégique, pour que ses données soient séquencées en même temps que les données ciblées. L'attaquant retrouvera alors des séquences qui l'intéressent mais qui ne lui appartiennent pas [94]. D'autre part, le séquençage en lui-même présente plusieurs failles de sécurité. Parmi elles, nous pouvons citer les vulnérabilités suivantes. La technologie du séquençage nanopore repose sur des analyses du courant, qui change selon les bases d'une molécule ADN. Un attaquant peut espionner ce processus et récupérer les variations de courant, et ainsi les données séquencées. D'autre part, afin d'améliorer son modèle nanopore et son logiciel de basecalling, un séquenceur nanopore transmet par défaut les résultats du séquençage au fabricant (ONT). Cela pose des problèmes de confidentialité des données. Enfin, la "flow cell", support du séquençage, est vulnérable à des attaques de récupération de données [95]. En effet, de l'ADN résiduel resté attaché dans la "flow cell" peut être récupéré et lu [95].

**Déni de service** Une source de risque est un Client qui souhaite empêcher un concurrent d'accéder à ses données. Pour cela, il peut envoyer de nombreuses requêtes prioritaires de récupération de données au Prestataire. Le séquençage étant un procédé qui prend plusieurs jours à la machine, il empêche ainsi d'autres Clients d'accéder à leurs données. Une autre menace repose sur le fait que l'Employé lit toutes les séquences d'ADN qu'il reçoit, faisant confiance au Prestataire qui lui envoie les données. Un Utilisateur ou Client peut tirer avantage de ceci, en générant un shellcode malveillant, qui sera traduit en séquence ADN et ensuite ajouté aux molécules stockées. Lorsque l'Employé reçoit ces molécules d'ADN, il les lit toutes. La lecture du shellcode décodé est une commande qui permet de prendre contrôle de l'ordinateur de l'Employé, comme démontré dans l'article [94]. L'attaque exploite donc le fait que le séquenceur à ADN ne sait pas ce qu'il séquence et de failles de sécurité des logiciels utilisés. L'impact de ce type d'attaque sur le long terme reste faible car les logiciels évoluent avec le temps. Cependant, il faut encore détecter ce type de faille. Néanmoins, il permet de mettre hors de fonction le séquenceur à ADN, et potentiellement d'avoir accès à des données sensibles de Clients et du Prestataire.

L'Employé étant un expert en biologie moléculaire, il connaît les étapes de manipulation d'ADN qui ont une forte incertitude et peuvent être sabotées. Il peut nuire à la disponibilité des données, en perdant du temps et en multipliant les manipulations. Par exemple, il peut effectuer une PCR avec les mauvaises amorces, saboter le séquençage en ajoutant de l'ADN, prétendre un problème matériel, etc. En conséquence, il perd plusieurs jours ou semaines et nuit à l'image de marque de son entreprise et prive provisoirement le Client des données demandées.

Une autre cible d'attaque permettant de menacer la disponibilité des données est le séquenceur à ADN. Le processus de séquençage introduit un pourcentage non négligeable d'erreurs, et la bonne récupération des données dépend des solutions chimiques, du logiciel de basecalling, etc. Tous ces paramètres doivent être optimisés afin de garantir la disponibilité et l'intégrité des données. En conséquence, une attaque sur les composants d'un séquenceur nanopore pourrait par exemple brouiller le signal électrique et donc la récupération des données. Leur disponibilité est menacée puisque le séquençage nanopore ne permet pas de lire une molécule plusieurs fois. Une erreur lors du processus de lecture entraîne donc la perte des molécules utilisées.

**Élévation de privilège** Une source de menace est un attaquant qui réussit à avoir un accès physique au lieu du séquençage. Prenons l'exemple du MinION, séquenceur nanopore de la taille du grosse clef USB. En s'introduisant dans le laboratoire pendant le séquençage, un attaquant peut discrètement en voler un. Notons que le MinION fonctionne généralement pendant 48 à 72 heures, durant lesquelles il lit et stocke les données de séquençage qu'il produit. Un attaquant a donc une fenêtre d'action de plusieurs jours pour le voler et récupérer des données confidentielles. L'attaquant porte aussi atteint à la disponibilité des données, puisque l'ADN est dans le séquenceur volé.

### 1.3.8 Décodage numérique

L'Employé est un bio-informaticien qui reçoit les fichiers contenant les résultats du séquençage d'un autre Employé, et les instructions de décodage et d'indexation du Client, et récupère les données stockées dans les séquences ADN.

**Altération** Tout comme à l'étape d'encodage, l'Employé peut falsifier les données d'un utilisateur d'un Client, via l'ajout, la suppression ou la modification d'informations. Il peut aussi altérer les paramètres d'encodage et d'indexation.

### 1.3.9 Vue d'ensemble des menaces

Dans les sections précédentes, nous avons détaillé les menaces pesant sur notre scénario d'usage (en Figure 1.10) de stockage de données d'un Client dans l'ADN à l'aide de plusieurs Prestataires. Le Tableau 1.3 présente les menaces classées selon la méthode STRIDE pour chaque service d'un Prestataire.

Notons que la menace d'usurpation d'identité n'est pas présente, car nous considérons que les échanges d'informations entre les acteurs du système se font de façon sécurisée après authentification.

## 1.4 Traitement du risque

Dans la section précédente, nous avons identifié des menaces qui pèsent sur la chaîne de stockage. Différents outils de sécurité peuvent être utilisés pour y répondre ou au moins réduire l'impact de ces risques. C'est l'objet de cette section.

### 1.4.1 Traçabilité

Lors de l'analyse de risques, nous avons vu que la traçabilité était menacée de plusieurs manières, soit pour des questions de fuites de données, soit pour des questions de non-répudiation des actions menées par un utilisateur.

Le watermarking [96] ou le tatouage numérique est une technique qui permet de marquer un document numérique avec des informations de vérification ou un copyright. Elle consiste à modifier quelques bits d'un document numérique afin d'y ajouter un message de vérification contenant, par exemple, le nom ou un identifiant du propriétaire ou de l'utilisateur du document. Dans le domaine du stockage de données dans l'ADN, cette question est ouverte car une molécule d'ADN peut être copiée exponentiellement à l'identique. Il existe différents travaux à ce sujet [97] [98] [99] [100], qui permettent de tatouer des organismes vivants sans modifier l'expression de leurs gènes. Le mécanisme d'expression des gènes permet de transcrire l'ADN (exprimé en séquences de bases A, C, G et T) en une suite d'acides aminés, qui forment une protéine. Chaque bloc de 3 bases appelé codon est traduit en un acide aminé. Plusieurs blocs de 3 bases différents peuvent correspondre au même acide aminé. Il est possible de prendre en compte cette particularité pour modifier certaines bases d'ADN sans changer la séquence d'acides aminés traduite. Par exemple dans [97], le tatouage consiste à modifier la troisième base de certains codons pour y encoder de l'information, sans modifier la traduction en acides aminés. Cela n'influence alors pas l'expression des gènes. Pour appliquer ces méthodes au stockage de données dans l'ADN, il faut encoder des données en des acides aminés. Il est alors possible de tatouer ces données sans les altérer.

Toutefois, à notre connaissance, il n'existe actuellement pas de technique permettant d'effectuer le tatouage avec des opérateurs biologiques, c'est-à-dire tatouer des molécules d'ADN en s'appuyant sur des processus chimiques ou biologiques. C'est une piste de recherche à explorer.

TABLE 1.3 – Vue d'ensemble des menaces pesant sur les étapes de notre scénario d'usage pour le stockage de données dans l'ADN. Les menaces sont classées selon la méthode STRIDE.

| STRIDE | Usurpation | Altération | Répudiation | Divulgation d'information | Déni de service | Elévation de privilège |
|---|---|---|---|---|---|---|
| A toute étape | | | | Mélange des index ou primers des données de plusieurs utilisateurs | | Vol d'une copie des données numériques ou biologiques |
| Encodage | | Sabotage de l'encodage par l'Employé | | Copie des données numériques par l'Employé | | |
| Synthèse | | Ajout de molécules d'ADN pouvant conduire à la création de virus (bioterrorisme) | Sabotage par modification indétectée de quelques bases | Espionnage accoustique de la machine de synthèse d'ADN | Mise hors service par Ddos avec de nombreuses reqûetes de synthèse d'ADN | |
| | | Sabotage par ajout d'ADN aléatoire associé à des primers existants | | | | |
| | | Sabotage de la conservation de l'ADN, non encapsulé et vulnérable aux changements de température | | | | |
| Préparation au stockage | | Ajout de molécules d'ADN pouvant conduire à la création de virus (bioterrorisme) | Vol indétectable d'un échantillon d'ADN par l'Employé | | | |
| | | Sabotage physique des capsules par l'Employé | | | | |
| Archivage | | Destruction d'ADN via les conditions externes de stockage (température, humidité,etc) | Vol indétectable d'un échantillon d'ADN | | Vol d'un actif via un accès physique : support de stockage d'ADN | Fuite de données à plusieurs Prestataires sans traçabilité des copies |
| | | Sabotage chimique ou biologique des molécules | | | | |
| | | Altération naturelle de l'ADN dans le temps | | | | |
| Séquençage | | Sabotage du processus de lecture par l'Employé | Sabotage par modification intraçable des résultats du séquençage | Récupération accidentelle de séquences d'un autre utilisateur lors d'un séquençage commun | Envoi de nombreuses requêtes pour empêcher un concurrent d'accéder à ses données | Vol d'un actif via un accès physique : séquenceur portable en fonction |
| | | Accident lors du séquençage causant la destruction d'ADN | Vol d'un échantillon d'ADN après PCR | Espionnage des données séquencées par une analyse de courant de la machine de séquençage | | |
| | | Sabotage de la conservation de l'ADN, non encapsulé et vulnérable aux changements de température | Copie du résultat du séquençage | Fuite de données par la transmission des données au fabricant nanopore. | Envoi d'ADN se traduisant en un shellcode malveillant qui paralyse le PC de l'Employé | |
| | | Attaque par faute sur les composants matériel du séquenceur pour brouiller le signal et donc les bases ADN lues | | | | |
| Décodage | | Sabotage par l'Employé | | Copie des données numériques par l'Employé | | |

La Gestion des Identités et des Accès (GIA) [101], aussi appelée IAM (pour "Identity and Access Management"), permet en cybersécurité de gérer les habilitations des utilisateurs d'un Système d'Information. Plus précisément, avec une GIA adaptée, les utilisateurs, *i.e.* les personnes, machines et composants logiciels, ont accès aux ressources appropriées au bon moment. Cela permet de contrer les attaques par élévation de privilège. En effet, tout utilisateur doit prouver son identité avant de pouvoir accéder ou utiliser des ressources. Une GIA permet de créer, stocker et gérer l'identité numérique des utilisateurs, et d'y associer des niveaux d'accès ou d'habilitation selon la confidentialité des ressources et le type d'utili-

sateur. L'authentification d'un utilisateur peut se faire par un guichet d'authentification unique ("Single Sign On") via un mot de passe par exemple, ou bien avec l'authentification multifacteur ("Multi-Factor Authentication") [102], ou authentification forte, afin d'augmenter la sécurité. Une GIA inclut aussi un volet de surveillance des actions effectuées dans le système par le biais de logs, fichiers qui enregistrent tous les événements et accès des utilisateurs. Cela permet de garantir le bon fonctionnement du système et d'évaluer les risques de sécurité en traçant les actions de chaque entité du Système d'Information, que ce soit au niveau des accès physiques que des accès numériques. Une GIA adaptée permet donc de gérer à la fois les accès informatiques aux données et aux logiciels du Système d'Information, mais aussi les accès aux machines de synthèse, de séquençage et au supports physiques de conservation de l'ADN encodé.

### 1.4.2 Intégrité

L'intégrité des données peut être menacée dans le domaine numérique comme biologique, par l'altération des processus d'encodage, la détérioration de l'ADN, l'ajout d'ADN parasite, ou le sabotage de l'indexation de l'information.

En informatique, l'intégrité des données est généralement certifiée/vérifiée grâce à des fonctions de hachage ou des codes d'authentification de message (MAC) [103]. Une fonction de hachage permet d'associer une valeur de petite taille fixe, une signature, à des données de taille quelconque. La signature obtenue est unique à chaque fichier. Ainsi, un mécanisme de vérification d'intégrité consiste à stocker un fichier avec sa signature. Lorsqu'un utilisateur récupère un fichier, il en recalcule la signature et vérifie qu'elle correspond à celle stockée. Si c'est le cas, c'est la preuve de l'intégrité du fichier. Ce mécanisme est sécurisé car les fonctions de hachage sont à sens unique : il est impossible d'obtenir des informations sur le fichier à partir de sa signature. Le MAC a un fonctionnement similaire aux fonctions de hachage, avec un mécanisme d'authentification supplémentaire : la signature du fichier est chiffrée avec une clef privée. Cela permet d'authentifier le message avec la clef publique et la signature ; sachant que la signature chiffrée avec une clef privée donnée ne peut être déchiffrée qu'avec la clef publique associée à cette clef privée. Le NIST ("National Institute of Standards and Technology") définit et donne des conseils d'utilisation d'algorithmes de cryptographie, notamment de chiffrement et d'authentification. Celui-ci conseille [104] d'utiliser les familles de fonctions de hachage SHA-2 [105] ou SHA-3 [106], ou bien un des 3 algorithmes de codes d'authentification de message (MAC) [107] : HMAC [108], KMAC [109] et CMAC [110]. Par exemple, le HMAC ( "Keyed-Hash MAC") est un protocole qui combine l'utilisation d'une fonction de hachage et d'une clef de chiffrement. Il est recommandé de l'utiliser avec la fonction SHA-256 [111], devenant alors HMAC-SHA-256. Le HMAC-SHA-256 génère un code d'authentification de message de 256 bits en effectuant plusieurs opérations, notamment des opérations de hachage, de XOR et de concaténation de la clef et du message. Ainsi, le hachage obtenu est résistant aux attaques par dictionnaire et par collision.

L'ajout de mécanismes de vérification d'intégrité lors des transmissions de données numériques peut donc permettre de rapidement reconnaître les données corrompues. Il est donc possible de stocker les données avec leur signature sur des molécules d'ADN pour en vérifier l'intégrité une fois ces informations récupérées.

Au niveau chimique ou biologique, il est toutefois difficile de vérifier l'intégrité des molécules d'ADN. En effet, la seule solution pour connaître le contenu précis de molécules d'ADN est le séquençage. Un

mécanisme de "sécurité" pour vérifier l'intégrité de données synthétisées est donc le séquençage de vérification. Le Prestataire qui synthétise l'ADN effectue un séquençage et vérifie que les données récupérées correspondent à celles qui lui ont été confiées. Il est possible d'utiliser à intervalles réguliers des méthodes de quantification de l'ADN, pour estimer la concentration d'ADN dans une solution, ou la taille moyenne des molécules d'ADN stockées. Ainsi, les résultats obtenus peuvent être comparés à ceux attendus. Une faible concentration d'ADN indiquera une détérioration des molécules, et des tailles de molécules d'ADN différentes de celles attendues signifiera que l'ADN a été modifié, possiblement pour accéder à l'information ou la détruire. Une mise en place d'observation de l'ADN stocké à intervalles régulier est donc possible, afin de vérifier l'intégrité des molécules. Un inconvénient notable est que cela nécessite d'extraire l'ADN de son support de stockage pour le manipuler, et ainsi de le dé-protéger des conditions environnantes. En conséquence, une perte d'une partie des molécules est à prévoir. Quoiqu'il en soit, vérifier l'intégrité des molécules sans revenir dans le monde numérique, est aussi un sujet de recherche ouvert.

Dans le cadre du stockage de données *in vivo*, il est important de garantir l'intégrité de l'ADN et la non-dangerosité des matériaux biologiques utilisés. Pour cela, la synthèse d'acides nucléiques codant des parties d'organismes pathogènes ou des protéines et des toxines nocives, ou des mutations modifiant les séquences ADN, doivent être évitées. Une méthode efficace pour éviter ces altérations est un scan rigoureux de chaque séquence ADN par les Prestataires produisant de l'ADN synthétique. Il est ainsi possible d'éviter la synthèse de séquences initiant la transcription ou la traduction de protéines, par exemple les codons d'initiation : les séquences $ATG$, $CTG$, $TTG$ et leurs inverses. Une étape de scan est déjà mise en place par la plupart des fournisseurs d'ADN synthétique, toutefois les bases de données de séquences problématiques sont souvent incomplètes [112]. Les Prestataires fournissant de l'ADN synthétique restent donc vulnérables à la production d'organismes dangereux.

### 1.4.3 Disponibilité

Dans notre chaîne de stockage, la disponibilité des données est menacée lorsque des attaques par déni de service surchargent les machines de synthèse et de séquençage, ou les logiciels et webservices chargés d'encoder et vérifier la viabilité de séquences ADN. Le vol des supports physique de stockage ou de séquençage, ou bien des informations d'indexation et de séquençage, est aussi un risque.

Afin de traiter ces risques, les problèmes matériels peuvent être minimisés avec des contrats de maintenance, des mises à jour régulières de logiciels pour corriger des failles, la formation des personnels, la prévention des risques naturels (tremblement de terre, incendie avec pour conséquences la détérioration de l'ADN), la mise en place de plans de récupération des données via des sauvegardes, etc. Il est possible d'adapter le Système d'Information pour prendre en compte une tolérance aux pannes, c'est-à-dire ajouter des composants redondants dans le système. Dans ce cas, les composants de secours prennent le relais en cas de problème. Ainsi, lors du séquençage des données demandées par un Utilisateur, deux séquenceurs peuvent travailler en parallèle sur des copies des mêmes données. En conséquence, une défaillance sur un séquenceur n'empêchera pas l'accès aux données. Il en est de même pour les données biologiques stockées, dont des copies peuvent être archivées dans plusieurs zones de stockage. Ainsi, un vol dans une zone de stockage n'impactera pas la disponibilité des données.

Les séquences ADN sont entachées d'erreur lors des procédés de lecture et d'écriture de l'ADN, ce qui menace la disponibilité des données stockées. Pour corriger ces erreurs, il est possible d'ajouter des codes

correcteurs avec de fortes capacités de correction. Nous y reviendrons dans le Chapitre 2.

Ces mesures permettent de limiter l'impact des risques sur la récupération des données, notamment sur les temps d'accès à l'information et la probabilité de récupération sans erreur.

### 1.4.4 Confidentialité

Lors de l'analyse de risques, nous avons vu que la confidentialité était menacée de plusieurs manières, avec des conséquences pour les clients et aussi pour les acteurs du système en termes de réputation et d'image de marque, de coûts financiers et stratégiques. Il peut s'agir d'erreurs de manipulations de données qui font fuiter les données d'un utilisateur dans les fichiers d'un autre, de vol à diverses étapes numériques ou biologiques de la chaîne, ou d'espionnage par l'interception de transmissions d'informations, ou encore des virus (*e.g.* cheval de Troie).

#### 1.4.4.1 Management du Système d'Information

Pour contrer ces risques, plusieurs mesures de sécurités peuvent être mises en place. Tout d'abord, l'utilisation d'accords de confidentialité ou "non-disclosure agreements" engageant les acteurs du système d'information à ne pas divulguer d'information sur ce système peut être utile. Un client et un prestataire doivent signer ce type d'accords de confidentialité, qui créent une obligation légale de respecter les conditions de confidentialité stipulées. Un tel accord a aussi un effet dissuasif pour toute partie prenante. Les employés doivent aussi signer une charte et être informés dans leur contrat de travail de leur devoir en termes de secret professionnel et des conséquences en cas de manquement afin de les dissuader de revendre des données encodées dans de l'ADN ou des secrets R & D à la concurrence.

Comme nous l'avons vu dans les sections précédentes, l'authentification permet de limiter les risques d'intrusion. Avec une Gestion des Identités et des Accès appropriée, tout Utilisateur du système doit s'authentifier de façon unique, permettant d'autoriser (ou non) et de tracer ses actions. L'utilisation de pare-feu permet aussi de renforcer la sécurité du réseau d'une organisation en filtrant le trafic entrant et sortant. La prévention contre les infections par virus est importante, que cela passe par de la prévention, de la détection, ou l'isolation des virus et le rétablissement du système. Un virus peut être introduit physiquement via une clef USB, ou par une connexion Internet, ou des programmes. Ainsi, pour prévenir la transmission de virus dans le Système d'Information, tous ces éléments doivent être testés, et isolés si jugés suspects. Les données à stocker dans des molécules d'ADN doivent être testées pour vérifier qu'elles ne contiennent pas de données malveillantes tel que le shellcode présenté dans [94]. De même, chaque dispositif doit être testé. Prenons par exemple, le séquenceur MinION portable qui a le volume d'une grosse clef USB. L'utilisation d'un anti-virus sur cet appareil est essentiel avant de le brancher à un PC contenant des données sensibles et de séquencer de l'ADN encodant des données confidentielles. La prévention contre les virus passe aussi par des mises à jour régulières de tous les actifs du système ainsi que la formation des personnels. Il est possible de suivre les spécifications et mesures d'améliorations de la confidentialité de normes telle que la norme ISO-27001 [84], introduite en Section 1.2.1.1. Elle spécifie les exigences pour planifier, mettre en œuvre, vérifier et mettre à jour un système de management de la sécurité de l'information. Parmi celles-ci, l'identification des types de données et leurs niveaux de sensibilité, l'évaluation des risques liés à la sécurité de l'information et la mise en place de contre-

mesures, dont la formation des personnels, les audits réguliers, et le chiffrement des données. Dans la section suivante, nous explorons plus en détail l'utilisation du chiffrement pour garantir la confidentialité des données.

#### 1.4.4.2 Chiffrement

La norme ISO-27001 [84] préconise une stratégie de chiffrement complète, couvrant le stockage, les transmissions et l'utilisation des données. Cette stratégie est complétée par une politique de gestion des clés robuste, qui implique notamment un stockage sécurisé. Afin de garantir la confidentialité des données dans la chaîne de stockage, elles peuvent donc être chiffrées avant de les stocker dans l'ADN, en utilisant des algorithmes de chiffrement standardisés par le NIST, par exemple. Les deux chiffrements par blocs approuvés sont AES (Advanced Encryption Standard)[113], et le Triple DES dont la recommandation expire toutefois en ce début d'année 2024. Cependant, le stockage sur ADN a pour but de conserver les données à très long terme, bien au delà de la durée de vie du niveau de sécurité de tels algorithmes. En effet, des faiblesses de sécurité d'un algorithme peuvent être découvertes et, suivant la loi de Moore, la puissance de calcul des ordinateurs doublant tous les 18 mois, un crypto-système aujourd'hui sûr peut être cassé en quelques années. Un algorithme de chiffrement utilisant une clef de 256 bits est aujourd'hui impossible à casser avec une attaque brute force, *i.e.* en testant toutes les clefs possibles, mais sa sécurité n'est pas assurée dans 10 ans. L'avènement de l'ordinateur quantique rendra obsolètes tout un ensemble d'algorithmes de chiffrement [114], dont le niveau de sécurité repose sur le temps nécessaire pour les casser. Des algorithmes de cryptographie post-quantique [115] sont aujourd'hui en développement et en cours de standardisation par le NIST [116]. L'objectif est d'avoir des solutions sécurisées à la fois contre les ordinateurs quantiques et classiques, et de pouvoir interopérer avec les protocoles et réseaux de communication existants.

En prenant en compte ces évolutions, il est donc nécessaire de développer des solutions de confidentialité qui ne s'appuient pas seulement sur les limitations en puissance de calcul des ordinateurs actuels. Pour répondre à cette demande, deux domaines exploratoires utilisent l'ADN comme support pour faire de la sécurité : la stéganographie ADN et la cryptographie ADN. Quelques expérimentations ont été effectuées en stéganographie ADN pour dissimuler de l'ADN stockant de l'information, et en cryptographie ADN pour chiffrer de l'ADN. Nous décrivons ces méthodes dans les sections suivantes.

#### 1.4.4.3 Stéganographie ADN

La stéganographie ADN est un domaine qui repose sur la dissimulation d'informations en utilisant l'ADN comme support. Stéganographie signifie écriture cachée en latin. L'idée sous-jacente est de dissimuler l'ADN contenant de l'information dans un support existant, par exemple parmi une multitude de molécules. Nous distinguons deux classes de méthodes : celles qui reposent sur la dissimulation d'ADN dans la nature, et celles qui utilisent la stéganographie *in vitro*.

Pour la première classe de méthodes, Clelland *et al.* [117] présentent en 1999 une première approche avec deux couches de stéganographie. Le message est encodé dans une séquence d'ADN de 100 bases. Il est tout d'abord encadré de deux amorces et synthétisé, puis mélangé au génome d'un humain (plusieurs milliards de paires de bases), préalablement traité et coupé en molécules d'ADN d'environ 100 bases.

L'échantillon ainsi produit est confiné dans un "DNA microdot", un point microscopique d'ADN attaché à une surface solide. Pour retrouver les données encodées dans l'ADN, il faut tous d'abord retrouver le point microscopique sur la surface, et connaître les amorces permettant une PCR pour isoler et lire l'information utile parmi toutes les molécules. Une autre méthode [14] propose de cacher l'ADN dans un organisme vivant, en ajoutant l'ADN encodé dans des plasmides (des molécules d'ADN circulaires) avant d'être transféré et caché dans l'organisme hôte. Pour retrouver les données cachées il faut effectuer une PCR connaissant les amorces encadrant l'information encodée.

La deuxième classe de méthodes repose sur de la stéganographie *in vitro*, sans cacher l'ADN dans des organismes vivants. Deux méthodes [118] [119] utilisent comme clef les amorces encadrant les séquences ADN. Dans la première [118], deux utilisateurs Alice et Bob souhaitent échanger de l'information de façon sécurisée. Pour cela, ils se partagent un dictionnaire de couples d'amorces. Pour envoyer un message, Alice encode son message dans une séquence ADN, l'encadre d'un couple d'amorces du dictionnaire, synthétise l'ADN et le mélange à une multitude d'autres molécules. Elle envoie cet ensemble à Bob, ainsi que la référence des amorces dans le dictionnaire. Bob peut alors faire une PCR avec les amorces adaptées et récupérer l'information. Alice peut aussi choisir de transmettre le couple d'amorces à Bob, avec des erreurs. En utilisant le dictionnaire, Bob peut retrouver les amorces sans erreurs à utiliser par maximum de vraisemblance. La deuxième méthode [119] repose sur un échange sécurisé d'amorces. Alice et Bob déterminent chacun une amorce, la chiffrent et se l'envoient. Lorsque Alice envoie un message à Bob, elle l'encode en une séquence ADN et l'encadre des deux amorces. Ensuite, elle la synthétise et la mélange à une multitude de molécules ADN leurre, et envoie le tout à Bob. Seul Bob peut faire la PCR avec les amorces secrètement échangées. Une autre méthode [120] permet de lire un message caché dans une molécule d'ADN sans passer par un séquençage. Elle repose sur la connaissance de séquences ADN de 26 bases $S_0$ et $S_1$ correspondants aux bits 0 et 1, respectivement. Un message binaire est chiffré par la concaténation des courtes séquences $S_0$ et $S_1$ correspondants aux bits. La séquence obtenue est encadrée de deux amorces *Forward* et *Reverse*, synthétisée, et mélangée à de l'ADN leurre. Pour connaître les positions des bits 0 dans la séquence, il faut effectuer une PCR avec comme amorces *Forward* et la séquence $S_0$. Les molécules amplifiées par cette PCR terminent toutes par la séquence $S_0$, une observation de leurs tailles, avec la technique d'électrophorèse par gel [121], permet donc de connaître les positions des bits 0 dans la séquence. Cette méthode permet, sans effectuer de séquençage, d'identifier les positions des bits dans une molécule d'ADN et donc de le déchiffrer. Cette solution est limitée par la taille des images gel sur lesquelles on observe les tailles des molécules d'ADN. Des messages de 32 bits maximum ont été trouvées avec cette technique. Une autre méthode [120] est la dissimulation d'un message sous forme de molécule d'ADN parmi un ensemble de molécules d'ADN de tailles différentes. La clef secrète est la taille du message. Pour déchiffrer, la technique d'électrophorèse [121] permet de séparer les molécules selon leurs tailles, et ainsi d'isoler la molécule de la taille recherchée pour la déchiffrer.

Il existe donc plusieurs méthodes permettant de cacher des données dans de l'ADN. Toutefois, pour les méthodes stockant des données *in vivo*, la nature imprévisible des mutations génomiques peut poser des problèmes en modifiant la structure de l'ADN. D'autre part, les méthodes utilisant les amorces comme clefs n'adaptent pas les amorces aux séquences ADN, et ne garantissent donc pas leur bon fonctionnement lors d'une PCR. De plus, ces méthodes sont vulnérables à un séquençage en force brute, qui lit toutes les molécules présentes en solution, car elles ne sont pas sécurisées. Enfin, la méthode reposant sur

l'association de courtes séquences d'ADN à un bit est limitée en termes de densité d'information (0,03 bits encodés par base) et de taille de message chiffré (32 bits au maximum). D'autre part, la stéganographie ADN présente un problème de mise à l'échelle. Un des avantages principaux de l'ADN en tant que support de stockage est la densité qu'il offre, de l'ordre d'un million plus dense que les disques durs. Si l'ADN contenant de l'information doit être caché parmi une multitude de molécules ou dans une grande structure qui n'ont pour but que de dissimuler l'ADN encodé, alors une grande partie du volume ne sert pas à encoder de l'information. C'est pourquoi nous présentons dans la section suivante des solutions de sécurité qui ne reposent pas sur la dissimulation de l'information dans un support existant, mais plutôt sur le chiffrement.

#### 1.4.4.4 Cryptographie ADN

Le domaine de la cryptographie ADN regroupe plusieurs types de méthodes de chiffrement de données. Une première classe de méthodes, que nous appellerons pseudo cryptographie ADN [122], repose sur les manipulations de séquences ADN pour effectuer des chiffrements numériques. Le terme "pseudo" provient du fait qu'une méthode de pseudo cryptographie ADN peut se baser sur des procédés biologiques, mais dans le but de les simuler et non de les réaliser expérimentalement. Ce sont donc des méthodes de chiffrement qui utilisent la structure de l'ADN pour effectuer des opérations dans le domaine numérique. La deuxième classe de méthodes présente des solutions de sécurité au niveau biologique, qui s'appuient sur des opérateurs biologiques.

Décrivons tout d'abord les solutions de pseudo cryptographie ADN. Elles s'appuient sur des opérations de deux types : les opérations inspirées de la biologie telles que l'amplification par PCR ; et, les opérations numériques sans lien avec des manipulations biologiques telles que la substitution, les permutations, le XOR, etc.

La première sous-classe de méthode utilise uniquement des opérations numériques en base 4 pour manipuler des séquences ADN [123] [124] [125] [126], dans le but d'augmenter la complexité d'algorithmes numériques. Ces solutions de chiffrement ne fonctionnent que dans le monde numérique et non au niveau biologique. Un exemple est le DNA-AES [124], une version en base 4 du célèbre algorithme standard de chiffrement AES. La structure des séquences ADN encodées ne prenant cependant pas en compte les contraintes structurelles de l'ADN, il n'est pas possible de garantir que celles-ci sont synthétisables, et si oui, lisibles par des séquenceurs. Il en va de même pour la méthode DNAX [126], qui propose d'utiliser des gènes existants comme clef de chiffrement. Le chiffrement utilisé est le masque jetable ou "One-Time pad". Pour chiffrer, un XOR est effectué entre la clef de chiffrement et des séquences ADN encodant de l'information, avant de les synthétiser et de les stocker. Cet article présente cependant la solution en numérique, sans preuve de concept biologique ou prise en compte de contraintes structurelles de l'ADN. Il n'y a donc aucune garantie que ces données chiffrées pourront être synthétisées ou séquençables.

La deuxième sous-classe de méthodes de pseudo cryptographie ADN s'appuie sur des fonctions biologiques, en présentant des simulations de ces processus [127] [128] [122]. Un exemple est la méthode [128] dont le chiffrement repose sur un grand nombre d'opérations sur une séquence ADN qui encode de l'information. Cette séquence ADN est encodée avec des tableaux dynamiques en séquence d'ARN messager, puis en acides aminés, et des opérations numériques tel que le XOR sont appliquées aux séquences. Ainsi, un algorithme complexe chiffre les données en s'appuyant sur des opérateurs inspirés de

la biologie tels que le procédé de transcription biologique de l'ADN en ARN messager. Un autre article [122] se base sur le procédé d'épissage, lors duquel certaines régions d'une séquence d'ARN transcrit sont éliminées, donnant l'ARN messager. Le chiffrement repose sur l'ajout de régions non codantes dans une séquence ARN codant de l'information, et la clef est l'ensemble des positions de ces régions. Ces méthodes s'inspirent de l'ADN et du vivant pour chiffrer des données en numérique mais n'ont pas pour but d'être utilisées dans le vivant.

Les méthodes de chiffrement de pseudo cryptographie ADN peuvent donc être regroupées en deux classe. La première repose sur l'utilisation de données numériques en base 4 pour ajouter de la complexité à des algorithmes de chiffrement numériques, qui s'appuient sur des opérateurs numériques. La deuxième regroupe les méthodes de chiffrement qui s'inspirent d'opérateurs biologiques pour effectuer des chiffrements numériques. Ces méthodes ne sont pas vouées à être utilisées pour manipuler l'ADN, elles mélangent d'ailleurs dans la plupart des cas des opérateurs biologiques et des opérateurs numériques non applicables au domaine biologique, tel que le XOR. Ces méthodes ne peuvent donc pas s'inscrire dans une chaîne de stockage de données dans l'ADN.

La seconde classe de méthodes de cryptographie ADN, composée de quelques méthodes, assure la sécurité des données au niveau biologique.

La méthode fondatrice est celle de Gehani *et al.* [129]. Cette méthode est une proposition théorique qui n'a pas été implémentée. Dans celle-ci, la clef est une longue molécule d'ADN ($10^6$ à $10^8$ bases) contenant une suite de blocs qui sont des messages en clair et chiffrés. Un bloc de 8 à 24 bases codant un message en clair est suivi d'un bloc codant ce message sous forme chiffrée, puis d'un codon stop. Afin d'obtenir le message en clair, il faut utiliser une ADN polymérase qui va s'attacher au bloc chiffré et parcourir le message en clair pour créer son brin complémentaire. La clef est donc ce long brin d'ADN. Elle est à usage unique, et doit être préalablement partagée de façon sécurisée entre deux personnes qui souhaitent s'échanger un message. Le procédé d'échange de clef sécurisée n'est pas précisé dans la méthode. Cette méthode nécessite, pour chaque transmission de message, la création d'une clef à usage unique qui est une très longue molécule d'ADN contenant tous les messages en clair et chiffrés possibles, et un échange sécurisé de cette molécule. De plus, le déchiffrement repose sur un décodage biologique l'un après l'autre de chaque petit bloc chiffré (de 8 à 24 bases) par une opération biologique, le séquençage des blocs en clair obtenus et leur assemblage. Ce déchiffrement n'est pas simulé ou testé expérimentalement.

Plusieurs méthodes [130] [131] [132] reposent sur l'utilisation de puces à ADN ("DNA chips"), constituées d'une surface solide, telle qu'une lame de verre, sur laquelle des brins simples d'ADN de 15 à 10000 nucléotides sont fixés. Ces brins sont aussi appelés sondes ADN, car leur fonction est de s'hybrider à des séquences ADN d'intérêt. Les puces à ADN fonctionnent généralement comme suit : Une puce est préparée en fixant à sa surface des sondes ADN à des positions précises. Ensuite, l'échantillon d'ADN à analyser est marqué, avec un label fluorescent ou radioactif, puis ajouté à la puce. L'ADN peut s'hybrider aux sondes ADN si leurs séquences sont complémentaires. Enfin, la puce est analysée pour détecter les positions des liaisons. Pour identifier l'hybridation, une photo X-ray est alors prise ou les rayons UV sont utilisés. Selon la concentration d'ADN s'hybridant aux sondes, l'intensité du signal sur la photo diffère. Les solutions de sécurité [130] [131] [132] proposent d'utiliser l'intensité des signaux à chaque spot pour encoder de l'information. Par exemple [131], un signal faible est associé au bit 0, et un signal intense au bit 1. Pour chiffrer, il faut donc créer des puces à ADN avec des sondes arrangées dans un ordre précis,

représentant ainsi une matrice de 0 et de 1, qui forme le message. Pour déchiffrer, il faut disposer de la clef, c'est-à-dire un ensemble de brins d'ADN qui vont s'hybrider aux sondes de la puce et traiter les signaux obtenus des images d'hybridation. Ces méthodes permettent de se passer de séquençage car pour lire l'information il faut étudier l'image d'hybridation obtenue. Toutefois, cela signifie qu'il faut générer une sonde (multiples fragments de 25 bases avec une séquence ADN précise) pour chaque bit à lire, et qu'un bit est codé par des milliers de nucléotides, ce qui réduit l'intérêt de l'approche. Or la synthèse est le procédé le plus coûteux actuellement, cette méthode est donc complexe à mettre à l'échelle.

La méthode proposée dans [133] est théorique. Elle repose sur l'assemblage de petits fragments d'ADN en une longue molécule d'ADN pour constituer un message. Ce message est ensuite coupé en fragments de tailles aléatoires. Un morceau de clef est ajouté à chaque fragment d'ADN. C'est le mélange de ces fragments qui constitue le message chiffré, la difficulté étant de les réorganiser. Pour retrouver la longue molécule d'ADN, il faut tout d'abord retirer les morceaux de clef de chaque fragment d'ADN, puis assembler les fragments en une longue molécule. Cette méthode présente une complexité intéressante car il est difficile de reconstituer une séquence ADN à partir de multiples fragments désordonnés. Toutefois, l'algorithme n'indique pas la méthode d'assemblage de fragments, ni les protocoles à mettre en place pour une manipulation *in vivo*. Or, l'assemblage [134] repose souvent sur la complémentarité des extrémités des fragments. Ainsi, il faut maîtriser la structure des fragments afin de pouvoir correctement les assembler.

La méthode [135] du même auteur présente une solution numérique de chiffrement des données à l'aide d'un "One-Time Pad" ; un chiffrement qui consiste en un XOR du message avec une clef secrète de la même taille ; avec une méthode de génération de clefs à partir de fragments ADN aléatoires. Plus clairement, la clef secrète est obtenue à partir de matériel génétique d'une bactérie. Elle est séquencée pour servir le chiffrement du message en numérique. Elle est ensuite implantée dans une bactérie pour être conservée de façon sûre. L'implantation de la clef dans une bactérie est simulée via un programme informatique et non testée expérimentalement. Cela soulève quelques questions. Tout d'abord, implanter la clé dans un plasmide passe par l'usage d'enzymes de manière à ce qu'elle puisse bien s'hybrider. Cela pose la contrainte de ne pas voir apparaître des motifs de nucléotides dans la clef ; motifs qui engendrent des comportements particuliers des enzymes. Nous y reviendrons dans le chapitre 3. A noter également, qu'il ne faut pas non plus que des parties de la clé encodent un virus. N'oublions pas non plus les contraintes liées au séquençage. La clé ne doit par exemple pas comporter d'homopolymères trop longs pour pouvoir garantir ce séquençage. Ces points ne sont pas discutés dans [135].

Nous pouvons remarquer qu'à part les méthodes reposant sur les puces à ADN, les autres méthodes sont théoriques. Elles n'ont pas été validées expérimentalement. Or, comme nous venons de le voir, plusieurs types de contraintes doivent être considérés lors du stockage de données dans l'ADN. Tout d'abord, des contraintes sur la structure de l'ADN s'imposent, notamment une taille maximale d'homopolymères et le taux de GC qui doit être relativement équilibré, afin de garantir le bon déroulement de la synthèse et du séquençage. De plus, l'utilisation de certains enzymes et d'amorces pour la PCR nécessite de maîtriser la structure des séquences ADN présentes dans une même solution, afin d'éviter des phénomènes inattendus tels que des hybridations et des amplifications non spécifiques.

Les méthodes reposant sur des puces à ADN sont limitées en terme de taille de message, car un ensemble de brins d'ADN de 25 bases correspond à un bit d'information encodée. De plus, pour lire un message il faut construire des sondes ADN qui nécessitent également de multiples copies d'une sonde

ADN pour un bit d'information.

Nous souhaitons nous inscrire dans une démarche de stockage de données dans l'ADN à long terme. Or, les méthodes [129] [130] [131] [132] reposent sur l'utilisation de petits brins simples d'ADN, qui sont beaucoup moins stables à long terme que des longs brins doubles d'ADN. En effet, un brin simple d'ADN peut se casser, et ne pourra pas être réparé contrairement à un brin double. D'autre part, un brin simple peut s'hybrider à une séquence complémentaire ou presque complémentaire, et créer des structures d'ADN inattendues et difficilement manipulables. De même, la méthode [135] repose sur l'implantation d'une clef secrète sous forme de molécule d'ADN dans une bactérie, cest-à-dire *in vivo*, ce qui peut poser des problèmes de stabilité à long terme.

## 1.5 Conclusion

L'objectif de cette thèse est d'explorer la sécurité des données stockées dans l'ADN en intégrant nativement des solutions à cette technologie, afin d'éviter les erreurs passées dans le développement de la technologie de stockage numérique difficile à sécuriser aujourd'hui. Ces solutions devront prendre en compte les problématiques spécifiques de ce stockage. Cela inclut les spécificités du stockage physique, des technologies utilisées pour encoder et accéder à l'information, et de la durée de vie très longue de l'ADN. En conséquence, les mesures de sécurité doivent être adaptées.

Dans ce chapitre, nous avons présenté comment stocker des données dans de l'ADN avec la chaîne de stockage. En particulier, nous avons pu définir l'ADN et les procédés biologiques utilisés pour le stockage, l'archivage et la récupération des données stockées dans l'ADN. Cette technologie soulève également de nombreuses questions en matière d'organisation de l'information, avec l'enjeu de l'indexation des fichiers et de leurs données. Elle impose aussi de nombreuses contraintes dans le codage de l'information. Elle souffre d'erreurs dans l'écriture (synthèse) et la récupération (séquençage) des données (erreurs de substitutions, délétions, injections de bases) et nécessite d'utiliser des algorithmes de consensus comme des codes correcteurs d'erreurs. Mais, nous l'avons aussi vu, il faut tenir compte de contraintes plus biologiques et chimiques. Par exemple, l'ADN simple brin se conserve moins bien que l'ADN double brin. L'ADN conservé *in vivo* peut muter et entraîner des pertes d'information. Certaines structures de séquences ADN sont très mal séquencées, voire illisibles si elles contiennent de longs homopolymères, si leur taux de GC est très déséquilibré, etc. Les structures complexes d'ADN ne se conservent pas dans le temps (même un double brin avec un brin légèrement plus grand que l'autre risque de perdre son extrémité saillante) et ne survivent pas à la PCR. Toutes ces contraintes jouent sur l'encodage des données. Nous le verrons plus précisément dans le Chapitre 2, où nous proposons une solution de codage de données binaires en séquences ADN qui prend en compte non seulement ces contraintes mais aussi la maîtrise de motifs présents dans les séquences. Cela nous permettra le déploiement d'un algorithme de chiffrement que nous proposons. Celui-ci fonctionne sur la base d'opérateurs biologiques qui modifient la structure de l'ADN en utilisant ces motifs, et sera présenté au Chapitre 3.

Dans la deuxième partie de ce chapitre, nous avons identifié la menace et les risques qui pèsent sur cette chaîne de stockage, sur la base d'un scénario d'usage relativement complexe mais que nous pensons représentatif du déploiement d'une telle chaîne. Nous avons présenté une analyse de risques STRIDE, qu'il nous a semblé importante car elle n'existe pas aujourd'hui. Celle-ci reste une première ébauche, qui

pourra être complétée en affinant ou modifiant le scénario d'usage. L'analyse STRIDE permet de classer des menaces sur la chaîne de stockage dans différentes catégories, où chaque lettre de l'acronyme STRIDE correspond à un type de menace et est associé à une propriété de sécurité. Par exemple, le risque de fuite de données est associé à la menace de divulgation d'informations, qui concerne la propriété de sécurité de confidentialité. Ainsi, pour chaque étape de la chaîne de stockage de données dans l'ADN, nous avons identifié différentes sources de menaces, mettant en avant les vulnérabilités de cette chaîne. Celle-ci peut être la cible d'attaquants avec des intérêts variés, allant de l'ex-employé qui souhaite se venger au service étatique dont le but est d'atteindre la souveraineté de leur recherche, en passant par le concurrent qui veut obtenir un avantage concurrentiel, ou encore le cyber-bioterroriste qui vise à altérer la composition des molécules d'ADN. Les sources de risque sont donc diverses, et toutes les étapes de la chaîne sont menacées, avec des enjeux en matière de confidentialité, de disponibilité, d'intégrité et de traçabilité des données comme des molécules et des manipulations biologiques.

En troisième partie, nous avons effectué un état de lieux des solutions de traitement du risque, en présentant les outils de sécurité exploitables. Nous y trouvons des méthodes classiques du numérique, telles que la Gestion des Identités et des Accès [101] et le watermarking [96] pour la traçabilité, les fonctions de hachage et les codes MAC [107] pour l'intégrité, la prévention des risques par des normes de management de système d'informations telles que ISO-27001 pour la disponibilité et la confidentialité, ainsi que des protocoles de chiffrement numérique [136] spécifiquement pour la confidentialité. Nous avons aussi présenté des solutions de sécurité spécifiques au données biologiques, telles que des méthodes de tatouage ADN, de cryptographie ADN et de stéganographie ADN. Les méthodes de tatouage ADN présentées permettent la traçabilité d'organismes vivants, qui sont tatoués sans modifier l'expression de leurs gènes. Une autre classe de méthodes repose sur la stéganographie ADN, qui dissimule de l'ADN stockant de l'information dans un support existant. La sécurité repose alors sur la dissimulation de l'ADN codant de l'information dans un organisme vivant, ou bien parmi une multitude d'autres molécules d'ADN ne codant pas d'information. La sécurité de ces méthodes repose sur la difficulté à récupérer les molécules d'ADN lorsqu'on ne connaît pas leurs tailles ou les amorces qui les encadrent. Ces méthodes nécessitent de stocker une petite quantité de données dans un grand support de stockage ou volume d'ADN qui ne sert pas à encoder de l'information. La stéganographie ADN présente donc un problème de mise à l'échelle, et ne permet de stocker qu'une quantité très limitée de données. De plus, dans une optique de stockage de données, un des avantages majeurs de l'ADN est la densité de l'information stockée, qu'il est important de préserver lors du développement de solutions de sécurité. Cela exclut donc l'utilisation des techniques de stéganographie.

Concernant la cryptographie ADN, des méthodes ont été proposées, qui visent majoritairement à sécuriser des données dans le domaine numérique. Peu de solutions visent à sécuriser au niveau biologique, et elles sont souvent théoriques, sans expérimentation des protocoles proposées et sans prise en compte des diverses contraintes biologiques. Cela soulève donc la question du caractère opérationnel de ces méthodes. Les méthodes reposant sur les puces à ADN présentent une approche intéressante qui permet de déchiffrer des données sans passer par un séquençage, mais elles sont limitées en termes de volume et de densité de données. De plus, ces solutions nécessitent le stockage à long terme de petits brins simples d'ADN, qui sont fragiles et peuvent se casser, être dégradés, ou s'hybrider de façon inattendue. Au contraire, de longs brins doubles d'ADN sont plus stables et les dégradations de leurs brins simples peuvent être réparés.

Les méthodes de l'état de l'art présentent donc des solutions de sécurité partielles, théoriques ou présentant un intérêt limité pour le stockage de données dans l'ADN. Il y a donc un intérêt à explorer comment assurer la confidentialité des données stockées dans l'ADN au travers de manipulations biologiques afin d'assurer la sécurité des données à long terme, tout en prenant en compte les contraintes liées à ce support.

Chapitre 2

# Encodage avec contraintes de données dans l'ADN

Comme nous l'avons vu au Chapitre 1, le stockage de données dans l'ADN est une nouvelle technologie. Il est donc possible d'intégrer l'aspect critique de la sécurité au coeur de son fonctionnement, contrairement à ce qui a été fait pour les dispositifs de stockage électroniques. Le travail présenté dans ce chapitre aborde spécifiquement ce problème. Il vise à sécuriser les données archivées dans les molécules d'ADN, notamment en termes de confidentialité dans l'ensemble de la chaîne de stockage. Comme présenté dans notre analyse de risques (*cf* Section 1.2), la confidentialité des données est menacée à toutes les étapes de la chaîne de stockage. En effet, un utilisateur non autorisé peut espionner le dispositif de séquençage [94] ou de synthèse [137], ou voler les molécules d'ADN stockées, et ainsi récupérer des données sensibles. Pour contrer ces menaces, une solution consiste à ajouter une étape de chiffrement des données à la chaîne. Comme illustré en Figure 2.1, le chiffrement est classiquement effectué sur les données binaires. Cependant, l'ajout d'une telle fonctionnalité n'est pas anodin. Il impacte l'encodage des données. Nous l'avons vu, il faut tenir compte à la fois des contraintes des molécules d'ADN et des contraintes des technologies de synthèse et de séquençage (*cf* Section 1.1.3.1). Or, il est impossible de prédire le résultat d'un cryptosystème. Ceux-ci convertissent des données en clair en un flux binaire chiffré, qui est en général une séquence de bits uniformément distribués. Il existe alors un risque que, une fois encodé en base 4, un flux binaire chiffré conduise à une séquence ADN qui ne respecte pas les contraintes imposées. Un encodage classique associant deux bits à une base ne permet par exemple pas de maîtriser la structure des séquences ADN. Pour prendre en compte ces contraintes, nous avons développé l'algorithme Dynamic Sliding Window Encoding (DSWE), qui a pour objectif de coder des flux de données chiffrées en base 4 en tenant compte de différentes contraintes : la création d'homopolymères de trop grande taille, d'un "G-C content" (ou taux de GC) global et local équilibré et l'apparition de séquences d'ADN qui correspondent par exemple à des amorces, *i.e.* des motifs interdits. Le DSWE s'appuie sur un encodage dynamique et un encodage avec fenêtre glissante. Le premier encode des blocs binaires de taille fixe en mots code ADN à taille variable, afin de limiter la taille des homopolymères à une valeur $N$ tout en optimisant la densité des données. Le second fait usage d'une fenêtre glissante afin de prévenir l'apparition de motifs interdits. De plus, notre encodage DSWE est développé de façon à préserver la distribution uniforme des données conférée par le chiffrement. En conséquence, la proportion de chaque base ADN et donc le taux de GC des séquences ADN encodées avec DSWE sont équilibrés, comme nous le montrerons dans la partie expérimentale.

Ce second chapitre est articulé en six parties. Tout d'abord, nous détaillons les contraintes pour

l'encodage de données dans de l'ADN et l'état de l'art des méthodes d'encodage proposées respectant ces contraintes, avant de présenter dans une seconde partie la notion de chiffrement et le cryptosystème choisi. Une troisième partie détaille notre propre méthode d'encodage, tandis que la quatrième partie présente plusieurs alternatives et optimisations à cette solution. Dans une cinquième partie, nous introduisons les codes correcteurs que nous allons tester avec notre DSWE. Enfin, les résultats expérimentaux sont présentés dans une sixième partie, mettant en avant les contributions originales de notre solution.

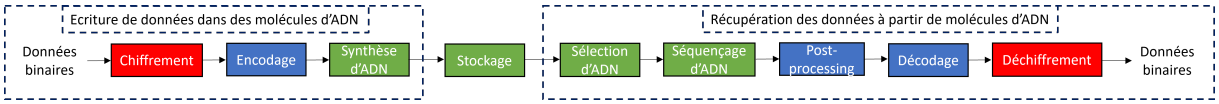



FIGURE 2.1 – Chaîne de stockage de données chiffrées dans l'ADN, avec des étapes de chiffrement et déchiffrement représentées dans des encarts rouges.

## 2.1 État de l'art des méthodes d'encodage avec contraintes

Les contraintes biologiques ont été prises en compte de différentes manières dans les solutions d'encodage existantes. Dans cette section, nous présentons ces contraintes trois grandes classes de méthodes d'encodage selon les contraintes qu'elles prennent en compte simultanément ou non.

### 2.1.1 Contraintes biologiques

Les contraintes sur la structure des séquences ADN proviennent de plusieurs sources. La première concerne le taux de GC, qui est la proportion de bases dans une séquence qui est un G ou un C. Il doit être contrôlé dans chaque sous-partie d'une séquence ADN, et pas seulement dans son ensemble. Une proportion très inégale de bases G-C et A-T entraînera des "dropouts" (bases manquantes) [33] lors du séquençage, et des erreurs lors de l'amplification par PCR [34]. En outre, dans le cas où le processus de synthèse des séquences dépend de l'assemblage de petites séquences en de longs brins d'ADN, les séquences à assembler doivent être proches en termes de taux de GC. Ainsi, le taux de GC de l'ADN peut être plus élevé que celui de l'ADN de base et doit être équilibré, dans une certaine plage de valeurs, d'environ 40% à 60% [33] pour les longues séquences, et 20% à 80% dans les régions de 100 à 200 bases, en général. Une deuxième contrainte concerne les polymères. Les processus de synthèse et de séquençage sont sensibles aux homopolymères, *i.e.* les répétitions de la même base. Plus ils sont longs, plus la probabilité de générer des erreurs est élevée. Contrairement au séquençage des génomes où la taille des homopolymères est imposée, dans la chaîne de stockage nous avons un contrôle total sur la nature des séquences synthétiques. La génération de longs homopolymères peut donc être évitée afin d'optimiser à la fois les étapes de synthèse et de séquençage. Une autre contrainte importante est la présence indésirable de certains motifs dans les séquences synthétiques. La synthèse est actuellement limitée à la génération d'oligonucléotides, c'est-à-dire des courtes séquences ADN d'une taille inférieure à 200 bases. Par conséquent, un fichier numérique doit être divisé en plusieurs séquences, chacune contenant un index qui indique sa position dans le fichier. Une séquence est également encadrée par des amorces [63], généralement d'une longueur de 18 à 30 bases, qui sont uniques pour une séquence ou un groupe de séquences. Ces amorces peuvent être utilisées, par

exemple, pour amplifier un sous-ensemble de séquences avant le séquençage par PCR (*cf.* Section 1.1.5). Si une amorce se trouve inopinément au milieu d'une séquence, seule une partie de la séquence sera amplifiée et des données seront perdues. Par conséquent, les index ou les amorces ne doivent pas se trouver dans la partie d'une séquence ADN qui encode les données d'un utilisateur. Il y aurait un risque de confondre une partie d'un fichier avec un motif, ce qui entraînerait une perte d'informations. D'autre part, dans le cas où l'on stocke l'ADN *in vivo*, certains motifs doivent être évités car ils pourraient être utilisés comme information génétique par l'organisme hôte. Nous pouvons citer les séquences d'initiation de transcription et les codons start de 3 bases qui initient la production de protéines. Enfin, lorsque l'on manipule l'ADN, il est utile de réserver des séquences, appelées sites de restriction, qui sont spécifiquement repérées par des enzymes de restriction [138] pour couper des fragments d'ADN double. Comme nous le verrons dans le Chapitre 3, nous proposons une solution de sécurité au niveau du stockage moléculaire basée sur des opérateurs biologiques qui profitent de ce type de sites. Une majorité de ces sites sont de taille 6, tel que ($GAATTC$) qui est le site de restriction de l'enzyme EcoRI. Les enzymes de restriction sont utilisés pour de nombreuses manipulations de l'ADN, par exemple pour insérer des fragments d'ADN dans des plasmides, molécules d'ADN circulaires, et ainsi les stocker *in vivo*.

Nous décrivons ci-après les méthodes d'encodage existantes les plus récentes qui tiennent plus ou moins compte de ces contraintes biologiques afin de préciser l'originalité de notre proposition.

### 2.1.2 État de l'art

#### 2.1.2.1 Méthodes avec une contrainte

La première classe de méthodes ne considère qu'une contrainte biologique : la taille maximale d'homopolymères [18] ou un taux de GC contrôlé [139]. Dans [18], les données sont encodées sans aucun homopolymère. Pour cela, les blocs de données binaires sont tout d'abord convertis en blocs ternaires (*i.e.* en base 3) à l'aide d'un code de Huffman, qui correspond en fait aussi à une étape de compression des données. Chaque symbole en base 3 est ensuite encodé en une base ADN avec le dictionnaire de la Table 2.2B. Comme nous pouvons le voir, cette solution encode un symbole ternaire en une base ADN qui est toujours différente de la précédente. Ainsi, cette encodage considère une taille maximale d'homopolymère de $N = 1$. Dans [140], l'encodage est généralisé pour de plus grandes valeurs de $N$. Pour cela, l'encodage repose sur une stratégie où la Table 2.2B restreignant les bases ADN possibles n'est pas utilisé pour encoder toutes les bases. Un code de Shannon-Fano est utilisé pour convertir des blocs binaires de longueur fixe en mots de code de longueur variable contenant des symboles en base 3 et 4. Il s'agit là aussi d'une étape de compression des données. Puis, les symboles de chaque mot de code sont encodés à l'aide de deux dictionnaires alternativement. En particulier, les $N-1$ premiers symboles sont encodés avec l'encodage classique (*cf* Table 2.2A) qui associe un symbole en base 4 (ou deux bits) à une des 4 bases ADN. Ensuite, le $N^{\text{ème}}$ symbole, qui est un symbole ternaire, est encodé avec le dictionnaire de la Table 2.2B. Cette dernière casse tout homopolymère de plus de $N$ bases. Dans la Section 2.3, nous verrons que l'encodage dynamique que nous proposons diffère de ces stratégies. En particulier, il n'inclut pas de stratégie de compression des données ou d'étape de conversion, mais seulement un encodage direct de données binaires en mots code ADN.

De son côté, la méthode [139] se focalise sur la contrainte de taux de GC. Les données en entrée

**A**

| Bloc binaire | Base ADN |
|---|---|
| 00 | A |
| 01 | C |
| 10 | T |
| 11 | G |

**B**

| Base ADN précédente | 0 | 1 | 2 |
|---|---|---|---|
| A | T | C | G |
| C | A | T | G |
| T | A | C | G |
| G | A | T | C |

FIGURE 2.2 – Tableaux de conversion de données en bases ADN. Le Tableau A présente l'encodage classique qui associe deux bits à une base ADN, et le Tableau B présent la conversion de symboles ternaires $\{0, 1, 2\}$ en bases ADN $\{A, C, G, T\}$, proposé à l'origine par Goldman [18].

sont encodées en séquences ADN à l'aide d'un dictionnaire ou "codebook" de mots de codes ADN. Le dictionnaire contient des mots de code de $p$ nucléotides, avec un taux de GC équilibré. Pour ce faire, des mots de codes Varshamov-Tenengolts [141] sont générés avec des préfixes et des suffixes ajoutés qui aident à équilibrer le contenu G-C du mot de code. Ensuite, chaque mot de code est associé à un message de $v$ bits.

Comme nous venons de le voir, ces méthodes considèrent soit une taille maximale d'homopolymères, soit un taux de GC contrôlé. Pourtant, ces deux contraintes doivent être considérées en même temps afin de récupérer correctement les données stockées dans des molécules d'ADN avec les technologies de PCR et de séquençage (*cf.* Section 1.1.5.2).

#### 2.1.2.2 Méthodes avec gestion d'une taille d'homopolymère et du taux de GC

Plusieurs solutions ont été proposées pour encoder des données dans l'ADN en tenant compte de ces deux contraintes.

Les premières méthodes ne prennent en compte qu'une longueur maximale fixe d'homopolymère. Dans [17], les auteurs présentent la première architecture à grande échelle pour stocker des données dans des molécules d'ADN. Cette approche repose sur l'encodage de chaque bit 0 en une base A ou C, et de chaque 1 en une base T ou G, afin de garantir un taux de GC de 50%. En outre, les bases sont choisies au hasard, sauf si elles peuvent créer un homopolymère de plus de 3 bases. L'encodage se fait en deux étapes : un flux binaire est d'abord encodé en une séquence d'ADN en associant de manière aléatoire chaque bit à l'une des deux bases d'ADN possibles. La deuxième étape consiste à scanner la séquence d'ADN obtenue. Lorsque plus de 3 bases à la suite sont identiques, la $4^{eme}$ base est remplacée par l'autre base correspondant à la même valeur de bit (A avec C, et G avec T). Dans [142], un dictionnaire de mots codes ADN de longueur fixe est créé de sorte que chaque mot code ADN ait une taille maximale d'homopolymère de 3 bases, un taux de GC de 40-60 %, et que la concaténation des mots codés ne crée pas de longs homopolymères. Une méthode itérative est utilisée pour créer tous les mots codes ADN, par concaténation de mots codés d'ADN plus petits respectant ces contraintes. Dans [36], les auteurs construisent des dictionnaires de mots code ADN de 8 à 12 bases, avec un taux de GC de $40-60\%$ et des homopolymères limités à 3 bases. Le codebook est construit à l'aide d'un diagramme états-transitions fini avec 4 symboles de transition correspondant aux 4 bases de l'ADN (c'est-à-dire A, C, G, T). Le passage d'un état du diagramme de transition à un autre permet de contrôler l'apparition d'homopolymères et de maintenir un taux de GC équilibré.

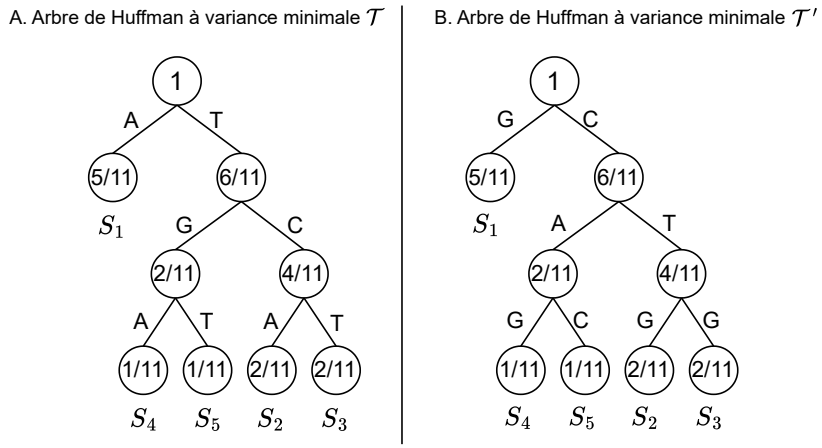



FIGURE 2.3 – Exemple d'arbres de Huffman à variance minimale, utilisés dans [144]. En Figure A, l'arbre $\mathcal{T}$ est construit avec des bords A et T aux distances impaires, et avec des bords G et C aux distances paires. En Figure B, l'arbre $\mathcal{T}'$ est construit de façon inverse.

Les auteurs de [39] encodent des données chiffrées avec un taux de GC contrôlé et une taille maximale d'homopolymère de 3 bases. Pour cela, ils commencent par chiffrer des données binaires à l'aide du cryptosystème AES, paramétré par une clé biométrique générée à partir des informations génétiques d'un utilisateur. Le flux binaire obtenu est converti en une séquence en base 47, en associant à deux octets trois éléments en base 47. Ensuite, ils appliquent aux données en base 47 deux codes correcteurs d'erreurs Reed-Solomon, un "inner code" et un "outer code". Chaque élément en base 47 est ensuite associé à un mot de code ADN de 3 bases à l'aide d'un dictionnaire. Ces mots code ADN de 3 bases ont été sélectionnés de manière à ce que leurs deux dernières bases ADN soient différentes. Cela permet d'éviter les homopolymères de longueur supérieure à 3. Dans [143], les contraintes d'absence d'homopolymères et de taux de GC valant exactement 50% sont garanties par l'alternance des bases A et T en positions paires, et des bases G et C en positions impaires dans les mots de code ADN. Pour ce faire, deux bits sont encodés dans deux bases.

Le schéma de [144] utilise des arbres de Huffman à variance minimale pour convertir des données binaires en mots code ADN de manière compressée. Ces mots de code ne contiennent pas d'homopolymère et présentent un taux de GC proche de 50%. La méthode commence avec la division des données binaires d'entrée en blocs de longueur fixe. Ceux-ci sont ensuite utilisés pour créer deux arbres de Huffman à variance minimale : $\mathcal{T}$ et $\mathcal{T}'$, présentés en Figure 2.3. Une base ADN est associée à chaque arête d'un arbre, de manière à éviter les répétitions de bases d'ADN. Par exemple, l'arbre $\mathcal{T}$ est construit comme suit : Si une feuille se trouve à une distance impaire de la racine, ses bords gauche et droit sont étiquetés respectivement $A$ et $T$, ou respectivement $G$ et $C$ si la feuille se trouve à une distance paire. L'arbre complémentaire $mathcalT'$, présenté en Figure 2.3B, est construit avec les instructions inverses : les étiquettes $A$ et $T$ sont associées aux feuilles situées à une distance paire de la racine. Pour coder des données binaires, si la dernière base codée est $G$ ou $C$, l'arbre $\mathcal{T}$ est utilisé ; ou l'arbre $\mathcal{T}'$ dans l'autre cas. Ainsi, ce schéma d'encodage garantit qu'aucun homopolymère n'est créé. En outre, il garantit un taux de GC proche de 50%.

JPEG-DNA [145] est un système de compression d'images inspiré de la norme JPEG ("Joint Photo-

graphic Experts Group"), qui crée des mots code ADN avec une taille maximale d'homopolymère $N = 3$. Comme dans JPEG, l'encodage commence par une transformation en cosinus discrète, calculée sur des blocs de $8 \times 8$ pixels. Ensuite, les nouveaux blocs sont quantifiés à l'aide d'une quantification scalaire uniforme, avant de calculer le codage différentiel sur les coefficients DC, et de lire les coefficients AC dans un ordre "zigzag". À cette étape, l'algorithme diffère de celui du JPEG, car les coefficients sont codés en mots code ADN. Tout d'abord, la valeur d'un coefficient est encodée à l'aide d'un dictionnaire PAIRCODE [146] $D_c$. Celui-ci limite les homopolymères à 3 bases et garantit un taux de GC équilibré. Le dictionnaire $D_c$ est choisi en fonction de l'intervalle ,ou catégorie, $c$ dans laquelle se trouve la valeur. Ensuite, cette catégorie est encodée avec la méthode de [18]. Tout d'abord, les données binaires sont converties en blocs de base 3 avec un arbre de Huffman ternaire. Dans un deuxième temps, le dictionnaire 2.2B est utilisé pour encoder chaque symbole en base 3 en une base ADN. L'utilisation de cette table garantit l'absence d'homopolymère dans la séquence d'ADN de sortie. La catégorie encodée et la valeur du coefficient sont ensuite toutes deux concaténées à la séquence ADN déjà encodée.

Dans [147], un dictionnaire de mots code ADN est construit de sorte que les homopolymères sont limités à 4 bases et que les sous-séquences complémentaires sont limitées à 2 bases. Cette dernière contrainte garantit qu'un brin d'ADN ne se replie pas sur lui-même, formant une structure secondaire qui peut être difficile à lire. Les mots codes ADN sont construits à l'aide de l'ensemble $\Sigma = \{AA, CC, AC, CA, TC\}$. Ces paires sont choisies pour éviter la création de sous-séquences complémentaires dans les mots codes. Ensuite, un mapping non surjectif prend en entrée chaque mot code et inverse certaines bases de l'ADN pour éviter les homopolymères de plus de 4 bases. Seuls les mots codes ADN sans sous-séquences complémentaires sont ajoutés au dictionnaire. Notons que cette méthode repose sur la création d'un dictionnaires de mots codes sans longues sous-séquences complémentaires, mais qu'elle ne tient pas compte des motifs interdits *a priori*.

Les approches précédentes présentent des méthodes d'encodage qui ne peuvent prendre en compte qu'une seule valeur de $N$, la taille maximale des homopolymères. Cela peut poser des problèmes car, selon le type et la génération des technologies de séquençage, la longueur maximale des homopolymères exigée pour récupérer les données diffère [33]. Il est donc important de créer une solution d'encodage qui puisse être adaptée à plusieurs valeurs de cette longueur maximale $N$. C'est ce que font les méthodes suivantes.

L'algorithme DNA Fountain proposé dans [35] applique la transformation de Luby (LT) et un test de sélection de manière itérative jusqu'à convertir un bloc binaire en une séquence ADN au taux de GC équilibré et sans homopolymères. En résumé, la première étape, *i.e.* la transformation de Luby, repose sur une graine aléatoire qui change à chaque itération. Celle-ci permet de produire une séquence binaire différente qui encode l'information. De son côté, le test de sélection vérifie si le bloc binaire converti en bases ADN valide les contraintes biologiques. Si ce n'est pas le cas, la LT est à nouveau appliquée. À l'issue de ce processus itératif, on obtient un mot de code ADN avec un taux de GC équilibré et sans homopolymères de plus de $N$ bases. La méthode d'encodage présentée dans [148] propose une construction de mots code ADN avec une taille maximale d'homopolymère $N$ et un taux de GC compris dans l'intervalle $[0,5 - \epsilon, 0,5 + \epsilon]$, pour tout $\epsilon$ et $N$.

Tout d'abord, des mots de code ADN de longueur fixe avec une longueur maximale d'homopolymère sont construits à l'aide de la technique de remplacement de séquence [149]. Cet algorithme supprime

itérativement tous les homopolymères interdits d'un mot code ADN et insère des sous-séquences spéciales à des positions prédéfinies pour indiquer où les bases ADN ont été supprimées. Ensuite, ces mots codes sont modifiés à l'aide d'une version généralisée de la technique d'équilibrage de Knuth [150]. Cette méthode modifie les valeurs de certaines bases ADN pour équilibrer le taux de GC des mots codes. En outre, cette solution comprend l'ajout d'un code correcteur d'erreurs de Varshamov-Tenengolts qui peut corriger une erreur par mot code.

Dans [37], les flux binaires sont encodés en des séquences ADN avec une taille d'homopolymère limitée, un taux de GC contrôlé et sans structure secondaire dans les molécules d'ADN. Pour ce faire, le codage se fait en trois étapes. Les données binaires sont d'abord fractionnées en segments indexés de longueur fixe. Ensuite, deux segments sont sélectionnés au hasard pour être incorporés dans une séquence d'ADN selon la méthode du "codex Yin-Yang". Enfin, la séquence d'ADN est examinée en fonction de contraintes préétablies. Si la séquence ADN ne répond pas à ces critères, le processus itératif est activé : un autre segment binaire généré aléatoirement est encodé avec le premier segment sélectionné, jusqu'à ce qu'une séquence d'ADN valide soit générée. Pour certaines données binaires en entrée, cette méthode peut poser des problèmes car il est difficile, voire impossible, de générer une séquence ADN valide. Cette méthode ne tient pas compte des motifs interdits *a priori*.

Dans [151], les données sont encodées en des séquences ADN à l'aide d'un code interne et d'un code externe. Le code interne fonctionne comme suit pour encoder une séquence binaire $S$ en une séquence ADN $D$. Tout d'abord, une séquence ADN pseudo-aléatoire $D_1...D_n$ est générée via une fonction de hachage avec certaines parties des données de la séquence binaire. Ensuite, chaque base d'ADN $D_i$ est codée en $C_i$ telle que $C_i = D_i + b_i \mathrm{mod} C$, avec $b_i$ le bit de la séquence binaire d'entrée $S$. $C$ est le nombre de valeurs autorisées que la base d'ADN $C_i$ peut prendre. Par exemple, pour éviter un homopolymère $AAAA$, la valeur $A$ peut être interdite à cette position, et donc $C = 3$. Le code externe est un code de Reed-Solomon, qui permet de reconstruire des séquences d'ADN qui n'ont pas pu être corrigées avec le code interne.

Comme nous pouvons le voir, ces méthodes ne considèrent pas la contrainte des motifs interdits. Ces motifs, comme les amorces, peuvent par exemple poser problème lors de l'amplification par PCR s'ils apparaissent au sein d'une molécule.

#### 2.1.2.3 Seule méthode concurrente considérant les motifs interdits en plus des homopolymères et du taux de GC

À notre connaissance, la solution mCGR présentée dans [38] est la première à considérer des motifs interdits ou des séquences d'ADN interdites comme une contrainte lors de l'encodage, en plus des contraintes de taux de GC équilibré et de taille maximale d'homopolymères. Fondamentalement, leur stratégie consiste à construire un dictionnaire qui associe un bloc binaire de longueur fixe à un mot code ADN de longueur fixe. Pour créer ce dictionnaire, ils suivent une stratégie qu'ils disent de type "fractale" pour parcourir l'espace des mots de code en base 4, afin de ne pas tester tous les mots code dans l'espace en base 4. Ils retiennent ensuite les mots code d'ADN qui satisfont aux contraintes d'homopolymère et d'équilibre de taux de GC, en vérifiant également qu'un mot de code sélectionné ne soit pas un motif interdit. Pour s'assurer que les mots respectent également les contraintes lorqu'ils sont concaténés, ce dictionnaire est parcouru pour supprimer tous les mots de code commençant et se terminant par au moins

la moitié d'un motif interdit ou d'un long homopolymère. Comparée à [35], cette méthode est beaucoup plus rapide [38]. Cependant, en raison de la suppression de nombreux mots code ADN du codebook lors du post-traitement, le taux d'information de cette méthode, *i.e.* le nombre de bits d'information codés par base ADN, chute de manière significative lorsque le nombre de motifs interdits augmente. De plus, comme [36] et [148], l'élaboration du dictionnaire reste une procédure très complexe, en particulier lorsque le nombre et la longueur des motifs interdits augmentent. En outre, comme ces dictionnaires contiennent de nombreux mots code ADN, la table de recherche est longue et l'association d'une entrée binaire à un mot de code ADN prend du temps.

la Table 2.1 résume toutes les méthodes présentées ci-dessus en fonction des contraintes qu'elles prennent en compte. Les contraintes sont les suivantes : un taux de GC contrôlé, un module de chiffrement, l'apparition de motifs interdits, une taille maximale d'homopolymère fixe ou variable. La première classe de méthodes ne prend en compte qu'une contrainte d'homopolymère ou de contenu G-C, tandis que la deuxième classe prend en compte les deux. La troisième classe de méthodes prend également en compte les motifs interdits.

| | | Homopolymères | | Taux de GC contrôlé | Chiffrement | Motifs interdits |
|---|---|---|---|---|---|---|
| Classe | Méthode | fixe | variable | | | |
| Méthodes avec une contrainte biologique | Cai [139] | | | X | | |
| | Goldman [18] | X | | | | |
| | Mishra [144] | X | | | | |
| | Pic [140] | X | X | | | |
| Méthodes avec des contraintes de taux de GC et d' homopolymères | Church [17] | X | | X | | |
| | Li [143] | X | | X | | |
| | Song [142] | X | | X | | |
| | Wang [36] | X | | X | | |
| | Dimopoulou [145] | X | | X | | |
| | Grass [39] | X | | X | X | |
| | Erlich [35] | X | X | X | | |
| | Nguyen [148] | X | X | X | | |
| | Press [151] | X | X | X | | |
| Méthodes avec 3 contraintes | Lochel [38] | X | X | X | | X |
| | Méthode proposée | X | X | X | X | X |

TABLE 2.1 – Comparaison des méthodes de l'état de l'art et de notre solution d'encodage DSWE. Homopolymère fixe et variable indique si l'encodage considère la contrainte d'une taille maximale d'homopolymère fixe ou variable, respectivement.

### 2.1.3 Contributions

Dans la suite de ce chapitre, nous proposons une approche nouvelle d'encodage pour stocker des données chiffrées sous forme d'ADN, en considérant les 3 contraintes suivantes :

— un taux de GC équilibré

— une taille maximale d'homopolymères $N$

— un nombre $M$ de motifs interdits

Son originalité repose sur une solution d'encodage qui convertit des données binaires en séquences ADN à longueur variable. Elle profite de l'utilisation d'un ensemble de dictionnaires de petites dimensions et d'une stratégie de décodage informée basée sur une fenêtre glissante. Plus précisément, un bloc binaire de longueur fixe est encodé en une séquence ADN de longueur variable à l'aide d'un dictionnaire, sélectionné en fonction de la dernière base d'ADN encodée, afin d'éviter les homopolymères, dans le cadre d'un processus appelé encodage dynamique ou DE.

Notre encodage DE peut prendre en entrée n'importe quelle valeur $N$, à une exception près. Il est important de noter que presque toutes les méthodes présentées dans la Table 2.1 ne prennent pas en compte les motifs interdits. Pour résoudre ce problème, nous proposons un encodage à fenêtre glissante, appelé SWE pour "Sliding Window Encoding". Lorsqu'un motif interdit est sur le point d'apparaître dans la séquence ADN, une base non codante est ajoutée à la séquence. Cette base ne code aucune information mais sera identifiée par le décodeur. Ce dernier répète en partie le processus d'encodage pour identifier le dictionnaire à utiliser et détecter les bases non codantes. Il est "informé". En outre, notre système est facile et souple à adapter lorsque l'on modifie la taille maximale des homopolymères, par rapport à de nombreuses solutions de codage qui ne prennent en compte qu'une seule longueur, comme le montre la Table 2.1. Nos choix de dictionnaires pour DE nous permettent aussi d'obtenir un taux de GC équilibré quand les données binaires en entrée sont uniformément distribuées. Tout cryptosystème peut donc être utilisé avant notre encodage, tant que les données chiffrées sont uniformément distribuées.

## 2.2 Chiffrement

Dans cette section, nous définissons tout d'abord le module de chiffrement dans la chaîne de stockage de données dans l'ADN (*cf.* Figure 2.1), avant de présenter le chiffrement choisi.

Pour assurer la confidentialité des données à chaque étape de la chaîne de stockage, les données binaires sont chiffrées (respectivement déchiffrées) avant l'encodage en base 4 (resp. après son décodage de base 4 en binaire) comme représenté en Figure 2.1 (cadres rouges). Le chiffrement est choisi de manière à maximiser l'entropie et produire des blocs de données indépendants et uniformément distribués. Cela signifie qu'un bit a la même probabilité d'être un "0" ou un "1" et qu'il est indépendant des autres bits du message chiffré.

Un cryptosystème est un ensemble composé des éléments suivants :

— Un algorithme de chiffrement $Enc$

— Un algorithme de déchiffrement $Dec$

— Un ensemble $Plaintexts$ de messages en clair possibles

— Un ensemble $Ciphertexts$ de messages chiffrés possibles

— Un ensemble $KeySpace$ de clefs possibles pour chiffrer et déchiffrer

Il existe deux types d'algorithmes de chiffrements, symétriques et asymétriques. Pour un chiffrement asymétrique, une paire de clef est utilisée : la clef publique permet de chiffrer et la clef privée correspondante permet de déchiffrer les données. Dans le cas d'un chiffrement symétrique, une clef secrète sert à la fois au chiffrement et au déchiffrement des données. Un algorithme de chiffrement peut traiter un message comme un flux continu de bits, ou bien diviser un message en blocs de taille fixe et traiter chaque bloc séparément. Nous nous intéressons ici au chiffrement symétrique par bloc.

Un cryptosystème par bloc est défini par une paire d'algorithmes $Enc$ pour le chiffrement et $Dec$ pour le déchiffrement. La fonction $Enc$ prend en entrée des blocs du message en clair et les chiffre en des blocs de message chiffré à l'aide d'une clef cryptographique, comme présenté ci-dessous.

$$C_i = Enc(P_i, K_{Enc}) \tag{2.1}$$

où : $P_i$ est le $i^{\text{ème}}$ bloc de message en clair à chiffrer, $C_i$ est le $i^{\text{ème}}$ bloc chiffré et $K_{Enc}$ est la clef de chiffrement.

La fonction de déchiffrement $Dec$ prend en entrée un bloc chiffré $C_i$ et la clef $K_{Enc}$, et produit en sortie un bloc de données en clair $P_i$. L'équation de déchiffrement est la suivante :

$$P_i = Dec(C_i, K_{Enc})$$

La fonction de chiffrement $Enc$ présentée à l'Equation 2.1 est déterministe, *i.e.* deux messages identiques auront les mêmes chiffrés. Cela pose un problème de sécurité car un attaquant peut retrouver le message en clair avec une attaque à message clair choisi [152]. Il est donc important d'introduire de l'aléa dans ce chiffrement, pour que chiffrer le même message clair deux fois produise deux chiffrés totalement différents. Lorsque cette propriété de sécurité connue sous le terme IND-CPA ("indistinguishability against Chosen Plaintext Attack"), le chiffrement est dit sémantiquement sûr. Ainsi, même si un attaquant devine le message en clair, il ne peut pas vérifier son hypothèse. Pour assurer cette propriété, plusieurs modes de chiffrement peuvent être utilisés. Ils spécifient comment les blocs de données sont combinés, chiffrés et déchiffrés. Le mode de chiffrement CBC ("Cipher Block Chaining") [153], présenté en Figure 2.4, est l'un des plus utilisés. Il repose sur l'utilisation d'un vecteur d'initialisation (VI) aléatoire et unique à chaque chiffrement, qui fournit une forme de "hasard" indépendant du message à chiffrer. Pour chiffrer le premier bloc $P_1$ d'un message, une opération XOR (ou exclusif) est effectuée avec ce bloc $P_1$ et le VI, puis ce résultat est chiffré avec $Enc$ en un bloc $C_1$. Ensuite, le chiffrement du bloc $P_2$ du message commence par un XOR entre $P_2$ et $C_1$, le bloc chiffré précédemment. Le bloc obtenu est ensuite chiffré avec $Enc$. Un bloc de chiffrement $C_i$ est donc obtenu par l'opération suivante :

$$C_i = Enc(P_i \oplus C_{i-1}, K_{Enc}) \tag{2.2}$$

où : l'opérateur $\oplus$ correspond à l'opération XOR ; $P_i$ est le $i^{\text{ème}}$ bloc du message en clair ; $C_{i-1}$ est le bloc chiffré précédent et $K_{Enc}$ est la clef de chiffrement. Pour obtenir le premier bloc chiffré $C_1$, le chiffrement $Enc$ est appliqué au résultat du XOR entre le premier bloc en clair $P_1$ et un Vecteur d'Initialisation $VI$.

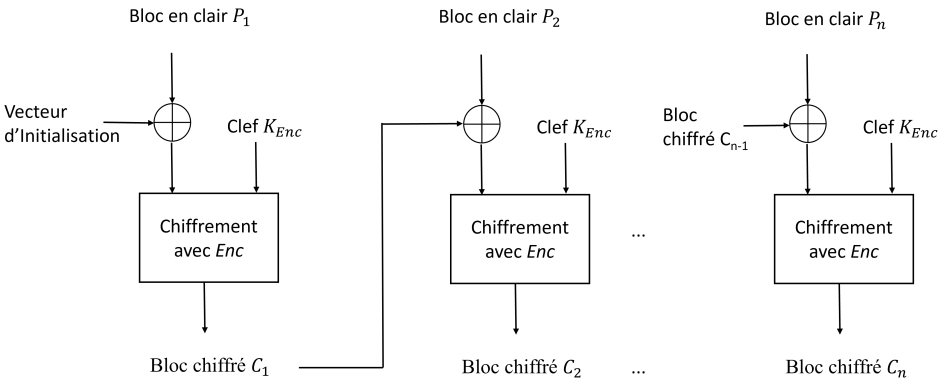



FIGURE 2.4 – Mode de chiffrement CBC ("Cipher Block Chaining") de blocs en clair $P_i$ en blocs chiffrés $C_i$, avec la clef secrète $K_{Enc}$.

Dans ce travail, à des fins d'expérimentation, nous avons opté pour le chiffrement Advanced Encryption Standard (AES) [136]. Il a été choisi par le "National Institute of Standards and Technologies" (NIST) et la "National Security Agency" (NSA), deux agences américaines, à l'issue d'un processus de normalisation en 2001 pour être l'algorithme de chiffrement symétrique standard. Il l'est toujours aujourd'hui. AES est un cryptosystème de chiffrement symétrique par blocs qui chiffre un bloc $P_i$ de 128 bits du message en clair en un bloc chiffré $C_i$ de même taille avec une clé de chiffrement $K_{AES}$ de 128, 192 ou 256 bits. Pour AES en mode CBC, le chiffrement fonctionne comme présenté à l'Equation 2.2, c'est-à-dire : $C_i = Enc(P_i \oplus C_{i-1}, K_{AES})$. AES étant un algorithme symétrique, la même clef est utilisée pour chiffrer et déchiffrer le message. En conséquence, si AES est utilisé pour sécuriser des communications, cette clef doit tout d'abord être transmise au destinataire via un protocole sécurisé d'échange de clefs. Un des avantages du chiffrement AES est la légèreté de l'algorithme, dans le sens où il est peu complexe et qu'il ne nécessite pas une grande quantité de mémoire. Actuellement, ce chiffrement est utilisé avec des clefs de 128 ou 256 bits pour résister aux attaques.

Comme les blocs en sortie de l'AES sont indépendants et identiquement distribués, il n'est pas possible de contrôler la structure des données chiffrées. En particulier, si l'on applique un encodage classique de l'ADN associant deux bits à une base d'ADN, chaque valeur de nucléotide $\{A, C, G, T\}$ a une probabilité de 25% d'apparaître à n'importe quelle position dans une séquence. Si cela permet de préserver le taux de GC, les risques d'homopolymères ne sont pas écartés. Un couple de nucléotides a une probabilité de 25% d'être une répétition de la même base, et les probabilités de générer à partir d'un flux binaire chiffré un homopolymère de 3 ou 4 bases sont de 6.25% et 1.56%, respectivement. Ces probabilités sont non négligeables lorsqu'il s'agit de coder des téraoctets de données chiffrées, comme l'envisage la technologie de stockage de l'ADN. Ainsi, pour contrôler la taille des homopolymères et l'apparition de motifs interdits, les données chiffrées doivent être encodées à l'aide d'un encodage de l'ADN avec contraintes, que nous décrivons dans la section suivante. Notons que notre solution d'encodage utilise des dictionnaires du binaire à l'ADN choisis pour préserver la distribution uniforme des données en entrée, et contrôler ainsi le taux de GC des séquences ADN encodées.

## 2.3 DSWE

Afin de prendre en compte les contraintes de motifs interdits et d'homopolymères tout en préservant un taux de GC équilibré, nous avons développé une solution d'encodage dynamique. Celle-ci prend en entrée un flux binaire chiffré et l'encode en une séquence ADN conforme aux contraintes énoncées plus haut.

Comme présenté en Figure 2.5, l'encodage dynamique à fenêtre glissante que nous proposons, appelé Dynamic Sliding Window Encoding (DSWE), consiste à appliquer itérativement deux encodages ADN à longueur variable : l'encodage dynamique (ou DE, *i.e.* "Dynamic Encoding") et l'encodage à fenêtre glissante (ou SWE, *i.e.* "Sliding Window Encoding"). Rappelons que DE et SWE permettent d'éviter les homopolymères plus longs que $N$ bases et l'apparition de motifs interdits de $m$ bases d'un dictionnaire $P_m$, respectivement.

Dans ce qui suit, nous décrivons tout d'abord l'encodage dynamique DE, avant de présenter l'encodage à fenêtre glissante SWE et son interaction avec DE pour obtenir notre solution DSWE.

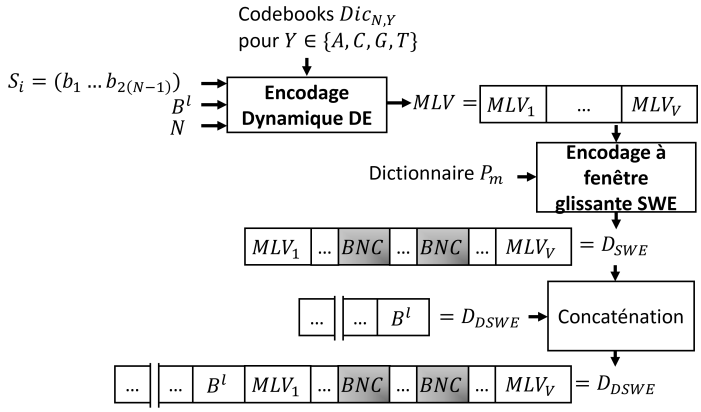



FIGURE 2.5 – Encodage dynamique avec fenêtre glissante DSWE d'un bloc binaire $S_i$. Tout d'abord, $S_i$ est encodé en un mot de code ADN à longueur variable $MLV$ avec l'encodage dynamique DE, en prenant en compte la dernière base encodée $B^l$ de la séquence ADN $D_{DSWE}$, la taille maximale d'homopolymère $N$ et les codebooks $Dic_{N,Y}$, avec $Y \in \{A, C, G, T\}$. Ensuite, l'encodage à fenêtre glissante SWE est appliqué, entrainant l'ajout ou non de bases non-codantes (des BNC) pour éviter des motifs interdits du dictionnaire $P_m$. Le mot de code ADN obtenu $D_{SWE}$ est concaténé à la séquence ADN déjà encodée $D_{DSWE}$.

### 2.3.1 Dynamic Encoding avec dictionnaires préservant la distribution uniforme

L'idée de base de notre solution d'encodage dynamique DE est de convertir des données binaires en mots code ADN en base 4, tout en évitant les homopolymères plus longs que $N$ bases. Notre solution d'encodage est conçue pour optimiser le taux d'information, c'est-à-dire le nombre de bits encodés dans une base ADN. Pour ce faire, des blocs binaires de taille fixe sont encodés en des mots code ADN de longueur variable. La taille de ces mots de code dépend de la dernière base d'ADN encodée, c'est-à-dire la dernière base du mot de code ADN précédent. Plus clairement, le DE repose sur l'utilisation de quatre dictionnaires ou codebooks, dont l'utilisation dépend de la dernière base encodée, *i.e.* ($\{A, C, G, T\}$).

Chaque dictionnaire contient des mots codes ADN de longueur variable (MLV) qui ont été conçus de telle sorte que, s'ils sont concaténés avec la dernière base ADN codée, il n'est pas possible de générer un homopolymère d'une taille supérieure à $N$. Chaque dictionnaire permet d'encoder le même nombre de messages ; *i.e.* dans le cas d'un message de $b$ bits à encoder, chaque dictionnaire contient $2^b$ mots-codes ADN. DE permet donc un encodage direct du binaire à l'ADN contrairement à d'autres stratégies [18] [140] qui appliquent d'abord un codage binaire à ternaire suivi d'un encodage Goldman qui convertit les symboles ternaires en bases ADN.

#### 2.3.1.1 Méthode d'encodage

Décrivons maintenant notre encodage dynamique. Comme expliqué plus haut, notre idée de base est d'encoder des données binaires de manière dynamique en utilisant différents dictionnaires ou codebooks en fonction de la dernière base de la séquence d'ADN qui a déjà été codée. Un tel dictionnaire associe directement des blocs binaires de longueur fixe à des Mots code ADN de Longueur Variable, que l'on appellera $MLV$. Pour toute valeur $N > 2$, les mots code ADN ont été conçus de telle sorte que, s'ils sont concaténés avec la dernière base ADN encodée, il n'est pas possible de générer un homopolymère d'une taille supérieure à $N$.

Notre encodeur dynamique DE fonctionne comme suit. Il utilise quatre dictionnaires ou codebooks, un pour chaque valeur possible de la dernière base ADN encodée $B^l$, *i.e.* $B^l \in \{A, C, G, T\}$. Un exemple de tels dictionnaires est présenté en Figure 2.6B pour une taille maximale d'homopolymères $N = 2$. DE prend en entrée une séquence binaire, la divise en blocs binaires de $2(N-1)$ bits et encode chaque bloc en un mot de code ADN à l'aide de l'un des dictionnaires. L'encodage des deux premiers bits, *i.e.* l'initialisation de la séquence ADN, se fait toujours avec l'encodage classique qui associe deux bits à une base (*cf.* Table 2.6 A).

En particulier, DE encode un bloc binaire $S_k$ de $2(N-1)$ bits $(b_1^k b_2^k ... b_{2(N-1)}^k)$ en un mot de code ADN de longueur variable $D_k$ de $N-1$ ou $N$ bases, c'est-à-dire $D_k = (B_1^k B_2^k ... B_{N-1}^k)$ ou $(B_1^k B_2^k ... B_N^k)$. Pour ce faire, le dictionnaire $Dic_{N,B^l}$ est choisi parmi les quatre possibilités, à savoir $Dic_{N,A}$, $Dic_{N,C}$, $Dic_{N,G}$ et $Dic_{N,T}$. Le Mot code ADN à Longueur Variable appelé MLV est ensuite concaténé à la séquence ADN déjà encodée. En ce qui concerne l'initialisation de l'encodage, c'est-à-dire l'encodage de la première base de la séquence ADN, nous utilisons l'encodage classique de l'ADN. Celui-ci associe deux bits à une base ADN comme présenté en Figure 2.6A.

Pour expliquer comment construire les codebooks, considérons le cas de $Dic_{N,A}$ utilisé pour éviter plus de $N$ répétitions du nucléotide $A$. Dans ce cas, les blocs binaires en entrée sont de $2(N-1)$ bits. Dans un premier temps, $Dic_{N,A}$ contient tous les mots code ADN encodés classiquement de taille $N-1$, dont la première base n'est pas $A$. Ces mots code ne causeront jamais un homopolymère s'ils sont concaténés avec la dernière base codée, *i.e.* $B^l = A$. Dans un second temps, certains mots code ADN classiques commençant par la base $A$ sont réencodés en mots code ADN de longueur variable afin d'éviter les homopolymères $(A...A)$. Pour ce faire, ces mots de code sont construits de telle sorte que $B_1^k = A$ et $B_2^k \in \{C, G, T\}$. Ainsi, $B^l = A$ ne sera pas répété plus d'une fois au début du mot de code. Pour générer ces mots de code en s'assurant qu'un mot de code n'est pas le préfixe d'un autre (afin de rendre possible le décodage sans ambiguïté), nous proposons un algorithme simple. Celui-ci tire parti des quatre codebooks $Dic_{2,B^l}$ représentés dans la Figure 2.6B qui évitent tous homopolymères. Simplement, la deuxième base

**A**

Encodage Classique

| Blocs binaires | Mots de code ADN classiques |
|---|---|
| 00 | A |
| 01 | C |
| 10 | T |
| 11 | G |

**B**

Dictionnaire $Dic_{2,A}$

| Blocs binaires | Mots de code ADN à longueur variable |
|---|---|
| 00 | CG |
| 01 | CT |
| 10 | T |
| 11 | G |

Dictionnaire $Dic_{2,C}$

| Blocs binaires | Mots de code ADN à longueur variable |
|---|---|
| 00 | AT |
| 01 | AG |
| 10 | T |
| 11 | G |

Dictionnaire $Dic_{2,T}$

| Blocs binaires | Mots de code ADN à longueur variable |
|---|---|
| 00 | A |
| 01 | C |
| 10 | GC |
| 11 | GA |

Dictionnaire $Dic_{2,G}$

| Blocs binaires | Mots de code ADN à longueur variable |
|---|---|
| 00 | A |
| 01 | C |
| 10 | TA |
| 11 | TC |

FIGURE 2.6 – Dictionnaires ou codebooks pour encoder les séquences ADN de façon à empêcher les homopolymères de taille 2, *i.e.* tous homopolymères. **A.** Dictionnaire de l'Encodage Classique utilisé pour encoder la première base de la séquence ADN. **B.** Codebooks $Dic_{N=2,B^l}$ dépendants de $B^l$, la dernière base de la séquence ADN déjà encodée. Les mots code ADN encodés classiquement sont encadrés en vert, tandis que ceux encadrés en rouge sont ré-encodés pour éviter la création d'homopolymères.

des mots code ADN commençant par $(AA)$ et $(AC)$, c'est-à-dire les bases $A$ et $C$, sera remplacée par les deux bases $CG$ et $CT$ respectivement, selon le dictionnaire $Dic_{2,A}$. Cela conduit à des mots code ADN commençant par $(ACG)$ et $(ACT)$, respectivement. Le remplacement de la deuxième base des mots de code par deux bases garantit qu'aucun mot ne sera le préfixe d'un autre, tout en assurant qu'aucun mot ne commence par un homopolymère $A$, quelque soit $N > 2$. En effet, les mots code ADN de $N-1$ bases peuvent être des homopolymères, mais ils sont concaténés avec un mot codé commençant par au plus une base en commun avec le bloc précédent. Cela garantit qu'aucun homopolymère d'une longueur supérieure à $N$ n'est créé lors de la concaténation de deux mots de code par DE. La Figure 2.7 présente les règles de ré-encodage des mots code ADN classiques afin d'éviter les homopolymères dans les dictionnaires $Dic_{N,B^l}$. Cette solution optimise la longueur des mots code ADN, car la plupart d'entre eux sont codés à l'aide de l'encodage ADN classique. Un exemple est illustré à la Figure 2.8 pour $N = 3$, les quatre codebooks encodent 4 bits en des mots code ADN classiques de 2 bases ou des mots de code ré-encodés de 3 bases, dans les cases vertes et rouges respectivement.

Codebook $Dic_{N,B^l=A}$

| Encodage Classique de $N-1$ bases | Mots de codes ADN *MLV* de $N$ bases |
|---|---|
| $AAB_2^k...B_{N-2}^k$ | $ACGB_2^k...B_{N-2}^k$ |
| $ACB_2^k...B_{N-2}^k$ | $ACTB_2^k...B_{N-2}^k$ |

Codebook $Dic_{N,B^l=C}$

| Encodage Classique de $N-1$ bases | Mots de codes ADN *MLV* de $N$ bases |
|---|---|
| $CAB_2^k...B_{N-2}^k$ | $CATB_2^k...B_{N-2}^k$ |
| $CCB_2^k...B_{N-2}^k$ | $CAGB_2^k...B_{N-2}^k$ |

Codebook $Dic_{N,B^l=T}$

| Encodage Classique de $N-1$ bases | Mots de codes ADN *MLV* de $N$ bases |
|---|---|
| $TTB_2^k...B_{N-2}^k$ | $TGCB_2^k...B_{N-2}^k$ |
| $TGB_2^k...B_{N-2}^k$ | $TGAB_2^k...B_{N-2}^k$ |

Codebook $Dic_{N,B^l=G}$

| Encodage Classique de $N-1$ bases | Mots de codes ADN *MLV* de $N$ bases |
|---|---|
| $GTB_2^k...B_{N-2}^k$ | $GTAB_2^k...B_{N-2}^k$ |
| $GGB_2^k...B_{N-2}^k$ | $GTCB_2^k...B_{N-2}^k$ |

FIGURE 2.7 – Ré-encodage de certains mots code ADN dans les codebooks $Dic_{N,B^l}$ afin qu'aucun mot de code ne commence par $(B^lB^l)$. Les dictionnaires encodent les blocs binaires avec l'Encodage Classique, sauf pour les mots code ADN commençant par $B^l$ et une base en rouge. Ces bases rouges sont ré-encodées en bases en vert, en utilisant $Dic_{N=2,B^l}$ *cf.* Figure 2.6.

Expliquons maintenant notre solution d'encodage pour une longueur maximale d'homopolymère de $N = 3$, pour éviter que plus de trois nucléotides successifs ne prennent la même valeur. L'initialisation de la

séquence ADN, c'est-à-dire l'encodage de la première base, est effectuée à l'aide de l'encodage classique de l'ADN, présenté en Figure 2.6A. Ensuite, comme le montre la Figure 2.8, notre encodeur prend en entrée : un bloc binaire $S_k$ de $2(N-1) = 4$ bits et la dernière base ADN $B^l$ de la séquence ADN déjà encodée. Il produit un mot de code ADN $D_k$ de longueur variable entre $N-1 = 2$ et $N = 3$ bases, *i.e.* $D_k = (B_1^k B_2^k)$ ou $D_k = (B_1^k B_2^k B_3^k)$, avec $B_p^k \in \{A, C, G, T\}$. $D_k$ est ensuite concaténé à la séquence d'ADN. $S_k$ est encodé en $D_k$ à l'aide de $Dic_{3,B^l}$, choisi parmi les quatre dictionnaires $\{Dic_{3,A}, Dic_{3,C}, Dic_{3,G}, Dic_{3,T}\}$ donnés dans la Figure 2.8.

Dictionnaire $Dic_{3,A}$

| Blocs binaires | Mots de code ADN à longueur variable |
|---|---|
| 0000 | ACG |
| 0001 | ACT |
| 0010 | AT |
| 0011 | AG |
| 0100 | CA |
| 0101 | CC |
| 0110 | CT |
| 0111 | CG |
| 1000 | TA |
| 1001 | TC |
| 1010 | TT |
| 1011 | TG |
| 1100 | GA |
| 1101 | GC |
| 1110 | GT |
| 1111 | GG |

Dictionnaire $Dic_{3,C}$

| Blocs binaires | Mots de code ADN à longueur variable |
|---|---|
| 0000 | AA |
| 0001 | AC |
| 0010 | AT |
| 0011 | AG |
| 0100 | CAT |
| 0101 | CAG |
| 0110 | CT |
| 0111 | CG |
| 1000 | TA |
| 1001 | TC |
| 1010 | TT |
| 1011 | TG |
| 1100 | GA |
| 1101 | GC |
| 1110 | GT |
| 1111 | GG |

Dictionnaire $Dic_{3,T}$

| Blocs binaires | Mots de code ADN à longueur variable |
|---|---|
| 0000 | AA |
| 0001 | AC |
| 0010 | AT |
| 0011 | AG |
| 0100 | CA |
| 0101 | CC |
| 0110 | CT |
| 0111 | CG |
| 1000 | TA |
| 1001 | TC |
| 1010 | TGC |
| 1011 | TGA |
| 1100 | GA |
| 1101 | GC |
| 1110 | GT |
| 1111 | GG |

Dictionnaire $Dic_{3,G}$

| Blocs binaires | Mots de code ADN à longueur variable |
|---|---|
| 0000 | AA |
| 0001 | AC |
| 0010 | AT |
| 0011 | AG |
| 0100 | CA |
| 0101 | CC |
| 0110 | CT |
| 0111 | CG |
| 1000 | TA |
| 1001 | TC |
| 1010 | TT |
| 1011 | TG |
| 1100 | GA |
| 1101 | GC |
| 1110 | GTA |
| 1111 | GTC |

FIGURE 2.8 – Dictionnaires $Dic_{N=3,B^l}$ encodant 4 bits en des mots code ADN à longueur variable de 2 or 3 bases, pour le cas $N = 3$. Les mots code ADN encodés classiquement sont encadrés en vert, tandis que les mots code en rouge sont les mots code créés pour éviter les homopolymères.

Ainsi, si $B^l = A$, l'encodeur utilisera le dictionnaire $Dic_{3,A}$, où l'on peut voir que les mots code peuvent commencer par au plus un $A$, mais pas par la séquence $AA$. Le respect de la même règle avec les autres valeurs de nucléotides $B^l$ garantit que nous n'encoderons jamais un homopolymère de taille supérieure à $N = 3$.

Une remarque importante est que la concaténation de mots code ADN n'évite pas les homopolymères de taille 3. Prenons l'exemple où $B^l = A$ et deux blocs binaires successifs à encoder tels que : $S_k = (0101)$ et $S_{k+1} = (0101)$. $S_k$ et $S_{k+1}$ sont encodés respectivement en $D_k = (CC)$ et $D_{k+1} = (CAG)$, et leur concaténation conduira à un homopolymère de taille 3 : $(CCCAG)$. Par contre, il n'y aura pas de risque d'homopolymères de taille 4.

Pour illustrer l'encodage DE avec $N = 3$, nous présentons un exemple d'encodage et de décodage d'une séquence binaire $S = (0111000001)$ en une séquence d'ADN $D$ dans la figure 2.9A. Pour initialiser la séquence ADN, les deux premiers bits de $S$, $(b_1^k b_2^k)$, sont codés avec l'encodage classique en la base C. Ensuite, les blocs de $2(N-1) = 4$ bits sont encodés à partir des quatre codebooks $Dic_{3,B^l}$. La séquence ADN encodée qui en résulte est $D = (CGAACT)$. La figure 2.9B présente le décodage de $D$. La première base de $D$ est décodée à l'aide du codage classique de l'ADN, tandis que le reste des nucléotides est décodé à l'aide de notre décodeur dynamique informé.

Notre encodage dynamique DE se fait donc en deux étapes :

— Initialisation : Le bloc binaire $S_1$, composé des deux premiers bits de la suite binaire $S$, est encodé

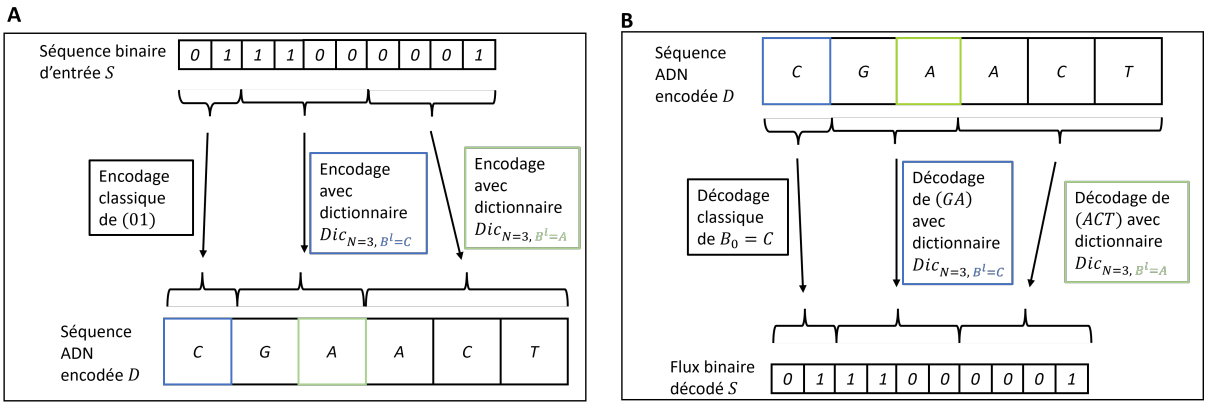



FIGURE 2.9 – Exemple de l'Encodage et du Décodage Dynamique pour $N = 3$. L'initialisation se fait toujours avec l'Encodage ou le Décodage Classique. **A.** Encodage d'une séquence binaire $S$ avec les dictionnaires $Dic_{N=3,B^l}$ en une séquence ADN $D$. **B.** Décodage d'une séquence ADN $D$ avec les dictionnaires $Dic_{N=3,B^l}$.

en une base ADN $D_1$ avec l'Encodage Classique présenté en Figure 2.6A.

— Encodage : Les bits suivants de la suite $S$ sont divisés en blocs $S_k$ de $2(N-1)$ bits. Chaque bloc $S_k$ est encodé en un mot code ADN de longueur variable $D_k$ de $N-1$ ou $N$ bases ADN. Pour cela, le dictionnaire $Dic_{N,B^l}$ est déterminé selon la dernière base encodée $B^l$ du mot code ADN précédent $D_{k-1}$. Le mot code obtenu est ensuite concaténé à la séquence ADN déjà encodée.

En dernier lieu, nous voudrions insister sur la différence entre notre proposition et les solutions basées sur la combinaison d'un encodage binaire à ternaire avec l'encodage de Goldman comme proposé dans [18], [140] et [145]. Ces solutions utilisent des algorithmes de codage entropique, tels que ceux de Huffman ou de Shannon-Fano, pour convertir les messages binaires en messages ternaires. Ces algorithmes sont dédiés à la compression de données sans perte. Ils construisent et utilisent une table de recherche de mots code de longueur variable pour coder des messages binaires de longueur fixe. C'est la source de la variabilité de la taille des séquences d'ADN codées avec les approches [18], [140], et [145]. Ceux-ci ne se contentent pas d'encoder les données de manière ternaire, ils les compressent également. Ce n'est pas le cas de DE. Ces approches encodent les données en base ternaire, puis appliquent le codage de Goldman, qui ne peut pas être utilisé pour coder directement des données binaires. En effet, comme le montre la Table Figure 2.2B, il ne respecte pas la règle du préfixe qui exige qu'aucun mot code ne soit le préfixe d'un autre mot code. Si cette règle n'est pas respectée, des ambiguïtés apparaissent lors du décodage. Prenons un exemple. Supposons que la dernière base codée soit $A$. Les messages binaires (00), (01) et (10) sont codés respectivement par $T$, $C$ et $G$. Quant au message (11), il sera codé dans deux bases $(CT)$. Cela provoquera une ambiguïté à l'étape du décodage car celui-ci commence par le nucléotide $C$ qui code également (01). Le décodeur ne saura pas s'il doit décoder $CT$ comme (11) ou comme (0101) (voir Fig.2.2B). DE satisfait cette règle de préfixe et peut donc être utilisé pour encoder directement des messages binaires en une séquence ADN, sans utiliser de compression de données. Comme nous le verrons, DE peut toutefois coder des données compressées, ce qui permet d'obtenir des performances meilleures ou équivalentes à celles de [18], [140] et [145] en termes de taux d'information.

#### 2.3.1.2 Calcul du taux d'information

Comme nous encodons des données chiffrées, qui, par définition, sont uniformément distribuées (*i.e.* $Pr(b_i^k = 0) = Pr(b_i^k = 1) = \frac{1}{2}$), il est possible d'estimer le taux d'information moyen $R$ de cette stratégie. $R$ est le nombre moyen de bits d'information encodés par base ADN. Sur cette base, toutes les valeurs des blocs binaires sont équiprobables. Il en va alors de même pour chaque mot de code ADN dans chaque dictionnaire. Le taux d'information moyen $R$ peut alors être défini comme suit :

$$R = \frac{|S_k|}{\sum_i p_i \times l_i} \tag{2.3}$$

où : $|S_k|$ correspond à la taille en bits du bloc binaire $S_k$ ; $p_i$ et $l_i$ sont les probabilités d'occurrence et la taille en bases du $i^{\text{ème}}$ mot de code ADN à longueur variable, respectivement. Pour $N = 2$, avec un bloc binaire $S_k$ de 2 bits nous avons $p_i = 1/4$, et 2 mots de codes de taille 2 ainsi que 2 de taille 1, qui conduisent à un taux d'information moyen de $R_{N=2} = 1,33$ BPB.

Comme le taux d'information décrit correspond au cas moyen, définissons les scénarios du pire et du meilleur cas. Ceux-ci correspondent aux cas où les mots code ADN sont toujours de taille $N$ et $N - 1$, respectivement. Ces taux ont une probabilité extrêmement faible d'apparaître pour des données uniformément distribuées, comme nous allons le montrer pour le cas $N = 3$. Dans le pire des cas, où un mot de code ADN MLV ayant la plus grande taille $l_{max} = N$ est toujours choisi , le taux $R_{pire}$ obtenu est :

$$R_{pire} = \frac{|S_k|}{l_{max}} \tag{2.4}$$

A l'opposé, dans le meilleur cas, où les mots de code à longueur variable ont toujours la plus courte taille possible $l_{min} = N - 1$, le meilleur taux d'information possible est défini par :

$$R_{meilleur} = \frac{|S_k|}{l_{min}} \tag{2.5}$$

Dans les mêmes conditions, il est possible d'estimer le taux d'information moyen pour toutes les valeurs de $N > 2$ dans le cas où l'on encode des données uniformément distribuées. Considérant les quatre dictionnaires $\{Dic_{3,A}, Dic_{3,C}, Dic_{3,G}, Dic_{3,T}\}$, des blocs de $2(N-1)$ bits sont encodés en un mot code ADN de $N - 1$ ou $N$ bases. En particulier et comme le montre la Figure 2.7, $\frac{7}{8}^{\text{èmes}}$ des blocs binaires sont encodés en $N - 1$ bases, et $\frac{1}{8}^{\text{ème}}$ en $N$ bases. Les données binaires étant équiprobables, le taux d'information théorique moyen (en BPB) de DE pour toute valeur de $N > 2$, calculé à partir de l'équation 2.3, est le suivant :

$$R^T_{DE(N)} = \frac{2 * (N-1)}{(N * \frac{1}{8} + (N-1) * \frac{7}{8})} = \frac{16 * (N-1)}{8 * N - 7} \tag{2.6}$$

Ainsi, le taux d'information moyen pour $N = 3$ est de l'ordre de $R^T_{DE(N=3)} = 1,882$ BPB, avec deux tailles de mots de code ADN $(N - 1) = 2$ et $N = 3$ bases.

Également, nous pouvons déterminer le pire taux d'information, où tous les blocs binaires sont encodés en mots code de $N$ bases, et le meilleur cas où les blocs sont codés en des mots code de $(N - 1)$ bases.

Partant de l'Équation 2.4, le pire taux d'information de DE selon N est défini par :

$$R_{pire,DE(N)} = \frac{2(N-1)}{N} \tag{2.7}$$

Et, comme défini par l'Équation 2.5, le meilleur taux possible est :

$$R_{meilleur,DE(N)} = \frac{2(N-1)}{N-1} = 2 \tag{2.8}$$

Dans ce dernier cas, le taux d'information idéal de 2 BPB est atteint.

Dans le cas $N = 3$, nous obtenons $R_{pire,DE(N=3)} = \frac{4}{3} = 1.33$ BPB et $R_{meilleur,DE(N=3)} = \frac{4}{2} = 2$ BPB. La probabilité de rencontrer la situation de pire cas est infime dans le cas de données uniformément distribuées à encoder. En effet, la probabilité que $q$ mots code consécutifs soient des mots de 3 bases est de $(\frac{1}{8})^q$, probabilité qui décroît très rapidement. Par exemple, dans le cas $q = 5$, cette probabilité est de 0,003%. A l'inverse, la probabilité est plus grande d'approcher le meilleur taux d'information possible, *i.e.* $R_{meilleur} = 2$ BPB. Par exemple pour DE(N=3), la probabilité que $q = 10$ mots consécutifs soient de taille 2 est de 26,3%.

Notons aussi que nos dictionnaires ont été choisis pour préserver la distribution uniformes des données chiffrées d'entrée. En effet, les dictionnaires $Dic_{N,B^l}$ sont constitués du même nombre de bases A,C, G et T. Pour cela, les données sont encodées de deux façons. Elles peuvent être encodées avec l'encodage classique qui préserve la distribution uniforme. Comme présenté en Figure 2.6 A, chaque base est présente le même nombre de fois. De plus, l'encodage classique donne aussi un taux de GC équilibré lorsqu'une séquence binaire est une suite de (01) et (10), ou une suite de (00) et (11). Ces suites correspondent respectivement à des suites de C et T, ou de A et G, ce qui garantit un taux de GC équilibré. Les données peuvent aussi être ré-encodées tel que présenté en Figure 2.7. Ce ré-encodage change une base d'un mot encodé classiquement par deux bases en s'assurant que, sur l'ensemble des dictionnaires, la proportion de chaque base est égale. Ainsi, des données binaires uniformément distribuées sont encodées en séquences ADN avec un taux de GC équilibré. Cet équilibre des bases dans les séquences encodées par DSWE est testé dans la partie expérimentale.

Nous avons ainsi présenté la méthode d'encodage dynamique DE de blocs binaires en mots de codes ADN à longueur variable , qui permet de limiter les homopolymères à $N$ bases. Dans la section suivante, nous présentons l'encodage à fenêtre glissante SWE et comment il s'associe au DE.

### 2.3.2 Encodage à fenêtre glissante SWE

L'objectif de ce schéma d'encodage à fenêtre glissante (SWE *i.e.* "Sliding Window Encoding") est de générer des séquences ADN sans créer de motifs interdits, tels que des amorces par exemple.

Considérons une séquence binaire $S$ progressivement encodée en une séquence ADN $D_{SWE}$ avec le codage classique et un dictionnaire $P_m$ qui contient $M$ motifs interdits de $m$ nucléotides. L'objectif de SWE est d'éviter que de tels motifs n'apparaissent lors de l'ajout à $D_{SWE}$ d'une nouvelle base $B_j$. Pour ce faire, nous proposons d'utiliser une fenêtre glissante $W$ qui contient les $m-1$ nucléotides précédemment encodés $(B_{j-m+1}...B_{j-1})$. Si $W$ n'est pas le préfixe d'un motif interdit, $B_j$ peut être ajoutée. Dans le cas contraire, $B_j$ doit prendre une valeur de base ADN qui évite l'apparition d'un motif non autorisé. Nous

appelons une telle base une base non codante (BNC). Une fois que $B_j$ est ajouté à $D_{SWE}$, la fenêtre glissante avance d'une base dans $D_{SWE}$.

De son côté, le décodeur suivra la même stratégie. Avant de considérer que la valeur $B_j$ qu'il observe encode des données, il vérifiera si les $m-1$ bases précédentes correspondent aux premières bases d'un motif interdit. Si c'est le cas, alors $B_j$ est une BNC, elle ne code pas d'information. Cette stratégie garantit ainsi que certains motifs spécifiques n'apparaissent pas. Telle qu'elle est définie, l'approche SWE est également une approche d'encodage dynamique puisqu'elle produit des séquences ADN de longueur variable.

Nous donnons une représentation schématiques de l'encodage et du décodage SWE en Figure 2.10 A et B, respectivement.

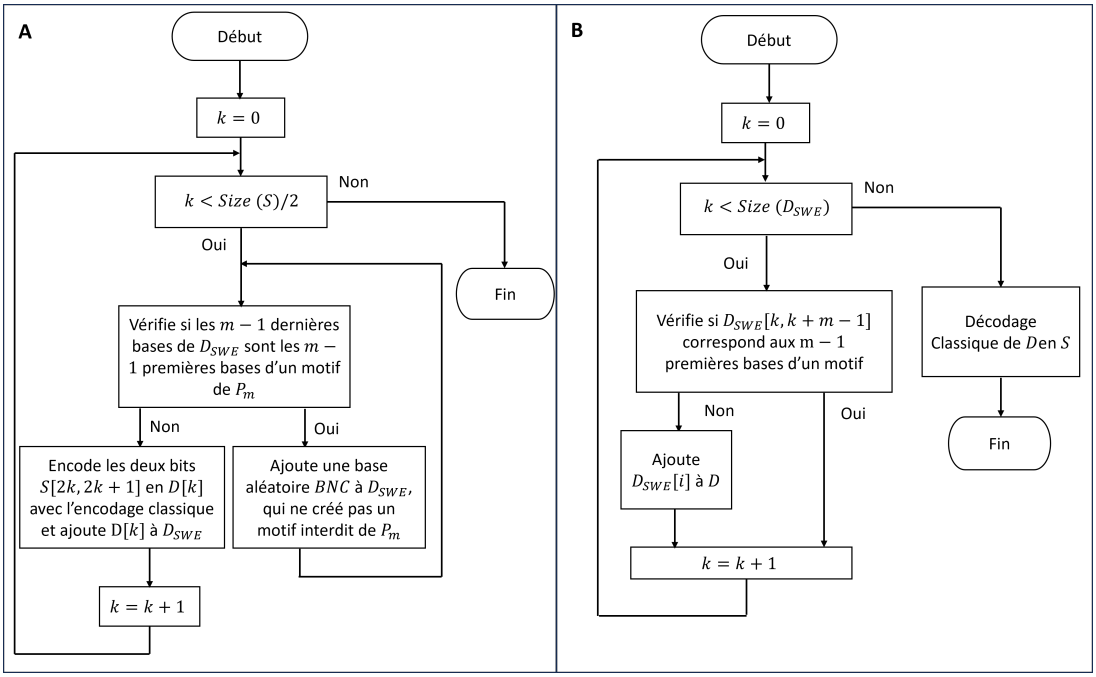



FIGURE 2.10 – Encodage avec fenêtre glissante SWE. A) Procédé d'encodage SWE d'une suite binaire $S$ en une séquence ADN $D_{SWE}$ sans motifs interdits du dictionnaire $P_m$. B) Procédé de décodage SWD.

Il est important de noter que cette approche contraint les motifs interdits possibles, du fait qu'elle s'assure que le dernier nucléotide d'un motif interdit n'est pas encodé. Considérons $M$ motifs interdits de taille $m$ : $M_{u(u=1...M)} = (B_1^u, ..., B_{m-1}^u, B_m^u)_{u=1...M}$. Une fois les valeurs des $m-1$ bases de poids fort $B_1^u$ ,...,$B_{m-1}^u$ fixées, on ne peut générer que trois motifs interdits de sorte que $B_m^u$ ne puisse prendre qu'une seule valeur non codante. C'est la seule contrainte imposée par SWE. Ainsi, pour une taille fixe de $m$ bases, jusqu'à $3 * 4^{m-1}$ motifs interdits peuvent être considérés.

La complexité de cette solution dépend de $M$ et $m$. En effet, plus les motifs interdits sont nombreux et longs, plus le nombre de tests à effectuer est important et plus la complexité est grande. Pour encoder une base, avec un dictionnaire $P_{mW}$, $|P_{mW}|$ tests doivent être effectués pour comparer la fenêtre glissante $W$ aux éléments du sous-dictionnaire $P_{mW}$. Parmi l'ensemble des $4^{m-1}$ séquences de $(m-1)$ bases, un nombre $|P_{mW}|$ de séquences de $m-1$ bases doivent être suivies d'une base non codante.

Calculons la probabilité qu'une base soit non codante. Elle est non codante lorsqu'elle est précédée d'un élément du sous-dictionnaire $P_{mW}$. De tels éléments représentent $|P_{mW}|$ parmi $4^{m-1}$ séquences de $(m-1)$ bases. Lorsque les données sont uniformément distribuées, il existe une probabilité de $\frac{|P_{mW}|}{4^{m-1}}$ qu'une base soit non codante. Si tous les motifs interdits ont des bases distinctes d'ordre élevé $m-1$, cette probabilité est égale à $\frac{M}{4^{m-1}}$.

### 2.3.3 Dynamic Sliding Window Encoding

Ce codage, noté $DSWE(N, M, m)$ combine l'encodage dynamique (DE), qui permet d'éviter la génération d'homopolymères d'une longueur supérieure à $N$, au SWE qui permet d'éviter $M$ séquences interdites de $m$ bases (contenues dans un dictionnaire $P_m$). DSWE consiste simplement a appliquer séquentiellement ces deux encodages.

#### 2.3.3.1 Méthode d'encodage

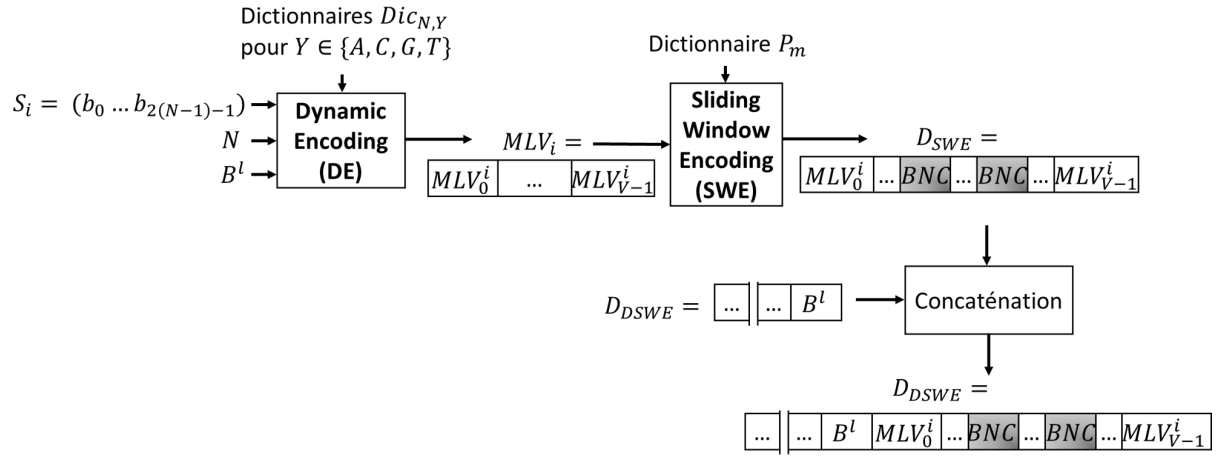



FIGURE 2.11 – Encodage dynamique avec fenêtre glissante DSWE d'un bloc binaire $S_i$. Tout d'abord, l'encodage dynamique DE est appliqué, pour obtenir un mot code ADN à longueur variable $MLV_i$ selon la taille maximale d'homopolymères $N$ et la dernière base encodée $B^l$ de la séquence ADN encodée $D_{DSWE}$. Ensuite, le mot code $MLV$ est encodé avec SWE, l'encodage à fenêtre glissante. Celui-ci entraîne ou non l'ajout de bases non codantes ($BNC$) pour éviter les motifs interdits du dictionnaire $P_m$. Enfin, le mot de code obtenu $D_{SWE}$ est concaténé à la séquence ADN déjà encodée $D_{DSWE}$.

Comme illustré en Figure 2.11, le codage d'un bloc binaire $S_i$ de $2(N-1)$ bits commence par l'encodage dynamique DE de $S_i$ en un mot code ADN de longueur variable $MLV_i$ de taille $N$ ou $(N-1)$ nucléotides, à l'aide de l'un des codebooks $\{Dic_{N,B^l}\}_{B^l=A,C,G,T}$ choisi en regardant la dernière base codée $B^l$. Considérant une séquence ADN de $d$ base déjà encodée : $D_{DSWE} = (D_1^{DSWE}, ..., D_d^{DSWE})$, la deuxième étape consiste à encoder $MLV_i$ avec l'encodage à fenêtre glissante SWE, en considérant le dictionnaire de $M$ motifs interdits $P_m$. Cela produit une séquence ADN à longueur variable $D_{SWE}$ où une ou plusieurs bases non-codantes ($BNC$) peuvent avoir été ajoutées si nécessaire. Enfin, $D_{SWE}$ est concaténé à la séquence déjà encodée $D_{DSWE}$. En résumé, les bases de $MLV$ sont ajoutées à $D_{DSWE}$ une par une en respectant l'encodage SWE. Ce dernier est appliqué jusqu'à ce que toutes les bases de $MLV$ aient été ajoutées à $D_{DSWE}$.

L'initialisation de DSWE est la même que pour DE et SWE. Les deux premiers bits de $S$ sont codés avec l'encodage classique. Une fois la séquence d'ADN $D_{DSWE}$ finalisée, elle est encadrée par des amorces et est prête pour la synthèse. Rappelons que les amorces font partie des motifs interdits.

**Algorithm 1:** Dynamic Sliding Window Encoding DSWE d'un bloc binaire $S_i$

**Inputs :** Le bloc binaire à encoder $S_i$, la séquence déjà encodée par DSWE $D_{DSWE}$, la taille maximale d'homopolymères $N$, le dictionnaire $P_m$ de $M$ motifs interdits de taille $m$
**Output :** La séquence ADN $D_{DSWE}$ contenant l'encodage par DSWE de $S_i$

---

% **Étape 1 : Encodage Dynamique DE de** $S_i$
$d = size(D_{DSWE}) - 1$ ;
% *Création de d, le dernier index de* $D_{DSWE}$ ;
$B^l = D_{DSWE,d}$ ;
% *Détermination de la dernière base encodée i.e. la dernière base de* $D_{DSWE}$ ;
$MLV_i = DE(S_i, N, B^l)$ ;
% *Encodage Dynamique de* $S_i$ *avec les 4 codebooks* $Dic_{N,Y}$ *avec* $Y \in A, C, G, T$ *pour déterminer un mot code ADN de longueur variable* $MLV_i$ *de* $N$ *ou* $N-1$ *bases* ;

% **Étape 2 : Encodage à fenêtre glissante SWE ("Sliding Window Encoding") du mot code ADN à longueur variable** $MLV_i$
$W = (D_{DSWE,d-m+2}, ..., D_{DSWE,d})$ ;
% *Création de la fenêtre glissante* $W$ *de* $m-1$ *bases* ;
$D_{SWE} = ()$ ;
% *Initialisation de* $D_{SWE}$ ;
$P_{mW} = SubDict(P_m, m)$ ;
% *Création d'un sous-dictionnaire* $P_{mW}$ *à partir de* $P_m$ *en prenant les* $m-1$ *bases d'ordre élevé de chaque motif de* $P_m$ ;
**for** $u \in \{0, \ldots, size(MLV_i) - 1\}$ **do**
    % *Parcours les bases du mot code* $MLV_i$ ;
    **while** $W \in P_{mW}$ **do**
        % *Teste si la fenêtre glissante est égale à un élément de* $P_{mW}$ ;
        $D_{SWE} = (D_{SWE}, BNC)$ ;
        % *Concaténation de* $D_{SWE}$ *avec une base non-codante BNC pour prévenir la génération d'un motif interdit* ;
        $W = (W_1, ..., W_{m-2}, BNC)$ ;
        % *Décale* $W$ *d'une base tout en lui concaténant BNC pour préserver la taille de la fenêtre glissante, i.e.* $(m-1)$ ;
    **end**
    $D_{SWE} = (D_{SWE}, MLV_u^i)$ ;
    % *Concaténation de* $MLV_u^i$*, la* $u^{eme}$ *base de* $MLV_i$*, avec* $D_{SWE}$ ;
    $W = (W_1, ..., W_{m-2}, MLV_u^i)$ ;
    % *Décale* $W$ *d'une base en lui concaténant* $MLV_u^i$ *pour préserver la taille de la fenêtre glissante, i.e.* $(m-1)$ ;
**end**
$D_{DSWE} = (D_{DSWE}, D_{SWE})$ ;
% *Concaténation du mot code obtenu par SWE* $D_{SWE}$ *à la séquence déjà encodée par DSWE* $D_{DSWE}$ ;

L'Algorithme 1 correspond à l'encodage $DSWE(N, M, m)$ d'un bloc binaire $S_i$ de $2(N-1)$ bits. Il prend en entrée une séquence binaire $S_i$ et une séquence ADN déjà encodée $D_{DSWE}$. En sortie, nous obtenons une séquence ADN qui correspond à la concaténation de $D_{DSWE}$ avec les nucléotides qui encodent $S_i$. Il repose sur deux fonctions principales :

— $DE(S_i, N, B^l)$ est l'Encodage Dynamique (DE) d'un bloc binaire $S_i$ avec le dictionnaire $Dic_{N,B^l}$, sélectionné selon $B^l$ qui est la dernière base encodée de $D_{DSWE}$, considérant une taille maximale d'homopolymères $N$. Cette fonction sort un mot code ADN de longueur variable de $N$ ou $N-1$ bases.

— $SubDict(P_m, m)$ est une fonction qui créé un sous-dictionnaire $P_{mW}$ à partir de $P_m$ pour regrouper les séquences de nucléotides qui correspondent aux $m-1$ bases d'ordre élevé des $M$ motifs de $P_m$. Puisque plusieurs motifs de $P_m$ peuvent avoir les mêmes $m-1$ premiers nucléotides, le nombre d'éléments de $P_{mW}$ est souvent plus petit que $M$.

#### 2.3.3.2 Méthode de décodage

Le processus de décodage de DSWE fonctionne de manière similaire. Supposons que DSWE cherche à décoder la $d^{\text{ème}}$ base $D_d$ de la séquence $D_{DSWE}$, ayant déjà décodé $d-1$ bases en une séquence binaire globale $S_{DSWD}$. Pour ce faire, le processus commence par appliquer le décodage SWE. Il prend comme entrée la fenêtre $W$ qui contient les $m-1$ bases qui précèdent $D_d$ dans $D_{DSWE}$. Si $W$ correspond aux $m-1$ bases de poids fort de l'un des motifs interdits de $P_m$, on considère que $D_d$ est une base non codante. Dans ce cas, DSWE passe au décodage de la base suivante. Dans le cas contraire $D_d$ porte de l'information et correspond à une base d'un mot de code ADN issu de l'encodage dynamique d'un bloc binaire. Un tel mot de code peut comporter $N$ ou $N-1$ bases. Son décodage DE dépend de la valeur de la base qui le précède dans $D_{DSWE}$ (c'est-à-dire la base $B^l$ du point de vue de l'encodage DSWE) pour sélectionner le dictionnaire de décodage approprié (*i.e.* $Dic_{N,B^l}$). La sortie du décodage d'un mot de code ADN à longueur variable est un bloc binaire de $2(N-1)$ bits qui est concaténé à la séquence binaire déjà décodée par DSWD $S_{DSWD}$. Il convient de noter que, comme pour le processus d'encodage DSWE, la première base de la séquence d'ADN $D_{DSWE}$ est décodée en deux bits selon le dictionnaire classique d'encodage de l'ADN.

**Algorithm 2:** Dynamic Sliding Window Decoding DSWD d'un mot code ADN à longueur variable commençant à la $d^{eme}$ base de $D$

**Inputs :** La séquence ADN à décoder $D$, ayant déjà décodé $d-1$ bases d'une séquence ADN $D$ en une suite binaire globale $S_{DSWD}$, l'index $d$ qui indique le début du décodage à la base $D_d$, la taille maximale d'homopolymères $N$, le dictionnaire $P_m$ de $M$ motifs interdits de taille $m$

**Output :** La séquence binaire $S_{DSWD}$ contenant le bloc binaire décodé avec DSWD $S_{DD}$ d'un mot code ADN commençant par la $d^{eme}$ base de $D$

---

% **Étape 1 : Décodage avec fenêtre glissante ("Sliding Window Decoding") SWD**
$W = (D_{d-m+1}, ..., D_{d-1})$ ;
% *Création de la fenêtre glissante $W$, correspondant aux $m-1$ dernières bases décodées* ;
$P_{mW} = SubDict(P_m, m)$ ;
% *Création d'un sous-dictionnaire $P_{mW}$ à partir de $P_m$ en prenant les $m-1$ bases d'ordre élevé de chaque motif de $P_m$* ;
**while** $W \in P_m$ **do**
    % *Détermine si $D_d$ est une $BNC$* ;
    $d = d + 1$ ;
    % *Incrémente d pour tester si la prochaine base $D_d$ est codante* ;
    $W = (D_{d-m+1}, ..., D_{d-1})$ ;
    % *Création de la nouvelle fenêtre glissante* ;
**end**
$W = (W_1, ..., W_{m-2}, D_d)$ ;
% *Création de la nouvelle fenêtre glissante* ;
$MLV_i = D_d$ ;
% *Initialisation du mot code ADN à longueur variable $MLV_i$ avec la $d^{eme}$ base codante de la séquence ADN $D$ à décoder* ;
$B^l = D_{d-1}$ ;
% *Définition de la dernière base encodée pour déterminer quel dictionnaire de décodage utiliser parmi les $Dic_{N,Y}$ pour $Y \in A, C, G, T$* ;
$d = d + 1$ ;
**while** $size(MLV) < N - 1$ **do**
    % *Construction d'un mot de code $MLV_i$ de $N-1$ bases codantes* ;
    **while** $W \in P_{mW}$ **do**
        % *Teste si la fenêtre glissante est égale à un élément de $P_{mW}$* ;
        $W = (W_1, ..., W_{m-2}, D_d)$ ;
        % *Création de la nouvelle fenêtre glissante* ;
        $d = d + 1$
    **end**
    $MLV_i = (MLV_i, D_d)$ ;
    % *Concaténation d'une base codante $D_d$ à $MLV_i$* ;
    $W = (W_1, ..., W_{m-2}, D_d)$ ;
    % *Création de la nouvelle fenêtre glissante* ;
    $d = d + 1$ ;
**end**
**if** $MLV_0^i = B^l$ **then**
    % *Vérifie si la base d'ordre le plus élevé de $MLV_i$, $MLV_0^i$, correspond à la dernière base encodée $B^l$* ;
    **if** ($MLV_0^i = A$ **AND** $MLV_1^i = C$ ) **OR** ($MLV_0^i = C$ **AND** $MLV_1^i = A$) **OR** ($MLV_0^i = G$ **AND** $MLV_1^i = T$) **OR** ($MLV_0^i = T$ **AND** $MLV_1^i = G$) **then**
        % *Vérifie si la taille de $MLV_i$ est $N$ selon les valeurs des deux bases d'ordres les plus élevés $MLV_0^i$ et $MLV_1^i$ ($MLV_1^i$ est la base avec le second ordre le plus élevé de $MLV_i$)* ;
        **while** $W \in P_{mW}$ **do**
            %*Teste si la fenêtre glissante est égale à un élément de $P_{mW}$* $W = (W_1, ..., W_{m-2}, D_d)$ ;
            % *Création d'une nouvelle fenêtre glissante* ;
            $d = d + 1$ ;
        **end**
        $MLV_i = (MLV_i, D_d)$ ;
        % *Concaténation d'une base codante $D_d$ à $MLV_i$ pour obtenir un mot code ADN de taille $N$* ;
        $d = d + 1$ ;
    **end**
**end**
% **Étape 2 : Décodage Dynamique DD du mot code ADN à longueur variable** $MLV_i$
$S_{DD} = DD(MLV_i, N, B^l)$ ;
% *Décodage Dynamique de $MLV_i$ avec le dictionnaire $Dic_{N,B^l}$ pour obtenir un bloc binaire $S_{DD}$ de $2(N-1)$ bits* ;
$S_{DSWD} = (S_{DSWD}, S_{DD})$ ;
% *Concaténation du mot code obtenu par DD $S_{DD}$ avec la séquence déjà décodée par DSWD $S_{DSWD}$*;

L'Algorithme 2 correspond au décodage DSWE d'un bloc binaire $S_{DD}$ de $2(N-1)$ bits à partir d'une séquence ADN $D$, considérant une taille maximale d'homopolymères $N$ et un dictionnaire $P_m$ de $M$ motifs interdits de $m$ nucléotides. Il part du principe que la séquence ADN a déjà été partiellement décodée en une suite binaire $S_{DSWD}$, et que nous commençons à décoder le bloc binaire à partir de la $d^{eme}$ base $D_d$ de la séquence. Le décodage repose sur les fonctions suivantes :

— $DD(MLV, N, B^l)$ est le décodage de DE d'un mot code ADN à longueur variables $MLV_i$ utilisant le dictionnaire $Dic_{N,B^l}$. Il donne en sortie un bloc binaire de $2(N-1)$ bits selon la $(d-1)^{eme}$ base de $D$ (*i.e.* $B^l$ du point de vue de l'encodage dynamique).

— $SubDict(P_m, m)$ est une fonction qui créé un sous-dictionnaire $P_{mW}$ à partir de $P_m$ pour regrouper les motifs de nucléotides qui correspondent aux $m-1$ bases d'ordre élevé des $M$ motifs de $P_m$. Puisque plusieurs motifs de $P_m$ peuvent avoir les mêmes $m-1$ premiers nucléotides, le nombre d'éléments de $P_{mW}$ est souvent plus petit que $M$.

#### 2.3.3.3 Performances théoriques de DSWE

Comme DSWE produit des séquences ADN à longueur variable, le taux d'information théorique est probabiliste. Il dépend du taux d'encodage de DE, de la taille maximale d'homopolymère $N$, ainsi que du nombre de bases non codantes ajoutées. Ce dernier est lié au dictionnaire $P_m$ avec $M$ motifs interdits de $m$ bases. Puisque nous encodons des données chiffrées qui sont uniformément distribuées, le taux d'information théorique moyen de DSWE est alors donné par :

$$R^T_{DSWE(N,m,M)} = R_N * P_{B_j} \tag{2.9}$$

où : $R_N$ est le taux d'information de l'encodage dynamique pour une taille maximale d'homopolymères $N$ ( *cf.* Équation 2.6) ; $|P_{mW}|$ est le nombre de motifs interdits avec les $m-1$ bases de poids fort distinctes ; et, $P_{B_j}$ est la probabilité que la base $B_j$ soit codante. $B_j$ est une base non codante si elle est précédée par les $m-1$ premières bases d'un motif interdit. Si il y a $|P_{mW}|$ séquences différentes correspondant aux $M$ motifs interdits, la probabilité qu'une séquence de $m-1$ bases soit dans $P_{mW}$ est de $\frac{|P_{mW}|}{4^{m-1}}$.

En conséquence, le taux d'information moyen est le suivant :

$$R^T_{DSWE(N,m,M)} = R_N * (1 - \frac{|P_{mW}|}{4^{m-1}}) \tag{2.10}$$

Ce taux d'information tend vers le taux idéal de 2 BPB quand $N$ tend vers l'infini. Pour $N = 5$, le taux d'information est déjà important avec $R = 1,94$ BPB, et dès que $N > 25$, le taux d'information est supérieur à $1,99$ BPB et donc très proche de l'idéal.

Avec SWE, lorsqu'une ou plusieurs bases non codantes sont ajoutées pour éviter les motifs interdits, il y a un risque d'introduire un homopolymère de taille supérieure à $N$. Pour s'assurer qu'aucun homopolymère n'est créé par l'ajout de bases non codantes, DSWE impose une autre contrainte. La base non codante doit également être différente de $B^u_{m-1}$ afin d'éviter la création d'un homopolymère. Ici, pour une séquence donnée $m-1$ de bases de poids fort, nous avons un motif interdit fixe $M^p_u = (B^u_1, ..., B^u_{m-1}, B^u_m = B^u_{m-1})$ et trois autres motifs dont deux pourraient être utilisés comme motifs interdits et le dernier comme motif autorisé (la valeur de sa dernière base correspond à la valeur d'une base non codante). En résumé, pour

une taille fixe de $m$ bases, il peut y avoir jusqu'à 3 motifs interdits avec les mêmes $m-1$ bases de poids fort, avec la contrainte que l'un de ces motifs ait les deux mêmes bases de poids faible (c'est-à-dire $M_u$ tel que $B_m^u = B_{m-1}^u$). Par conséquent, jusqu'à $3 * 4^{m-1}$ motifs interdits peuvent être considérés.

L'algorithme DSWE permet de conserver les proportions de bases ADN et donc d'encoder des séquences ADN avec un taux de GC équilibré. Comme nous l'avons vu dans la Section 2.3.1, l'encodage dynamique DE encode des données binaires avec des dictionnaires choisis pour être équilibrés. De son côté, SWE ajoute des bases non codantes BNC lorsque nécessaire, qui sont choisies aléatoirement parmi les bases possibles. Cet aspect aléatoire permet d'équilibrer les proportions de bases BNC ajoutées.

Dans cette section, nous avons donc présenté notre encodage dynamique DSWE, qui permet de limiter la taille des homopolymères à $N$ bases, évite l'apparition de $M$ motifs de $m$ bases, et génère des séquences ADN avec un taux de GC équilibré lorsque les données en entrée sont uniformément distribuées.

## 2.4 Alternatives à DSWE

Dans cette section, nous présentons deux alternatives à notre solution DSWE pour optimiser le taux d'information, au prix d'une augmentation de la complexité et d'une réduction du choix et nombre des motifs que l'on peut interdire.

### 2.4.1 Gestion des homopolymères par la fenêtre glissante

#### 2.4.1.1 Méthode d'encodage

Cet encodage par fenêtres glissantes gère les motifs interdits et les homopolymères. Avec cette méthode, les données binaires sont tout d'abord encodées en une séquence ADN avec l'encodage classique, puis un encodage avec deux fenêtres glissantes limite la taille des homopolymères à $N$ bases et interdit les motifs de $m$ bases du dictionnaire $P_m$. Cet encodage à fenêtres glissantes est noté $SWE^{(4)}_{N,P_m}$.

L'encodage fonctionne de la façon suivante. Considérons une séquence binaire $S$ progressivement encodée en une séquence ADN $D$ avec le codage classique, un dictionnaire $P_m$ qui contient $M$ motifs interdits de $m$ nucléotides, et une taille maximale d'homopolymères $N$. L'objectif de l'encodage $SWE^{(4)}$ est d'éviter que des homopolymères de $N+1$ bases (contenus dans le dictionnaire $P_{HP}$) et des motifs de $m$ bases de $P_m$ n'apparaissent lors de l'ajout à $D$ d'une nouvelle base $B_j$.

Pour ce faire, nous utilisons deux fenêtres glissantes $W_N$ et $W_{m-1}$ qui contiennent respectivement les $N$ et $m-1$ nucléotides précédemment encodés. Nous avons donc $W_N = (B_{j-N}...B_{j-1})$ et $W_{m-1} = (B_{j-m+1}...B_{j-1})$. Pour que $B_j$ soit ajoutée à $D$, il faut qu'aucune des fenêtres ne soit le préfixe d'un élément de $P_m$ ou $P_{HP}$. La plus petite fenêtre est testée en premier. Si une fenêtre correspond au préfixe d'un élément d'un dictionnaire, $B_j$ doit prendre une valeur de base ADN qui évite l'apparition de cet élément du dictionnaire (*i.e.* un homopolymère ou un motif interdit). Nous appelons une telle base une base non codante (BNC). Une fois que $B_j$ est ajouté à $D$, les fenêtres glissantes avancent d'une base dans $D$. Le décodage fonctionne selon la même stratégie. Avant de considérer que la valeur $B_j$ encode des données, on vérifie si les $m-1$ bases précédentes correspondent aux premières bases d'un élément de $P_m$ ou $P_{HP}$. Si c'est le cas, alors $B_j$ est une BNC, elle ne code pas d'information.

#### 2.4.1.2 Performances théoriques de $SWE^{(4)}$

Les performances de cet encodage dépendent du nombre et de la taille des éléments des dictionnaires. Définissons tout d'abord le taux d'information de cette solution lorsque $P_m = \emptyset$, c'est-à-dire lorsque la seule contrainte est la taille maximale $N$ d'homopolymères. Dans ce cas, le dictionnaire $P_{HP}$ contient les 4 homopolymères de $N+1$ bases A, C, G ou T. La fenêtre $W_N$ est donc de taille $N$, elle entraîne l'ajout d'une base non codante lorsqu'elle correspond à un des 4 homopolymères de $N$ bases. En conséquence, le taux d'information de $SWE_N^{(4)}$ est calculé comme le taux de DSWE (*cf.* Équation 2.10). Il est alors tel que :

$$R^T_{SWE_N^{(4)}} = R * (1 - \frac{4}{4^N}) \tag{2.11}$$

avec $R = 2$ BPB le taux d'information de l'encodage classique en bits par bases et $|P_m W|$ le nombre de motifs interdits avec $N$ premières bases distinctes.

Notons que $SWE_N^{(4)}$ ne permet pas de gérer le cas $N = 1$ où l'on souhaite ne créer aucun homopolymère. En effet, le principe de la fenêtre glissante est d'ajouter une base non codante préventive pour éviter un motif. Ici, la fenêtre glissante ajouterait des bases non codantes en boucle pour éviter des homopolymères de 2 bases. Cette méthode présente un taux d'information de $1,5$ BPB pour $N = 2$ contre $1,33$ BPB pour l'encodage dynamique sans homopolymères $DE(N = 1)$. Les deux méthodes présentent des taux d'information similaires pour $N = 3$. Mais, à partir de $N = 4$, $SWE^{(4)}$ démontre des performances légèrement meilleures, comme nous le détaillerons dans la partie expérimentale.

Lorsque l'encodage interdit les homopolymères (d'un dictionnaire $P_{HP}$ contenant les 4 homopolymères de $N+1$ bases A, C, G ou T) ainsi que les motifs de taille $m$ d'un dictionnaire $P_m$, deux fenêtres glissantes doivent être utilisées, dans l'ordre croissant de leur taille. Le calcul du taux d'information dépend alors des tailles des éléments des dictionnaires $P_{HP}$ et $P_m$. Des éléments d'un dictionnaire peuvent être inclus dans les éléments de l'autre. En conséquence, certains éléments du second dictionnaire ne seront jamais pris en compte par la seconde fenêtre. Prenons le cas où $m > N + 1$. La probabilité qu'une base soit non codante est calculée en faisant le produit des probabilités qu'une base soit codante pour chacune des deux fenêtres, ce qui donne :

$$P(BNC) = \frac{4}{4^N} \times \frac{|P_{mW}|}{4^{m-1} - 4^{m-N-1} \times (m - N - 1)} \tag{2.12}$$

où $|P_{mW}|$ est le nombre de motifs interdits avec les $m-1$ bases de poids fort distinctes, et $4^{m-1} - 4^{m-N-1} \times (m - N - 1)$ le nombre de séquences de $m-1$ bases qui ne contiennent pas d'homopolymères de taille $N$.

Le taux d'information est alors de :

$$R^T_{SWE^{(4)}_{N,P_m}} = 2 * (1 - \frac{4}{4^N} \times \frac{|P_{mW}|}{4^{m-1} - 4^{m-N-1} \times (m - N - 1)}) \tag{2.13}$$

Cette méthode permet donc d'utiliser l'encodage à fenêtres glissantes pour la gestion des homopolymères, au prix d'un nombre important de tests. La comparaison avec DSWE en terme de complexité et de taux d'information est présentée dans la partie expérimentale.

### 2.4.2 Fenêtre glissante binaire

L'idée de cette solution est d'utiliser des fenêtres glissantes pour les motifs interdits et les homopolymères comme dans la méthode précédente, mais cette fois-ci sur les données en binaire. La séquence binaire obtenue est ensuite encodée avec l'encodage classique, qui convertit deux bits en une base ADN. L'objectif est d'optimiser le taux d'information, car les fenêtres ajoutent un bit non codant au lieu d'une base non codante.

#### 2.4.2.1 Méthode d'encodage

Le fonctionnement de cette méthode, que nous appelons $SWE^{(bin)}$, est présenté en Figure 2.12. Chaque motif ADN interdit du dictionnaire $P_m^{(4)}$ est converti en une suite binaire avec l'encodage classique, formant le dictionnaire $P_m^{(2)}$. Dans un second temps, l'encodage avec fenêtre glissante est appliqué : la séquence binaire à encoder est parcourue de gauche à droite, et des bits non codants peuvent être ajoutés pour éviter l'apparition d'un motif interdit binaire de $P_m^{(2)}$. Cela fait, la séquence binaire obtenue est convertie en bases ADN avec l'encodage classique.

L'intérêt de cet encodage dynamique appelé $SWE^{(bin)}$ est d'optimiser la fenêtre glissante utilisée dans DSWE. Celle-ci repère normalement les $m-1$ premières bases d'un motif interdit de $m$ bases, et ajoute une base non codante (BNC) afin de ne jamais obtenir le motif interdit de $m$ bases. Avec $SWE^{(bin)}$, la fenêtre glissante repère les $2m-1$ premiers bits d'un motif avant encodage classique. Dans le cas où cette séquence apparaît, on ajoute un bit non codant *bnc* au lieu d'une base, ce qui permet de gagner en densité d'information par rapport à DSWE.

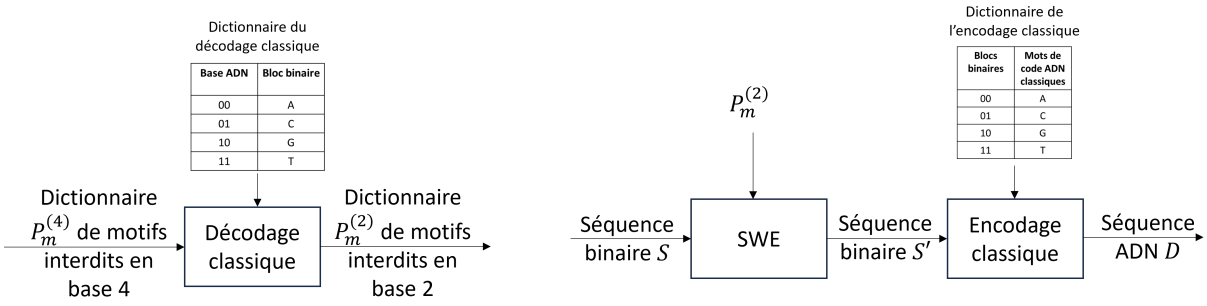



FIGURE 2.12 – Encodage $SWE^{(bin)}$. A gauche, préparation du dictionnaire $P_m^{(4)}$ en un dictionnaire en base 2 $P_m^{(2)}$. A droite, encodage de la séquence binaire $S$ en deux étapes : Encodage avec la fenêtre glissante et le dictionnaire $P_m^{(2)}$, puis encodage classique de bits en bases ADN.

Décrivons le cas simple où la seule contrainte est la taille maximale d'homopolymères $N$. Les homopolymères interdits sont donc de taille $N+1$ : ils forment le dictionnaire $P_{HP}$. Chaque homopolymère $H_i$ de $N+1$ bases est associé à une suite binaire $H_i^{(2)}$ de $2N+2$ bits correspondant à son décodage classique. Un dictionnaire $P_{HP}^{(2)}$ de couples $\{s_i^{(2)}, bnc_i\}$ est créé pour chaque motif, avec $s_i^{(2)}$ la suite binaire contenant les $2N+1$ bits de poids fort de $H_i^{(2)}$, et $bnc_i$ le bit inverse du dernier bit de $H_i^{(2)}$. Par exemple si un motif interdit est $H_i = (AAA)$, alors $H_i^{(2)} = (000000)$, et donc le couple $\{s_i^{(2)}, bnc_i\} = \{00000, 1\}$. Le principe de cet encodage repose alors sur l'utilisation d'une fenêtre glissante de $2N+1$ bits, que l'on compare aux suites binaires $s_i^{(2)}$ du dictionnaire $P_{HP}^{(2)}$. La fenêtre glissante parcourt la séquence binaire $S$ à encoder de

gauche à droite. Si une suite binaire $s_i^{(2)}$ est repérée, alors le bit non codant $bnc_i$ est ajouté à la séquence. La fenêtre est alors décalée pour prendre en compte ce dernier bit, et tester à nouveau les suites binaires. Une fois la séquence binaire $S$ scannée, celle-ci est encodée en séquence ADN avec l'encodage classique. Le décodage informé fonctionne de la même façon, en commençant par le décodage de la séquence ADN en flux binaire. La fenêtre permet ensuite de détecter si un bit est codant ou non. Chaque bit codant est ajouté à une séquence binaire, qui correspond à l'information encodée.

Lorsque l'encodage interdit des homopolymères de $N+1$ bases ainsi que des motifs de $m$ bases, il fonctionne de la même façon, avec cette fois-ci deux fenêtres glissantes. Une fenêtre de $2N+1$ bits est associée aux homopolymères, et une fenêtre de $2m-1$ bits correspond aux motifs de $m$ bases du dictionnaire $P_m$. La fenêtre la plus petite est toujours appliquée en premier. Ceci entraîne des contraintes sur la liste des motifs interdits. En effet il ne faut pas qu'un motif interdit soit le préfixe d'un autre, ou que l'ajout d'un bit $bnc_i$ pour éviter un motif interdit en créé un autre. Prenons le cas où $(AAA)$ est un motif interdit. Les bits correspondant à ce motif sont 6 $'0'$ consécutifs. Alors, lorsque la fenêtre glissante détecte 5 $'0'$ consécutifs, un bit non codant 1 est inséré. Le corrigé de $(AAA)$ est donc $(AAC)$, obtenu avec l'encodage classique associant (00) à $A$ et (01) à $C$. En conséquence, $(AAC)$ étant le corrigé d'un motif interdit, il ne peut pas lui-même être un motif interdit. Cela restreint le choix des motifs interdits, pour $m-1$ premières bases fixées, SWE binaire permet d'avoir au maximum un motif de $m$ bases qui finit par un A ou un C, et un qui finit par G ou T. Au contraire, notre solution DSWE permet toujours de choisir 2 motifs interdits avec les $m-1$ premières bases, et 3 dans certains cas.

#### 2.4.2.2 Performances théoriques

L'intérêt de cette méthode est d'ajouter un bit non codant et pas une base non codante, et cela moins souvent. Pour illustrer ceci, prenons l'exemple d'un motif interdit $(AAAA)$. Avec $SWE^{(4)}$, on ajoute une base non codante quand $(AAA)$ apparaît, c'est-à-dire dans $\frac{1}{4^3} = \frac{1}{64}$ cas. Avec $SWE^{(bin)}$, on ajoute un bit non codant quand (0000000) apparaît, ce qui correspond à $\frac{1}{2^7} = \frac{1}{128}$ cas. Ainsi, cette nouvelle méthode permet d'ajouter un seul bit de redondance au lieu d'une base, et à une fréquence deux fois moins importante.

Si les homopolymères de taille maximale $N$ sont la seule contrainte, alors la fenêtre glissante de $2N+1$ bits entraine l'ajout d'un bit non codant dans 4 cas. Le taux d'information théorique de $SWE_N^{(2)}$ est donc :

$$R^T_{SWE_N^{(2)}} = 2 * (1 - \frac{4}{2^{2N+1}}) = 2 * \frac{2^{2N-1}-1}{2^{2N-1}} \tag{2.14}$$

Comme l'encodage $SWE^{(4)}$, $SWE_N^{(2)}$ ne permet pas de gérer le cas $N=1$. Toutefois, pour des valeurs de $N$ plus importantes, le taux d'information de $SWE^{(bin)}$ est proche de l'idéal. Par exemple, il est de $1,937$ BPB pour $N=3$, contre $1,882$ BPB pour $DE(N=3)$.

Dans le cas où les contraintes sont $M$ motifs interdits de taille $m$ et une taille maximale d'homopolymères $N < m-1$, le taux d'information est calculé similairement à l'équation 2.13 de $SWE^{(4)}$. Nous avons alors le taux suivant :

$$R^T_{SWE^{(2)}_{N,P_m}} = 2 * (1 - \frac{4}{2^{2N+1}} \times \frac{|P_{mW}|}{2^{2m-1} - 2^{2m-2N-1} \times (2m-2N-1)}) \tag{2.15}$$

Une comparaison plus complète à DSWE est donnée dans la partie expérimentale.

## 2.5 Codes correcteurs d'erreurs

Dans le Chapitre 1, nous avons vu que les technologies de séquençage et de synthèse ne sont pas parfaites et qu'elles introduisent des erreurs dans la récupération des données : erreurs de substitution, de délétion et d'insertion de nucléotides. L'encodage DSWE, qui prend en compte plusieurs contraintes structurelles telles que le taux de GC et la taille des homopolymères, permet de limiter l'apparition de telles erreurs, mais pas de les éliminer totalement. Pour remédier à ce problème, plusieurs techniques peuvent être utilisées. Une première consiste à générer et stocker plusieurs copies d'une même séquence ADN. Les copies sont lues (séquençage) puis traitées par un algorithme bio-informatique de consensus (*cf.* Section 1.1.5.3), qui permet de reconstruire une séquence consensus. Une autre solution consiste à utiliser des codes correcteurs, comme présenté en Section 1.1.3.1. Ils sont appliqués sur les données binaires ou sur les séquences ADN, *i.e.* en base 4.

Dans cette Section, nous présentons plusieurs codes correcteurs : codes LDPC appliqué aux données binaires, aux séquences ADN, et structure de deux codes GRS ("Generalized Reed-Solomon"). Nous avons choisi ces codes car ils ont déjà été utilisés dans le cadre d'encodage de données dans l'ADN, et ont permis de corriger des erreurs causées par les étapes biologiques de la chaîne de stockage dans l'ADN. Toutefois, ces codes correcteurs ont été utilisés avec des encodages à taille fixe, tandis que notre encodage DSWE est dynamique. Dans la partie expérimentale, nous allons évaluer la capacité de correction de ces codes appliqués avec notre DSWE, et l'impact de son aspect dynamique sur la correction d'erreurs.

### 2.5.1 Codes LDPC

Les codes LDPC (Low-Density Parity-Check), ou codes de parité à faible densité ont été introduits en 1963 [154], et sont utilisés dans les systèmes de communication pour corriger les erreurs de substitution, par exemple dans le standard Wi-fi 802.11. Ils sont populaires de par leurs performances de corrections d'erreur et leur complexité : ils peuvent être décodés en un temps proportionnel à leur longueur à l'aide de techniques itératives de propagation d'erreurs. Les codes LDPC sont construits à partir du principe élémentaire de code de parité. Une équation de parité relie $n$ données binaires entre elles par l'opérateur "ou exclusif" (XOR), noté $\oplus$. Elle est vérifiée (*i.e.* vaut 0) si le nombre total de 1 dans l'équation est pair ou nul. Un code LDPC s'applique à une séquence d'information unique, binaire ou ADN, comme nous le présentons dans les paragraphes suivants.

#### 2.5.1.1 LDPC binaire

Dans le cas binaire, un code LDPC $[n, k]$ [44] est défini par une matrice de contrôle de parité binaire clairsemée $H$ de taille $(n - k) \times n$, telle que présentée en Figure 2.13. Une séquence $c$ est un mot codé si elle satisfait à l'équation linéaire $c.H^t = 0$. Ainsi, chaque ligne de $H$ correspond à une équation de contrôle de parité où toutes les opérations arithmétiques sont modulo 2 (reste de la division), et chaque colonne de $H$ correspond à un bit du mot de code $c$. Dans la Figure 2.13, le graphe bipartite lie les noeuds

de variables, qui sont les bits de mots de code $c_i$ (colonnes de $H$), aux noeuds de parité correspondant aux équations de parité $l_i$ *i.e.* les lignes de $H$.

Pour générer un mot de code $c$ à partir de la séquence $x$, une matrice génératrice $G$ est dérivée de $H$, par exemple par une élimination de Gauss-Jordan, de sorte que $G.HT = 0$ et $c = x.G$. Ainsi, la séquence binaire $x$ est encodée en $c$.

Pour décoder, il est possible d'utiliser la propagation de croyance, ou "Belief Propagation" (BP). Ce décodage est itératif et se fait en temps linéaire par rapport à la taille de ses blocs pour des codes LDPC. Chaque noeud de variable envoie un message sur la valeur estimée du bit aux noeuds de parité auxquels il est associé. Le traitement successif des noeuds de variable puis de parité constitue une itération. À chaque itération, il y a donc un échange bilatéral de messages entre les nœuds de parité et les nœuds de variable, sur les arcs du graphe bipartite représentant le code LDPC. Le nombre d'itération est plafonné, même si toutes les équations de parité ne sont pas satisfaites. Dans ce cas, les bits décodés sont estimés.

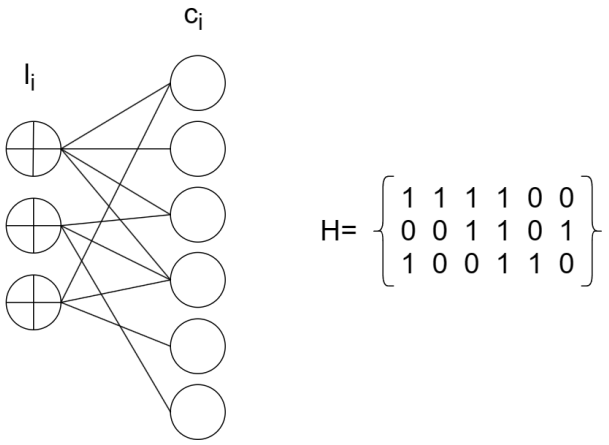



FIGURE 2.13 – Graphe biparti et matrice représentant la matrice de contrôle de parité, avec 3 bits en entrée et 3 bits de contrôle.

Le rendement d'un code correcteur est $\frac{k}{n}$, c'est-à-dire le nombre de bits contenant de l'information divisé par le nombre de bits codés, *i.e.* la taille des mots code.

#### 2.5.1.2 LDPC quaternaire

Il est possible d'étendre la définition d'un code LDPC avec des données non binaires. Ici, nous nous intéressons au cas du code LDPC quaternaire, utilisé notamment pour la correction d'erreurs dans des séquences ADN après leur séquençage [47]. Cela nous permet de corriger des erreurs directement dans une séquence ADN, sans passer par une étape de décodage de base 4 en binaire qui peut introduire des erreurs supplémentaires.

Dans ce cas, les éléments de la matrice de parité sont des éléments de $GF(4)$, le corps fini (ou "Galois field") [155] de 4 éléments. Pour tout corps fini $GF(q)$, $q$ est une puissance non nulle d'un nombre premier, et toute opération arithmétique avec des éléments du corps $GF(q)$ est effectuée modulo $q$, donnant un élément de $GF(q)$. Si $q = p^\lambda$ avec $\lambda > 1$, chaque élément de $GF(q)$ est représenté par un polynôme de degré de degré inférieur ou égal à $\lambda$ : $pol(X) = \alpha_0 + \alpha_1 X^1 + ... + \alpha_{\lambda-1} X^{\lambda-1}$, avec $pol(X) \in GF(q = p^\lambda)$

et les coefficients $\alpha_i \in GF(p)$. Un élément de $GF(q)$ est donc représenté par un vecteur de longueur $\lambda$. Pour le corps fini $GF(4 = 2^2)$ utilisé pour le code LDPC en base 4, ses éléments sont représentés par le polynôme $pol(X) = \alpha_0 + \alpha_1 X^1 \in GF(4)$. Les 4 éléments du corps $\{0, 1, 2, 3\}$ correspondent aux vecteurs $\{(0,0), (0,1), (1,0), (1,1)\}$, dont les représentations polynomiales sont $\{0, 1, X, 1 + X\}$.

Pour définir les codes LDPC dans $GF(4)$, les éléments non nuls de la matrice de contrôle de parité $H$ prennent des valeurs dans $GF(4)$. Ensuite, tout mot de code $c$ avec des éléments de $GF(4)$ vérifie $c.H^t = 0$, et les opérations arithmétiques sont évaluées sur $GF(4)$. Le procédé de décodage par propagation de conviction, ou Belief Propagation (BP), est similaire à celui des codes LDPC binaires. En effet, un message $m$ est reçu et le but est de trouver un mot de code $c$ proche de $m$ et tel que $c.H^t = 0$. Le format est différent car les calculs se font avec des polynômes, et les probabilités d'erreurs sont basées dans [42] sur des données expérimentales stockées dans l'ADN.

Dans la partie expérimentale, nous allons tester ces codes LDPC binaires et quaternaires avec notre encodage DSWE. Notre but est d'évaluer l'impact des codes correcteurs sur la récupération des données après leur passage dans un canal simulant les erreurs causées par la chaîne de stockage de données dans l'ADN.

### 2.5.2 Codes GRS concaténés

Comme nous l'avons présenté en Section 1.1.3.1 du Chapitre 1, il est possible d'appliquer aux données deux codes correcteurs concaténés pour améliorer la capacité de correction. Dans ce cas, un code externe et interne sont appliqués successivement aux données, comme représenté schématiquement en Figure 2.14. Ces codes peuvent être différents, par exemple un code convolutif et un code LDPC en base 4 [156], ou bien deux codes de Reed-Solomon (RS) [32]. Cette dernière solution consiste à appliquer un code RS externe sur des séquences individuelles, puis un deuxième code interne sur l'ensemble des séquences. Les données obtenues (information et codes correcteurs) sont ensuite encodées en séquence ADN. Une méthode existante [32] repose sur deux codes RS en base 47, choix lié à l'encodage ADN utilisé. Nous avons testé cette dernière solution avec nos propres codes de Reed-Solomon en base 2, afin de pouvoir encoder les données obtenues avec DSWE. C'est pour cela que nous définissons dans les sections suivantes les codes de Reed-Solomon et la structure de codes concaténés.

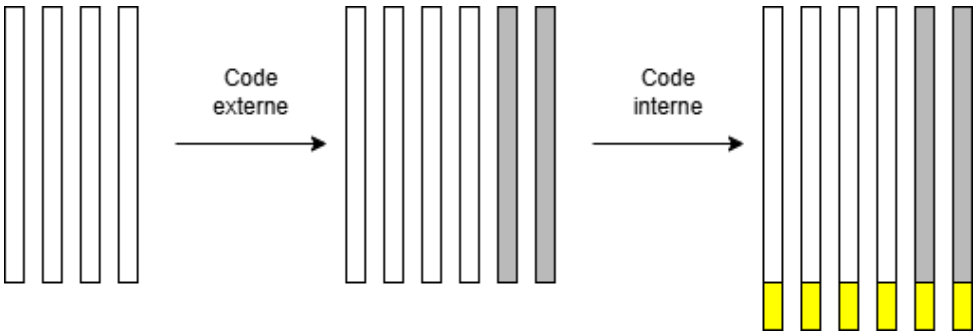



FIGURE 2.14 – Représentation schématique d'un code concaténé. Un code externe produit de la redondance sous forme de nouvelles séquences ADN, puis un code interne ajoute de la redondance à chaque séquence.

#### 2.5.2.1 Codes de Reed-Solomon

Les codes de Reed-Solomon [157] ont été développés par IS. Reed et G. Solomon en 1959. En raison de leurs paramètres optimaux et de leur facilité de décodage, ces codes ont été beaucoup utilisés dans l'industrie, pour les CD et DVD par exemple, ou encore pour les communications satellites.

Définissons formellement un code de Reed-Solomon. Soit $GF(q)$ un corps [155], et soient $x_1, ..., x_n$ $n$ éléments distincts de $GF(q)^\star$. Pour $k \leq n$, $P_k$ est l'ensemble des polynômes de $GF(q)[x]$ de degré au plus $k-1$. Un code de Reed-Solomon est composé de l'ensemble des mots $c(f) = (f(x_1), ..., f(x_n))$, pour $f \in P_k$. Un code de Reed-Solomon de longueur $n$ sur $GF(q)$ et de paramètre $k$ est appelé code $RS(n,k)_q$. Les codes RS sont des codes $[n, k, n-k+1]$, ce qui veut dire que le code est de longueur $n$, de dimension $k$, et la distance minimale entre deux mots de code est $n-k+1$. Cette distance est la plus grande possible pour un tel code, ce qui en fait un code MDS (maximum distance séparable). En cela, un code RS est un code linéaire optimal (très peu de redondance inutile).

Pour décoder un code de Reed-Solomon, on reçoit un mot de code entaché d'erreur et on le corrige en reconstruisant le polynôme correspondant au mot. En particulier, soit $c = c(f)$ un mot d'un code $RS(n,k)_q$ construit à partir de $f(x)$ avec $deg(f) \leq k-1$. Soit maintenant un mot reçu $r = c + e$ avec une erreur de poids $w_H(e) \leq t =$. Si l'on construit le polynôme $Q(x,y) = Q_0(x) + yQ_1(y)$, le polynôme d'interpolation bivarié défini précédemment à partir du mot reçu $r$, alors on peut retrouver le polynôme $f(x)$ associé à $c(f)$ par $f(x) = \frac{-Q_o(x)}{Q_1(x)}$.

La capacité de correction du code RS dépend de sa distance minimale $n-k+1$. Il peut corriger jusqu'à $\frac{n-k}{1}$ symboles erronés, dits erreurs. Si les positions des erreurs sont connues, ce sont des effacements, et le code RS peut alors corriger deux fois plus d'effacements que d'erreurs. La capacité de correction des codes de Reed-Solomon est donnée par la formule $2R + F \leq n - k$, avec $R$ le nombre d'erreurs et $F$ le nombre d'effacements.

**Code de Reed-Solomon généralisé (GRS)** Les codes de Reed-Solomon généralisés correspondent à des codes de Reed-Solomon dont les colonnes sont multipliées par des éléments non nuls de $GF(q)$. Ces codes présentent les mêmes paramètres et la même capacité de correction que les codes RS. Définissons formellement un code GRS $RS_k(\alpha, \nu)$ sur $GF(q)$ de longueur $n \leq q-1$ et de dimension $k$ est défini par un vecteur $\nu = (\nu_1, ..., \nu_n)$ d'éléments non nuls de $GF(q)$ et un ensemble $\alpha = (\alpha_1, ..., \alpha_n)$ d'éléments distincts non nuls de $GF(q)$. Alors

$$RS(\alpha, \nu) = \{(\nu_1 f(\alpha_1), ..., \nu_n f(\alpha_n)) | f \in GF(q)[x], deg(f) < k\} \tag{2.16}$$

**Codes concaténés GRS** Nous définissons dans cette section la structure de codes correcteurs GRS concaténés. Les deux codes de Reed-Solomon généralisés utilisés pour cette structure sont le code interne (ou "inner code") $C_{in}$ et le code externe (ou "outer code") $C_{out}$. Ils seront appliqués l'un après l'autre aux données. Soit $q$ une puissance d'un nombre premier, Soit $\alpha$,$\beta \in GF(n_{out})$ et $\gamma, \tau \in GF(n_{in})$, $C_{out} = GRS(\alpha, \beta)$ sur $GF(q_{out})$ est le "outer code" de longueur $n_{out}$ et de dimension $k_{out}$, tandis que $C_{in} = GRS(\gamma, \tau)$ sur $GF(q_{in})$ est le "inner code" de longueur $n_{in}$ et de dimension $k_{in}$. On note $C = C_{out} \otimes C_{in}$ le code concaténé, obtenu en appliquant successivement $C_{in}$ puis $C_{out}$ aux données. Dans notre cas, la structure des codes est représentée en Figure 2.15 sous forme de matrice. Le code externe en gris est tout

d'abord appliqué sur l'ensemble des données, chaque ligne étant un élément de $GF(q_{out})$. Ensuite, un index représenté dans un cadre violet est ajouté à chaque colonne de la matrice obtenue. Enfin, le code interne $C_{in}$ en jaune est appliqué séparément à chaque colonne, qui contient des éléments de $GF(q_{in})$. Nous appellerons chaque colonne encodée une séquence. Chaque séquence peut être transmise séparément dans un canal bruité.

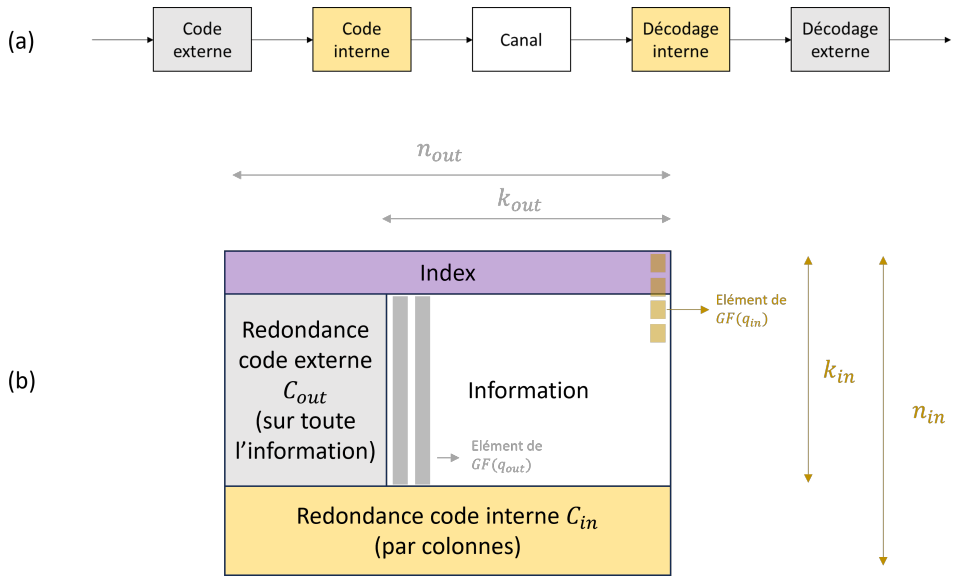



FIGURE 2.15 – Définition de codes produit. Deux codes sont appliqués successivement sur les données, un code externe puis un code interne, représentés en gris et jaune respectivement en Figure (a). Au contraire, l'opération de décodage débute par le décodage interne, suivi du décodage externe. La figure (b) représente la structure des données, l'information en blanc est encodée avec le code externe, avant d'être indexée (en violet), et le code interne est appliqué sur l'ensemble de ces données, considérant chaque colonne comme un élément pour l'encodage.

Le décodage se fait dans l'ordre inverse. Le décodage interne agit en premier, corrigeant quelques erreurs isolées dans chaque séquence individuelle. Les index ainsi retrouvés permettent de reformer la structure de matrice présentée dans la Figure 2.15. Enfin, le décodeur externe est appliqué pour corriger les rafales d'erreurs. Présentons ce procédé de décodage plus en détail.

Les codes interne $C_{in}(n_{in}, k_{in})$ et externe $C_{out}(n_{out}, k_{out})$ ont une capacité de correction de $2R + F \leq n - k$, avec $R$ le nombre d'erreurs de substitution et $F$ le nombre d'effacements. Ainsi, le inner code peut corriger jusqu'à $F = n_{in} - k_{in}$ erreurs de substitution dans une séquence de $n_{in}$ éléments, puisqu'il n'en connaît pas la position. Ensuite, chaque séquence est associée à un des deux scénarios suivants :

1. La séquence contient au maximum $F$ substitutions, elle est corrigée par le décodeur interne $GRS_{in}(n_{in}, k_{in})$ et son index est récupéré.
2. La séquence contient trop d'erreurs pour être corrigée ou est corrigée en une séquence de mauvaise taille, elle est effacée.

Une matrice est construite avec les séquences exactes retrouvées indexées. Cette matrice contient des effacements, des séquences entières effacées. Le décodeur externe $GRS_{out}(n_{out}, k_{out})$ considère chaque séquence comme un élément, et peut retrouver jusqu'à $F = n_{out} - k_{out}$ séquences effacées.

Nous avons donc vu comment fonctionnent les codes de Reed-Solomon, et la concaténation de ces codes

pour créer une matrice permettant de corriger un grand nombre d'erreurs, notamment des effacements de séquences entières. En raison du grand nombre et de la variabilité des erreurs causées par la chaîne de stockage, un tel dispositif peut s'avérer utile pour garantir la récupération des données stockées dans l'ADN. Dans la partie expérimentale, nous allons présenter notre propre structure de codes de Reed-Solomon généralisés associée à notre encodage DSWE, et comparer ses performances à la structure présentée par Grass [32]. Nous testerons aussi les codes LDPC binaire et quaternaire présentés précédemment avec DSWE.

## 2.6 Résultats expérimentaux

Nous présentons les résultats expérimentaux de notre solution d'encodage DSWE et la comparons à l'état de l'art, ainsi qu'à ses alternatives que nous avons présenté dans la section précédente. Nous donnons aussi plusieurs résultats pour DSWE combiné avec des codes correcteurs et des algorithmes de compression.

### 2.6.1 Paramètres

Notre jeu de données se compose de l'image test "Lena.png" de 48 Kbytes, d'un texte aléatoire *lorem ipsum* de 95 Kbytes ainsi que les fichiers pdf de deux articles scientifiques : [47] et [39]. Les fichiers texte ont été utilisés pour évaluer les taux d'information, le taux de GC et les taux d'erreur, tandis que l'image a été utilisée pour évaluer les temps d'exécution des processus d'encodage. Nous avons implémenté nos algorithmes en Python3 (version 3.8.10).

Le benchmark mis en œuvre est le suivant. Les données ont d'abord été chiffrées 5 fois à l'aide de 5 clés AES-256 distinctes. Ensuite, chacun des 5 flux binaires chiffrés a été séparé en 10000 blocs binaires de taille fixe de $X$ bits, avec $X \in \{100, 200, 1000, 2000\}$, puis codés séparément en séquences d'ADN à l'aide de notre codeur DSWE. Les séquences obtenues ont été fournies au simulateur de [47], décrit en Section 1.1.6 du Chapitre 1, afin d'évaluer le nombre de copies nécessaires pour récupérer les données sans erreur. I Notre schéma DSWE a été mis en œuvre en considérant les valeurs des paramètres $N = \{2, 3, 4\}$, $M = \{0, 10, 20, 30, 40, 50, 60, 70, 76\}$ et $m = 6$, en utilisant un dictionnaire de référence de 76 motifs interdits avec $m - 1$ bases d'ordre élevé distinctes. Ces motifs ont été choisis car ils correspondent à des sites de restriction dont nous nous servirons au Chapitre 3 pour sécuriser le stockage au niveau moléculaire et qui ne doivent pas être utilisés pour coder de l'information utile. Les résultats qui suivent sont donc donnés en moyenne pour chaque longueur de séquence. Par souci de simplicité, la notation suivante $DSWE(N, M, m)$ est utilisée pour indiquer le paramétrage de notre schéma d'encodage.

Nous comparons notre solution aux méthodes de pointe décrites dans la section 2.1 en termes de taux d'information. Notamment, une comparaison approfondie avec la méthode concurrente mCGR [38] a été faite considérant différents critères de performance :

— Le taux d'information

— Le temps d'exécution

— Les performances de récupération des séquences d'ADN

Rappelons que [38] génère des dictionnaires de mots code ADN de taille $l$ en tenant compte de trois types de contraintes : un contenu G-C fixe ou dans un intervalle de valeurs ; des homopolymères de taille maximale $N$ ; et une liste de motifs interdits comprenant jusqu'à $l$ bases. À notre connaissance, [38] est le seul autre travail existant qui prend en compte les motifs interdits. Nous utilisons sa mise en œuvre accessible au public en tenant compte des éléments suivants : un équilibre du contenu G-C de $40-60\%$ ; une taille maximale d'homopolymère de $N=3$ ; et la génération d'un livre de codes avec des mots code ADN de 10 nucléotides. Un livre de codes de 12 bases est proposé dans [38]. Cependant, compte tenu du fait que [38] s'appuie sur une opération de nettoyage du codebook ayant pour but d'éliminer les mots de code dont la concaténation pourrait conduire à la création d'homopolymères ou de motifs interdits, la différence entre les codebooks dans le cas de bases de 10 et de 12 est négligeable. C'est la raison pour laquelle nous n'avons pas considéré ce paramètre dans la suite.

### 2.6.2 Résultats de DSWE

Dans cette Section, nous montrons les performances de DSWE en terme de taux d'information et de taux de GC.

#### 2.6.2.1 Taux d'information

Nous évaluons les performances du Dynamic Sliding Window Encoding (DSWE) et comparons son taux d'information théorique moyen $R^T_{DSWE}$ (*cf.* Équation 2.10) au taux d'information expérimental $R^E_{DSWE}$ (voir Équation 2.17) en considérant 3 tailles maximales d'homopolymères $N \in \{2,3,4\}$ et nombres $M$ de motifs interdits de taille $m=6$ avec $M \in [0,76]$. Le taux d'information expérimental $R^E_{DSWE}$ que nous calculons est la valeur moyenne des taux d'information obtenus avec toutes les séquences encodées. Il est exprimé en bits par base et défini tel que :

$$R^E_{DSWE} = \frac{\sum_{i=1}^{e} \frac{|S_i|}{|D_i|}}{e} \tag{2.17}$$

où : $e$ est le nombre de blocs binaires ; $S_i$ est le $i^{\text{ème}}$ bloc binaire du flux binaire chiffré ; $D_i$ est la séquence ADN encodée avec DSWE à partir de $S_i$ ; et, $|x|$ est la valeur absolue de $x$, *i.e.* le nombre de bits ou de nucléotides que contient $x$ selon que $x$ est un bloc binaire ou une séquence ADN, respectivement.

Les taux d'information théoriques et expérimentaux sont donnés dans la Table 2.2 en fonction de la longueur maximale des homopolymères $N$ et du nombre $M$ de motifs interdits. On pourra remarquer que les taux d'information expérimentaux pour DSWE correspondent bien aux taux théoriques pour $N \in \{2,3,4,5\}$ et $M = \{0,20,40,76\}$, avec $m=6$. Ce résultat valide l'hypothèse selon laquelle : i) l'AES produit des données chiffrées uniformément distribuées, hypothèse que nous avons prise en compte dans le calcul du taux d'information théorique ; ii) les mots code ADN de nos dictionnaires sont générés en proportions égales, ce qui préserve la distribution des données en entrée.

Une légère différence de 0.5% existe entre les taux théoriques et expérimentaux. Cela peut s'expliquer par le fait que, lors de la génération de notre dictionnaire de 76 motifs interdits, nous n'avons pas considéré qu'ils devaient avoir une distribution uniforme des bases. Notons que des valeurs plus grandes de $N$ garantissent un taux d'information moyen plus stable avec un écart-type plus faible.

TABLE 2.2 – Taux d'information expérimental $R^E_{DSWE(N,M,6)}$ et théorique $R^T_{DSWE(N,M,6)}$ de DSWE donnés en bits par base (BPB) selon la taille maximale d'homopolymères $N$ et le nombre $M$de motifs interdits de $m = 6$ nucléotides. Les taux d'information expérimentaux sont donnés en moyenne avec leurs écarts-type.

| $M$ | 0 | | | | 20 | | | | 40 | | | | 76 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $N$ | 2 | 3 | 4 | 5 | 2 | 3 | 4 | 5 | 2 | 3 | 4 | 5 | 2 | 3 | 4 | 5 |
| $R^E_{DSWE(N,M,6)}$ (en BPB) | 1.334 | 1.882 | 1.921 | 1.939 | 1.302 | 1.839 | 1.88 | 1.902 | 1.239 | 1.803 | 1.843 | 1.864 | 1.167 | 1.73 | 1.768 | 1.787 |
| Écart-type de $R^E_{DSWE(N,M,6)}$ (in BPB) | 0.015 | 0.01 | 0.008 | 0.005 | 0.017 | 0.011 | 0.01 | 0.006 | 0.019 | 0.011 | 0.011 | 0.07 | 0.02 | 0.013 | 0.011 | 0.009 |
| $R^T_{DSWE(N,M,6)}$ (en BPB) | 1.333 | 1.882 | 1.92 | 1.939 | 1.288 | 1.846 | 1.883 | 1.902 | 1.242 | 1.809 | 1.845 | 1.863 | 1.16 | 1.743 | 1.778 | 1.795 |

Comme le montre la Figure 2.16, les taux d'information expérimentaux pour les valeurs $N = 2$, $N = 3$, $N = 4$ et $N = 5$ diminuent linéairement en fonction de $M$. Dans le cas où DSWE fonctionne avec $M = 76$ motifs interdits par rapport à aucun, nous avons une perte de taux d'information de 0.15 BPB en moyenne. Quoi qu'il en soit, même si DSWE entraîne certaines pertes, il préserve toujours un taux d'information moyen élevé, la plupart des séquences ADN étant très proches du taux théorique, comme le montre le faible écart-type. En outre, DSWE préserve la distribution uniforme des données chiffrées pour tout nombre $M$ de motifs interdits, comme cela a été testé pour plusieurs tailles maximales d'homopolymères $N$ ($N \in \{2, 3, 4, 5\}$).

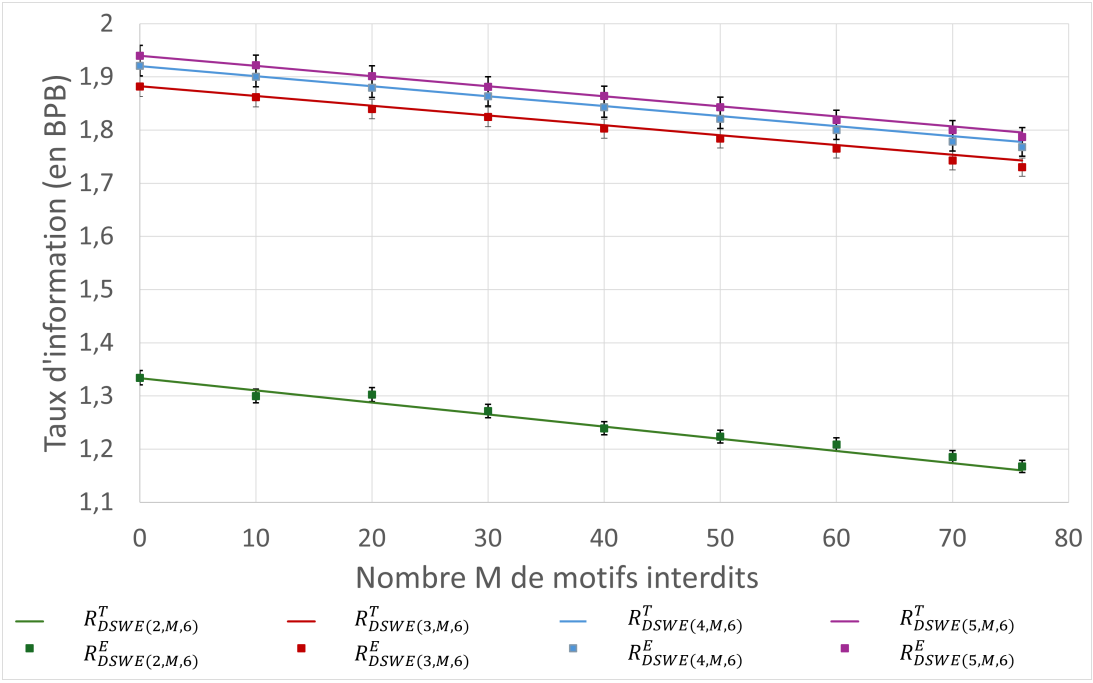



FIGURE 2.16 – Taux d'information expérimentaux et théoriques de $DSWE(2, M, 6)$, $DSWE(3, M, 6)$, $DSWE(4, M, 6)$ et $DSWE(5, M, 6)$. Les taux d'information expérimentaux sont donnés en moyenne en bits par base (BPB) avec leurs écarts-type respectifs, pour 10000 séquences de 1000 bits et selon le nombre $M$ de motifs interdits.

#### 2.6.2.2 Taux de GC

Pour être correctement synthétisées et lues, les séquences ADN doivent avoir une proportion de bases équilibrée, avec un pourcentage de 40% à 60% de bases G et C [33], appelé taux de GC.

Comme le montre la Table 2.3, la valeur moyenne du taux de GC obtenu avec DSWE est très proche de 50% pour toutes les valeurs testées de $N$ et $M$, avec un écart type de 1.32 à 2.02%. Toutes les séquences ADN testées ont un taux de GC compris entre $42,6$ et 57%, la plupart des séquences étant proches de 50%. Ces résultats correspondent aux exigences [33]. En outre, le faible écart type montre que la modification de la clé de chiffrement n'a pas d'incidence sur le taux de GC, car la distribution uniforme est préservée. Il convient de noter que les séquences ADN générées ont une proportion de bases A,C,G, et T équilibrée en général en raison de l'encodage des données chiffrées. Pour résumer, le DSWE préserve la distribution des données.

De plus, un taux de GC équilibré est obtenu quel que soit le nombre $M$ de motifs interdits. Par exemple, le taux de GC est de 49.8% pour $DSWE(N, M, m) = DSWE(3, 20, 6)$ et de 49.6% pour $DSWE(3, 76, 6)$. En moyenne, le taux de GC diffère légèrement selon $M$. Cette différence peut également s'expliquer par le fait que notre dictionnaire de motifs interdits ne contient pas de séquences d'ADN avec une distribution uniforme des bases. Mis à part cette petite marge, l'utilisation d'un Sliding Window Encoding, c'est-à-dire l'ajout de bases non codantes pour prévenir l'apparition de motifs interdits comme discuté dans la section 2.3.2, n'a pas d'impact sur la distribution uniforme des bases dans les séquences d'ADN.

TABLE 2.3 – Taux de GC moyen et déviation standard de séquences ADN encodées avec DSWE selon le nombre $M$ de motifs interdits. Les résultats sont donnés en pourcentages, pour 10000 séquences de 1000 bits, avec $M \in \{0, 20, 40, 76\}$, $N = 3$ et $m = 6$.

| $M$ | 0 | | | | 20 | | | | 40 | | | | 76 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $N$ | 2 | 3 | 4 | 5 | 2 | 3 | 4 | 5 | 2 | 3 | 4 | 5 | 2 | 3 | 4 | 5 |
| Taux de GC moyen (en %) | 50,0 | 50,0 | 50,0 | 50,0 | 50,0 | 50,0 | 49,8 | 49,9 | 50,0 | 49,8 | 50,0 | 49,9 | 49,9 | 49,6 | 49,6 | 49,6 |
| Taux de GC minimal (en %) | 48,0 | 46,6 | 46,2 | 46,0 | 46,2 | 46,6 | 46,0 | 46,0 | 48,6 | 46,4 | 45,8 | 46,0 | 48,0 | 46,1 | 45,6 | 45,6 |
| Taux de GC maximal (en %) | 52,0 | 54,6 | 54,6 | 54,1 | 50,7 | 54,1 | 54,5 | 54,0 | 52,3 | 54,1 | 54,0 | 53,9 | 51,8 | 54,0 | 53,3 | 53,2 |
| Écart-type (en %) | 1,3 | 1,9 | 2,0 | 2,1 | 1,4 | 1,9 | 2,0 | 1,9 | 1,5 | 1,9 | 2,0 | 2,0 | 1,5 | 1.9 | 2,0 | 2,0 |

Pour aller plus loin, nous avons aussi évalué le taux de GC pour des petites séquences ADN. En effet, comme démontré dans [33], les séquences ADN contenant des régions de 100 à 200 bases avec un taux de GC inférieur à 20% ou supérieur à 80% peuvent poser des problèmes à l'étape de séquençage. C'est la raison pour laquelle GeneArt String Assistant, un outils d'analyse qui associe un score de complexité à une séquence ADN donnée, rejette les séquences avec des régions de 20 et plus bases dont le contenu est déséquilibré en termes de A et T, ou G et C, en particulier les régions comportant uniquement des bases A et T, ou uniquement des bases G et C. L'expérience que nous avons menée pour vérifier que nos séquences ADN encodées par DSWE respectent ces contraintes est la suivante. Le texte [47] a été chiffré avec une clé AES-256 et le flux binaire chiffré obtenu a été séparé en 10000 blocs binaires de 100 bits et également en 10000 blocs de 40 bits. Ensuite, ces blocs ont été codés en séquences ADN à l'aide de $DSWE(N, M, 6)$ pour plusieurs valeurs de $M$, $M \in \{20, ..., 76\}$ et de $N$, $M \in \{2, 3, 4, 5\}$.

Comme le montre le Tableau 2.4, les blocs binaires de 100 bits sont encodés en séquences de 52

à 85 bases ADN. Dans ces séquences, le taux de GC est toujours compris entre $28,0\%$ et $71,1\%$, ce qui est conforme aux exigences. Notons que les premiers et troisièmes quartiles sont toujours inclus dans l'intervalle $45,5-54,0\%$, ce qui montre que la majorité des séquences ADN a un taux de GC équilibré. Le taux de GC moyen est très proche de 50% pour toutes valeurs de $N$ et $M$, avec de meilleures performances pour des plus petites valeurs de $N$. Cela est causé par notre choix de dictionnaires dans l'encodage dynamique DE. Plus nos dictionnaires sont utilisés, plus la proportion de bases ADN est équilibrée. Le nombre $M$ de motifs influe aussi sur le taux de GC. En effet, les motifs ont été choisis pour des raisons pratiques, sans se soucier de l'équilibre des bases. Certaines bases sont donc ajoutées plus souvent par la fenêtre glissante de SWE que d'autres, ce qui déséquilibre légèrement le taux de GC. Globalement,le taux de GC est contrôlé et conforme aux exigences pour des séquences de moins de 100 bases.

TABLE 2.4 – taux de GC pour 10000 séquences de 100 bits encodées par $DSWE(N, M, 6)$ pour plusieurs valeurs de $M$ et $N$. Les résultats sont donnés en pourcentage et en moyenne.

| $M$ | 20 | | | | 40 | | | | 76 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $N$ | 2 | 3 | 4 | 5 | 2 | 3 | 4 | 5 | 2 | 3 | 4 | 5 |
| Longueur moyenne (en bases) | 77 | 54 | 53 | 52 | 81 | 55 | 54 | 53 | 85 | 57 | 56 | 55 |
| taux de GC minimal (en %) | 38,4 | 31,3 | 28,0 | 29,4 | 37,0 | 31,5 | 28,0 | 29,4 | 36,2 | 30,4 | 30,0 | 28,8 |
| Premier quartile (en %) | 47,9 | 46,2 | 45,5 | 45,4 | 46,8 | 45,6 | 45,5 | 45,3 | 47,7 | 45,9 | 45,6 | 45,5 |
| taux de GC moyen (en %) | 50,0 | 50,0 | 49,9 | 49,9 | 49,2 | 49,8 | 49,7 | 49,8 | 49,9 | 49,7 | 49,7 | 49,6 |
| Troisième quartile (en %) | 52,1 | 53,8 | 53,8 | 54,0 | 51,3 | 53,7 | 53,7 | 53,8 | 51,9 | 53,4 | 53,6 | 53,7 |
| taux de GC maximal (en %) | 61,8 | 67,3 | 71,1 | 69,2 | 62 | 70,9 | 69,1 | 69,2 | 65,0 | 67,2 | 71,4 | 68,4 |

Le Tableau 2.5 révèle que pour des blocs binaires de 40 bits, les séquences ADN sont en moyenne d'une longueur de 20 à 31 bases, et le taux de GC varie de $9,5\%$ à $90,0\%$. Les séquences contiennent donc toujours des bases A-T ainsi que G-C, ce qui est conforme aux exigences. Le taux de GC moyen est équilibré, de $49,7\%$ à $50,0\%$. Les premiers et troisièmes quartiles sont proches de 50%, ce qui prouve que la plupart des séquences ont un taux de GC équilibré. Par exemple, pour $N = 3$ et $M = 40$, le taux de GC varie de $15,8\%$ à $85,7\%$ pour des séquences ADN de 21 bases en moyenne. Le taux de GC moyen de ces séquences est $50,0\%$, avec des quartiles proches de cette valeur : $42,8\%$ et $57,1\%$. Comme pour les blocs binaires de 100 bits, les performances sont meilleures pour des valeurs moindres de $N$ et $M$, c'est-à-dire lorsque l'encodage DE avec dictionnaires est le plus utilisé. DE permet donc bien de réguler le taux de GC.

TABLE 2.5 – G-C content pour 10000 séquences de 40 bits encodées par $DSWE(N, M, 6)$ pour plusieurs valeurs de $M$ et $N$. Les résultats sont donnés en pourcentage et en moyenne.

| M | 20 | | | | 40 | | | | 76 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N | 2 | 3 | 4 | 5 | 2 | 3 | 4 | 5 | 2 | 3 | 4 | 5 |
| Longueur moyenne (en bases) | 29 | 20 | 20 | 20 | 30 | 21 | 21 | 20 | 31 | 21 | 21 | 21 |
| G-C content minimal (en %) | 25,0 | 15,8 | 10,5 | 10,5 | 25,0 | 15,8 | 10,0 | 15,0 | 24,0 | 15,0 | 9,5 | 15,8 |
| Premier quartile (en %) | 46,4 | 42,8 | 42,8 | 42,8 | 46,4 | 42,8 | 42,8 | 42,8 | 46,4 | 42,8 | 42,8 | 42,8 |
| G-C content moyen (en %) | 49,9 | 50,0 | 50,0 | 50,0 | 50,0 | 50,0 | 50,0 | 50,0 | 49,9 | 49,7 | 49,7 | 49,7 |
| Troisième quartile (en %) | 53,6 | 57,1 | 57,1 | 57,1 | 53,6 | 57,1 | 57,1 | 57,1 | 53,3 | 56,5 | 56,5 | 57,1 |
| G-C content maximal (en %) | 73,9 | 85,7 | 90 | 85,7 | 73,9 | 85,7 | 90,0 | 85,7 | 78,7 | 81,8 | 81,8 | 82,6 |

En conclusion, notre encodage DSWE préserve un taux de GC équilibré, pour toute taille maximale

d'homopolymères $N$ et tout nombre de motifs interdits $M$. Si avec DSWE, plus une séquence ADN est longue plus l'équilibre tend vers 50%, nous montrons aussi que les résultats sont probants pour de petites séquences ADN, avec un taux de GC conforme aux contraintes de séquençage et de synthèse.

#### 2.6.2.3 Complexité algorithmique

Avec l'encodage dynamique DE, un mot de code de $N$ ou $N-1$ bases est encodé sans créer d'homopolymères plus long que $N$ avec une recherche dans un dictionnaire de petite taille. En effet, les deux premiers bits d'un mot sont encodés en une ou deux bases selon le dictionnaire de $N = 2$ présenté en Figure 2.6B. Ensuite, deux bits sont toujours encodés en une base selon l'encodage classique (Figure 2.6). Chaque dictionnaire est de taille 4, d'où une complexité de $\theta(4)$ par base encodée, donc encoder un mot de code de $N$ bases est une opération de complexité $\theta(4 * N)$. Une suite binaire de $b$ bits a donc une complexité comprise entre $\theta(4 * b)$ et $\theta(4 * b)$, selon les dictionnaires utilisés. Dans le pire cas, ce sont toujours les dictionnaires de $N = 2$ qui sont utilisés et ils encodent 2 bits en 2 bases, tandis que dans le meilleur des cas 2 bits encodent toujours une base.

La complexité de l'encodage par fenêtre glissante dépend du nombre de bases à encoder, de la taille $m$ et du nombre $M$ de motifs interdits. Chaque comparaison de la fenêtre au début d'un motif de $m-1$ bases a une complexité de $\theta(m-1)$. Donc, avant chaque ajout de base, l'ensemble des comparaisons a une complexité de $\theta(M \times (m-1))$. Au total, pour un mot de $N$ bases, la complexité est de $\theta(N \times M \times (m-1))$. En général, pour une séquence de $d$ bases, la complexité associée est linéaire selon $d$ : $\theta(d \times M \times (m-1))$. Pour une suite binaire de $b$ bits en entrée, la pire complexité est donc de $\theta(b \times M \times (m-1))$.

La complexité de DSWE est calculée à partir de celles de DE et SWE. Elle est linéaire selon la taille de la séquence binaire en entrée, avec $\theta(4 \times b + b \times M \times (m-1))$.

### 2.6.3 Performances comparées

Dans cette section, DSWE est comparé aux méthodes de l'état de l'art considérant des contraintes biologiques pour l'encodage de données décrites à la section 2.1.

Pour ce faire, les résultats des performances sont donnés pour les classes de méthodes présentées dans la Section 2.1, c'est-à-dire les méthodes tenant compte d'une, de deux ou de trois des contraintes suivantes : longueur maximale des homopolymères, taux de GC contrôlé et motifs interdits. La Table 2.6 montre les performances de ces méthodes, en indiquant le taux d'information de chaque encodage pour plusieurs valeurs de la taille maximale d'homopolymères $N$ si possible, si un taux de GC contrôlé est pris en compte, et la longueur minimale des mots code ADN encodées. Dans ce qui suit, nous discutons ces résultats par classe de méthodes, en fonction du nombre de contraintes qu'elles prennent en compte.

#### 2.6.3.1 Comparaison aux méthodes satisfaisant 1 à 2 contraintes biologiques

Comme présenté dans la Section 2.3, les méthodes de [18] et [140] encodent des données binaires en séquences ADN en combinant un encodage de données ternaires (avec le dictionnaire de la Table 2.2B) et une étape de compression. [18] a été conçu pour $N = 2$, c'est-à-dire qu'aucun homopolymère n'est toléré. Grâce à l'algorithme de compression de Huffman, le codage de [18] présente un taux d'information de

TABLE 2.6 – Comparaison des méthodes d'encodage de méthodes d'état de l'art et du schéma DSWE proposé . Les taux d'information sont donnés en bits par base (BPB), selon la taille maximale d'homopolymères $N$. '-' signifie Non Applicable, *i.e.* la méthode ne considère pas cette contrainte donc aucun résultat ne peut être donné.

| Classe | Méthode | Taux sans HP | Taux pour $N = 3$ | Taux pour $N = 4$ | Taux pour $N = 5$ | taux de GC contrôlé | Taille minimale des mots code ADN |
|---|---|---|---|---|---|---|---|
| Méthodes avec une contrainte biologique | Cai [139] | Dépendant de la taille des code-words | - | - | - | oui | variable |
| | Goldman [18] | 1.58 | - | - | - | - | variable |
| | Pic [140] | 1.58 | 1.839 | 1.877 | 1.900 | - | N |
| Méthodes avec contraintes d'homopolymères et de taux de GC | Church [17] | - | 1 | - | - | oui | 2 |
| | Song [142] | - | 1.9 | - | - | oui | 10 |
| | Wang [36] | - | 1.917 | - | - | oui | 12 |
| | Grass [39] | - | 1.77 | - | - | oui | 9 |
| | Li [143] | 1 | - | - | - | oui | variable |
| | Erlich [35] | 1.65 – 1.75 | 1.65 – 1.75 | 1.65 – 1.75 | 1.65 – 1.75 | oui | 3 |
| | Nguyen [148] | 1.84 | 1.92 | 1.92 | 1.92 | oui | 200 |
| | Press [151] | variable | variable | variable | variable | oui | 100 |
| Méthodes avec contraintes d'homopolymères, de taux de GC et de motifs interdits | mCGR [38] | 1.8 | 1.8 | 1.9 | 1.9 | oui | 10 |
| | DSWE | 1.33 | 1.883 | 1.92 | 1.94 | oui | $N - 1$ |

1.58 BPB pour les séquences ADN sans homopolymères, comparé à 1.33 BPB pour DSWE en utilisant nos dictionnaires de la Figure 2.6B.

[140] est une généralisation de [18] pour des valeurs plus longues de $N$. Elle repose sur l'algorithme de compression de Shannon-Fano. Ses performances, fournies par ses auteurs, dépendent de la source des données à encoder. Par exemple, l'expérience suivante est menée. Tout d'abord, l'ensemble de données d'images Kodak est pris en entrée. À partir de ces images, les coefficients AC sont obtenus à l'aide de la transformée en cosinus discrète de $8 \times 8$ blocs de pixels. Ensuite, le codage est appliqué à une source qui suit une table de fréquence de ces coefficients AC. Avec ces paramètres, [140] atteint une meilleure densité de données que le codage de Goldman [18]. En effet, la longueur attendue pour un mot de code ADN codant un coefficient AC est de 1.43 nucléotides par symbole de la source pour la méthode de [140] avec une longueur maximale d'homopolymère de $N = 3$, contre une moyenne de 1.64 pour Goldman [18] avec $N = 1$. Quoi qu'il en soit, ces méthodes utilisent des algorithmes de compression de données pour encoder des données binaires en symboles ternaires, alors que DSWE encode des données binaires en mots de code ADN. Plus clairement, DSWE n'utilise pas de compression de données. Nous verrons plus loin dans la section 2.6.6 que, combiné à la compression de données, DSWE est soit équivalent soit meilleur.

La méthode de [139] ne prend en compte que la contrainte de taux de GC et son taux d'information dépend fortement de la taille des mots codes ADN. Elle commence à bien fonctionner pour des tailles de mots de code très importantes, avec les taux d'information suivants : $1,51$ BPB pour des séquences de 50 bases, $1,75$ BPB pour 100 bases et $1,87$ BPB pour 200 bases. DSWE offre des performances similaires

quelle que soit la taille des mots codes de l'ADN, tout en tenant compte d'un plus grand nombre de contraintes.

Comme présenté dans la Table 2.1, certaines méthodes, *i.e.* [17] [142] [36] [39] [143], ne considèrent qu'une taille maximale possible d'homopolymères $N$. En particulier, deux méthodes atteignent un taux d'information légèrement meilleur que DSWE pour $N = 3$ et un taux de GC de 40 à 60%, avec $1,9$ BPB pour [142] pour des mots de code de 10 bases, et $1,917$ BPB pour [36] avec des mots de code de 12 bases, comparé à $1,88$ BPB pour DSWE avec des mots de code de 2 à 3 bases. Cependant, ces méthodes ne prennent en compte qu'une taille d'homopolymère fixe par rapport à DSWE qui est flexible. Cela peut poser des problèmes car, selon le type et la génération des technologies de séquençage, la longueur maximale d'homopolymères requise pour récupérer les données diffère [33]. En outre, ces solutions s'appuient sur une étape de génération de codebook pour déterminer tous les mots de code tenant compte des contraintes biologiques. Comme nous le verrons dans notre comparaison avec [38], la prise en compte des motifs interdits entraînerait une baisse significative du taux d'information, car tous les mots de code commençant ou se terminant par au moins la moitié d'un motif doivent être éliminés du codebook.

Comparons maintenant notre méthode à des solutions d'encodage considérant une taille maximale d'homopolymères variable et un taux de GC contrôlé. Bien que la méthode de [35] présente un bon taux d'information expérimental entre $1,65$ et $1,75$ BPB [38], elle est d'une complexité assez élevée car elle est statistique et, comme le rapportent ses auteurs, il existe certaines contraintes sur le code de correction d'erreur à vérifier (redondance minimale) afin de récupérer correctement les données à l'étape du décodage. Le taux d'information maximal théorique de cette solution est de 1.98 BPB. Dans la pratique, toutes les données ne peuvent pas être décodées à l'aide de cette méthode. Des paquets supplémentaires sont donc utilisés pour encoder les données, ce qui conduit à un taux d'information de 1.65 à 1.75 BPB. La méthode présentée dans [148] offre elle un bon taux d'information pour des longs mots code ADN. Par exemple, avec des homopolymères limités à 3 bases et un taux de GC de $40 - 60\%$, des mots code ADN de 200 bases conduisent à un taux d'information de $1,92$ BPB. Cette méthode présente une légère amélioration par rapport à notre taux de $1,88$ BPB, au prix d'une taille de mot de code fixe de 200 bases peu pratique. En effet, les technologies de synthèse sont encore en développement et ne peuvent donc créer que de petites séquences ADN [158], en particulier la synthèse enzymatique de l'ADN est limitée à moins de cent bases pour assurer une haute fidélité [50]. D'autre part, cette méthode présente de bonnes performances pour les longs mots code ADN, mais le taux d'information diminue avec leur longueur. Par exemple, pour des mots de code de 100 bases, le taux d'information chute à 1.82 BPB pour $N = 3$ et un taux de GC de $40 - 60\%$. De plus, comme cette méthode utilise un codebook de mots code ADN de taille fixe, il est difficile de considérer des motifs interdits sans que le taux d'information ne diminue de manière significative. Le taux d'information de [151] est déterminé a priori, par exemple le "demi-taux" ("half-rate") typique présenté dans l'article encode chaque bit en entrée en une base ADN, ce qui conduit à un taux d'information de 1 BPB. Le code interne est fait de la fusion d'un code correcteur d'erreurs et de l'encodage classique des bits en bases ADN ; il est donc impossible d'obtenir un taux d'information sans inclure une capacité de correction d'erreurs. En raison de la structure de codage, l'algorithme n'est pas adapté aux séquences ADN de moins de quelques centaines de bases et nécessite de coder les données dans 255 séquences ADN minimum, alors que notre méthode peut prendre en compte n'importe quelle

taille et n'importe quel nombre de séquences d'ADN. En outre, cette méthode présente une complexité de décodage qui croît de manière exponentielle avec la taille de l'entrée.

En résumé, même si certaines méthodes ont un taux d'information légèrement meilleur pour des valeurs de $N \leq 4$, aucune d'entre elles ne prend en compte les motifs interdits. Dans la section suivante, nous comparons les performances de notre solution à la seule autre méthode prenant en compte les homopolymères, le taux de GC ainsi que les motifs interdits : [38].

#### 2.6.3.2 Comparaison à mCGR

Comme décrit en Section 2.1, la stratégie adoptée par [38] consiste à éliminer tous les mots de code commençant et se terminant par au moins la moitié d'un motif interdit. Cette méthode n'est pas optimale et conduit à une baisse significative du taux d'information. Dans la section suivante, nous comparons nos résultats expérimentaux à la méthode concurrente en tenant compte du taux de GC, de la taille maximale d'homopolymères et des motifs interdits.

**Taux d'information expérimental** Comparons tout d'abord le taux d'information des deux solutions selon la taille maximale d'homopolymères $N$. Pour cela, nous avons encodé l'image "Lena" de taille 256×256. Les taux d'information expérimentaux de [38] et de DSWE sont donnés dans la Table 2.7, en fonction du nombre $M$ de motifs interdits de taille $m = 6$ bases et d'une taille maximale d'homopolymères $N$. Le taux $R^E_{DSWE(N,M,6)}$ est donné en moyenne. Évaluons d'abord l'influence de la longueur maximale des homopolymères $N$ pour un nombre fixe de motifs $M$ sur le taux d'information de DSWE et de [38]. Pour ce faire, nous avons fixé le nombre de motifs interdits à $M = 60$ et $N \in \{2, 3, 4\}$. Comme le montre la Table 2.7, DSWE atteint un taux d'information de 1.20 BPB, 1.77 BPB et 1.80 BPB pour $N = 2$, 3 et 4, respectivement, tandis que [38] atteint 1.5 BPB pour $N = 2$ et 1.6 BPP pour $N = 3$ et 4. Même si [38] est meilleur dans le cas $N = 2$, DSWE est plus intéressant pour des valeurs plus grandes de $N$ ; des valeurs qui sont plus susceptibles de se produire en pratique en terme de taille maximale d'homopolymères. Notre gain en performance tient principalement à la procédure de nettoyage du dictionnaire de [38] qui supprime un grand nombre de mots code ADN pour un motif interdit donné. De plus, le taux de DSWE tend vers le taux idéal de 2 BPP avec l'augmentation de $N$ alors que le taux de [38] plafonne à $R_{mCGR} = 1.9$ BPB. En effet, à mesure que $N$ augmente, la proportion de bits encodés de manière classique (2 bits dans une base) avec notre méthode augmente également. Notons également que notre solution est plus flexible et peut prendre en compte toute valeur pour $N$, à l'exception de $N = 2$, tandis que [38] ne peut pas prendre en compte des valeurs de $N$ supérieures à la longueur $l$ de ses mots code ADN.

La deuxième expérience que nous avons menée porte sur l'impact du nombre de motifs interdits $M$ sur le taux d'information de DSWE et de [38]. À cette fin, la taille maximale d'homopolymères $N$ a été fixée à 3 et la taille des motifs interdits à $m = 6$, tout en considérant $M\ dans\{10, 20, 30, 40, 50, 60, 70, 76\}$. Dans ce cadre, deux tailles de mots code ADN $l = 6$ et $l = 10$ ont été considérées pour [38], ce qui a conduit à la génération de deux codebooks distincts. Nous choisissons de ces valeurs car une longueur de mot de code de 6 bases correspond à la taille choisie pour les motifs interdits, ce qui nous permet d'effectuer une comparaison équitable entre [38] et DSWE avec une fenêtre glissante de 6 bases. Le codebook avec des mots de code de 10 bases est utilisé pour comparer DSWE à la version optimale de [38]. Les taux d'information expérimentaux pour [38] et DSWE en fonction du nombre $M$ de motifs

TABLE 2.7 – Taux d'information expérimentaux $R^E_{DSWE(N,M,6)}$ de DSWE et $R^E_{mCGR}$ de [38] selon la taille maximale d'homopolymères $N$ et le nombre $M$ de motifs interdits de taille $m = 6$. Le taux d'information expérimental $R^E_{DSWE(N,M,6)}$ est donné en moyenne avec chaque écart-type associé. Le taux expérimental $R^E_{mCGR}$ de [38] est donné pour des mots code ADN de $l = 10$ bases.

| $M$ | $N$ | $R^E_{DSWE(N,M,6)}$ (en BPB) | Écart-type de $R^E_{DSWE(N,M,6)}$ (en BPB) | $R^E_{mCGR}$ (en BPB) for $l = 10$ |
|---|---|---|---|---|
| 0 | 2 | 1.334 | 0.015 | **1.8** |
| | 3 | **1.882** | 0.010 | 1.8 |
| | 4 | **1.921** | 0.008 | 1.9 |
| | 5 | **1.939** | 0.005 | 1.9 |
| 20 | 2 | 1.302 | 0.017 | **1.7** |
| | 3 | **1.839** | 0.011 | 1.8 |
| | 4 | **1.880** | 0.01 | 1.8 |
| | 5 | **1.902** | 0.006 | 1.8 |
| 40 | 2 | 1.239 | 0.019 | **1.6** |
| | 3 | **1.803** | 0.011 | 1.7 |
| | 4 | **1.843** | 0.011 | 1.7 |
| | 5 | **1.864** | 0.007 | 1.7 |
| 76 | 2 | 1.167 | 0.020 | **1.2** |
| | 3 | **1.730** | 0.013 | 1.3 |
| | 4 | **1.768** | 0.011 | 1.3 |
| | 5 | **1.787** | 0.009 | 1.4 |

interdits sont indiqués en Figure 2.17. Nous pouvons constater que la longueur des mots code ADN a une forte incidence sur le taux d'information de la solution donnée dans [38]. Pour les deux tailles de mots de code, $l = 6$ ou $l = 10$, le taux d'information de cette méthode diminue lorsque plus de $M = 30$ motifs sont considérés avec une diminution plus importante pour des valeurs supérieures à $M = 60$. Ce n'est pas le cas de DSWE, dont le taux d'information diminue linéairement (voir la Table 2.7) mais lentement et qui présente de meilleurs performances [38] pour de grandes valeurs de $M$. Par exemple, lorsque $M = 76$, $DSWE(3, 76, 6)$ atteint un taux d'information expérimental de 1.73 BPP alors que [38] obtient un taux de 0.75 BPP et 1.3 BPP avec des mots code ADN de taille $l = 6$ et $l = 10$, respectivement. Comme expliqué précédemment, le fait que [38] s'effondre avec un grand nombre de motifs interdits provient de sa procédure de nettoyage de codebook. En effet, pour tout motif interdit, [38] supprime tout mot de code du codebook commençant ou se terminant par au moins la moitié de ce motif.

En résumé, DSWE démontre un meilleur taux d'information que mCGR [38] pour $N > 2$ pour n'importe quel nombre $M$ de motifs interdits.

**Temps d'exécution** Dans cette section, nous comparons les temps d'exécution de l' encodage pour les deux solutions, en fonction de la longueur maximale des homopolymères $N$ et du nombre $M$ de motifs interdits. Pour les comparer, l'image "Lena" de taille 256×256 a été encodée 100 fois pour chaque jeu de paramètres et chaque solution. Il est important de noter que nous n'avons pas mis en œuvre une version optimisée de notre proposition. Les temps d'exécution de notre proposition et de [38] sont donnés en moyenne dans la Table 2.8 pour différentes valeurs de $N$ et $M$, avec $m = 6$. On peut tout d'abord remarquer que, pour une valeur donnée de $M$, le temps d'exécution de l'encodage de notre proposition diminue lorsque $N$ augmente. Ceci est dû au fait que plus $N$ est grand, plus le nombre de fois où l'Encodage Classique de l'ADN est utilisé est important. Il ressort aussi clairement que la durée d'exécution de notre encodage augmente avec le nombre $M$ de motifs interdits, alors que la durée

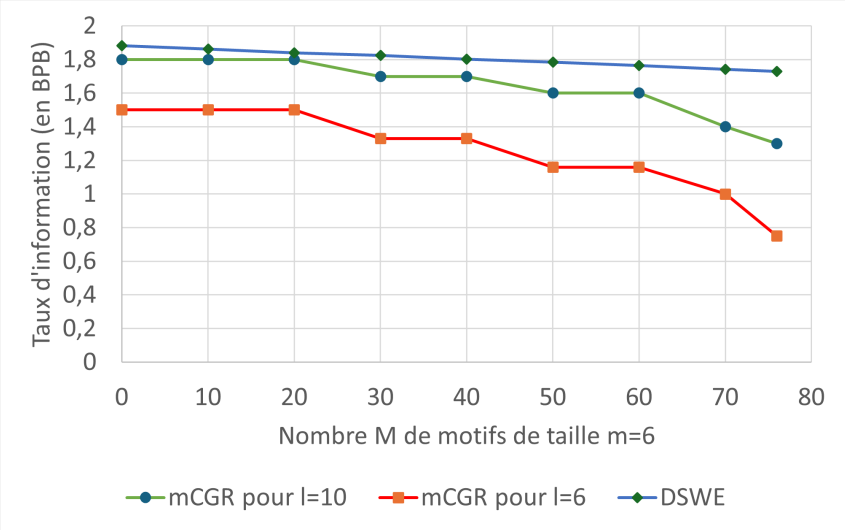



FIGURE 2.17 – Taux d'information expérimentaux $R^E_{DSWE(3,M,6)}$ de $DSWE(N=3,M,m=6)$ et $R^E_{mCGR}$ de [38] selon le nombre $M$ de motifs interdits, avec $N=3$ et $m=6$. [38] a été implémenté avec deux tailles de mots code ADN $l=6$ et $l=10$.

d'exécution de [38] est relativement constante. Concernant [38], même si son temps d'exécution augmente avec $N$ en raison de l'augmentation de la taille de son codebook, il obtient de meilleures performances que DSWE. Cette différence de performance résulte principalement de la fenêtre glissante de DSWE qui effectue $M$ tests à chaque position dans la séquence ADN (voir la Section SWE 2.3.2) alors que [38] utilise simplement son codebook de mots code ADN. Rappelons cependant que notre proposition atteint un taux d'information bien supérieur à [38], avec environ 25% de gain pour $M=76$, et que son implémentation peut être optimisée en termes de nombre de tests, en particulier. En effet, notre algorithme d'encodage a une complexité de $\mathcal{O}(d*m*M)$, avec $d$ la taille en bases de la séquence ADN encodée, $m$ et $M$ la taille et le nombre de motifs interdits, respectivement. Une optimisation possible est l'utilisation d'arbres de préfixes pour tester si une fenêtre correspond au début d'un motif interdit. La complexité de notre solution croît linéairement avec la taille de l'entrée pour des paramètres $N$, $m$ et $M$ fixes.

En conclusion, le temps d'encodage de DSWE augmente avec la hausse des motifs interdits considérés et la diminution de la longueur maximale des homopolymères $N$. [38] obtient de meilleures performances en termes de temps d'exécution par rapport à notre implémentation non optimisée, mais avec un taux d'information beaucoup plus faible que DSWE pour toutes les valeurs de $N>2$ et $M$ (voir la section précédente). Notons également que, contrairement à [38], notre solution DSWE inclut un module de chiffrement pour garantir la confidentialité des données.

**Récupération des séquences ADN** Dans cette section, nous évaluons la robustesse des schémas d'encodage aux erreurs courantes qui existent dans une chaîne de stockage de l'ADN (voir la section 1.1.2). En particulier, nous souhaitons démontrer que, malgré le caractère dynamique de notre encodage DSWE, celui-ci offre au moins les mêmes performances que le schéma de [38] qui génère des séquences d'ADN de longueur fixe.

TABLE 2.8 – Comparaison des temps d'exécution de l'encodage de l'image test Lena par mCGR et DSWE exprimé en secondes selon le nombre de motifs interdits $M$ et la longueur maximale des homopolymères $N$, ainsi que le temps de génération de dictionnaire (en secondes) pour mCGR.

| $M$ | 0 | | | 20 | | | 40 | | | 60 | | | 76 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $N$ | 2 | 3 | 4 | 2 | 3 | 4 | 2 | 3 | 4 | 2 | 3 | 4 | 2 | 3 | 4 |
| Encodage avec DSWE | 1,9 | 1,4 | 1,3 | 3,3 | 2,7 | 2,6 | 4,7 | 3,9 | 3,6 | 5,5 | 4,0 | 3,9 | 6,9 | 4,6 | 5,0 |
| Encodage avec mCGR | 0,6 | 0,9 | 1,1 | 0,4 | 0,6 | 0,8 | 0,3 | 0,4 | 0,5 | 0,2 | 0,3 | 0,3 | 0,2 | 0,2 | 0,2 |
| Temps de génération de dictionnaires de mCGR | 18 | 26 | 33 | 15 | 19 | 23 | 14 | 17 | 18 | 13 | 17 | 18 | 13 | 17 | 17 |

Pour cette expérience, les fonctionnalités biologiques d'une chaîne de stockage d'ADN sont simulées à l'aide du simulateur proposé dans [47] (voir le Chapitre 1, Section 1.1.6). Le simulateur tient compte du fait que les erreurs ne sont pas distribuées de manière identique dans les séquences d'ADN. En effet, certains motifs sont plus sujets aux erreurs que d'autres, et il existe une dépendance entre les erreurs successives. Pour obtenir une simulation d'erreur précise, le modèle a été entraîné sur des données génomiques, ainsi que sur des données expérimentales qui ont été récupérées après avoir été stockées dans l'ADN. Ces deux types de données introduisent en moyenne 3% et 10% d'erreurs, respectivement. De cette manière, le modèle de canal est basé sur les probabilités d'erreurs expérimentales, principalement causées par le séquenceur nanopore MinION en version 9.4. Le simulateur prend en entrée une séquence ADN $D$ et produit $X$ copies de lecture potentiellement erronées/incorrectes/entachées d'erreurs de $D$ dans un fichier fastq. Ces copies sont post-traitées avec la procédure de consensus de [78] en une séquence ADN reconstruite/restaurée. Il est à noter que si trop de copies erronées sont fournies à la procédure de consensus, celle-ci peut décider de ne pas fournir de séquence ADN en sortie. En outre, un tel consensus ne garantit pas qu'il fournisse une séquence d'ADN exempte d'erreurs. Elle peut encore contenir des nucléotides mal décodés. Par exemple, lorsque l'on fait passer dans le simulateur des séquences ADN encodées classiquement, *i.e.* sans aucune contrainte, celles-ci sont lues avec beaucoup d'erreurs. Nous avons testé des séquences de 50 bases ADN, encodant donc 100 bits d'information. Pour ces petites séquences, 45 copies sont nécessaires. Tandis que pour des séquences encodant 1000 bits d'information, nous n'avons pas pu récupérer l'information sans erreur même avec 50 copies.

Les tests suivants visent à évaluer l'influence de la taille maximale d'homopolymères ($N$) et du nombre $M$ de motifs interdits d'une longueur de $m$ bases, ainsi que du nombre de copies séquencées $X$, sur la capacité des trois deux d'encodage à récupérer les données sans erreur. Le texte chiffré [47] a d'abord été séparé en séquences de 1000 bits qui ont ensuite été codées en séquences ADN avec [38] et DSWE de telle sorte que : $N \in \{2, 3, 4\}$, $M \in \{0, 20, 76\}$, $m = 6$. [38] a également été paramétrée pour garantir un taux de GC de $40-60\%$ avec des mots code ADN d'une longueur de $l = 10$ bases. La Table 2.9 indique l'impact de $N$ et $M$ sur le nombre de copies $X$ nécessaires pour obtenir 100% de séquences d'ADN correctement récupérées en sortie du simulateur.

Comme le montre la Table 2.9, quelles que soient les valeurs de $M$ et de $N \leq 4$, les deux méthodes parviennent à récupérer les séquences ADN encodées avec pratiquement le même nombre de copies. De plus, pour ce faire, moins de $X = 35 \sim 40$ copies sont nécessaires. Ce nombre est assez faible si l'on considère que l'amplification des séquences ADN par PCR (*cf.* Chapitre 1 Section 1.1.5) donne accès à un nombre beaucoup plus élevé de copies. Par conséquent, on peut considérer que la propriété dynamique de notre système d'encodage n'a pas d'incidence sur la récupération des séquences ADN, tout en garantissant

TABLE 2.9 – Comparaison du nombre $X$ de copies/reads nécessaires pour récupérer sans erreur les séquences ADN encodées avec mCGR et DSWE après leur passage dans le simulateur publié dans [47] et l'algorithme de consensus de [78], selon une taille maximale d'homopolymères $N$ et un nombre $M$ de motifs interdits de 6 nucléotides. Ces valeurs de $X$ ont été calculées en considérant 1000 séquences ADN qui encodent les mêmes données (1000 bits d'information).

| $M$ | 0 | | | 20 | | | 76 | | |
|---|---|---|---|---|---|---|---|---|---|
| $N$ | 2 | 3 | 4 | 2 | 3 | 4 | 2 | 3 | 4 |
| DSWE | 32 reads | 29 reads | 28 reads | 32 reads | 28 reads | 29 reads | 34 reads | 30 reads | 30 reads |
| mCGR | 28 reads | 26 reads | 26reads | 30 reads | 26 reads | 28 reads | 30 reads | 28 reads | 28 reads |

un taux d'information bien meilleur (voir la figure 2.17). Ainsi, notre encodage dynamique ne semble pas être plus vulnérable à la propagation d'erreurs que les encodages basés sur des mots de code à longueur fixe.

### 2.6.4 Alternatives à DSWE

Comme nous l'avons présenté dans la section 2.4, il est possible de prendre en compte les homopolymères avec une fenêtre glissante au lieu des dictionnaires de DSWE. L'encodage se fait alors uniquement avec des fenêtres, en base 4 ou en base 2, avec des méthodes appelées $SWE^{(4)}$ et $SWE^{(bin)}$, respectivement. Nous comparons dans cette partie les trois encodages selon les critères suivants :

— Le taux d'information

— Le taux de GC

— La complexité de l'algorithme d'encodage

Pour ce faire, 10000 séquences de 40, 100 et 1000 bases ont été séquencées pour plusieurs valeurs de $N$, $N \in \{2, 3, 4, 5\}$, et nombre $M$ de motifs de taille $m = 6$, $M \in [20, 76]$.

Dans la Table 2.10, nous présentons le taux d'information des trois solutions en fonction de deux critères : la taille maximale d'homopolymère autorisée $N$ et un nombre $M$ de motifs interdits de taille $m$. Lorsque seuls les homopolymères sont pris en compte, $SWE^{(4)}$ permet d'obtenir de meilleures performances que DE pour des valeurs de $N$ supérieures à 3, tandis que $SWE^{(bin)}$ présente le plus important taux d'information pour tout $N > 1$. L'utilisation de la fenêtre glissante en binaire permet donc un taux d'information très proche du taux idéal de 2 BPB. Lorsque des motifs sont pris en compte, l'impact est le même pour DSWE et $SWE^{(4)}$. Par exemple, pour ces deux méthodes, la prise en compte de $M = 76$ motifs supplémentaire entraîne une perte de $0,15$ BPB. Cela est dû au fait que les deux méthodes utilisent la même fenêtre glissante $SWE$. La méthode reposant sur une fenêtre en base 2 permet elle de n'avoir une perte que d'environ $0,07$ BPB pour $M = 76$ motifs. Ainsi, $SWE^{(bin)}$ présente les meilleures performances en terme de densité d'information, pour tout $N$ et $M$.

Le taux de GC de séquences encodées par SWE binaire et SWE quaternaire est lui aussi testé, sur de longues et courtes séquences. Le taux de GC de séquences encodant 1000 bits est similaire à celui de DSWE, avec un taux moyen variant de $49,5\%$ à $50,0\%$. Concernant les taux de GC minimaux et maximaux ils sont toujours compris entre $42,5\%$ et $56,6\%$ pour les 2 méthodes, ce qui est comparable aux résultats de DSWE présentés dans le Tableau 2.3, variant entre $45,6\%$ et $54,6\%$.

Nous évaluons aussi le taux de GC des séquences de petites tailles, en particulier les séquences de 40 et 100 bits. L'encodage de séquences de 40 bits ne résulte jamais en des séquences avec seulement des

TABLE 2.10 – Taux d'information pour une taille maximale d'homopolymères de $N$ bases et un nombre $M$ de motifs de taille 6, pour trois méthodes : l'encodage dynamique avec dictionnaires $DE(N)$, la fenêtre glissante en base 4 $SWE_N^{(4)}$ et la fenêtre glissante en base 2 $SWE_N^{(2)}$.

| $M$ | 0 | | | | 20 | | | | 76 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N | 1 | 3 | 4 | 5 | 1 | 3 | 4 | 5 | 1 | 3 | 4 | 5 |
| $R^E_{DSWE(N,M,6)}$ (en BPB) | 1.334 | 1.882 | 1.921 | 1.939 | 1.302 | 1.839 | 1.88 | 1.902 | 1.167 | 1.73 | 1.768 | 1.787 |
| $R^T_{SWE^4(N,M,6)}$ (en BPB) | X | 1,875 | 1,968 | 1,992 | X | 1,836 | 1,929 | 1,953 | X | 1,727 | 1,820 | 1,844 |
| $R^E_{SWE^4(N,M,6)}$ (en BPB) | X | 1,905 | 1,976 | 1,994 | 1,483 | 1,870 | 1,938 | 1,955 | 1,38 | 1,764 | 1,827 | 1,842 |
| $R^T_{SWE^{(bin)}(N,M,6)}$ (en BPB) | 1 | 1,9375 | 1,984 | 1,996 | 0,990 | 1,918 | 1,964 | 1,976 | 0,962 | 1,864 | 1,910 | 1,922 |
| $R^E_{SWE^{(bin)}(N,M,6)}$ (en BPB) | 1,006 | 1,94 | 1,988 | 1,995 | 0,994 | 1,921 | 1,966 | 1,979 | 0,961 | 1,866 | 1,92 | 1,925 |

bases A-T ou G-C, ce qui est conforme aux exigences. Toutefois, certaines séquences s'en approchent : par exemple, pour $N = 5$ et $M = 20$, le taux de GC minimal de $SWE^4$ est de $5,2\%$, ce qui correspond à une seule base. Ce seuil n'est jamais atteint dans nos 10000 séquences encodées avec DSWE, comme présenté dans le Tableau 2.5 : Le taux de GC est toujours compris entre $10,5\%$ et $90,0\%$.

TABLE 2.11 – taux de GC pour 10000 séquences de 40, 100 et 1000 bits encodées avec DSWE, $SWE^4$ et $SWE^{(bin)}$. Chaque taux est donné en moyenne pour des paramètres $N \in \{2,3,4,5\}$ et $M \in \{0,20,40,76\}$

| | 1000 bits | | | 100 bits | | | 40 bits | | |
|---|---|---|---|---|---|---|---|---|---|
| | G-C minimum | G-C moyen | G-C maximal | G-C minimum | G-C moyen | G-C maximal | G-C minimum | G-C moyen | G-C maximal |
| DSWE | 46,4 | 49,9 | 53,4 | 31,5 | 49,8 | 67,7 | 16 | 49,9 | 82,3 |
| $SWE^4$ | 43,9 | 49,7 | 55,2 | 24,8 | 49,7 | 72,4 | 12,1 | 49,6 | 86 |
| $SWE^{(bin)}$ | 44,3 | 49,7 | 55,1 | 24,9 | 49,7 | 72,6 | 12 | 49,5 | 85,8 |

Le Tableau 2.11 présente les moyennes des taux de GC pour DSWE, $SWE^4$ et $SWE^{(bin)}$ pour plusieurs valeurs de $N$ et $M$, pour des séquences de 40, 100 et 1000 bits. Nous observons qu'en moyenne, DSWE permet d'obtenir des séquences avec un taux de GC plus proche de 50%, et des petites séquences plus équilibrées. En effet, les taux de GC minimum et maximaux sont plus proches des extrêmes pour $SWE^4$ et $SWE^{(bin)}$ que pour DSWE. L'encodage avec dictionnaires DE joue donc un rôle d'équilibrage des bases ADN, et donc de régulation du taux de GC.

Évaluons maintenant la complexité des trois méthodes d'encodage. Les encodages $SWE^4$ et $SWE^{(bin)}$ reposent sur un encodage classique ainsi qu'une utilisation d'une ou plusieurs fenêtres. L'encodage classique a une complexité de $\theta(4)$, encoder une suite de $b$ bits a donc une complexité de $\theta(4 * \frac{b}{2}) = \theta(2 * b)$. Les fenêtres utilisées pour empêcher les homopolymères font 4 tests à chaque ajout de base, des tests d'une complexité de $\theta(N)$ en base 4 ou bien $\theta(2N+1)$ en base 2. Ainsi, lorsque la seule contrainte est la taille maximale d'homopolymères $N$, les complexités des trois solutions sont présentées dans le Tableau 2.12. Pour encoder une suite de $b$ bits, DSWE a une pire complexité de $\theta(4 * b)$ pour DSWE, comparé à $\theta((2 * N + 2) * b)$ et $\theta((4 + 4N) * b)$ pour $SWE^4$ et $SWE^{(bin)}$ respectivement. Notre solution d'encodage dynamique DSWE est donc la plus performante pour tout $N$ et $M = 0$.

Si les contraintes incluent $M$ motifs de $m$ bases, une fenêtre glissante est ajoutée aux trois solutions. Elle peut être en base 4, c'est alors la fenêtre de complexité $\theta(M * (m-1) * b)$ utilisée par DSWE, ou bien en base 2. Dans ce cas, l'encodage avec fenêtre glissante a une complexité plus importante de $\theta(M * (2m-1) * b)$, car la fenêtre est de taille $2m-1$ et non $m-1$. Comme présenté dans le Tableau 2.12, la complexité de DSWE est légèrement meilleure que $SWE^4$ et $SWE^{(bin)}$.

TABLE 2.12 – Complexité algorithmique de DSWE et des alternatives, pour encoder une suite binaire de $b$ bits

| | Contrainte : homopolymères de $N$ bases | Contraintes : homopolymères de $N$ bases et $M$ motifs de $m$ bases |
|---|---|---|
| DSWE | $\theta(4 * b)$ | $\theta(4 * b + M * (m-1) * b)$ |
| $SWE^4(N, M, 6)$ | $\theta((2N+2) * b)$ | $\theta((2N+2) * b + M * (m-1) * b)$ |
| $SWE^{(bin)}(N, M, 6)$ | $\theta((4+4N) * b)$ | $\theta((4+4N) * b + M * (2m-1) * b)$ |

Pour toute valeur de $N$ et $M$, l'encodage DSWE présente donc de meilleures performances. Ceci est dû à l'utilisation de l'encodage DE pour limiter le nombre de tests faits par la fenêtre et au fait que la fenêtre glissante soit en base 4. Notons que DSWE présente un avantage supplémentaire : la préparation de dictionnaire est simple. En effet, les contraintes sur le dictionnaire de mots interdits de DSWE sont moindres. On peut toujours avoir au moins 2 mots interdits avec les mêmes $m-1$ premières bases, quels qu'ils soient. Au contrainte, l'encodage $SWE^{(bin)}$ nécessite une préparation de dictionnaires bien plus contraignante, ainsi que la conversion en base 2 du dictionnaire et de la séquence à encoder.

Pour conclure, les méthodes alternatives à DSWE permettent un gain en taux d'information par rapport à DSWE, pour $SWE^4$ pour tout $N > 3$ et pour $SWE^{(bin)}$ pour tout $N > 1$. De plus, l'encodage $SWE^{(bin)}$ permet d'effectuer toutes les opérations en binaire. Toutefois, il nécessite une préparation du dictionnaire de motifs et impose des contraintes fortes sur les motifs possibles. En effet, avec DSWE, il est possible d'interdire $(AAA)$, $(AAC)$, et $(AAG)$, ou en général au moins deux motifs au hasard ayant les mêmes $m-1$ premières bases. Avec $SWE^{(bin)}$, seuls deux motifs sur 4 peuvent être interdits, et ils doivent correspondre au critère suivant : un motif se termine par $A$ ou $C$, et l'autre par $G$ ou $T$. Les contraintes sur le nombre et le choix des motifs sont donc plus fortes que pour DSWE. De plus, l'encodage DE utilisé dans DSWE permet de contrôler le taux de GC de séquences de toutes tailles, ce qui n'est pas le cas pour les deux encodages alternatifs. Enfin, la méthode DSWE présente la complexité la plus basse, grâce à l'utilisation de dictionnaires de petites tailles pour éviter des homopolymères de plus de $N$ bases.

### 2.6.5 Codes correcteurs et encodage dynamique

Dans cette section, notre objectif est de limiter le nombre de copies nécessaires de chaque séquence ADN pour retrouver l'information. Pour cela, nous avons testé plusieurs solutions de codes correcteurs avec notre encodage DSWE. Ceux-ci permettent de corriger des erreurs, au prix d'un ajout de redondance : des bases ADN ne codant pas d'information. Nous avons évalué ces solutions selon plusieurs critères :

— La redondance causée par les codes correcteurs (taux d'information)

— La capacité de correction de chaque code correcteur

— Le nombre de copies nécessaires pour retrouver l'information sans erreurs après passage dans un canal bruité (simulateur de la chaîne de stockage [42])

Pour chacune de ces solutions, nous avons encodé des données et testé les séquences ADN produites avec le simulateur présenté en Section 1.1.6. Celui-ci produit un nombre $X$ de copies potentiellement erronées de chaque séquence, qui sont traitées avec un algorithme de consensus [78]. La séquence en sortie est ensuite décodée avec le code correcteur adapté et le DSWE.

#### 2.6.5.1 Codes LDPC et encodage dynamique

Nous avons testé les codes LDPC afin de corriger des erreurs de substitution et d'effacements dans les séquences (voir Section 2.5.1). Pour cette expérimentation, le fichier de l'article [47] est utilisé comme jeu de données et le DSWE a été paramétré avec $N = 3$ et $M = 20$. Nous avons vu dans la section précédente que l'absence ou la présence de motifs interdits influait de façon négligeable sur le nombre de copies nécessaires pour retrouver l'information. Nous ne faisons donc pas varier ce paramètre dans cette expérimentation. Le code LDPC binaire provient de la librairie Python pyldpc, et le code LDPC quaternaire est celui de l'article [47]. Un des paramètres du simulateur est la taille des séquences en entrée, nous l'avons fixé à 1000 bases pour le code LDPC binaire ainsi que quaternaire. Nous avons aussi choisi d'utiliser le même ratio de redondance, chaque code LDPC ajoutant 50% de symboles de redondance. Voici le processus d'encodage pour chaque code LDPC :

— Code LDPC quaternaire : des suites binaires de 900 bits ont été encodées en des séquences ADN de 490 bases en moyenne. Un base-padding est ensuite appliqué pour obtenir des séquences de 500 bases. Le code LDPC quaternaire avec une redondance de 500 bases est appliqué, ce qui permet d'obtenir des séquences ADN de 1000 bases. Ici, le code correcteur est donc appliqué après le codage DSWE.

— Code LDPC binaire : il est appliqué à des séquences de 918 bits avec une redondance de $\frac{1}{2}$, ce qui donne des séquences binaires de 1836 bits. Après encodage avec $DSWE(3, X, Y)$, les séquences ADN font en moyenne 1000 bases de longueur. La taille de chaque séquence est conservée en mémoire, car l'algorithme de consensus a besoin de ce paramètre. En résumé, le codage DSWE est appliqué après le code correcteur d'erreur.

Un second paramètre du simulateur est le nombre $X$ de lectures à fournir en sortie du séquençage à l'algorithme de consensus [78]. Cet algorithme tente de reconstruire une séquence à partir des $X$ copies potentiellement erronées. Nous avons fait des tests pour des valeurs de $X$ entre 1 et 100. Les situations suivantes peuvent se produire en sortie de consensus :

— Aucun consensus n'est trouvé

— Une séquence erronée est trouvée

— La séquence exacte est retrouvée

Lorsqu'une séquence erronée est trouvée en sortie du consensus, nous évaluons l'impact du code correcteur en donnant la proportion de séquences corrigées par ce code.

Le simulateur prend donc en entrée une séquence ADN $D$ de 1000 bases environ et un paramètre $X$, et sort $X$ copies possiblement erronées de $D$. Pour chaque $X$ entre 1 et 100, nous évaluons 100 séquences différentes avec l'algorithme de consensus.

Dans la Figure 2.18, nous évaluons la proportion de séquences correctement retrouvées après l'étape de consensus, lorsqu'elles sont encodées avec le code LDPC binaire (en Figure 2.18A) ou avec le code LDPC

quaternaire (en Figure 2.18B). Les séquences encodées avec DSWE et le code quaternaire nécessitent bien plus de copies (plus de 50 copies) pour retrouver l'information sans erreur après consensus que la solution avec le code LDPC binaire. Cela est dû au fait que le code LDPC quaternaire ajoute 50% de bases de redondance à la séquence sans respecter les contraintes structurelles (*i.e. homopolymères, taux de GC*). Par exemple, nous avons pu observer que chaque séquence de 1000 bases encodée avec le LDPC quaternaire contient en moyenne deux longs homopolymères de taille $N \geq 6$. Au contraire, les séquences encodées avec le code LDPC binaire ne contiennent aucun homopolymère de taille supérieure à 3, car l'encodage DSWE est appliqué après le code correcteur. Avec ces paramètres, il est possible de retrouver l'information sans erreur à partir de 28 copies.

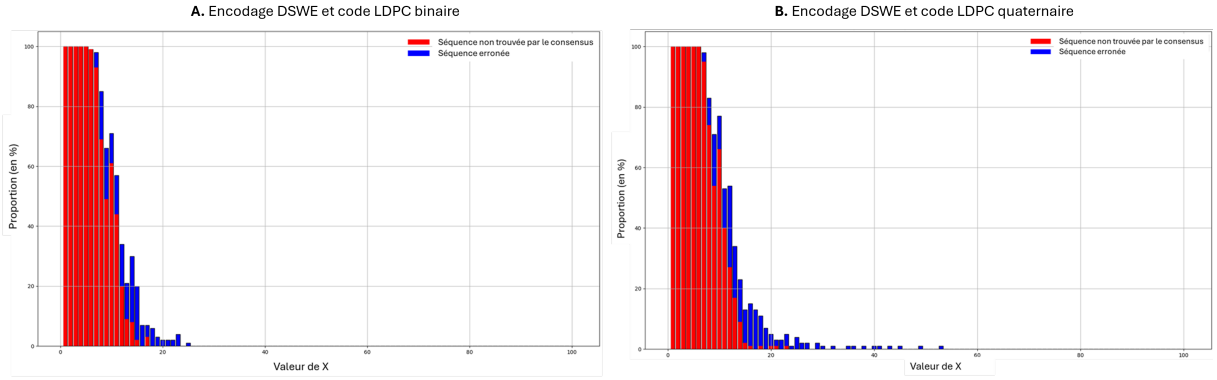



FIGURE 2.18 – Proportion de séquences de 1000 bases correctement retrouvées après simulation et consensus [78] selon le nombre $X$ de copies générées par le simulateur. Pour chaque graphique, la proportion de séquence erronée en sortie du consensus est représentée en bleu, et la proportion de séquence non retrouvée est en rouge. Les séquences encodées avec le code LDPC binaire sont présentées en Figure A, et en Figure B pour le code LDPC quaternaire.

Évaluons maintenant la capacité de correction de chaque code correcteur. Nous donnons en Figure 2.19 la proportion de séquences retrouvées après le décodage LDPC binaire (*cf.* Figure 2.19A) et quaternaire (*cf.* Figure 2.19B). Les graphiques montrent qu'une proportion non négligeable des séquences erronées (en bleu) est corrigée par les codes correcteurs. Pour le code LDPC quaternaire, la proportion moyenne de séquences erronées passe de $3,5\%$ avant le décodage à $1,5\%$. Concernant le code LDPC binaire, cette proportion est de $5,8\%$ avant le décodage, et $3,9\%$ après. Les codes correcteurs permettent donc de retrouver environ la moitié des séquences qui sont erronées en sortie du consensus. En moyenne, le code LDPC binaire permet de corriger $2,12\%$ du total des séquences, tandis que le code LDPC quaternaire permet d'en corriger 3%. Le code LDPC binaire a moins d'impact car il faut faire le décodage dynamique avant la correction d'erreur. Or, une erreur dans la séquence ADN peut causer une avalanche d'erreurs dans la suite binaire, qui devient indécodable. Notons que malgré les corrections des codes LDPC, le nombre $X$ de copies nécessaires à la récupération des données ne baisse pas. Comme présenté dans la Table 2.13, pour le code LDPC quaternaire, il faut au moins 54 copies, contre 28 pour le code LDPC binaire. C'est moitié moins de copies.

En conclusion, les codes LDPC testés permettent de corriger environ la moitié des séquences qui sont erronées en sortie du consensus. Toutefois, cette correction ne permet pas de diminuer le nombre $X$ de copies nécessaire à la récupération des séquences. Comme présenté dans la Table 2.13, autant de copies sont nécessaires pour retrouver des séquences ADN encodées avec DSWE, avec et sans code LDPC binaire.

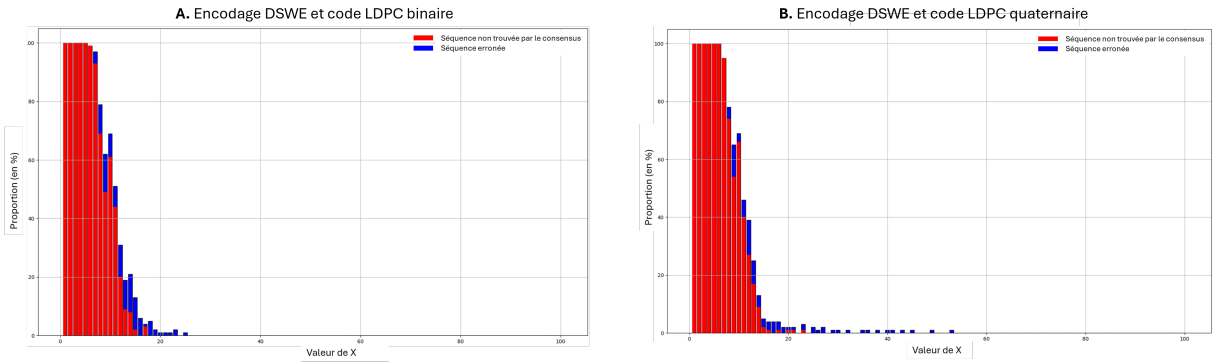



FIGURE 2.19 – Proportion de séquences de 1000 bases correctement retrouvées après simulation, consensus [78] et décodage LDPC selon le nombre $X$ de copies générées par le simulateur. Pour chaque graphique, la proportion de séquence erronée après le décodage est représentée en bleu, et la proportion de séquence non retrouvée est en rouge. Les séquences encodées avec le code LDPC binaire sont présentées en Figure A, et en Figure B pour le code LDPC quaternaire.

TABLE 2.13 – Nombre $X$ de copies nécessaires pour retrouver l'information sans erreur et taux de séquences bien corrigées, avec des codes de LDPC binaires et quaternaires associés à l'encodage DSWE. Test sur des séquences de 1000 bases en moyenne.

| Correction d'erreur | Seuil de lecture (nombre minimal $X$ de copies) | Taux moyen de séquences bien corrigées |
|---|---|---|
| Aucun | 28 | 0 |
| LDPC binaire | 28 | $2,12\%$ |
| LDPC quaternaire | 54 | $3,00\%$ |

De plus, ces deux codes nécessitent une redondance de 50%, divisant par deux le taux d'information des séquences ADN. Il est important de noter que le code LDPC quaternaire ne permet pas de maîtriser les contraintes structurelles sur une partie de la séquence ADN, ce qui dégrade la qualité des copies. Ainsi, il faut plus de 50 copies pour retrouver l'information sans erreur. Dans la section suivante, nous testons une solution de correction d'erreurs qui permet de corriger des effacements de séquences ADN entières, principale cause d'erreurs en sortie du simulateur. Pour cela, comme nous allons le voir, une structure de deux codes correcteurs concaténés est appliquée sur plusieurs séquences ADN.

#### 2.6.5.2 Création d'une structure de codes concaténés binaires : 2 codes de Reed-Solomon

Dans cette section, nous utilisons une structure de deux codes correcteurs concaténés qui permet de corriger des séquences ADN entières. Pour cela, nous nous inspirons de la structure de codes correcteurs concaténés en base 47 de Grass [32] utilisée avec un encodage ADN fixe, pour construire notre propre structure de codes GRS binaires à laquelle nous appliquons notre encodage dynamique DSWE.

La solution de Grass utilise deux codes de Reed-Solomon en base 47 et un dictionnaire qui traduit un élément en base 47 en 3 bases ADN. La structure des codes concaténés de Grass est présentée en Figure 2.20A. L'encodage fonctionne comme suit :

— Un code externe $GRS(713, 594)$ est appliqué à tout le bloc de données (en blanc), c'est-à-dire 594 colonnes de 30 éléments en base 47. En sortie, il y a 713 colonnes de 30 éléments.

— Un index de 3 éléments en base 47 (en jaune) est ajouté au début de chacune de ces 713 colonnes.

— Le code interne $GRS(39, 33)$ est appliqué séparément à chaque colonne. En sortie de ce deuxième code GRS, une matrice de 713 colonnes de 39 éléments en base 47 est obtenue.

— Chaque colonne est convertie en séquence ADN selon le dictionnaire fixe présenté précédemment : un élément en base 47 est converti en un mot de 3 bases ADN. Ce dictionnaire permet de limiter la taille des homopolymères à $N = 3$.

Pour récupérer l'information, les séquences lues sont tout d'abord converties en base 47 puis décodées en commençant par le code interne, afin de corriger séparément les erreurs individuelles de chaque séquence. Les séquences sont ensuite ré-organisées à l'aide de leurs index (*cf.* Figure 2.20A) pour permettre le décodage externe et la correction des séquences entières (colonnes de 30 éléments en base 47), et la récupération des séquences complètement perdues.

Notre structure de codes correcteurs est présentée en Figure 2.20B. Elle repose aussi sur deux codes de Reed-Solomon, mais ceux-ci sont en base 2, afin de pouvoir appliquer notre encodage dynamique. Afin de nous comparer à la structure de Grass, nous avons choisi des paramètres les plus proches possible. Les paramètres de chaque méthode sont listés en Table 2.14. Comme présenté en Figure 2.20, notre code externe est un code $GRS(400, 332)$ sur $GF(2^{297})$. Chaque élément en entrée du code externe est donc une colonne de 297 bits. Ce code entraîne une redondance de 17%, comparé à $16,69\%$ pour Grass. Les index sont des séquences de 18 bits, *i.e.* des éléments de $2^{18}$, ce qui représente $4,9\%$ de la séquence, contre $7,7\%$ pour Grass. Concernant le code interne, notre code $GRS(369, 315)$ est appliqué séparément à chaque colonne de 315 bits et génère des séquences de 369 bits. Cela correspond à une redondance de $14,63\%$, comparé à $15,38\%$ pour le code interne de Grass. Une fois les codes correcteurs appliqués aux données, l'encodage DSWE est appliqué à chaque colonne. En sortie, les séquences ADN font environ 196 bases, contre 117 pour Grass. Ces tailles d'un ordre de grandeur similaire permettront de comparer les méthodes avec le simulateur de la chaîne de stockage. Notons toutefois que les séquences plus longues ont souvent plus d'erreurs et nécessitent plus de copies pour être retrouvées sans erreur.

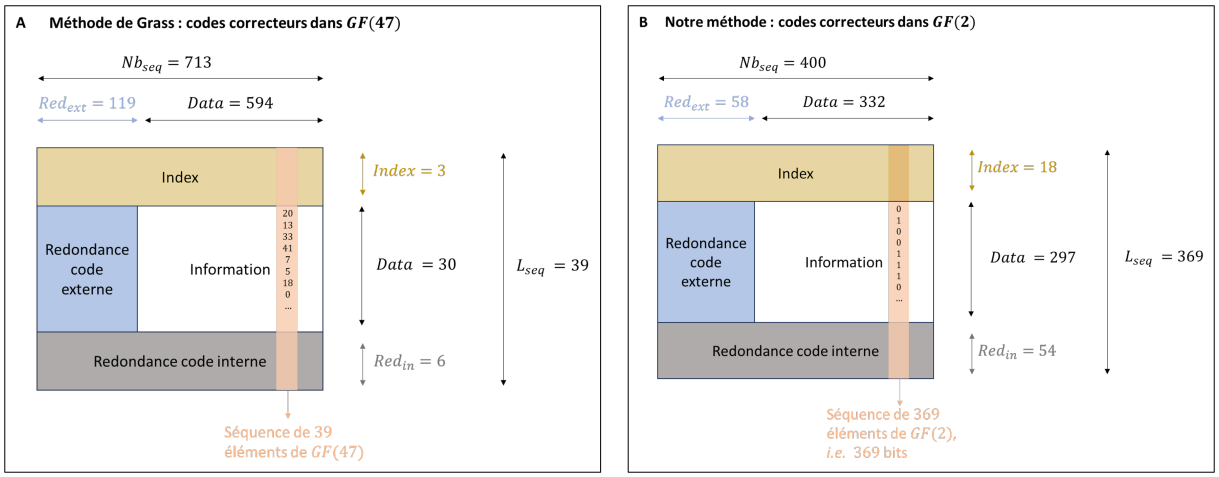



FIGURE 2.20 – Structures de codes correcteurs avec deux codes de Reed-Solomon (externe et interne) appliqués successivement à un bloc de données (bloc "Information"). Le schéma A présente la structure de Grass [32] pour des éléments en base 47, tandis que notre version pour des éléments en base 2 est présentée au schéma B. Dans les deux cas, le code externe représenté en bleu est appliqué en premier, avant d'indexer chaque colonne et d'appliquer le code RS interne en gris. Chaque colonne résultante, représentée en orange, sera encodée en une séquence ADN selon l'encodage choisi. La méthode de Grass utilise un dictionnaire fixe, tandis que notre méthode fait usage de notre encodage dynamique DSWE.

Le taux global d'information de la méthode de Grass est défini par :

$$R_{Grass} = R_{out} \times R_{in,index} \times R_{dico} = \frac{594}{713} \times \frac{33-3}{39} \times \frac{\log_2(47)}{3} = 0,64 \times \frac{\log_2(47)}{3} \tag{2.18}$$

où : $R_{out}$ et $R_{in,index}$ sont les ratio de redondance des codes correcteurs externe et interne (avec index), respectivement. $R_{dico}$ est le taux d'information fixe de la méthode de Grass, encodant un élément en base 47 en 3 bases ADN.

Contrairement à l'encodage fixe de Grass, notre méthode repose sur un encodage dynamique, les valeurs de taux d'information seront donc données en moyenne.

Le taux d'information moyen de notre solution est le suivant :

$$R_{DSWE_{N,M,m},CC} = R_{out} \times R_{in,index} \times R_{DSWE_{N,M,m}} = \frac{332}{400} \times \frac{315-18}{369} \times R_{DSWE_{N,M,m}} = 0,668 \times R_{DSWE_{N,M,m}} \tag{2.19}$$

Ainsi, pour $N = 3$ et $M = 0$, la solution de Grass démontre un taux d'information fixe de $1,187$ BPB, comparé à $1,257$ BPB en moyenne pour notre solution avec le DSWE.

Pour évaluer notre solution avec le simulateur, nous avons encodé les mêmes données, *i.e.* le fichier [47] chiffré avec AES-256, avec les structures de codes correcteurs puis l'encodage ADN de chaque solution. 8 blocs de données sont encodés. Chaque bloc contient 400 séquences d'environ 196 bases pour notre solution, et 713 séquences de 117 bases pour la solution de Grass. Chacune des séquences est fournie au simulateur, paramétré pour produire un nombre $X$ de lectures de la séquence entre 1 et 100. Les $X$ copies sont ensuite traitées par l'algorithme de consensus [78]. En sortie du consensus, sans appliquer de code correcteur, les séquences peuvent être retrouvées sans erreur pour les deux méthodes pour $X \geq 30$ copies.

Après le consensus, les deux codes correcteurs sont décodés. La Figure 2.21 donne les résultats reçus par le décodeur externe, pour Grass en Figure 2.21A et pour DSWE en Figure 2.21B. En ordonnée est représenté le nombre de séquences effacées, et en abscisse le nombre $X$ de copies utilisées pour le consensus. Grâce à la capacité de correction du décodeur externe, dès qu'il y a moins de 58 séquences effacées pour notre méthode, et moins de 119 pour celle de Grass, la totalité des données peut être retrouvée. Cela se traduit par le plafond de capacité de correction en rouge : l'information peut être retrouvée dès $X = 9$ copies pour Grass (*cf.* Figure 2.21A), et dès $X = 11$ pour notre méthode (*cf.* Figure 2.21B), ce qui est une amélioration significative par rapport aux $25 - 30$ copies nécessaires avant le décodage.

Ainsi, la structure de codes concaténés permet de diminuer significativement le nombre $X$ de copies nécessaires pour récupérer l'information sans erreur. L'encodage de Grass et notre encodage dynamique nécessitent en moyenne 9 et 11 copies, respectivement. Les codes concaténés ont donc une forte capacité de correction, et leur capacité de correction a été peu impactée par l'aspect dynamique de notre DSWE lors de nos tests. Toutefois, les codes concaténés entraînent une redondance d'environ 33%, impactant le taux d'information de $0,7$ BPB environ pour les valeurs testées. De plus, la structure des codes présente une contrainte sur le volume de données, qui est fixé par le choix des codes correcteurs. Pour encoder des volumes plus petits, il faut choisir de nouveaux paramètres de codes Reed-Solomon généralisés et passer par une étape complexe de génération de polynômes. Notons que l'utilisation de DSWE implique aussi des contraintes. Tout d'abord, il faut stocker la longueur de chaque séquence encodée par DSWE.

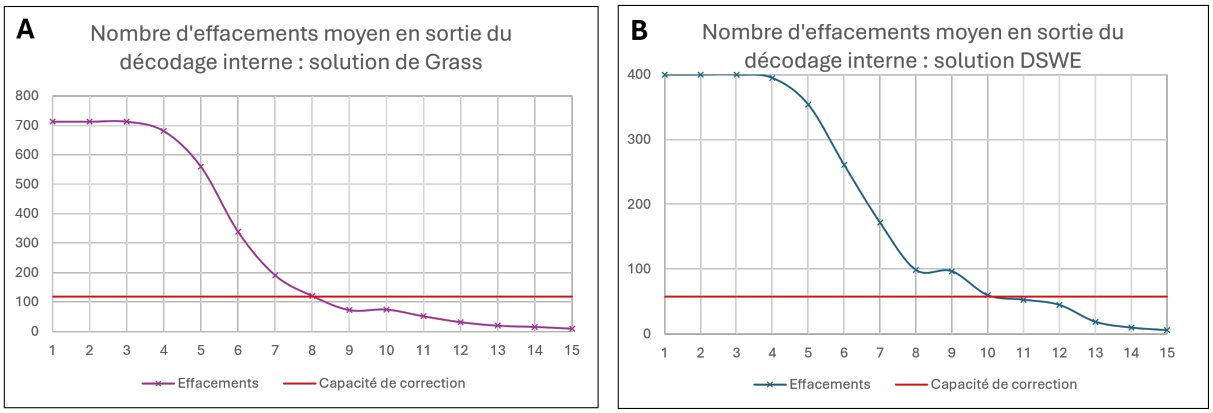



FIGURE 2.21 – Nombre moyen de séquences effacées en sortie du décodage interne avec les méthodes de Grass (Figure A) et notre méthode DSWE (Figure B). Le nombre moyen d'effacements est donné selon le nombre $X$ de copies utilisées pour le consensus. La capacité de correction du décodage externe est représentée en rouge, elle correspond à 119 séquences parmi 713 pour Grass, et 58 séquences parmi 400 pour notre méthode.

D'autre part, le décodage dynamique de DSWE est très sensible aux erreurs. Une simple erreur peut donc compromettre le décodage de tout une séquence. C'est pourquoi, même si les résultats en sortie du consensus sont similaires pour la méthode de Grass et DSWE, notre méthode présente des résultats légèrement moins bons après les décodages.

TABLE 2.14 – Comparaison des solutions d'encodage avec codes correcteurs en matrice de Grass [32] et notre solution avec DSWE

| Critères | Méthode de Grass | Notre méthode |
|---|---|---|
| Chiffrement | AES 256 | AES 256 |
| Entrée (bits) | 98983 | 98604 |
| Outer code | $[713, 594, 120] \in GF(47^{30})$ | $[400, 332, 69] \in GF(2^{297})$ |
| Redondance de l'outer code $Red_{out}$ | $16,69\%$ | $17\%$ |
| Nombre d'application de l'outer code | 1 | 1 |
| Inner code | $[39, 33, 7] \in GF(47)$ | $[41, 35, 7] \in GF(2^9)$ |
| Redondance de l'inner code $Red_{in}$ | $15,38\%$ | $14,63\%$ |
| Nombre d'applications de l'inner code | 713 | 400 |
| Taille des index | 3 éléments de $GF(47)$ | 18 éléments de $GF(2)$ |
| Nombre d'indices disponibles | $47^3 = 103823$ | $2^{18} = 262144$ |
| Redondance d'index dans une séquence | $7,7\%$ | $4,9\%$ |
| Taille maximale d'homopolymères $N$ | 3 | 3 |
| Taux d'information | 1,18 BPB | 1,22 BPB |
| Taille des séquences (avec amorces) | 117(158) bases | 201(268) bases |
| Padding | 23760 | 394416 |

### 2.6.6 Compression et encodage DSWE

Dans l'idée d'optimiser la densité des données stockées dans les molécules d'ADN, il est possible de compresser les données avant de les encoder avec DSWE. En effet, compte tenu du prix de la synthèse de l'ADN, un gain de taux d'information peut réduire considérablement les coûts. De plus, la compression maximise l'entropie des données, tout comme le chiffrement, qui fournit un flux de bits indépendants et identiquement distribués. Cette propriété garantit que le taux de GC est proche de 50% dans les séquences encodées. Dans ce qui suit, nous comparons notre méthode à deux solutions d'encodage présentées en Section 2.1 qui combinent une étape de compression à une méthode d'encodage pour produire des séquences ADN. La première méthode [144] s'appuie sur une compression avec sur un arbre de Huffman à

variance minimale, tandis que la deuxième [145] s'appuie sur un encodage JPEG modifié.

Pour une comparaison équitable, nous avons compressé les données à l'aide d'un code de Huffman standard à variance minimale ou l'algorithme JPEG standard, appliqués avant d'utiliser DSWE. En termes de critères de comparaison, nous avons considéré le taux de compression $r$ :

$$r = \frac{|S|}{|D|} BPB \tag{2.20}$$

où : $S$ est une séquence binaire à compresser, et $D$ la séquence ADN produite.

Les jeux de données utilisés sont les mêmes que ceux dans [144] et [145] : les images Lena ($512 \times 512$ pixels, 8 bits/pixels) et Jet ($256 \times 256$ pixels, 8 bits/pixels) ; le texte "Where the mind is without fear" [159] de 490 caractères ASCII comme dans [144], et 7 images JPEG avec différents niveaux de qualité comme dans [145]. Pour une juste comparaison, la taille maximale d'homopolymères $N$ de DSWE a été fixée à 1 pour [144] et à 3 pour [145], et les motifs interdits ne sont pas pris en compte.

Comparons d'abord DSWE précédé d'une étape de compression de Huffman à la méthode de [144]. Cette dernière peut construire des arbres de Huffman pour n'importe quelle taille de symboles en entrée. Par souci de simplicité, nous donnons des résultats expérimentaux en utilisant un arbre de Huffman à variance minimale codant 8 bits par symbole, c'est-à-dire un pixel par symbole, pour les deux méthodes. En outre, les données sont codées sans homopolymères. En utilisant le texte de test utilisé dans [144], notre solution montre une amélioration de 0.69 BPB. Par contre, notre solution d'encodage présente un taux de compression légèrement inférieur à la méthode concurrente de [144] pour les deux images, de $0,16$ et $0,17$ BPB. Rappelons cependant que DSWE n'est pas optimisée pour des tailles maximales d'homopolymères $N$ inférieures à 3. Ainsi, le simple fait de changer la taille maximale d'homopolymères $N$ de 1 à 3 donne des taux de compression de 2.305 BPB pour l'image Jet et de 2.017 BPB pour l'image Lena, ce qui représente une amélioration significative. Ces résultats sont importants, car les tailles maximales d'homopolymères $N \in [3, 6]$ sont des valeurs utilisées actuellement [33].

TABLE 2.15 – Comparaison des ratios de compression en BPB (bits par base) du schéma d'encodage de [144] et DSWE précédé par la compression avec Huffman

| Data | Ratio de [144] | Ratio de DSWE |
|---|---|---|
| Texte [159] | 1.86 | **2.55** |
| Lena | **1.59** | 1.43 |
| Jet | **1.81** | 1.63 |

Comparons maintenant DSWE avec la méthode de [145], *i.e.* JPEG-DNA. Pour ce faire, notre méthode consiste à ajouter une étape de compression d'image JPEG standard avant l'encodage DSWE pour $N = 3$. Pour une comparaison équitable, le taux de compression de JPEG-DNA est calculé en utilisant uniquement le flux binaire de l'image. Les taux de compression des deux méthodes sont donnés en fonction du PSNR ("Peak Signal to Noise Ratio") de chaque image compressée. Le PSNR est l'une des mesures standard de la qualité d'une image. Comme le montre la Table 2.16, notre solution donne de meilleurs résultats en termes de taux de compression pour les 7 images pour un PSNR correspondant à une qualité d'image de 50%. En moyenne, elle présente un gain de $37,5$ % de taux de compression sur les 7 images testées par rapport aux résultats de [145].

Ces résultats montrent que notre solution d'encodage DSWE peut facilement être associée à des

TABLE 2.16 – Comparaison des ratios de compression en BPB (bits par base) de JPEG-DNA [145] et DSWE précédé par la compression avec JPEG . Pour chaque image, les ratios de compression sont comparés pour le même PSNR, équivalents à des niveaux de qualité de 50%.

| | baboon (grayscale) | gray21 | jet | lena | baboon (color) | Kodim 23 | pepper |
|---|---|---|---|---|---|---|---|
| PSNR (dB) | 28.2 | 48.0 | 36.1 | 35.8 | 28.6 | 38.5 | 34.3 |
| JPEG-DNA | 8.19 | 78.83 | 17.22 | 18.47 | 18.64 | 49.00 | 43.67 |
| DSWE | **10.81** | **91.40** | **22.38** | **23.79** | **29.47** | **81.33** | **57.44** |

algorithmes de compression et conduire à des gains significatifs en termes de densité de données.

Dans cette partie expérimentale, nous avons montré que notre solution d'encodage dynamique DSWE présente un taux d'information élevé pour plusieurs tailles maximales d'homopolymères $N$ et nombres $M$ de motifs de 6 bases, qu'il préserve la distribution uniforme des données chiffrées, et qu'il démontre une complexité linéaire selon la taille des données en entrée. Comparé aux méthodes de l'état de l'art, notre méthode peut présenter un taux d'information légèrement inférieur à certaines d'entre elles, mais elle est une des deux seules à considérer la contrainte des motifs interdits. Comparée à la seule autre méthode de ce type, mCGR [38], nous constatons que notre méthode DSWE a un meilleur taux d'information pour tout $N > 2$ et $M$. En termes de temps d'encodage, mCGR obtient de meilleures performances, mais requiert une étape supplémentaire de génération de dictionnaire. Enfin, nous avons montré via des tests avec un simulateur que les deux méthodes parviennent à récupérer les séquences ADN encodées avec pratiquement le même nombre de copies. Ainsi, notre encodage dynamique ne semble pas être plus vulnérable à la propagation d'erreurs que les encodages basés sur des mots de code à longueur fixe. D'autre part, nous avons comparé DSWE à deux alternatives reposant sur des fenêtres glissantes. Nous avons montré que, bien qu'elle obtiennent de meilleures performances en termes de taux d'information, l'absence d'encodage par dictionnaires entraîne un taux de GC plus déséquilibré, une plus grande complexité algorithmique, ainsi que des choix de motifs interdits plus restreints. Enfin, nous avons montré qu'il est possible d'associer à notre DSWE des codes correcteurs et des algorithmes de compression, afin d'optimiser la récupération des données et le taux d'information, respectivement.

## 2.7 Conclusion

Dans ce chapitre, nous avons développé une solution d'encodage permettant de maîtriser la structure de l'ADN, notamment des motifs présents dans les séquences, tout en prenant en compte les contraintes imposées par les étapes biologiques du stockage sur ADN. Ainsi, comme nous le verrons dans le chapitre 3 qui suit, les séquences ADN encodées pourront être manipulées via des opérations biologiques, sans poser de problèmes quant au codage de l'information.

Dans la première partie de ce chapitre, nous avons détaillé les contraintes biologiques à respecter pour encoder des données en séquences ADN, avant de présenter un état de l'art des méthodes existantes. Plusieurs classes de méthodes se distinguent, selon le nombre de contraintes considérées parmi : une taille maximale d'homopolymères de $N$ bases, un taux de GC équilibré, un nombre $M$ de motifs interdits de $m$ bases. Une seule méthode, mCGR, prend en compte ses trois contraintes. Cette méthode s'appuie sur une construction de dictionnaire de mots de code de taille fixe respectant les contraintes souhaitées. En conséquence, lorsqu'un mot de code commence ou se termine par au moins la moitié d'un motif interdit

ou d'un long homopolymère, le mot est supprimé du dictionnaire. Cela entraîne une forte baisse du taux d'information de mCGR lorsque le nombre de motifs interdits augmente. Cette méthode d'encodage fixe n'est donc pas optimale lorsque de nombreux motifs sont pris en compte.

Afin d'optimiser la densité d'information stockée dans nos séquences ADN, nous avons présenté dans la deuxième partie un nouveau schéma d'encodage dynamique avec fenêtre glissante qui permet d'encoder des données binaires chiffrées par AES en des séquences ADN de longueur variable. Paramétré avec $DSWE(N, M, m)$, il permet de satisfaire les trois contraintes principales : une taille maximale d'homopolymères de $N$ bases ; un taux de GC équilibré proche de 50% ; et $M$ motifs interdits de $m$ nucléotides. Il s'appuie tout d'abord sur quatre dictionnaires sélectionnés pour l'encodage des données en fonction de la dernière base encodée, avec comme objectif d'éviter la génération d'homopolymères de taille $N + 1$. Chacun de ces dictionnaires associe des blocs binaires de longueur fixe à des mots de code ADN de longueur variable. En second lieu, DSWE tire parti d'une fenêtre glissante de taille $m - 1$ bases qui est exploitée pour détecter le début des motifs interdits. Une base non codante peut être ajoutée pour éviter les motifs interdits. Ceux-ci peuvent être utilisés pour l'indexation des données ou comme primers. Comme nous l'avons démontré, notre proposition préserve également la distribution uniforme des données d'entrée chiffrées par AES, ce qui permet d'obtenir des séquences d'ADN avec un taux de GC équilibré. Notre méthode d'encodage est flexible et peut être facilement adaptée à n'importe quelle valeur de $N$ et $m$, contrairement à mCGR qui est limité à des tailles de $N$ et $m$ au plus égales à la taille de ses mots de code, c'est-à-dire jusqu'à 12 bases. De plus, notre méthode d'encodage nous permet d'atteindre un meilleur taux d'information que la méthode concurrente pour $N > 2$ et quel que soit le nombre $M$ de motifs interdits. Enfin, pour un nombre fixe de motifs interdits $M$, le taux d'information de notre solution tend vers le taux idéal de 2 BPB lorsque la taille maximale d'homopolymères $N$ augmente. Pour $N$ fixé, ce taux d'information diminue lentement de façon linéaire en fonction de $M$.

Dans ce travail, jusqu'à 76 motifs interdits de 6 bases ont été considérés. Notre méthode peut être généralisée pour interdire un plus grand nombre de motifs avec une faible diminution du taux d'information. Ce n'est pas le cas de [38], dont le taux d'information chute rapidement lorsque de nombreux motifs interdits sont considérés. En outre, la méthode de [38] est limitée à des longueurs de motifs interdits égales à la longueur du mot de code ADN, c'est-à-dire jusqu'à 12 bases, alors que la nôtre n'a pas de limite supérieure. Il est à noter que notre code en Python3 n'est absolument pas optimisé et pourrait être beaucoup plus rapide. Mais le temps d'exécution de l'encodage/décodage augmente avec le nombre de motifs interdits, car le nombre de comparaisons avec la fenêtre glissante augmente. Ce n'est pas le cas de [38] qui est plus rapide car ses temps d'exécution d'encodage/décodage sont constants. En effet, ils sont basés sur la taille du dictionnaire. Quoi qu'il en soit, comme le montre la section de comparaison des performances, notre approche peut être utilisée efficacement en pratique.

De plus, nous avons montré dans la partie expérimentale qu'il est possible d'optimiser la récupération des données en utilisant une structure de deux codes correcteurs concaténés. Celle-ci permet de diviser par 2 le nombre $X$ de copies d'une séquence nécessaires au décodage sans erreur. Notre encodage dynamique peut donc être utilisé avec des codes correcteurs et permet d'optimiser la densité de l'information. Nous avons aussi montré qu'il est possible de précéder DSWE d'une étape de compression par le code de Huffman ou l'algorithme JPEG, et d'obtenir un gain significatif en termes de taux d'information.

Pour aller plus loin, notre schéma peut être utilisé pour gérer des motifs interdits d'une taille supérieure

à 6 bases au prix d'une augmentation de la complexité du codage. Une façon de surmonter ce problème est de considérer des amorces de longs motifs interdits qui sont la combinaison de motifs plus petits. Par exemple, on peut construire des amorces de 18 et 24 bases qui sont la concaténation de motifs interdits de 6 bases. En outre, notre solution peut être généralisée pour prendre en compte des motifs interdits de tailles distinctes, chacun avec une fenêtre glissante adaptée, au prix toutefois d'une augmentation de la complexité.

Notre méthode d'encodage est indépendante de l'algorithme de chiffrement choisi, tant que ce dernier produit des données uniformément distribuées pour garantir un taux de GC équilibré dans les brins ADN encodés. Il s'agit d'une propriété que possèdent les cryptosystèmes parfaits. Cette possibilité de changer de cryptosystème dans notre système présente un grand intérêt car, dans quelques décennies, AES pourrait ne plus être sûr. Avec le temps, AES peut simplement être remplacé par un cryptosystème plus efficace sans modifier les principes de base de notre encodage dynamique avec fenêtre glissante DSWE.

Dans ce chapitre, nous avons donc proposé une nouvelle méthode d'encodage des données chiffrées stockées sous forme d'ADN. Elle tient compte de certaines contraintes biologiques, à savoir le taux de GC et les homopolymères, et les motifs interdits de 6 nucléotides utilisés pour l'indexation des données. Cette solution repose sur un nouvel encodage dynamique par fenêtre glissante (DSWE) qui permet : i) d'éviter les homopolymères de plus de $N$ bases en encodant des données binaires de taille fixe dans des mots code ADN de longueur variable, et ii) d'empêcher l'apparition des motifs interdits pendant l'encodage de la séquence ADN à l'aide d'une fenêtre glissante. Par rapport aux solutions les plus récents, notre méthode permet d'obtenir de meilleurs taux d'information pour différentes longueurs maximales d'homopolymères $N$, tout en garantissant un taux de GC équilibré. De plus, en utilisant un récent simulateur de chaîne de stockage d'ADN, nous montrons que la récupération des données n'est pas affectée par notre DSWE.

Nous avons défini une solution d'encodage qui permet de stocker des données chiffrées dans de l'ADN et de les retrouver sans erreur. Toutefois, cette solution est purement numérique et pourrait devenir vulnérable lorsque les puissances de calcul permettront de déchiffrer notre chiffrement. Qu'en est-il alors d'une solution de sécurité dans le domaine biologique ? C'est l'objet du Chapitre 3, dans lequel nous allons présenter une solution de chiffrement des données qui conditionne le déchiffrement des données à des manipulations biologiques de l'ADN. Plus précisément, nous allons définir des fonctions biologiques permettant de modifier la structure de molécules d'ADN encodées avec notre DSWE, et présenter un algorithme de déchiffrement biomoléculaire à partir de ces fonctions.

Chapitre 3

# Une chaîne de stockage sécurisé de données dans l'ADN, du numérique au moléculaire

L'objectif de ce chapitre est d'intégrer nativement une solution de sécurité à la technologie de stockage dans l'ADN. Cette dernière est développée dans une perspective de stockage longue durée. Or, les solutions actuelles de sécurité reposent sur des chiffrements numériques, voués à devenir obsolètes à court ou moyen terme, notamment avec l'arrivée de l'ordinateur quantique. La sécurité des données stockées dans l'ADN à long terme reste donc une question ouverte. Quelques solutions exploratoires du domaine de la cryptographie et de la stéganographie ADN visant à sécuriser dans le domaine biologique ont été proposées. Elles sont présentées dans le Chapitre 1 (*cf.* Section 1.4). Certaines méthodes de stéganographie cachent des molécules portant de l'information dans un volume plus important de molécules non-porteuse d'information. Elles n'exploitent donc pas la densité de l'ADN pour cacher les données. Les quelques solutions qui proposent du chiffrement dans le domaine biologique n'ont pas été validées expérimentalement et ne prennent pas en compte un certain nombre de contraintes biologiques identifiées au Chapitre 2 (*e.g.* homopolymères, séquences de nucléotides interdites, ou encore un taux de GC équilibré) ; contraintes pour lesquelles nous avons développé le codage DSWE (*cf.* Section 2.3 du Chapitre 2).

Dans ce chapitre, nous proposons un nouveau protocole permettant de modifier la structure d'une molécule ADN encodée avec DSWE à l'aide d'opérateurs biologiques. Comme nous le verrons, certains opérateurs reposent sur des enzymes de restrictions qui reconnaissent des sites de restrictions, *i.e.* des séquences de nucléotides particulières, et permettent par exemple de sectionner ou coller des molécules entre elles. C'est en fait cette approche qui nous a poussé à développer le codage DSWE, car ces sites de restrictions ne peuvent pas servir à coder de l'information. Ce sont des séquences interdites. Plus précisément, nous proposons une solution de déchiffrement biologique, qui conditionne la récupération des données stockées dans l'ADN à l'utilisation de manipulations biologiques. Pour récupérer l'information stockée dans l'ADN, il faut d'abord déchiffrer en manipulant les molécules avec des opérateurs biologiques, avant de les séquençer et de récupérer l'information stockée.

Ce chapitre est structuré en 3 parties. Dans une première partie, nous présentons sous forme de fonctions les différents opérateurs biologiques que nous avons pu identifier, avec leurs avantages et limites. A partir de certains de ces opérateurs biologiques élémentaires, nous présentons ensuite deux fonctions plus avancées. La première est une rotation biologique, qui permet d'échanger les positions de deux blocs de données adjacents dans une molécule d'ADN. La seconde est une permutation biologique entre

deux molécules d'ADN, qui échange des blocs de données entre elles. Dans une seconde partie, nous donnons la solution de sécurité, avec la chaîne complète de stockage incluant le chiffrement numérique et le déchiffrement biologique des molécules d'ADN. Nous verrons qu'elle s'appuie sur plusieurs fonctions de rotation et de permutation différentes appliquées à des copies d'une même séquence ADN. Les molécules d'ADN obtenues sont synthétisées et mélangées dans la même solution. Pour retrouver l'information, il faut déchiffrer chaque version de la molécule d'ADN avant le séquençage, sinon l'opération de consensus après ce dernier n'est pas possible. Dans une troisième partie, nous présentons les résultats expérimentaux obtenus sur la base de simulations afin de tester la robustesse de cet algorithme de chiffrement biologique, et sur des manipulations biologiques montrant qu'une fonction de rotation sur des molécules d'ADN est possible. Cette partie se conclut par des pistes d'améliorations possibles de cette fonction biologique.

## 3.1 Définition des opérateurs biologiques

Notre idée pour sécuriser des données au niveau moléculaire est de forcer l'utilisateur à devoir modifier la structure des molécules avant de les séquencer et pouvoir accéder aux données. Partant de ce qui a été proposé pour la sécurité en numérique et ses opérateurs élémentaires, nous avons cherché à identifier des processus chimiques ou biologiques : des opérateurs biologiques pour manipuler les molécules d'ADN.

Nous avons identifié plusieurs classes d'opérateurs biologiques :

1. **Les modifications des extrémités d'une molécule d'ADN double brin** qui peuvent favoriser ou empêcher que les molécules se lient à leurs extrémités. Lorsqu'une molécule a des extrémités saillantes, c'est-à-dire une extrémité plus longue que l'autre comme présenté en Figure3.1 (a), elle ne peut se lier qu'aux extrémités saillantes avec les bases complémentaires (selon les paires A-T ou G-C). Il est aussi possible de modifier ces extrémités saillantes pour les transformer en extrémités franches, *cf.* Figure 3.1(b). Selon l'opérateur utilisée, cela revient à concaténer ou supprimer les premiers ou derniers bits d'un bloc de données.

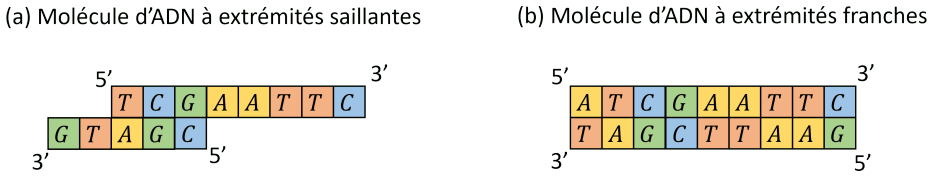



FIGURE 3.1 – Molécule d'ADN double brin. (a) Molécule avec des extrémités saillantes, *i.e.* l'extrémité d'un brin est plus longue que l'autre. (b) Molécule avec des extrémités franches.

2. **Les coupes**. Ces opérateurs séparent une molécule en plusieurs fragments double brins distincts, avec des enzymes de restriction, qui reconnaissent des séquences de nucléotides dans les molécules, et les coupent d'une façon précise. Il est donc possible d'obtenir plusieurs blocs de données à partir d'un bloc unique, en suivant des motifs précis.
3. **Les ligations** qui sont semblables à des concaténations numériques. Une ligation peut agir sur toutes molécules à extrémités franches, ou bien lier spécifiquement les molécules d'ADN à extrémités saillantes complémentaires.

4. **Les modifications de bases**. Les bases de l'alphabet $\{A, C, G, T\}$ des séquences ADN peuvent être transformées en de nouvelles bases. Ces modifications peuvent être réversibles ou non. L'opérateur numérique associé est la substitution.
5. **La fonction d'amplification de molécules d'ADN par PCR**, et les opérateurs associés.
6. **Les opérateurs de séparation de molécules spécifiques**. Il est possible d'utiliser plusieurs critères pour séparer des molécules d'ADN présentes dans une solution : la longueur, la présence de sous-séquences dans l'ADN, la présence de molécules organiques appelées biotines aux extrémités de l'ADN. Ces opérateurs permettent de mieux maîtriser le contenu de solutions. L'ADN peut aussi être séparé de tout contaminant ou impureté dans la solution, avec un opérateur de purification.

Dans ce qui suit, nous présentons ces différents opérateurs biologiques avec leur intérêt, leur fonctionnement, la fonction numérique associée ainsi que les contraintes de cette fonction dans son utilisation. Nous verrons comment plusieurs de ces opérateurs peuvent être utilisés pour définir deux fonctions avancées : la rotation qui échange la position de deux blocs de données adjacents dans une molécule d'ADN, et la permutation qui échange deux blocs de deux molécules différentes.

Pour définir ces fonctions, nous proposons d'utiliser différentes notations qui sont les suivantes. Tout d'abord, un brin double d'ADN $D$ est composé de deux brins simples complémentaires $S$ et $\overline{S}$. Le brin simple $S = (B_1 B_2 ... B_{n-1} B_n)$ est lu dans le sens 5' vers 3', de gauche à droite, où $B_i \in \{A, C, G, T\}$ est la $i^{\text{ème}}$ base ADN de $S$. Son brin complémentaire $\overline{S}$ est tel que $\overline{S} = (\overline{B_n B_{n-1}} ... \overline{B_2 B_1})$, où la base $\overline{B_i}$ est le complémentaire de la base $B_i$. Rappelons que les deux couples de bases complémentaires sont les couples Watson-Crick, *i.e.* A-T et G-C.

Un brin double d'ADN $D$ peut ainsi être noté :

$$D = \left(\frac{S}{\overline{S}}\right) = \left(\frac{B_1 \ \ldots \ B_{n-1} \ B_n}{\overline{B_1} \ \ldots \ \overline{B_{n-1}} \ \overline{B_n}}\right)$$

où : la première ligne de bases est lue de gauche à droite, et la seconde ligne de droite à gauche, de façon à toujours lire les bases d'un brin simple dans le sens 5' vers 3'.

### 3.1.1 Classe 1 - Modifications des extrémités de l'ADN

Dans cette première classe, nous pourrons distinguer 5 opérateurs.

- **Production d'extrémités franches**

  Le "Blunting", ou production d'extrémités franches, transforme les extrémités saillantes de brins doubles d'ADN en extrémités franches. Selon l'enzyme utilisé, l'extrémité la plus longue peut être supprimée, ou à l'inverse l'extrémité la plus courte peut être allongée avec les nucléotides complémentaires au brin existant. Un enzyme peut aussi effectuer les deux actions en parallèle. Les molécules à extrémités franches obtenues peuvent se lier avec toutes autres molécules à extrémités franches.

  Le protocole [160] proposé par le fournisseur de telles enzymes, New England Biolabs, présente un taux d'erreur très faible, de l'ordre de $1,8 \times 10^{-5}$ à $1 \times 10^{-6}$ selon l'enzyme utilisé parmi les 3 possibles (*cf.* le fournisseur NEB [160]). Toutefois, si les conditions d'incubation ne sont pas adaptées (température trop élevée, quantité excessive d'enzymes, temps trop long), l'enzyme peut

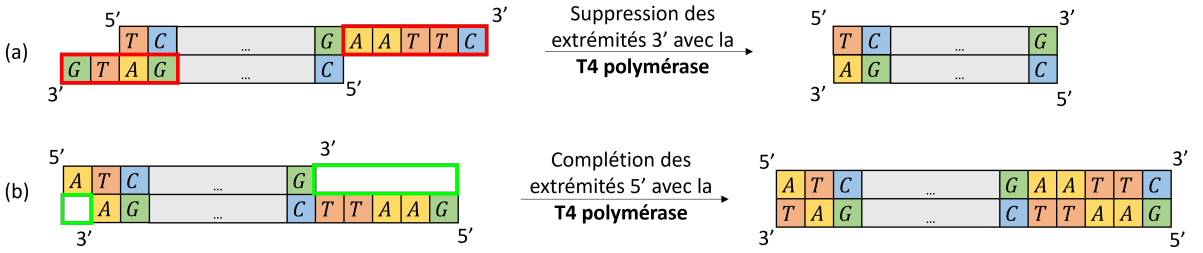



FIGURE 3.2 – Production d'extrémités franches ou "Blunting" de molécules d'ADN, via deux actions simultanées de l'enzyme T4 polymérase. (a) Action de suppression des extrémités 3' (encadrées en rouge). (b) Action de complétion des extrémités 5' (avec les bases complémentaires, dans les cadres verts).

supprimer trop de bases de l'extrémité 3'. Ce protocole peut être effectué sur des molécules jusqu'à 20000 paires de bases.

Nous présentons tout d'abord la fonction *Blunting*, qui créé des molécules d'ADN à extrémités franches à partir de toutes molécules à extrémités saillantes, avant d'introduire deux autres fonctions avec des actions plus réduites dans les sections suivantes.

La fonction *Blunting* effectuée par l'enzyme T4 polymérase a deux actions simultanées :

1. La dégradation des extrémités 3' jusqu'à obtenir des extrémités franches, comme présenté en Figure 3.2(a). Nous pouvons lui associer la fonction $SupExt3'$ définie telle que :

$$(\{D_i^{s3'}\}, E) = SupExt3'(\{D_i\}, E)$$

avec en entrée $\{D_i\}$ un ensemble de brins doubles et $E$ une enzyme T4 polymérase, et en sortie : $\{D_i^{s3'}\}$ l'ensemble de brins doubles sans les extrémités saillantes côté 3'.

2. La complétion des extrémités 5', comme illustré en Figure 3.2(b), qui peut être exprimée par la fonction $ComplExt5'$ :

$$(\{D_i^{c5'}\}, E) = ComplExt5'(\{D_i\}, E)$$

qui prend en entrée un ensemble de brins doubles $\{D_i\}$ et $E$ une enzyme T4 polymérase, et qui a en sortie l'ensemble de brins doubles $\{D_i^{c5'}\}$, mais les extrémités saillantes côté 5' ont été complétées par des bases côté 3' pour former des extrémités franches.

Nous pouvons ainsi résumer la fonction *Blunting* de la manière suivante :

$$(\{D_i^{s3'c5'}\}, E) = Blunting(\{D_i\}, E) = SupExt3'(\{D_i\}, E)\ o\ ComplExt5'(\{D_i\}, E)$$

Si $D$ a à la fois des extrémités saillantes 5' et 3', la fonction supprimera les extrémités 3' (*cf.* Figure 3.2 (a)) et complétera les extrémités 5' (*cf.* Figure 3.2(b)).

- **Complétion d'extrémités 5'**

Il est possible de faire seulement la complétion des extrémités 5' avec l'enzyme Bst DNA polymérase

[161]. La fonction associée est $BstPol$, telle que :

$$(\{D_i^{c5'}\}, E) = BstPol(\{D_i\}, E)$$

qui prend en entrée un ensemble de brins doubles $\{D_i\}$ et $E$ une enzyme Bst DNA polymérase, et produit les brins $\{D_i^{c5'}\}$ dont les extrémités saillantes côté 5' ont été complétées pour former des extrémités franches.

- **Suppression d'extrémités 5'**

  La Mung Bean Nuclease [162] est un enzyme qui supprime les extrémités saillantes côté 5' de molécules d'ADN. La fonction associée est $SupExt5'$, telle que :

  $$(\{D_i^{s5'}\}, E) = SupExt5'(\{D_i\}, E)$$

  avec : $\{D_i\}$ l'ensemble de brins doubles de la solution à traiter et $E$ la Mung Bean Nuclease, et $\{D_i^{s5'}\}$ sans les extrémités saillantes côté 5'.

  Cette digestion enzymatique se fait en 30 minutes d'incubation à 30°C, en suivant le protocole de NEB.

- **Ajout d'une base A**

  Comme illustré en figure 3.3, le "A-tailing" [65] est l'ajout d'une base A à l'extrémité 3' d'ADN double brin à extrémités franches, à l'aide d'un enzyme (la Taq DNA polymérase). Cet opérateur est utilisé avant le séquençage nanopore pour lier les barcodes à l'ADN, comme nous l'avons présenté en Section 1.1.5 du Chapitre 1.

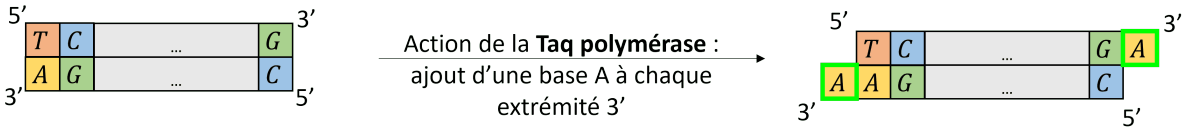



FIGURE 3.3 – A-tailing, ou ajout d'une base A aux extrémités 3' d'un brin double d'ADN à extrémités franches.

L'étape de A-tailing peut aussi servir pour préparer de l'ADN double brin avant de l'insérer dans une molécule d'ADN circulaire avec des extrémités saillantes 3' T, dite "T-Vector".

La fonction associée est telle que :

$$(\{D_i^{ATail3'}\}, E) = ATailing3'(\{D_i\}, E)$$

avec $\{D_i\}$ des brins doubles avec des extrémités franches et $E$ la Taq DNA polymérase, et $\{D_i^{ATail3'}\}$ les mêmes brins dont les extrémités côté 3' ont été complétées par un A.

- **Phosphorylation**

  Deux molécules d'ADN peuvent se lier entre elles si et seulement si au moins une de leurs extrémités 5' ont un groupe phosphate. La fonction de phosphorylation permet donc de préparer l'ADN à une ligation, tandis que la fonction de déphosphorylation empêche toute ligation.

Ainsi, la phosphorylation, présentée en Figure 3.4, est effectuée grâce à l'action de l'enzyme T4 PNK, qui ajoute un groupement phosphate à chaque extrémité 5' d'une molécule d'ADN. Cela est utile car certaines opérations peuvent endommager l'ADN et le déphosphoryler, ce qui pourrait empêcher des ligations avec cet ADN. Notons que déphosphoryler (*cf.* Figure 3.5 ) permet aussi d'empêcher une molécule d'ADN de se refermer sur elle-même si ses deux extrémités sont complémentaires. La déphosphorylation est effectuée à l'aide d'un enzyme phosphatase qui décompose les groupes phosphate.

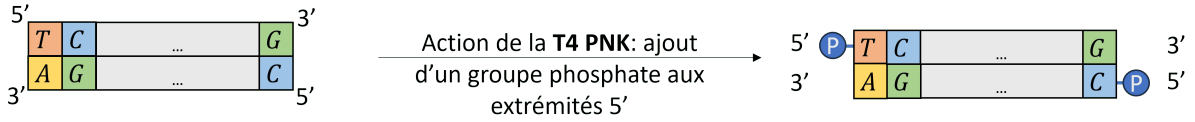



FIGURE 3.4 – Phosphorylation grâce à l'action de l'enzyme T4 PNK. Un groupe phosphate représenté dans un rond bleu est ajouté aux extrémités 5' de l'ADN. Ces extrémités peuvent alors se lier à d'autres extrémités de molécules d'ADN.

Sur cette base nous pouvons définir la fonction *Phos*, qui prépare l'ADN à une ligation, avec les paramètres suivants :

$$(\{D_i^{phos}\}, \{S_j^{phos}\}, E) = Phos(\{D_i\}, \{S_j\}, E)$$

avec $\{D_i\}$ des brins doubles et $\{S_j\}$ des brins simples et $E$ l'enzyme T4 PNK, et $\{D_i^{phos}\}$ et $\{S_j^{phos}\}$ les mêmes brins avec les extrémités 5' phosphorylées.

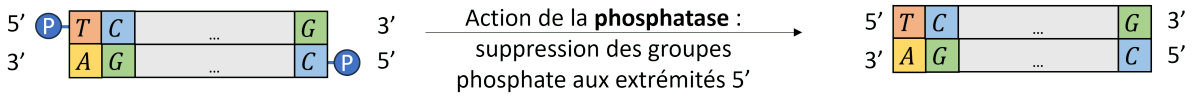



FIGURE 3.5 – Déphosphorylation grâce à l'action de l'enzyme phosphatase. Les groupes phosphate aux extrémités 5' de l'ADN sont supprimés.

La fonction *Dephos*, qui empêche toute ligation, a les paramètres suivants :

$$(\{D_i^{dephos}\}, \{S_j^{dephos}\}, E) = Dephos(\{D_i\}, \{S_j\}, E)$$

avec $\{D_i\}$ des brins doubles et $\{S_j\}$ des brin simples et $E$ l'enzyme phosphatase, et $\{D_i^{dephos}\}$ et $\{S_j^{dephos}\}$ les mêmes brins avec les extrémités 5' déphosphorylées.

Les opérateurs de phosphorylation et déphosphorylation reposent sur des réactions biochimiques simples et fonctionnent très bien lorsque les conditions de réaction (solution tampon, température, temps de réaction, etc.) sont optimisées.

### 3.1.2 Classe 2 - Coupes d'ADN

Nous avons identifié deux types d'opérateurs : ils coupent des brins doubles d'ADN en plusieurs fragments double brin distincts, ou bien entaillent un des brins simples d'une molécule double brin.

- **Coupe de brins doubles**

  Une molécule peut être coupée en plusieurs fragments double brin distincts à l'aide d'enzymes de restriction. Nous nous intéressons en particulier aux enzymes de restriction de type II. Un tel enzyme $E$ identifie sur un double brin d'ADN une séquence de quelques bases (généralement 4 à 10), appelée site de restriction, et coupe le brin double selon un motif précis créant ainsi plusieurs brins doubles distincts. Cet opérateur permet de fragmenter de l'ADN double brin et pourra, comme nous le verrons, contribuer à modifier la structure de blocs de données dans une molécule d'ADN.

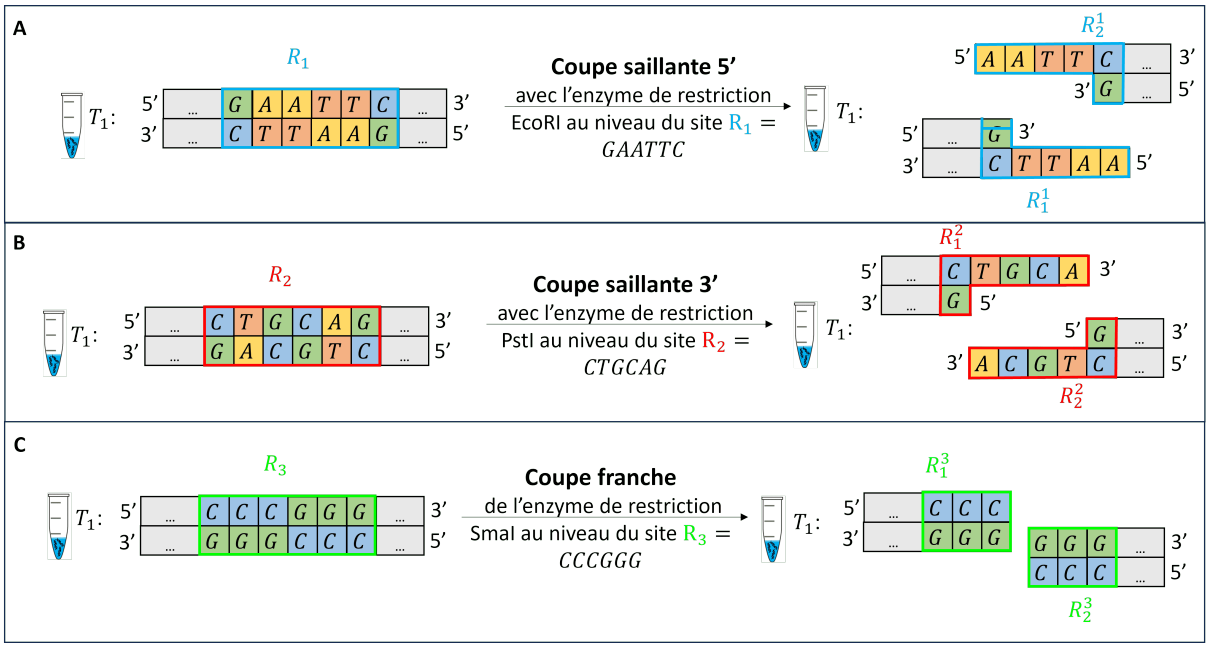



FIGURE 3.6 – Enzymes de restriction coupant leurs sites respectifs pour créer des extrémités saillantes (a et b) ou franche (c).

Il existe des centaines d'enzymes différents, associés à des sites de restrictions et des coupes spécifiques. Un enzyme peut couper l'ADN de deux façons en créant :

— des extrémités saillantes, *cf* Figure 3.6A et B, avec un brin plus long que l'autre. Les extrémités saillantes créées sont dites compatibles ou encore cohésives : elles peuvent se lier sous l'action d'une enzyme de ligation, comme nous le verrons dans la prochaine section. Le brin le plus long peut être côté 5' (*cf* Figure 3.6A) ou 3' (*cf* Figure 3.6B). La fonction associée à la coupe créant des extrémités saillantes côté 5' est telle que :

$$(\{D_l^{R_k^1,5'}\},\{D_m^{R_k^2,5'}\},\{S_j\},\{E_k\}) = CutS5'(\{D_i\},\{S_j\},\{E_k\})$$

avec $\{D_i\}$ des brins doubles, $\{S_j\}$ des brins simples et $\{E_k\}$ un ensemble d'enzymes de restriction, et $(\{D_l^{R_k^1,5'}\},\{D_m^{R_k^2,5'}\})$ des brins doubles avec au moins une extrémité saillante 5' résultant de la coupe du site de restriction $\{R_k\}$.

La fonction associée à la coupe créant un brin plus long côté 3' est définie similairement, telle

que :

$$(\{D_l^{R_k^1,3'}\},\{D_m^{R_k^2,3'}\},\{S_j\},\{E_k\}) = CutS3'(\{D_i\},\{S_j\},\{E_k\})$$

avec $\{D_i\}$ des brins doubles, $\{S_j\}$ des brins simples et $\{E_k\}$ un ensemble d'enzymes de restriction, et $(\{D_l^{R_k^1,3'}\},\{D_m^{R_k^2,3'}\})$ des brins doubles avec au moins une extrémité saillante 3' résultant de la coupe du site de restriction $\{R_k\}$.

— des extrémités franches comme présenté en Figure 3.6C. La fonction associée est définie telle que :

$$(\{D_l^{R_k^1,F}\},\{D_m^{R_k^2,F}\},\{S_j\},\{E_k\}) = CutF(\{D_i\},\{S_j\},\{E_k\})$$

avec $\{D_i\}$ des brins doubles, $\{S_j\}$ des brins simples et $\{E_k\}$ un ensemble d'enzymes de restriction, et $(\{D_l^{R_k^1,F}\},\{D_m^{R_k^2,F}\})$ des brins doubles avec au moins une extrémité franche résultant de la coupe du site de restriction $\{R_k\}$.

Le protocole de digestion enzymatique, c'est-à-dire de réaction de l'enzyme choisi avec les molécules d'ADN, dépend de plusieurs facteurs. Un premier est la durée, généralement fixée à une heure pour maximiser le nombre de molécules rencontrées et coupées par l'enzyme. Si ce temps peut être plus long, l'enzyme peut faire des coupes non spécifiques, c'est-à-dire couper l'ADN à des positions non prévues. Certains enzymes sont optimisés et certifiés High-Fidelity (HF), ils garantissent un très faible taux de coupes non spécifiques et agissent en peu de temps, nécessitant seulement 5 à 15 minutes.

Selon l'outil Enzyme Finder de NEB (New England Biolabs), 667 enzymes de restriction sont actuellement disponibles, dont 362 créant des extrémités saillantes 5', 173 créant des extrémités 3', et 127 créant des extrémités franches. Ces chiffres sont susceptibles d'évoluer dans le temps, avec la découverte de nouveaux enzymes. Bien entendu, ces enzymes ont des propriétés et des comportements différents. Un enzyme peut par exemple identifier plusieurs sites de restriction différents. L'enzyme BsteII est associé aux sites $GGTAACC, GGTCACC, GGTGACC, GGTTACC$, où on peut voir que toutes les bases sont identiques sauf la 4ème base qui peut être un $A$, un $C$, un $G$ ou un $T$. On note une telle base $N$. Pour cette raison, le site de restriction associé à l'enzyme BsteII est noté ($GGTNACC$) et est dit dégénéré. Il est aussi possible que plusieurs enzymes reconnaissent le même site de restriction. Ils sont appelés isoschizomères s'ils coupent l'ADN de la même façon, ou néoschizomères s'ils coupent l'ADN de façons différentes. Par exemple, SmaI et XmaI reconnaissent tous deux le site ($CCCGGG$), mais le premier enzyme crée des extrémités franches, tandis que le deuxième crée des extrémités saillantes côté 5' ($CCGGG$). D'autre part, certains enzymes (dits de type IIS) coupent le double brin à côté du site de restriction qu'ils reconnaissant. Par exemple, l'enzyme BsaI est présenté en Figure 3.7, il créé des extrémités saillantes en coupant le brin double à une distance de 1 à 5 bases du site de restriction reconnu. Enfin, un type d'enzyme que nous utiliserons pour nos fonctions de rotation et de permutation est l'isocaudomère. Deux enzymes sont dits isocaudomères s'ils reconnaissent des sites légèrement différents l'un de l'autre mais produisent les mêmes extrémités saillantes.

Notons que quelques enzymes peuvent couper des brins simples d'ADN. Toutefois, cette coupe s'ef-

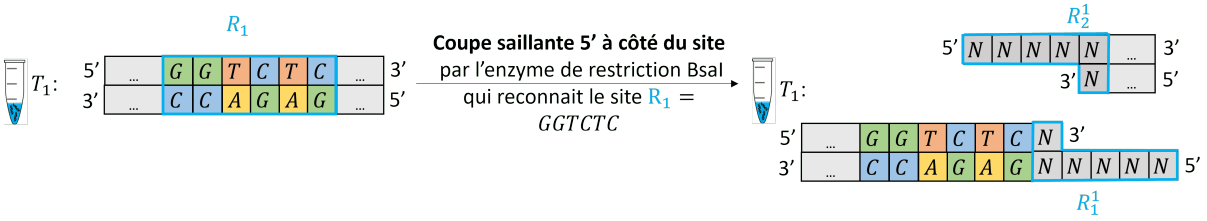



FIGURE 3.7 – Coupe de l'enzyme de restriction BsaI, qui reconnaît le site ($GGTCTC$) et créé des extrémités saillantes 5' en dehors de ce site. La notation $N$ correspond à n'importe quel nucléotide parmi $\{A, C, G, T\}$.

fectue probablement lorsque le brin simple se replie sur lui-même car il contient des sous-séquences complémentaires, formant alors un brin double. A ce jour, 17 enzymes coupant de l'ADN brin simple sont reconnus, la liste est disponible sur la base de données des enzymes de restrictions REBASE de NEB.

- **Entailles**

  Quelques enzymes permettent de créer des entailles dans de l'ADN double brin, comme illustré en Figure 3.8. Ils ne coupent qu'un seul des deux brins simples, contrairement aux enzymes de restriction qui coupent les deux brins. Plus précisément, ils coupent les liaisons entre deux nucléotides d'un même brin, au niveau ou à côté du site de restriction spécifique à l'enzyme. Ils sont appelés "nicking enzymes", ou endonucléases de césure. 15 sont connus à ce jour [138].

  Un tel enzyme peut être utilisé pour retirer quelques nucléotides d'une molécule d'ADN double brin et les remplacer par leur équivalent fluorescent [163], par exemple. Ainsi, certaines régions de l'ADN peuvent être observées via leur fluorescence sans avoir besoin de séquencer.


FIGURE 3.8 – Création d'une entaille par l'enzyme Nt.BbvCI dans le site de restriction ($CCTCAGC$).

Par exemple, l'enzyme Nt.BbvCI présenté à la Figure 3.8 reconnaît le site ($CCTCAGC$) et coupe le brin contenant ce site entre le 'C' et le 'T'.

La fonction d'entaille est telle que

$$(\{D_i^{nick_{R_j}}\}, \{E_j\}) = Nick(\{D_i\}, \{E_j\})$$

avec $\{D_i\}$ des brins doubles et $\{E_j\}$ des enzymes créant des entailles, et $\{D_i^{nick_{R_j}}\}$ des brins doubles entaillés aux sites $\{R_j\}$ correspondant aux enzymes $\{E_j\}$.

### 3.1.3 Classe 3 - Ligations

Cette classe d'opérateurs utilise des enzymes d'ADN ligase pour lier les extrémités de doubles brins d'ADN. La liaison se fait entre le groupe phosphate d'un double brin d'ADN et le sucre de l'autre. On

distingue 3 opérateurs qui lient des types de molécules différents.

- **Ligation d'extrémités saillantes compatibles**

  L'enzyme T7 DNA ligase fait uniquement la ligation de brins doubles d'ADN à extrémités saillantes compatibles. La fonction associée, illustrée en Figure 3.9, est telle que :

  $$(\{S_k\}, \{D_j\}_{j \leq i}, E) = LigS(\{S_k\}, \{D_i\}, E)$$

  avec : $S_k$ des brins simples et $D_i$ des brins doubles, et $E$ un enzyme de ligation, et $D_j$ un ensemble de brins doubles contenant les brins $D_i$ liés lorsque leurs extrémités saillantes sont compatibles, d'où : $j \leq i$.


FIGURE 3.9 – Ligation de fragments d'ADN double brin à extrémités saillantes complémentaires

  D'après le fournisseur NEB, la T7 DNA Ligase peut lier des fragments d'ADN lorsque leurs extrémités saillantes dépassent de 2 bases ou plus. La ligation d'extrémités saillantes d'une unique base fonctionne très mal. La longueur optimale est de 4 bases.

- **Ligation d'extrémités saillantes compatibles et franches**

  L'enzyme T4 fait la ligation de tous types de fragments d'ADN double brin, que leurs extrémités soient franches ou saillantes. Les extrémités saillantes doivent être compatibles pour être liées. La fonction associée est telle que :

  $$(\{S_k\}, \{D_j\}_{j \leq i}, E) = Lig(\{S_k\}, \{D_i\}, E)$$

  avec $S_k$ des brins simples, $E$ un enzyme de ligation et $D_i$ des brins doubles, et $D_j$ les brins doubles à extrémités saillantes complémentaires liées, ainsi que certains brins doubles à extrémités franches, d'où : $j \leq i$.

  Cette ligation présente cependant plusieurs biais. Plus les molécules sont de petites longueurs, plus elles ont de chance de se rencontrer et donc de se lier. La longueur minimale des molécules est de 8 paires de bases [164]. D'autre part, les extrémités saillantes sont plus facilement liées que les extrémités franches. Enfin, les extrémités saillantes avec un taux de GC plus important se lient plus efficacement [165]. Notons que plus il y a de molécules différentes à lier, moins le rendement est bon. Le taux de réussite de l'assemblage de petits fragments en une longue molécule devient très mauvais si il y a plus de 5-10 fragments à lier.

- **Ligation d'entailles**

  Un double brin peut avoir une entaille ou césure, *i.e.* une absence de liaison entre deux nucléotides adjacents sur un de ses brins simples. Les liaisons peuvent se défaire naturellement, ou sous l'action


FIGURE 3.10 – Ligation par l'enzyme T4 d'ADN double brin. Cette enzyme peut lier des extrémités franches ainsi que des extrémités saillantes compatibles.

d'enzymes (cf. Section 3.1.2). Cette entaille est problématique car, si le double brin est dénaturé (*i.e.* séparé en deux brins simples), alors le brin simple contenant une entaille se séparera en plusieurs fragments.

Un enzyme ADN ligase peut généralement réparer de telles entailles. Pour cela, il parcourt un double brin et ajoute les éventuelles liaisons manquantes entre deux bases ADN du même brin simple. Nous pouvons lui associer la fonction $NickRepair(D)$ qui parcourt le double brin $D$. Si elle repère une absence de liaison, représentée par $||$, elle la supprime. Dans l'exemple suivant, l'ADN ligase identifie cette entaille et lie $B_j$ et $B_{j+1}$ ensemble.

$$\left(\frac{B_1}{B_1}\frac{\dots}{\dots}\frac{B_j}{B_j}\frac{B_{j+1}}{B_{j+1}}\frac{B_{j+2}}{B_{j+2}}\frac{\dots}{\dots}\frac{B_n}{B_n}\right) = NickRepair(\left(\frac{B_1}{B_1}\frac{\dots}{\dots}\frac{B_j|}{B_j}\frac{|B_{j+1}}{B_{j+1}}\frac{B_{j+2}}{B_{j+2}}\frac{\dots}{\dots}\frac{B_n}{B_n}\right))$$

La fonction NickRepair est donc telle que :

$$(\{S_k\}, \{D_i^{NoNick}\}, E) = NickRepair(\{S_k\}, \{D_i\}, E)$$

avec $S_k$ des brins simples, $E$ un enzyme de ligation et $D_i$ des brins doubles, et $\{D_i^{NoNick}\}$ les brins doubles sans entailles.


FIGURE 3.11 – Réparation d'entailles

Un enzyme qui effectue la réparation d'entailles est l'enzyme T4 ligase. Cet enzyme a deux actions :

- *NickRepair* : la réparation d'entailles dans des brins doubles d'ADN.
- *Lig* : la ligation de brins doubles à extrémités saillantes compatibles et de brins doubles à extrémités franches, présentée au paragraphe précédent.

Ainsi, l'enzyme T4 ligase est associée à la fonction suivante :

$$(\{S_k\}, \{D_j^{NoNick}\}_{j\leq i}, E) = T4Ligase(\{S_k\}, \{D_i\}, E) = NickRepair(\{S_k\}, \{D_i\}, E)\ o\ Lig(\{S_k\}, \{D_i\}, E)$$

avec $S_k$ des brins simples, $E$ l'enzyme T4 DNA ligase, $D_i$ des brins doubles, et $\{D_j^{NoNick}\}_{j \leq i}$ les brins doubles sans entailles, obtenus par ligations des brins doubles à extrémités saillantes complémentaires liées, ainsi que par ligations de brins doubles à extrémités franches, d'où : $j \leq i$.

### 3.1.4 Classe 4 - Modifications de bases

Il est possible de modifier les nucléotides A, C, G et T et d'augmenter la taille de l'alphabet de bases composant des séquences ADN, et d'agir sur le fonctionnement d'autres opérateurs qui ne fonctionnent que lorsque des bases sont modifiées. Par exemple, certaines fonctions de coupe (présentées en Section 3.1.2) nécessitent la présence de bases ADN modifiées dans une molécule.

Nous avons identifié deux types de modifications des bases ADN.

- **Méthylation et hydroxyméthylation**

  La méthylation [166] est l'ajout d'un groupe méthyle à certaines bases de l'ADN, qui va modifier la façon dont ces bases sont lues lors du séquençage. Un tel processus fait partie du domaine émergent de l'épigénétique. Les travaux de recherche sur la méthylation sont encore en évolution, notamment pour améliorer l'efficacité des processus, et découvrir de nouvelles méthylations.

  Parmi les 4 bases de l'ADN, seules les bases 'C' et 'A' peuvent être méthylées. Cette opération peut-être considérée comme une substitution de bases. En particulier, une base 'C' peut être méthylée ou hydroxyméthylée [167], devenant respectivement '5-mC' ou '5-hmC' comme présenté en Figure 3.12A. Une base 'A' peut seulement être méthylée, devenant une base '6-mA' (*cf.* Figure 3.12 B). Ce procédé peut être utile car certains enzymes y sont sensibles. Par exemple, les enzymes de restriction peuvent reconnaître un site de restriction seulement si les bases 'C' sont méthylées, ou au contraire seulement si elles ne le sont pas. C'est le cas de l'enzyme DpnI qui reconnaît le site GATC si et seulement si il y a une méthylation sur la base 'A' de chaque brin.

  Il est possible d'obtenir une base 'T' comme présenté en Figure 3.12C en méthylant une base uracile 'U' ; base qui ne fait pas partie de l'alphabet originel de l'ADN. Cette base 'U' peut être obtenue par une opération sur une base 'C', la désamination, que nous présentons dans le paragraphe suivant.

  Une méthylation peut être effectuée via des enzymes, les DNA methyltransferase ("DNMT"), ou des réactions chimiques spécifiques à chaque modification.

  Il est possible d'ajouter des bases déjà méthylées à une séquence d'ADN lors de la synthèse, ou bien de méthyler une molécule d'ADN déjà formée. On peut supprimer toute méthylation d'une séquence via réaction chimique, ou bien déméthyler chaque type de méthylation avec un enzyme particulier. L'hydroxyméthylation se fait grâce à l'intervention de protéines [166].

  Nous pouvons ainsi définir les fonctions de méthylation et déméthylation suivantes :

  - $MethylC$ et son inverse $DeMethylC$ : Modification des bases 'C' en bases 'mC', et inversement
  - $HydroxymC$ : Modification des bases 'mC' en 'hmC'
  - $HmethylC$ : Modification des bases 'C' en bases 'hmC'
  - $MethylA$ et son inverse $DeMethylA$ : Modification des bases 'A' en 'mA' et inversement
  - $MethylU$ : Modification des bases 'U' en 'T'

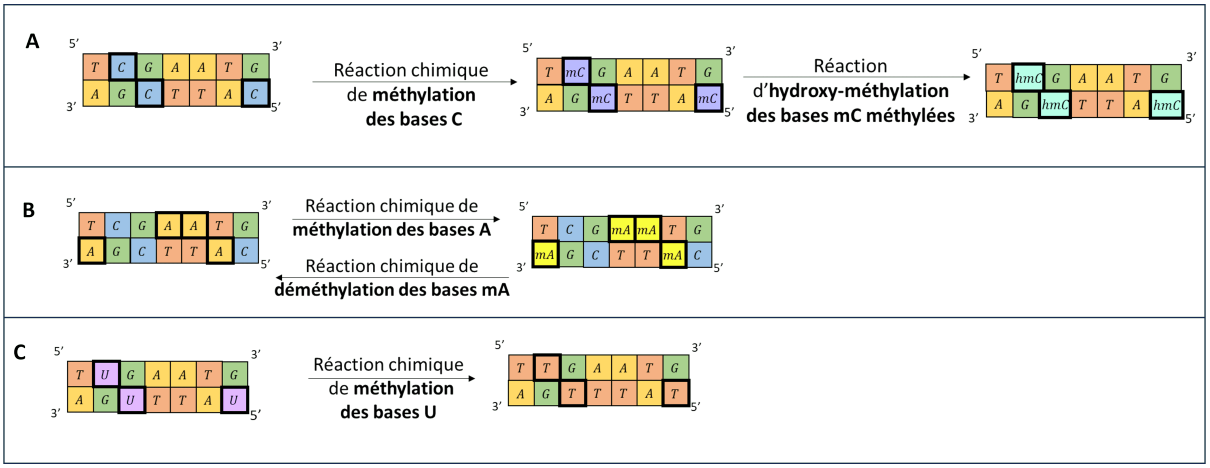



FIGURE 3.12 – Modifications de bases de l'ADN par méthylation et hydroxyméthylation. A) Méthylation et hydroxyméthylation des bases 'C'. B) Méthylation et déméthylation des bases 'A' et 'mA', respectivement. C) Méthylation des bases 'U'.

- *DeMethyl* : Déméthylation chimique des bases méthylées 'mA' et 'mC' : modification en bases 'A' et 'C', respectivement.

Il est important de noter que ces fonctions, lorsqu'elles sont appliquées, agissent sur toutes les molécules de la solution. Également, ces modifications ne résistent pas à l'amplification par PCR : les copies d'une molécule méthylée ne seront pas méthylées.

- **Désamination**

  La réaction chimique irréversible de désamination permet de modifier certaines bases. En particulier, il est possible de désaminer les bases 'C' et 'mC', les transformant alors respectivement en bases 'U' et 'T', comme représenté en Figure 3.13A et B.

  La fonction *Desamin* a donc deux actions :

  - La transformation des bases 'C' en bases 'U', exprimée par la fonction $CtoU$, définie telle que :

$$(\{D_i^{CtoU}\}, React) = CtoU(\{D_i\}, React)$$

    avec $React$ un réactif chimique, $\{D_i\}$ des brins doubles d'ADN et $\{D_i^{CtoU}\}$ les mêmes brins doubles avec les bases 'C' transformées en bases 'U'.

  - La transformation des bases 'mC' en bases 'T', correspondant à la fonction suivante :

$$(\{D_i^{mCtoT}\}, React) = mCtoT(\{D_i\}, React)$$

    avec $React$ un réactif chimique, $\{D_i\}$ des brins doubles d'ADN et $\{D_i^{mCtoT}\}$ les mêmes brins doubles avec les bases 'mC' transformées en bases 'T'.

  On peut ainsi résumer la fonction *Desamin* de la manière suivante :

$$\{D_i^{CtoU,mCtoT}\} = Desamin(\{D_i\}) = CtoU(\{D_i\})\ o\ mCtoT(\{D_i\})$$

Une fonction $Desamin^{CtoU}$ peut être cependant effectuée séparément par un traitement au disulfite de sodium permettant de désaminer uniquement les bases 'C' en bases 'U', sans modifier les bases 'mC' en 'T'.

La désamination est irréversible, il n'y a pas de procédé permettant de récupérer les bases 'C' et 'mC' à partir des bases 'U' et 'T'. Toutefois, comme nous l'avons vu dans la section précédente, l'opération de méthylation permet d'obtenir un 'T' à partir d'un 'U'.

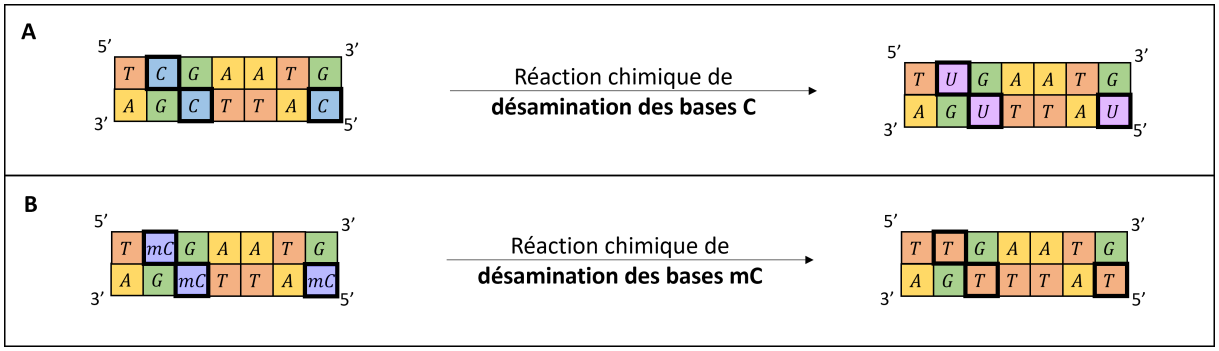



FIGURE 3.13 – Réaction chimique de désamination. A) Désamination des bases 'C', formant de nouvelles bases 'U'. B) Désamination des bases 'mC' pour obtenir des bases 'T'.

### 3.1.5 Classe 5 - Amplification

L'amplification est la copie d'une ou plusieurs molécules d'ADN. Pour ce faire, la PCR ("polymerase chain reaction") effectue des cycles de 3 opérations sur l'ADN : la dénaturation, l'hybridation et l'élongation. Nous proposons d'y associer des opérateurs et de définir un opérateur d'amplification pour la PCR dans sa globalité.

- **Dénaturation**

  La dénaturation, présentée en Figure 3.14, sépare un double brin en simples brins via une augmentation de température au delà d'un certain seuil (94 à 98°C). La température impacte les liaisons faibles (liaisons hydrogènes) qui se défont. Choisir le bonne température dépend de plusieurs facteurs, comme le taux de GC, la taille des molécules, etc.

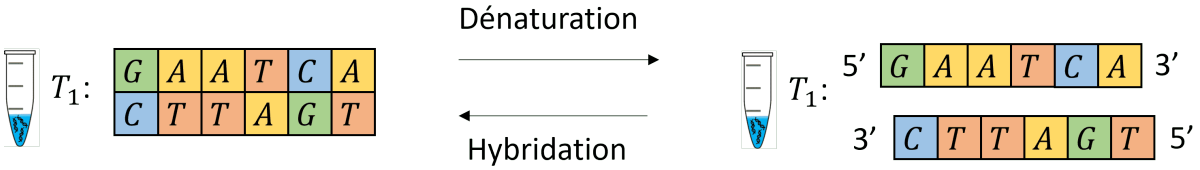



FIGURE 3.14 – Dénaturation de brins doubles d'ADN en brins simples, et son opération inverse d'hybridation.

La fonction associée est *Denat*, définie telle que :

$$(\{S_i, \overline{S_i}\}, \{S_j\}) = Denat(\{D_i\}, \{S_j\})$$

avec $\{D_i\}$ des brins doubles et $\{S_j\}$ des brins simples, et $\{S_i, \overline{S_i}\}$ les brins simples résultant de la dénaturation des brins doubles $\{D_i\}$.

La dénaturation a un rendement quasiment parfait car les liaisons hydrogènes sont fragiles et faciles à défaire. Toutefois, les brins séparés peuvent s'hybrider à nouveau si la température est baissée dans un certain intervalle de température. Pour minimiser les renaturations, il est aussi possible de plonger les tubes à essai contenant les molécules dénaturées dans la glace, pour inhiber leur mobilité.

- **Hybridation**

  L'hybridation permet de lier les brins simples complémentaires. Baisser la température au delà d'un certain seuil (45-60°C) favorise la liaison de brins simples complémentaires. Cet opérateur, présenté en Figure 3.14, est l'inverse de la dénaturation. Il est utile car certaines fonctions biologiques ne fonctionnent que sur des brins doubles et pas sur des brins simples, par exemple les fonctions de coupe (*cf.* Section 3.1.2).

  La fonction d'hybridation $Hy$ peut être définie telle que :

$$(\{D_i\}, \{D_j\}, \{S_k\}) = Hy(\{D_i\}, \{S_j\}, \{\overline{S_j}\}, \{S_k\})$$

  avec : $\{D_i\}$ des brins doubles et $\{S_j\}, \{\overline{S_j}\}, \{S_k\}$ des brins simples, et $\{D_j\}$ les brins doubles résultant de l'hybridation des brins simples complémentaires $\{S_j\}$ et $\{\overline{S_j}\}$.

  Il y plusieurs contraintes à prendre en compte avec l'hybridation. Si plusieurs brins simples ont des régions similaires mais non identiques, ils peuvent s'hybrider quand même et créer une forme alternative d'ADN. L'hybridation est aussi plus sensible que la dénaturation. Pour que deux brins simples complémentaires s'hybrident bien, les températures et temps de réaction doivent être définis précisément.

- **PCR**

  La PCR ("polymerase chain reaction", ou réaction en chaîne par polymérase) est une technique permettant de multiplier exponentiellement un double brin d'ADN si l'on connaît les amorces qui l'encadrent. Elle repose sur une répétition de 20 à 35 cycles de baisses et d'augmentations de température. Un cycle consiste en 3 étapes : la dénaturation, l'hybridation et l'élongation. La dénaturation permet de séparer le brin double d'ADN en brins simples. Ensuite, l'étape d'hybridation consiste à introduire des amorces et à baisser la température pour que les amorces s'hybrident aux extrémités des brins simples qui leur sont complémentaires. Enfin, lors de l'élongation, des enzymes ADN polymérase vont compléter les brins doubles composés d'un brin simple et d'une amorce, pour obtenir un brin double avec deux brins simples de même taille. Ici, des nucléotides sont ajoutés à la solution pour compléter les brins doubles.

  Sur cette base nous définissons la fonction de PCR :

$$(\{D_i\}, \{D\}, For_i, Rev_i) = PCR(\{D\}, For_i, Rev_i)$$

  avec $\{D\}$ des brins doubles d'ADN et $(For_i, Rev_i)$ le couple d'amorces encadrant la ou les séquences à amplifier, et $D_i$ les brins doubles amplifiés par la PCR.

Après $n$ cycles de PCR, le nombre de copies en sortie de la PCR peut être estimé de la façon suivante :

$$C(n) = C_0 \times (1 + E)^n$$

avec $C_0$ le nombre initial de copies de la séquence cible $\{D_i\}$, $E$ l'efficacité de la PCR (comprise entre 0 et 1), $n$ le nombre de cycles de PCR, et $C(n)$ le nombre final de copies de $\{D_i^U\}$ après $n$ cycles. L'efficacité d'une PCR est généralement haute, on obtient environ $2^n$ copies de la séquence ADN visée après $n$ cycles.

Un point fort de la PCR est le fait qu'elle introduit très peu d'erreurs. Toutefois celles-ci ne sont pas à négliger, notamment si l'on fait un grand nombre de cycles, ou plusieurs PCR consécutives. Le taux d'erreur de la PCR dépend de l'enzyme utilisé [62] : l'ADN polymérase Q5 présente un taux de $5,3 \times 10^{-7}$ erreurs par base par cycle de PCR, selon le fournisseur NEB. Il est possible d'estimer le pourcentages de copies correctes obtenues par PCR selon l'enzyme choisi, le nombre de cycles et la taille de la région d'intérêt, avec l'outil "PCR Fidelity Simulator" de NEB.

La PCR amplifie généralement mieux les molécules d'ADN de petite taille [168]. Pour limiter ce phénomène, elle doit être optimisée selon la taille des molécules : chaque cycle dure environ 100 secondes pour des molécules de 5000 paires de bases et 10 à 20 secondes pour des molécules de 500 paires de bases. Ainsi, plus les molécules sont longues plus il faut du temps pour qu'elles s'hybrident, se dénaturent, ou soient complétées.

Des contraintes sont aussi prendre en compte sur la structure des amorces utilisées. D'après le fabricant d'ADN synthétique Thermofisher [169], les amorces doivent faire 18 à 30 bases pour qu'elles s'hybrident efficacement aux molécules visées et qu'elles puissent se différencier les unes des autres. Les homopolymères doivent être limités à une certaine taille (3 pour ce fournisseur). Le taux de GC doit être compris entre 40 et 60%. Enfin, l'extrémité 3' d'une amorce doit finir en un G ou un C. Respecter ces contraintes permet d'optimiser le rendement de la PCR.

Il existe des variantes à la PCR. Sans être exhaustif, en voici quelques unes :

— Une PCR avec **une seule amorce** est possible. Elle entraîne alors une amplification linéaire et non exponentielle.

— Une PCR peut être effectuée avec des amorces couplées à des biotines, dites **amorces biotinylées**. Cela permet d'ajouter des biotines aux extrémités 5' de l'ADN créé par PCR.

— La **Multiplex PCR** est l'amplification simultanée dans un seul tube à essai en une réaction de plusieurs molécules d'ADN $D_i$, avec un couple différent d'amorces $For_i$ et reverse $Rev_i$ pour chaque cible. Le choix des amorces est ici crucial car les températures d'hybridation et les durées d'élongation optimales des PCR en dépendent. Les amorces doivent aussi être hautement spécifiques, c'est-à-dire que deux amorces ne peuvent pas être quasi-identiques. Par exemple, les auteurs de [22] choisissent des amorces de 20 bases avec au minimum 6 bases différentes de toute autre amorce. Pour optimiser cette PCR, la taille des molécules d'ADN à amplifier ne doit pas varier non plus.

### 3.1.6 Classe 6 - Séparation de molécules spécifiques

Il est possible de récupérer des molécules d'ADN spécifiques. Les opérateurs que nous avons identifiés permettent d'isoler les molécules selon leur longueur ou la présence ou non de sous-séquences.

- **Capture par amorces biotinylées et PCR biotinylée**

  Cette méthode permet de séparer les molécules selon leurs extrémités. Elle consiste à ajouter des biotines aux extrémités des molécules puis à capturer ces molécules biotinylées à l'aide de sondes.

  L'ADN n'est pas naturellement biotinylé. Pour ajouter des biotines à une ou deux extrémités 5', une PCR avec des amorces elle-mêmes biotinylées est effectuée. Ensuite, ces molécules sont récupérées grâce à des sondes, qui sont des billes magnétiques couplées à de la streptavidine. Cette dernière forme une liaison très forte avec la biotine de chaque molécule d'ADN. Comme la streptavidine est liée à des billes magnétiques, il est possible de récupérer les molécules d'ADN avec des aimants. La Figure 3.15 représente schématiquement les deux étapes de séparation avec amorces biotinylées. La PCR avec des amorces biotinylées, en Figure 3.15A, permet d'ajouter des biotines (ronds jaunes) aux amorces $For$ et $Rev$. La deuxième étape est la capture des molécules ADN biotinylées grâce à la liaison entre biotine (en jaune) et streptavidine (en noir) couplée à une bille magnétique. Dans la solution, les molécules d'ADN peuvent se trouver dans deux états, appelés phases, comme présenté en Figure 3.15B. La phase mobile est l'ensemble des molécules non liées aux billes, qui vont donc rester en solution. La phase stationnaire est l'ensemble des molécules immobilisées par des billes, et attirées contre la paroi du tube à essai par des aimants. Ainsi, la phase stationnaire est immobilisée à la paroi du tube à essai, tandis que la phase mobile est retirée de la solution par pipetage. De cette façon, les molécules biotinylées sont séparées des molécules non biotinylées. Elles sont ensuite mises en solution dans deux tubes à essai distincts.

  Nous proposons d'associer à ce procédé la fonction *SepBiotin*. Elle a pour but de séparer toutes les molécules d'ADN avec une extrémité biotinylée du reste de l'ADN présent dans un tube à essai. Elle est définie telle que :

$$(\{D_j^{biot}\}_{T_1}, \{D_k\}_{T_2}) = SepBiotin(\{D_i\}, For_j^{biot}, Rev_j^{biot})$$

  avec $\{D_i\}$ des brins doubles d'ADN et $(For_{biot}^j, Rev_{biot}^j)$ des couple d'amorces biotinylées, et $\{D_j^{biot}\}_{T_1}$ les brins doubles biotinylés isolés dans le tube $T_1$ et $\{D_k\}_{T_2}$ des brins doubles non biotinylés isolés dans $T_2$.

  Les billes "Dynabeads Streptavidin for Target Enrichment" d'un diamètre d'un micromètre du fournisseur ThermoFisher peuvent être utilisées.

  Deux sous-fonctions sont associées à ce procédé :

  1. **Suppression par extrémités biotinylées**

     Cet opérateur permet de supprimer l'ADN contenant des amorces spécifiques. Pour cela, les molécules liées aux billes magnétiques sont extraites de la solution. La fonction associée est telle que :

$$(\{D_k^{\neg biot}\}, For_j^{biot}, Rev_j^{biot}) = SupBiotin(\{D_i\}, For_j^{biot}, Rev_j^{biot})$$

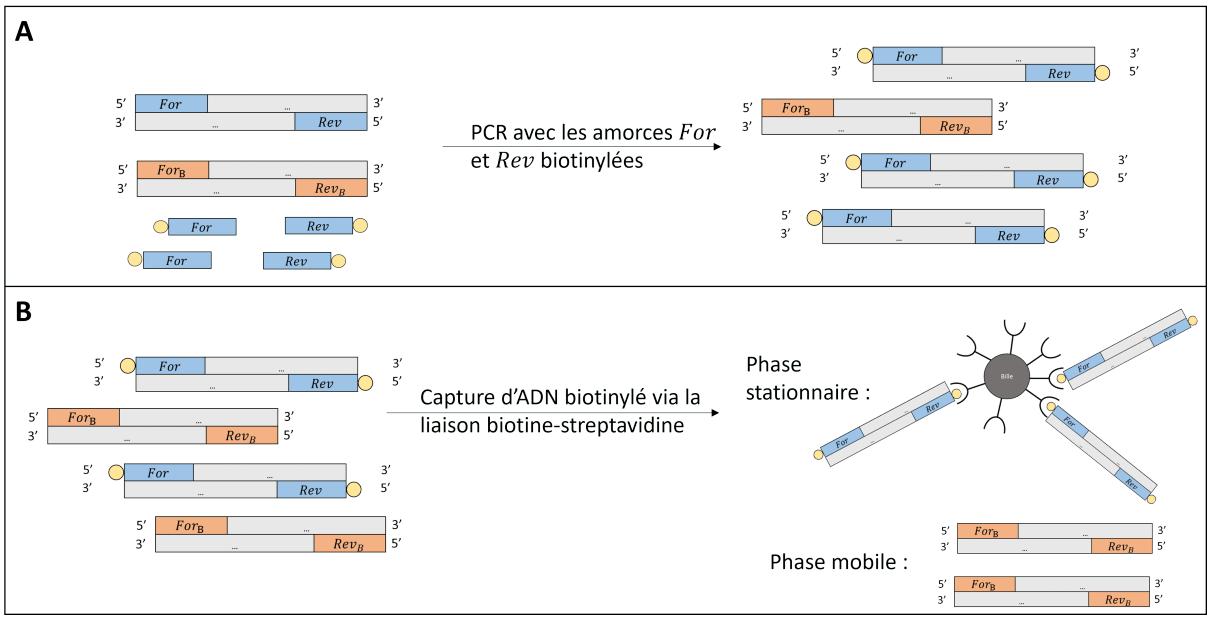



FIGURE 3.15 – Représentation schématique d'une capture d'ADN biotinylé grâce à la liaison entre les biotines des extrémités 5' de l'ADN et la streptavidine couplée aux billes magnétiques. Ces molécules d'ADN sont ensuite recupérées grâce à des aimants.

avec $\{D_i\}$ des brins doubles et $(For_j^{biot}, Rev_j^{biot})$ des couples d'amorces biotinylées, et $\{D_k^{\neg biot}\}$ un ensemble de brins doubles qui ne contiennent pas d'amorces $For_j$ et $Rev_j$.

2. **Tri par extrémités biotinylées**

   Cet opérateur permet de ne conserver que l'ADN contenant un ou plusieurs amorces spécifiques. Plus clairement, seuls les brins d'ADN liés aux billes magnétiques sont conservés. Cette fonction $TriBiotin$ est définie par :

$$(\{D_k^{biot}\}, For_j^{biot}, Rev_j^{biot}) = TriBiotin(\{D_i\}, For_j^{biot}, Rev_j^{biot})$$

   avec $\{D_i\}$ des brins doubles et $(For_j^{biot}, Rev_j^{biot})$ des couple d'amorces biotinylées, et $\{D_k^{biot}\}$ les brins doubles qui contiennent des amorces $For_j$ et $Rev_j$.

Un avantage de cette méthode est qu'elle est réalisable avec des brins doubles ou bien simples, tant que les extrémités 5' de l'ADN ciblé sont biotinylées. De plus, la capture se fait directement entre la biotine attachée à l'ADN et la streptavidine couplée aux billes magnétiques, l'ADN n'a pas besoin d'être modifié, via une dénaturation par exemple. Néanmoins elle présente plusieurs contraintes. Tout d'abord, les extrémités de l'ADN visé doivent être biotinylées, ce qui peut nécessiter une PCR avant l'étape de capture. D'autre part, la séparation des molécules se fait selon les sous-séquences présentes aux extrémités des molécules d'ADN, *i.e.* les amorces. Il n'est pas possible de séparer les molécules selon la présence de sous-séquences qui ne sont pas aux extrémités de l'ADN. Notons également que ces opérations ne sont jamais totales. Il restera toujours des molécules de la forme non désirée dans un tube à essai.

- **Séparation par hybridation** Si l'on souhaite récupérer de l'ADN contenant une sous-séquence

spécifique, et ce à n'importe quelle position dans une molécule, il est possible d'utiliser l'opérateur de séparation par hybridation. Celui-ci s'applique à des brins simples d'ADN.

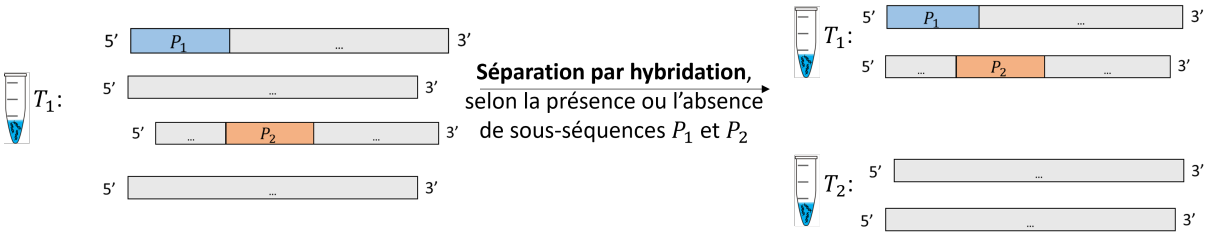



FIGURE 3.16 – Présentation de la séparation par hybridation : les molécules d'ADN sont séparées dans deux tubes à essai distincts, selon la présence ou l'absence de sous-séquences $P_1$ et $P_2$.

La récupération des brins simples d'ADN contenant des sous-séquences $P_j$ se fait grâce à des brins simples appelés sondes ADN, qui sont couplés à des billes magnétiques. Les sondes contiennent les sous-séquences $\overline{P_j}$, complémentaires aux $P_j$. Le principe de cette méthode repose sur l'hybridation des sondes aux brins d'ADN contenant les sous-séquences $P_j$. Pour récupérer les brins d'ADN hybridés, un aimant est collé à la paroi du tube à essai, isolant ainsi l'ensemble des séquences immobilisées par des billes, comme pour la méthode précédente de capture présentée en Figure 3.15.

Plus en détails, le protocole d'utilisation des sondes avec billes magnétiques (billes Dynabeads Streptavidin [170]) fonctionne de la manière suivante. Il débute par la préparation de la librairie et des billes. Celles-ci doivent être nettoyées pour supprimer la streptavidine présente en excès dans la solution. De l'ADN simple brin, dont les séquences sont des $\overline{P_j}$, est biotinylé : l'ADN est marqué avec la biotine à une de ses extrémités, à l'aide d'un kit commercial de marquage. Ensuite, les sondes biotinylées sont liées aux billes magnétiques par des liaisons non covalentes, avec une incubation à température ambiante. Les billes sont nettoyées, puis le procédé d'hybridation commence. Les billes avec sondes et les molécules visées sont introduites dans la solution, où les sondes ADN vont se lier aux brins simples complémentaires. Ces liaisons forment la phase stationnaire, immobilisée par les billes. Les molécules d'ADN non hybridées aux sondes, appelées phase mobile, sont retirées par pipetage du tube. Les billes immobilisées par aimant sont nettoyées, puis une phase d'élution permet de récupérer l'ADN capturé.

Cette fonction de séparation par hybridation $SHy$ peut être exprimée sous la forme :

$$(\{S_k\}_{T_1}^{\{P_j\}}, \{S_l\}_{T_2}^{\neg\{P_j\}}, , \{Sonde_{\overline{P_j}}\}) = SHy(\{S_i\}, \{Sonde_{\overline{P_j}}\})$$

avec : $\{S_i\}$ des brins simples ; $\{Sonde_{\overline{P_j}}\}$ des sondes ADN contenant les sous-séquences $\overline{P_j}$ ; $\{S_k\}_{T_1}^{\{P_j\}}$ les brins simples contenus dans le tube à essai $T_1$, qui contiennent au moins une sous-séquence parmi $\{P_j\}$ ; et, $\{S_l\}_{T_2}^{\neg\{P_j\}}$ les brins simples dans le tube $T_2$, qui ne contiennent aucune sous-séquence parmi $\{P_j\}$.

Cette fonction sous-entend en fait 3 opérations : capture de l'ADN avec les billes magnétiques couplées aux amorces, séparation, et élution, ce qui multiplie le risque de mauvais rendement.

De plus, lorsque l'ADN est sous forme de doubles brins, des étapes doivent être ajoutées pour pouvoir effectuer la séparation sur une solution contenant uniquement des brins simples : Il faut dénaturer les brins double d'ADN en augmentant la température. Cette étape est délicate car si la température est baissée à nouveau sous un certain seuil, les brins simples vont s'hybrider. Il faut donc plonger les tubes à essai dans la glace pour immobiliser l'ADN. A l'issue de l'étape de séparation, les brins simples isolés doivent être renaturés. La renaturation entraîne des erreurs car les brins ne se lient pas toujours correctement, notamment si des brins simples ont des régions similaires.

A partir de la fonction de séparation par hybridation $SHy$, deux autres opérateurs opérateurs peuvent être définis. Nous les décrivons dans les sections suivantes :

1. **Suppression par hybridation**

   Cet opérateur permet de supprimer les brins simples contenant une ou plusieurs sous-séquences spécifiques, par hybridation puis capture des brins à supprimer à l'aide d'aimants. Cette fonction $SupHy$ est définie comme suit :

   $$(\{S_k\}^{\neg P_j}, \{Sonde_{\overline{P_j}}\}) = SupHy(\{S_i\}, \{Sonde_{\overline{P_j}}\})$$

   avec $\{S_i\}$ des brins simples et $\{Sonde_{\overline{P_j}}\}$ des sondes ADN contenant les sous-séquences $\overline{P_j}$ , et $\{S_k\}^{\neg S_j}$ les brins simples qui ne contiennent aucune sous-séquence parmi $\{S_j\}$.

2. **Tri par hybridation**

   Cet opérateur permet de ne conserver que les brins simples contenant une ou plusieurs sous-séquences spécifiques. Pour cela, les brins capturés sont déposés dans un nouveau tube à essai. Cette fonction $TriHy$ est telle que :

   $$(\{S_k\}\{P_j\}, \{Sonde_{\overline{P_j}}\}) = TriHy(\{S_i\}, \{Sonde_{\overline{P_j}}\})$$

   avec $\{S_i\}$ des brins simples et $\{Sonde_{\overline{P_j}}\}$ des sondes ADN contenant les sous-séquences $\overline{P_j}$, et $\{S_k\}^{\{P_j\}}$ les brins simples qui contiennent des sous-séquences parmi $\{P_j\}$.

- **Séparation par taille avec électrophorèse sur gel d'agarose.** Cette méthode [171] permet de séparer les molécules d'ADN par taille, en plusieurs catégories, et de récupérer les molécules d'une catégorie de taille souhaitée.

  Pour séparer les brins d'ADN, ils sont déposés sur une plaque avec gel d'agarose, elle-même disposée dans une cuve contenant une solution tampon de migration. Un courant électrique est appliqué entre l'anode et la cathode disposées de part et autre du gel d'agarose. L'ADN étant une molécule chargée négativement, elle va naturellement migrer dans le gel d'agarose en direction de la cathode. La vitesse de migration dans le gel dépend de la taille des molécules d'ADN considérées : plus une molécule est grande, plus la migration est lente. Ainsi, à l'issue de la migration, nous avons sur la plaque un profil composé de différentes bandes, où chaque bande contient des molécules d'ADN de tailles proches.

  Après environ une heure, il est possible d'observer quelle proportion d'ADN a migré et à quelle vitesse (par comparaison avec une échelle d'ADN de tailles connues). Les molécules dans la fourchette

de taille souhaitée sont alors récupérables sur le gel. Pour cela, sur une table à UV, avec un scalpel, la bande d'ADN de la taille correspondante est découpée dans le gel et purifiée. Cette méthode fonctionne sur des fragments d'ADN dont la taille est comprise entre 10 et 50000 paires de bases [172]. Un colorant fluorescent, le bromure d'éthidium permet de détecter les fragments d'ADN sur le gel. Ainsi, des quantités très faibles d'ADN peuvent être visualisées en lumière UV ($5 - 10ng$).

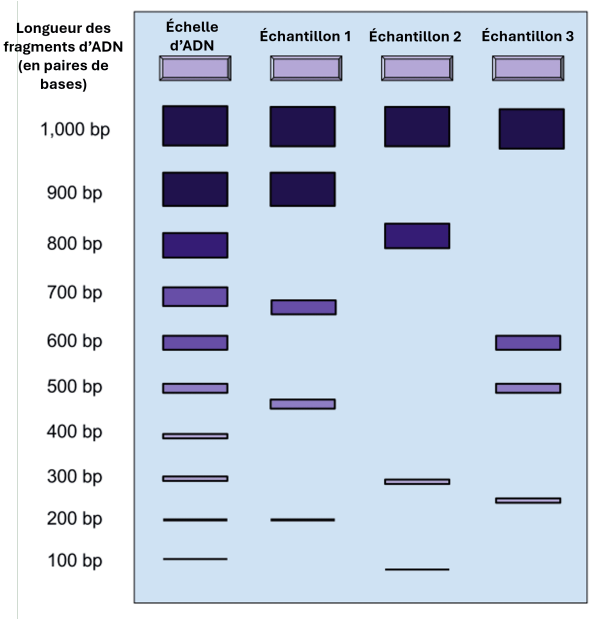



FIGURE 3.17 – Représentation schématique d'un résultat d'électrophorèse de l'ADN. A gauche, une échelle d'ADN est utilisée comme contrôle et comme référence pour la longueur de l'ADN (en paires de bases). À sa droite, on trouve trois exemples d'échantillons d'ADN différents : Échantillon 1, Échantillon 2 et Échantillon 3. *Source : Wikimedia Commons CC BY-SA 4.0.*

Un exemple de résultat d'électrophorèse est présenté en Figure 3.17. Une échelle d'ADN permet de visualiser les tailles des molécules d'ADN présentes dans les 3 échantillons. Les échantillons d'ADN sur la plaque sont bien séparés par blocs de tailles différentes.

La concentration du gel peut être adaptée pour modifier la vitesse de migration selon la taille des molécules. Les gels couramment utilisés [172] varient de $0,5\%$ à $5,0\%$, donnant la quantité d'agarose en grammes pour 100 mL de volume final. Les différentes concentration de gel et les longueurs de molécules associées sont présentées dans le Tableau 3.1. Par exemple, les molécules de 25 à 1000 paires de bases peuvent être séparées avec un gel concentré à $3,0\%$. Comme nous pouvons l'observer dans le Tableau 3.1, un gel avec un pourcentage plus important d'agarose,*i.e.* , plus dense, sépare des molécules de tailles plus proches.

La fonction associée à la séparation par électrophorèse sur gel est la suivante :

$$(\{D_k\}_{[V_1,V_2]}) = SepGel(\{D_i\}, [V_1, V_2])$$

avec $\{D_i\}$ des brins doubles et $[V_1, V_2]$ intervalle de longueur de molécules ADN souhaité, et $\{D_k\}_{[V_1,V_2]}$ des brins doubles d'ADN dont la taille est comprise entre $V_1$ et $V_2$ paires de bases.

TABLE 3.1 – Concentrations de gel d'agarose pour l'électrophorèse d'ADN et plages de séparation d'ADN en paires de base associées, selon l'entreprise ThermoFisher [172].

| Concentration du gel d'agarose en % | 0,5 | 0,6 | 0,8 | 1,0 | 1,2 | 1,5 | 2,0 | 3,0 | 4,0 | 5,0 |
|---|---|---|---|---|---|---|---|---|---|---|
| Taille minimale des fragments d'ADN (en paires de bases) | 2000 | 1000 | 800 | 400 | 300 | 200 | 100 | 25 | 10 | 10 |
| Taille maximale des fragments d'ADN (en paires de bases) | 50000 | 20000 | 10000 | 8000 | 7000 | 3000 | 2000 | 1000 | 500 | 300 |

Cet opérateur présente plusieurs contraintes. Pour pouvoir effectuer une électrophorèse, il faut connaître l'échelle de taille recherchée pour choisir la concentration de gel adaptée. De plus, la résolution est mauvaise pour des molécules de tailles proches. D'autre part, cette opération est fastidieuse car elle est difficilement automatisable. En conséquence, il faut identifier à l'oeil les bandes sur la piste, et découper le gel au scalpel avant élution pour récupérer les molécules de la taille souhaitée. Il y a une incertitude de quelques dizaines de bases sur la taille des molécules récupérées. Enfin, l'ensemble des étapes dure d'une à trois heures.

- **Séparation par taille avec chromatographie**

  Cette méthode sépare les molécules d'ADN par taille. Pour cela, la solution contenue dans le tube à essai passe dans une colonne avec un gel de billes sphériques contenant des pores d'une taille spécifique. Plus les molécules d'ADN sont petites, plus elles mettent du temps à passer dans les pores. Si les molécules sont trop grandes, elles ne passent dans aucuns pores et passent très vite à travers la colonne. Selon la colonne utilisée, on connaît le temps que prennent les molécules à migrer. Il est donc possible de récupérer les molécules qui sont dans un certain intervalle de taille. Par exemple, les colonnes de type Superdex™ 75 Increase 10/300 GL de l'entreprise Cytiva permettent de séparer les fragments d'ADN selon leur taille inférieure ou supérieure à $L = 107$ paires de bases, environ (d'un poids moléculaire inférieur à 70000 Dalton). La colonne Superdex™ 200 Increase du même fabricant sépare les molécules d'ADN selon une taille approximée à $L = 923$ paires de bases (poinds moléculaire de 600000 Da), environ.

  Cet opérateur de séparation nécessite 3 étapes qui sont : le filtrage, la récupération des molécules dans les pores, et la récupération des molécules éluées. Nous lui associons la fonction $SepChroma$ définie telle que :

  $$(\{D_j\}_{T_1}^{\leq L}, \{D_k\}_{T_2}^{\geq L}) = SepChroma(\{D_i\}, L)$$

  avec $\{D_i\}$ des brins doubles et $L$ un seuil de longueur, $\{D_j\}_{\leq T}$ et $\{D_k\}_{\geq T}$ des brins doubles d'ADN dont la taille est respectivement inférieure et supérieure à $L$ paires de bases.

  Cet opérateur requiert 10% de différence de poids moléculaire minimum pour bien trier les molécules, et ne fonctionne pas si on requiert une taille précise. De plus, il ne permet pas de séparer des molécules de petite taille.

- **Purification**

La purification ou "Clean-up" est une étape de nettoyage qui purifie l'ADN en éliminant des déchets, contaminants ou du matériel inutile tel que des enzymes, des amorces ou des protéines. En effet, ceux-ci peuvent interférer avec certaines opérations comme les coupes d'ADN. La purification permet de changer la solution dans laquelle se trouve l'ADN. Pour cela, une colonne est placée dans un tube à essai, et la solution est versée sur cette colonne. Celle-ci est choisie pour laisser passer les petites séquences ADN (*i.e.* de moins de 50 bases) tandis que les autres restent coincées dans une membrane. Les longues molécules d'ADN ainsi immobilisées dans la membrane sont récupérées par lavage avec une solution spécifique. De cette façon, des petites séquences telles que les amorces utilisées lors de la PCR sont éliminées.

Un kit commercial "Oligo Clean-Up" et le protocole associé de NEB (New England Biolabs) peuvent être utilisés. Une membrane filtrante peut traiter au maximum 5 $\mu g$ d'ADN simple brin de plus de 200 bases et double brin de plus de 50 paires de bases (cette capacité dépend de la taille des molécules d'ADN). Le rendement de cette manipulation n'est pas de 100% car une partie de l'ADN n'est pas immobilisé dans la membrane, ou bien reste dans celle-ci lors de l'étape de lavage.

La fonction de purification est définie comme suit :

$$(\{D_i\}_{l\geq 50pb}, \{S_j\}_{l\geq 50pb}) = Purif(\{D_i\}, \{S_j\}, Contam)$$

avec $\{D_i\}$ des brins doubles d'ADN et $\{S_j\}$ des brins simples d'ADN, et $Contam$ que des contaminants tels que des amorces ou des enzymes, et $(\{D_i\}_{l\geq 50pb}, \{S_i\}_{l\geq 50pb})$ des brins doubles et simples de plus de 50 paires de bases.

- **Mélange de deux solutions**

  Cet opérateur est le mélange de deux solutions, ou de deux ensembles de molécules d'ADN, dans une unique solution.

  Pour qu'une opération sur la solution obtenue par mélange soit efficace, les molécules doivent être présentes à proportions équivalentes. Un dosage peut donc être nécessaire. Il permet de déterminer quels volumes des différentes solutions sont mélangés.

  La fonction de mélange de plusieurs solutions est définie comme suit :

$$(\{D_k\}, \{S_k\})_{T_k} = Mix((\{D_1\}, \{S_1\})_{T_1}, ..., (\{D_j\}, \{S_j\})_{T_j})$$

  avec $(\{D_1\}, \{S_1\})_{T_1}, ..., (\{D_j\}, \{S_j\})_{T_j}$ des brins doubles et simples dans leurs tubes à essai respectifs $T_1$ à $T_j$, et $(\{D_k\}, \{S_k\})_{T_k}$ l'ensemble des brins doubles et simples dans un seul tube à essai $T_k$.

L'ensemble des opérateurs biologiques élémentaires que nous avons identifié dans les sections précédentes est présenté dans les Tableaux 3.18 et 3.19, avec pour chacun la description et une représentation schématique de sa fonction.

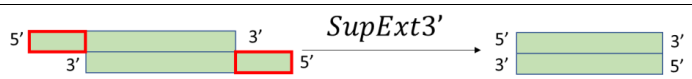
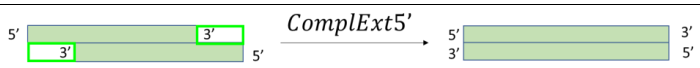
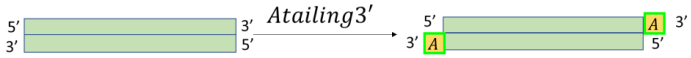
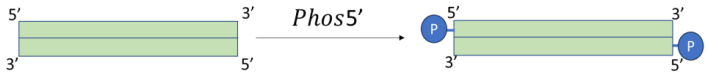
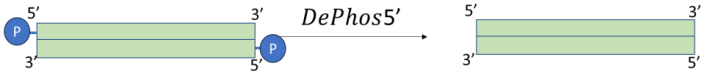
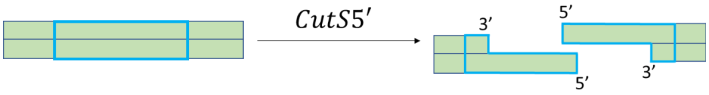
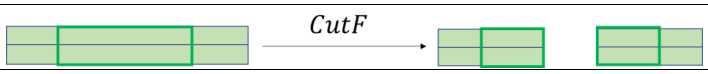
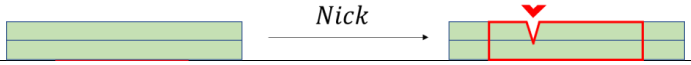
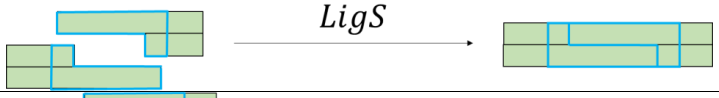
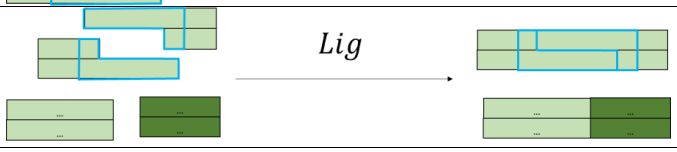
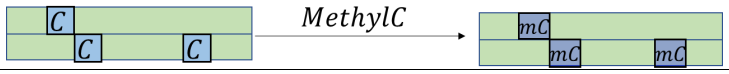
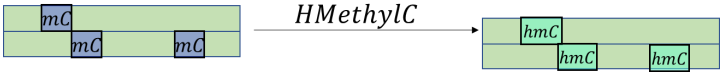
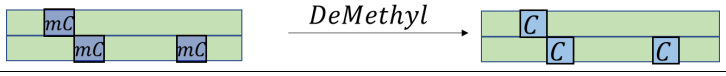
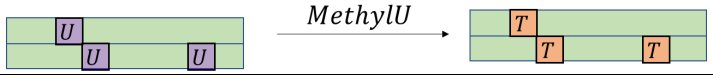
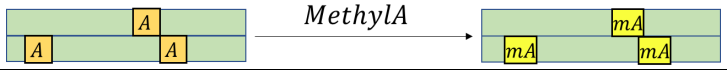
| Classe 1 : Modification des extrémités de l'ADN | | |
|---|---|---|
| **Fonction** | Description | Schéma |
| **SupExt3'** | Suppression des extrémités saillantes 3' |   |
| **ComplExt5'** | Complétion des extrémités saillantes 5' |   |
| **Blunting** | SupExt3' et ComplExt5' | |
| **ATailing3'** | Ajout de bases A aux extrémités 3' |   |
| **Phos** | Ajout d'un groupe phosphate aux extrémités 5' |   |
| **DePhos** | Suppression des groupes phosphates aux extrémités 5' |   |
| **Classe 2 : Coupes de l'ADN** | | |
| **CutS5'** | Coupe saillante formant des extrémités saillantes 5' |   |
| **CutS3'** | Coupe saillante formant des extrémités saillantes 3' |   |
| **CutF** | Coupe formant des extrémités franches |   |
| **Nick** | Création d'entaille dans un des brins d'une molécule |   |
| **Classe 3 : Ligations de l'ADN** | | |
| **NickRepair** | Réparation d'entailles dans une molécule |   |
| **LigS** | Ligation de molécules avec extrémités saillantes compatibles |   |
| **Lig** | Ligation de molécules avec extrémités saillantes compatibles et avec extrémités franches |   |
| **Classe 4 : Modifications de bases ADN** | | |
| **MethylC** | Transformation des bases C en bases méthylées mC |   |
| **HMethylC** | Transformation des bases mC en bases hydroxy-méthylées hmC |   |
| **DeMethyl** | Transformation des bases mC en bases C par déméthylation |   |
| **MethylU** | Transformation des bases U en bases T par méthylation |   |
| **MethylA** | Transformation des bases A en bases méthylées mA |   |

FIGURE 3.18 – Opérateurs biologiques élémentaires identifiés, répartis dans 6 classes, avec pour chaque opérateur le nom de la fonction associée, une courte description, et une représentation schématique. La deuxième partie des opérateurs est présentée dans le Tableau 3.19.

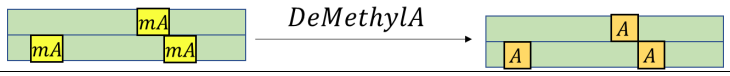
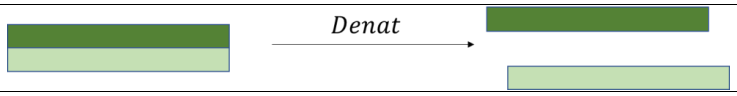
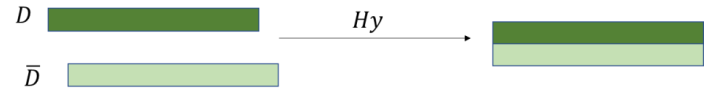
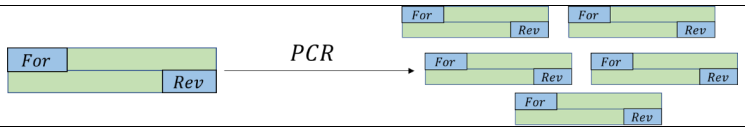
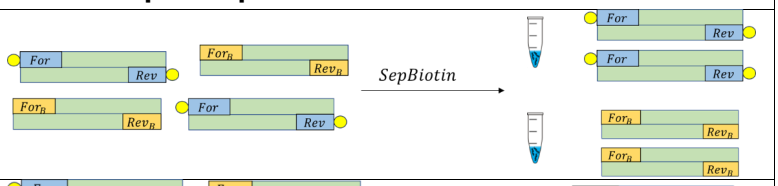
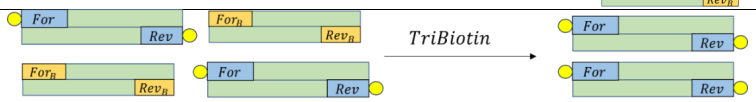
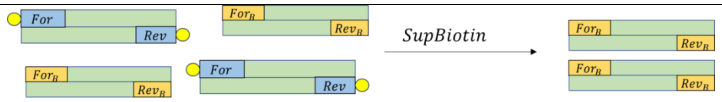
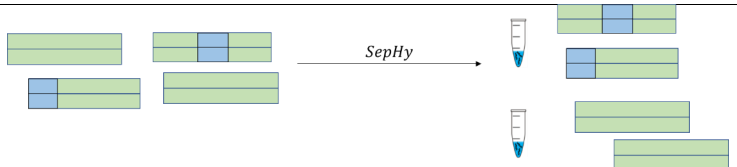
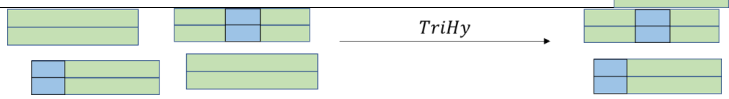
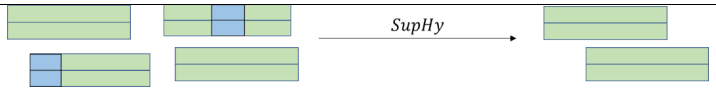
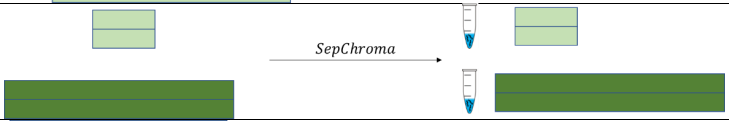
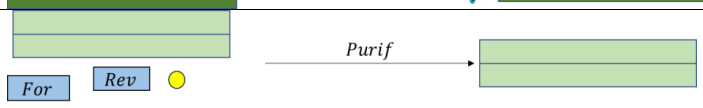
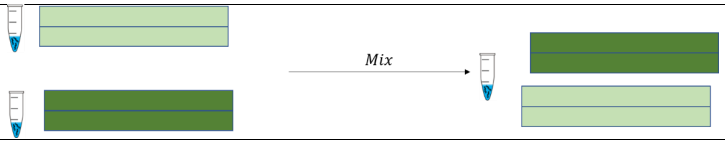
| | | |
|---|---|---|
| **DeMethylA** | Transformation des bases mA en bases A par déméthylation |   |
| **Classe 5 : Amplification** | | |
| **Denat** | Dénaturation des brins doubles en brins simples d'ADN |   |
| **Hy** | Hybridation des brins simples complémentaires |   |
| **PCR** | Amplification des molécules encadrées par les amorces For et Rev |   |
| **Classe 6 : Séparation de molécules spécifiques** | | |
| **SepBiotin** | Séparation de l'ADN dans deux solutions selon la présence d'extrémités biotinylées |   |
| **TriBiotin** | Tri des molécules pour conserver celles avec des extrémités biotinylées |   |
| **SupBiotin** | Suppression des molécules avec des extrémités biotinylées |   |
| **SepHy** | Séparation de l'ADN dans deux solutions selon la présence de sous-séquences spécifiques |   |
| **TriHy** | Tri des molécules pour conserver celles contenant des sous-séquences spécifiques |   |
| **SupHy** | Suppression des molécules contenant des sous-séquences spécifiques |   |
| **SepGel** | Séparation des molécules dans un intervalle de taille par électrophorèse sur gel d'agarose |   |
| **SepChroma** | Séparation des molécules plus ou moins longues qu'un seuil défini, par chromatographie |   |
| **Purif** | Purification de l'ADN présent dans une solution, supprimant tout déchet, amorce, etc. |   |
| **Mix** | Mélange de l'ADN présent dans deux solutions |   |

FIGURE 3.19 – Opérateurs biologiques élémentaires identifiés, répartis dans 6 classes, avec pour chaque opérateur le nom de la fonction associée, une courte description, et une représentation schématique. La première partie des opérateurs est présentée dans le Tableau 3.18.

### 3.1.7 Définition d'une fonction avancée de rotation biologique

Dans les sections précédentes, nous avons présenté des opérateurs biologiques de type élémentaires (*cf.* Tableaux 3.18 et 3.19). Nous allons maintenant voir qu'il est possible à partir de ces derniers de définir des fonctions biologiques plus avancées permettant notamment de modifier la structure d'une molécule d'ADN double brin. Dans cette section, nous présentons une fonction de rotation qui exploite donc plusieurs de ces opérateurs élémentaires. Elle a pour objectif d'échanger les positions de deux blocs adjacents de nucléotides dans une molécule d'ADN double brin.

Soit une molécule ADN double brin $D$ divisée en blocs $d_i$ de paires d'au minimum quelques dizaines de bases ADN et encadrée par des blocs $For_D$ et $Rev_D$, telle que :

$$D = \Big( For_D d_1 d_2 d_3 d_4 Rev_D \Big)$$

avec $d_i = \left( \frac{B_1^i \; B_2^i \; \cdots}{B_1^i \; B_2^i \; \cdots} \right)$ des blocs contenant des bases ADN $B_j^i$ et les bases complémentaires $\overline{B_j^i}$, et $For_D$ un bloc de paire de bases composé de l'amorce $For$ et son complémentaire $\overline{For}$, tel que $For_D = \left( \frac{For}{\overline{For}} \right)$. Le bloc $Rev_D$ correspond lui à l'amorce $Rev$ et son complémentaire, tel que $Rev_D = \left( \frac{\overline{Rev}}{Rev} \right)$.

La fonction de rotation proposée repose sur des opérations élémentaires de coupe, de ligation, de séparation selon des amorces biotinylées et de mélange de solutions. Avec ces opérateurs, nous coupons la molécule $D$ et isolons le bloc $d_2$, avant de l'insérer après $d_3$ dans $D$. Ainsi, les positions des blocs $d_2$ et $d_3$ sont échangées.

Plus précisément, le principe de la rotation est d'alterner des coupes, pour séparer de la molécule les blocs que l'on souhaite déplacer, et des ligations, pour ré-associer les blocs aux positions désirées dans la molécule. Cela ne peut cependant pas se faire sans des opérations permettant d'isoler certaines molécules, de purifier le contenu d'une solution et de mélanger les molécules contenues dans plusieurs solutions.

Comme présenté, de manière à pouvoir échanger les positions des deux blocs $d_2$ et $d_3$ présentés en Figure 3.20, nous allons leur associer des sites de restriction (*cf.* Section 3.1.2) : des blocs de $r$ paires de bases (généralement 6 à 10 bases) que des enzymes pourront identifier pour des opérations de découpage. Ainsi, comme illustrée par la molécule $D$ double brin prête pour une rotation représentée en Figure 3.20 (a), deux sites de restriction $R_1$ encadrent le bloc $d_2$, et un site $R_2$ est ajouté après le bloc $d_3$. L'idée est de découper le brin selon ces sites. $D$ a la structure suivante :

$$D = (For_D d_1 R_1 d_2 R_1 d_3 R_2 d_4 Rev_D)$$

Une fois le brin d'ADN $D$ synthétisé, l'opération est réalisée au niveau moléculaire. Pour ce faire, nous proposons d'utiliser les opérateurs élémentaires suivants :

— Ligation d'ADN double brin $\{D_i\}$ avec extrémités saillantes complémentaires $LigS(\{D_i\})$.

— Mélange de solutions contenues dans des tubes à essai $(T_1, ..., T_i)$ dans un seul tube $T_j$ : $Mix((\{D_1\}, \{S_1\})_{T_1}, ..., (\{D_i\}, \{S_i\})_{T_i})$.

— Séparation de molécules dans deux tubes à essai selon la présence ou l'absence d'amorces $(For_j^{biot}, Rev_j^{biot})$ biotinylés : $SepBiotin(\{D_i\}, For_j^{biot}, Rev_j^{biot})$

— Coupe saillante $CutS(\{D_i\}, E_j)$ effectuée avec un enzyme $E_j$ coupant l'ADN double brin $\{D_i\}$

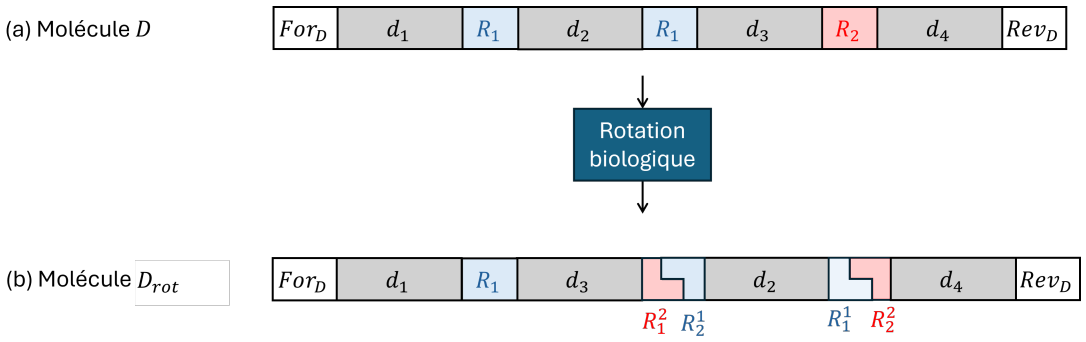



FIGURE 3.20 – Rotation de deux blocs $d_2$ et $d_3$ dans une molécule d'ADN double brin.

selon le site de restriction $R_j$.

Plus précisément, l'opérateur de Coupe saillante $CutS$ est utilisé avec deux enzymes particuliers. Ils ont des sites de restriction différents, mais créent les mêmes extrémités saillantes lorsqu'elles coupent l'ADN. Par exemple, les enzymes de restriction NheI et SpeI coupent des sites de restriction différents, $(\begin{smallmatrix}GCTAGC\\CGATCG\end{smallmatrix})$ et $(\begin{smallmatrix}ACTAGT\\TGATCA\end{smallmatrix})$ respectivement, et créent toutes les deux des extrémités saillantes $(CTAG)$ du côté 5'. En faisant ce choix, on peut lier deux molécules distinctes coupées par deux enzymes différents.

Décrivons en détails le fonctionnement de la fonction de Rotation, dont les 9 étapes sont présentées dans la Figure 3.21. La molécule d'ADN $D$ est tout d'abord coupée selon $R_1$, ce qui permet d'obtenir trois molécules distinctes. Chaque site $R_1$ est coupé en deux extrémités saillantes compatibles $R_1^1$ et $R_1^2$. A la deuxième étape, les molécules contenant une amorce $For$ ou $Rev$ sont isolées dans le tube à essai $T_1$, tandis que les autres molécules sont placées dans $T_2$. Dans $T_1$, une troisième étape permet de lier les deux molécules, par leurs extrémités saillantes compatibles $R_1^1$ et $R_1^2$, représentées en bleu. La molécule d'ADN obtenue est ensuite coupée avec l'enzyme $E_2$ à l'étape 4, formant deux molécules distinctes. Le site $R_2$ est coupé en deux extrémités compatibles $R_2^1$ et $R_2^2$. Le fragment avec une amorce $For$ est capturé à l'étape 5 et reste dans le tube $T_1$, tandis que l'autre molécule est placée séparément dans le tube $T_3$. A l'étape 6, les contenus des tubes $T_1$ et $T_3$ sont mélangés et liés par leurs extrémités saillantes compatibles, $R_2^1$ et $R_1^2$. L'étape 8 permet de mélanger le contenu des deux tubes à essai $T_1$ et $T_3$. Enfin, les deux molécules sont liées à l'étape 9 pour obtenir la molécule avec rotation de $d_2$ et $d_3$ : $(For_D d_1 R_1 d_3 R_2^1 R_1^2 d_2 R_1^1 R_2^2 d_4 Rev_D)$.

En Annexe A, l'algorithme 4 présente la fonction $Rot(D, d_2, d_3)$ de rotation des deux blocs adjacents $d_2$ et $d_3$ d'une même molécule d'ADN double brin.

La fonction biomoléculaire de rotation que nous venons de décrire peut donc être définie telle que :

$$D_{rot} = Rot(D, E_1, E_2)$$

avec $E_1$ et $E_2$ des enzymes de restriction et $D_{rot}$ la molécule présentée en Figure 3.20 (b), telle que :

$$D_{rot} = (For_D d_1 R_1 d_3 d_2 R_{1,2} d_4 Rev_D)$$

On observe bien l'échange de la position des blocs $d_2$ et $d_3$. Notons que cette rotation n'est pas parfaite, les sites de restriction $R_i$ permettant les coupes ont légèrement changé, en raison des recombinaisons de

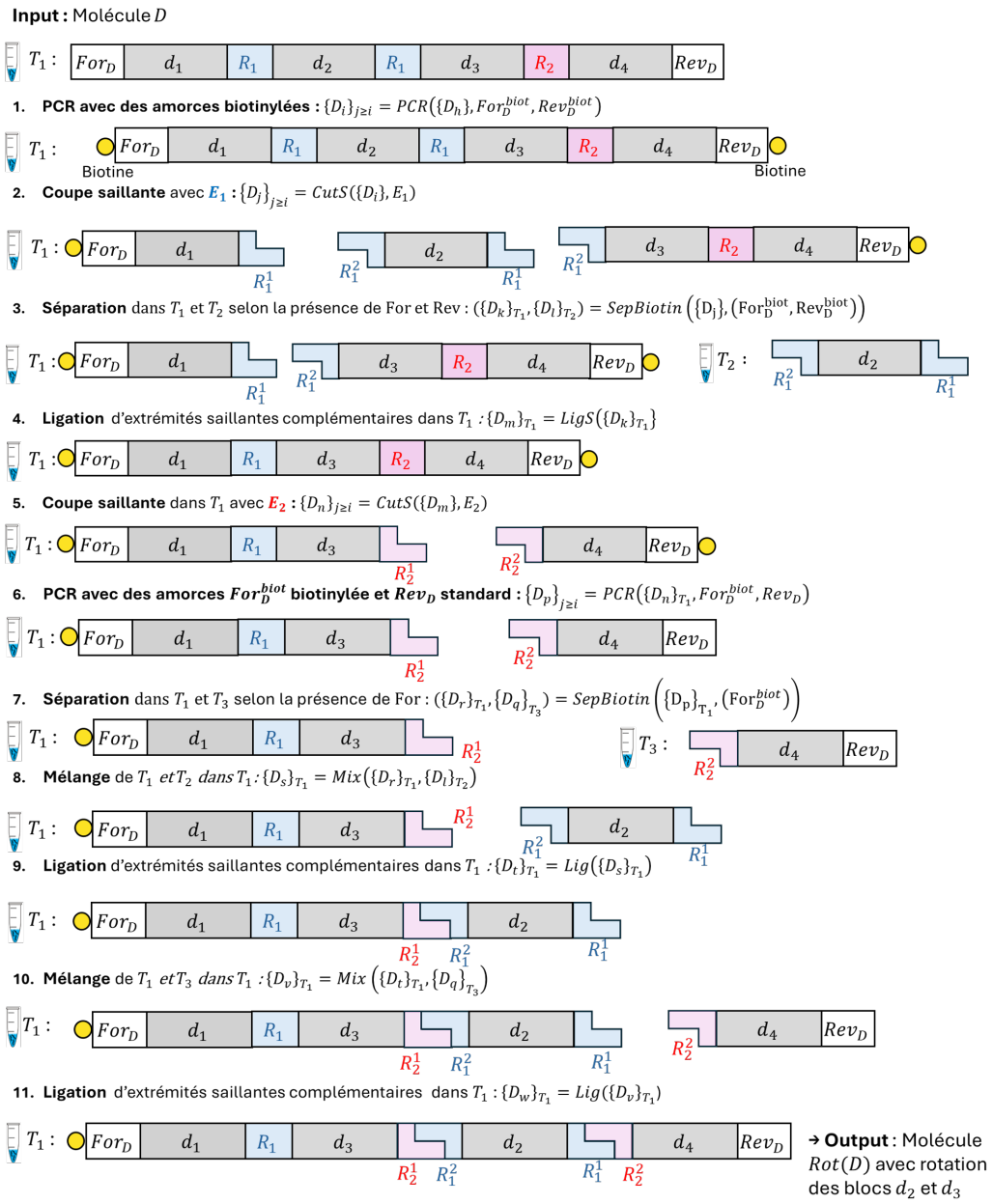



FIGURE 3.21 – Représentation schématique de la fonction de rotation en 9 opérations biologiques, qui permet d'échanger la position des deux blocs $d_2$ et $d_3$ de la molécule d'ADN double brin $D$.

sites de restriction $R_1$ et $R_2$ lors des ligations. Les sites obtenus par ligations sont $R_1^1R_2^2$ et $R_2^1R_1^2$.

Cette fonction de rotation repose donc sur plusieurs opérateurs avec des contraintes et des rendements divers. Pour sa part, l'opérateur coupes saillantes fonctionne bien et a un bon rendement en peu de temps. Une grande majorité des molécules sont coupées et il y a peu de coupes non prévues. L'opérateur de ligations d'extrémités saillantes est rapide et présente un bon rendement. Toutefois, certaines ligations non désirées peuvent avoir lieu et certaines extrémités compatibles ne se lieront pas. Deux mélanges de

deux solutions sont aussi utilisés. Cet opérateur est rapide et facile à mettre en place mais peut nécessiter un dosage de la concentration d'ADN dans chaque solution pour mélanger des quantités équivalentes d'ADN. La séparation par extrémités biotinylées est l'opérateur le plus complexe. Il y en a deux dans la fonction de rotation. En fait, la séparation n'est jamais totale. Il y aura des molécules non désirées dans chaque solution. De plus, elle implique plusieurs étapes : la préparation des billes magnétiques, la capture de l'ADN et la séparation de l'ADN dans 2 solutions. Si les extrémités de l'ADN n'ont pas de biotines, il est nécessaire d'effectuer une PCR avec amorces biotinylées avant la capture. Pour résumer, il est important de rappeler que les manipulations biologiques ne sont pas parfaites, chaque opérateur biologique a un taux d'erreur et peut créer des molécules avec des structures inattendues, qui vont se multiplier tout le long du processus.

### 3.1.8 Définition d'une fonction avancée de permutation de deux blocs de deux molécules différentes

La fonction de permutation que nous présentons repose sur plusieurs opérateurs élémentaires présentés précédemment. Elle permet l'échange de blocs de données de deux molécules d'ADN différentes encadrées par les mêmes amorces. Comme illustrée en figure 3.22, la fonction associée *Permut* prend en entrée deux molécules $D_1$ et $D_2$, et échange leurs derniers blocs $d_2^1$ et $d_2^2$.

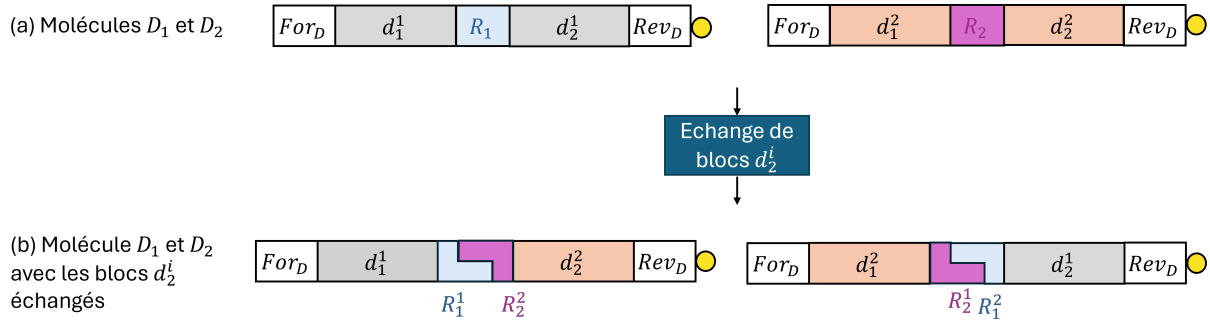



FIGURE 3.22 – Permutation des blocs $d_2^1$ et $d_2^2$ entre deux molécules $D_1$ et $D_2$ d'ADN double brin.

Plus en détail, soient des molécules d'ADN $D_i$ composées de deux blocs $d_1^i$ et $d_2^i$ de paires de bases, et encadrée d'amorces $For_D$ et $Rev_D$. Pour permettre la permutation de blocs entre deux molécules $D_1$ et $D_2$, il faut ajouter un site de restriction à chaque molécule séparant les deux blocs les constituants (*cf.* Figure 3.22 (a)). Ces sites de restriction sont choisis pour permettre des coupes saillantes et des ligations pour ré-associer les molécules (cf. Figure 3.22 (b)). $D_1$ et $D_2$ sont donc de la forme suivante :

$$D_1 = (For_D d_1^1 R_1 d_2^1 Rev_D)\ D_2 = (For_D d_1^2 R_2 d_2^2 Rev_D)$$

Dans sa mise en oeuvre au niveau biomoléculaire, la permutation repose sur les opérateurs biologiques élémentaires suivants :

— La coupe saillante *CutS*, avec deux enzymes de restriction qui coupent l'ADN en créant des extrémités saillantes compatibles.

— La séparation de molécules dans deux tubes à essai avec l'opérateur *SepBiotin*.

— La ligation d'extrémités saillantes cohésives avec $LigS$.

— Le mélange de solutions avec la fonction $Mix$.

— L'opérateur de déphosphorylation (voir Section 3.1.1) qui permet de supprimer les groupes phosphate aux extrémités 5' de l'ADN, empêchant les molécules déphosphorylées de se lier à d'autres molécules.

La fonction biomoléculaire de permutation est donc définie de la façon suivante :

$$(D'_1, D'_2) = Permut(D_1, D_2, R,_1, R_2)$$

avec $D'_1$ et $D'_2$ les molécules $D_1$ et $D_2$ dont les blocs $d_2^1$ et $d_2^2$ ont été permutés. Elle produit donc les molécules suivantes (*cf.* Figure 3.22(b)) :

$$D'_1 = (For_D d_1^1 R_1^1 R_2^2 d_2^2 Rev_D)\ D'_2 = (For_D d_1^2 R_2^1 R_1^2 d_2^1 Rev_D)$$

Cette fonction s'effectue en 10 étapes, présentées en Figure 3.23 :

1. Ajout de biotines aux extrémités $Rev_D$ de $D_1$ et $D_2$ avec $PCR(D_1, D_2, For_D, Rev_D^{biot})$, une PCR avec des amorces $Rev_D$ biotinylées. Les biotines sont représentées par des ronds jaunes dans la Figure 3.23.
2. : Coupe de la molécule $D_1$ selon $R_1$ avec $CutS(D_1, E_1)$, ce qui permet d'obtenir deux molécules distinctes. Chaque site $R_1$ est coupé en deux extrémités compatibles $R_1^1$ et $R_2^1$, qui sont naturellement phosphorylées (*cf.* groupe phosphate représenté par un rond bleu).
3. Séparation des molécules avec des extrémités biotinylées dans un tube $T_2$, avec $SepBiotin(\{D\}, Rev_D^{biot})$.
4. Dans les tubes $T_1$ et $T_2$, les extrémités de l'ADN sont déphosphorylées avec $Dephos(\{D\}_{T_1})$ et $Dephos(\{D\}_{T_2})$. Nous allons ainsi pouvoir utiliser la propriété qui dit que deux extrémités déphosphorylées ne pourront pas être liées, et donc empêcher le reformation de la molécule $D_1$.
5. Coupe dans le tube $T_2$ selon le site de restriction $R_2$ avec $CutS(\{D\}_{T_2}, E_2)$ : la molécule $D_2$ est coupée en deux molécules avec des extrémités saillantes compatibles phosphorylées, *i.e.* avec des groupes phosphate aux extrémités 5'.
6. Dans le Tube $T_2$, les molécules sont séparées selon si leurs extrémités sont biotinylées ou non, avec $SepBiotin(\{D\}_{T_2}, Rev_D^{biot})$.
7. Les molécules avec les extrémités biotinylées sont mélangées au tube $T_1$ avec $Mix(\{D\}_{T_1}, \{D\}_{T_2})$.
8. Dans le tube $T_1$, une ligation est effectuée avec $LigS(\{D\}_{T_1})$. Une liaison peut se faire seulement si au moins une extrémité est biotinylée, donc seules les molécules $d_1^1$ et $d_2^2$ se lient.
9. Le tube $T_3$ est mélangé au tube $T_1$ avec $Mix(\{D\}_{T_1}, \{D\}_{T_3})$
10. Une ligation est effectuée dans $T_1$ avec $LigS(\{D\}_{T_1})$, elle permet de lier les molécules $d_1^2$ et $d_2^1$.

A la fin de ce protocole, nous obtenons deux molécules $D_1$ et $D_2$ avec les blocs $d_2^i$ échangés. Grâce à la représentation de nos opérateurs élémentaires sous forme de fonction, il est possible de représenter la fonction $Permut(D_1, D_2, R_1, R_2)$ d'échange de blocs entre les molécules $D_1$ et $D_2$ sous la forme d'un algorithme, comme illustré par l'Algorithme 5, en Annexe B.

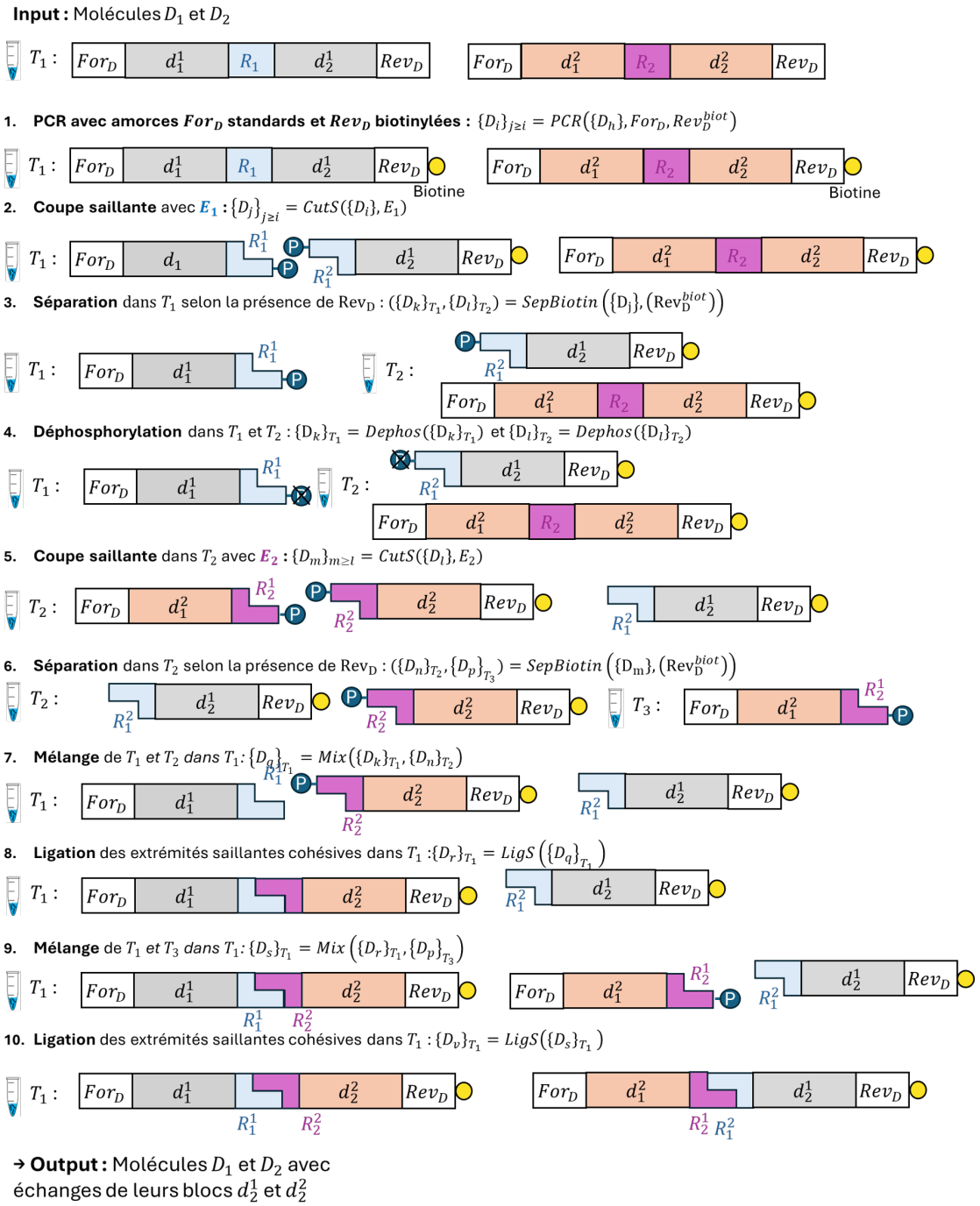



FIGURE 3.23 – Représentation schématique de la fonction de permutation de blocs de 2 molécules différentes en 11 opérations biologiques, qui permet d'échanger deux blocs $d_2^1$ et $d_2^2$ entre les deux molécules $D_1$ et $D_2$ d'ADN double brin. Les biotines sont représentées par des ronds jaunes, tandis que les groupes phosphates sont représentés par des ronds bleus.

Tout comme pour la rotation, cette permutation n'est pas parfaite : les sites de restriction séparant $d_1^i$ et $d_2^i$ dans chaque molécule ont légèrement changé, car les sites $R_1$ et $R_2$ sont recombinés lors des

ligations. Pour lier le bloc $d_2^1$ de $D_1$ au bloc $d_1^2$ de $D_2$, les extrémités saillantes de deux sites de restriction différents, $R_1$ et $R_2$, sont liées. Cela donne un nouveau site de restriction, de la forme $R_1^1 R_2^2$. Donc, les nouveaux sites $R_1^1 R_2^2$ et $R_2^1 R_1^2$ précèdent les blocs échangés $d_2^2$ et $d_2^1$ dans $D_1$ et $D_2$.

La fonction de permutation de blocs entre deux molécules repose sur plusieurs opérateurs avec des rendements divers. Elle utilise en grande partie les opérateurs de la fonction de Rotation, dont nous avons déjà discuté les limitations dans la section précédente. La nouveauté est l'opérateur de déphosphorylation, un opérateur biochimique simple, qui fonctionne très bien lorsque les conditions sont optimisées, notamment les températures et temps de réaction.

A l'issue de la première partie de ce chapitre, nous avons identifié plusieurs opérateurs biologiques permettant des opérations élémentaires sur des molécules d'ADN. Sur cette base, nous avons également montré qu'il était possible d'élaborer des fonctions plus avancées : une fonction de rotation de séquences de nucléotides, ou de manière équivalente de blocs d'information, au sein d'une même molécule ; et une fonction de permutation de blocs d'information entre des molécules. Ces fonctions sont essentielles à la solution de sécurité que nous proposons dans la deuxième partie de ce chapitre. Elle a pour idée d'obliger un utilisateur à devoir effectuer des manipulations biologiques sur les molécules avant de pouvoir les séquencer et lire correctement l'information.

## 3.2 Solution de sécurité

Dans cette section, nous présentons une solution de sécurité DNACipher qui impose l'utilisation de manipulations biologiques pour déchiffrer les molécules d'ADN avant de pouvoir les séquencer. En particulier, nous verrons que DNACipher repose sur des rotations et des permutations différentes appliquées à des copies d'une même séquence ADN avec pour but de perturber l'étape de consensus qui suit le séquençage. Cette étape de consensus prend en entrée le séquençage de plusieurs copies de la même molécule (*i.e.* encadrées des mêmes amorces) pour produire une seule séquence consensus par vote majoritaire (tel que nous l'avons présenté dans la chaîne de stockage, Chapitre 1 Section 1.1.2). Si ces opérations ne sont pas inversées, il sera difficile de retrouver les molécules par consensus car, bien qu'elles aient les mêmes amorces, elles ne peuvent être alignées.

Plus schématiquement, l'idée consiste à voir le consensus comme prenant en entrée une matrice, où chaque ligne est une copie d'une même molécule séquencée. Pour perturber le consensus, l'approche de DNACipher consiste à synthétiser des copies d'une même molécule sur lesquelles des rotations et des permutations différentes ont été appliquées. Si le séquençage est réalisé sans inverser ces perturbations, la matrice de consensus est désynchronisée et les conditions pour un consensus correcte ne sont pas réunies. Notre solution de sécurité repose donc sur un chiffrement en numérique pour préparer les copies d'une même molécule, et sur un déchiffrement, appelé DNADecipher, qui est lui biomoléculaire.

Notons qu'avec cette méthode, nous privilégions une forte densité d'information stockée. En effet, nous n'utilisons pas d'algorithme de stéganographie, consistant à cacher des données dans un grand volume, ni de molécules d'ADN leurre ne codant pas de données. Nous ajoutons seulement quelques courts motifs ne codant pas d'information dans chaque séquence, qui nous permettent d'utiliser certains opérateurs biologiques. En particulier, chaque fonction de rotation et de permutation interdit l'utilisation de 4 motifs : $R_1$, $R_2$, $R_1^1R_2^2$ et $R_1^2R_2^1$.

Dans cette section, nous présentons tout d'abord la chaîne de stockage globale permettant d'encoder des données dans de l'ADN et de les récupérer. Nous détaillerons ensuite les processus de chiffrement numérique et de déchiffrement biomoléculaire et numérique, avant de discuter de la complexité de notre solution.

### 3.2.1 Chaîne de stockage de données dans l'ADN

Nous nous plaçons dans le scénario où un tube à essai contient les données de plusieurs utilisateurs. Comme présenté dans la Figure 3.24, un fichier $F_i^U$ d'un utilisateur $U$ est contenu dans une molécule d'ADN double brin $D^{U,i}$ à taille variable. Une telle molécule est encadrée par un couple unique d'amorces $(For_i^U, Rev_i^U)$.

Dans une chaîne de stockage classique avec un chiffrement et un déchiffrement numériques, l'écriture de données chiffrées dans l'ADN repose sur un encodage des données binaires en ADN et une synthèse. Pour récupérer l'information, les étapes biologiques sont la sélection d'ADN par PCR et le séquençage.

Avec notre solution de sécurité, comme illustré en Figure 3.25, deux étapes sont ajoutées à cette chaîne. Elles sont encadrées en rouge. La première, DNACipher, est numérique et se positionne avant la synthèse. Elle prend en entrée une molécule qui encode un fichier, et va générer des copies différentes sur lesquelles ont été appliquées des rotations et des permutations. La seconde, DNADecipher est biologique et est

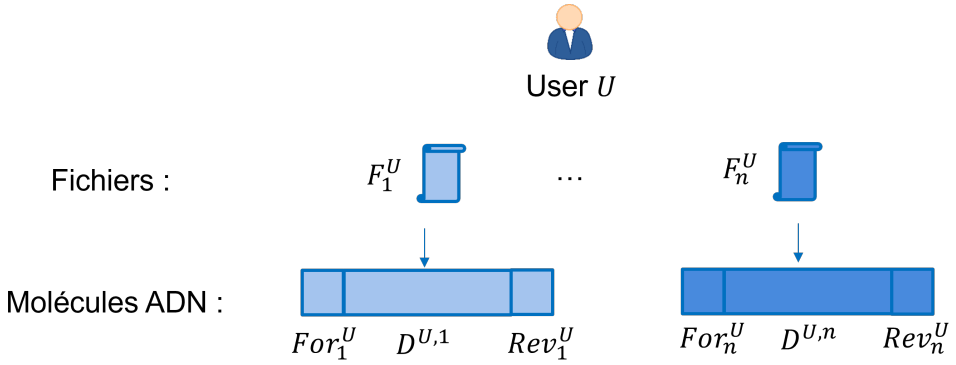



FIGURE 3.24 – Indexation des données de l'utilisateur $U$, où un fichier est encodé dans une molécule d'ADN avec son couple unique d'amorces.

ajoutée avant de pouvoir récupérer les données encodées dans l'ADN : Notre algorithme de déchiffrement biomoléculaire DNADecipher doit obligatoirement être effectué avant le séquençage. Ce déchiffrement est basé sur des opérateurs biologiques qui inversent les rotations et de permutations précédemment introduites.

Décrivons plus en détails cette nouvelle chaîne de stockage sécurisée. Le processus d'écriture des données dans l'ADN est présenté en Figure 3.25 (a). Les données binaires que l'on souhaite stocker sont tout d'abord chiffrées avec un crypto-système numérique, puis encodées avec l'algorithme DSWE en une séquence ADN. L'algorithme DSWE prend en compte l'interdiction de plusieurs motifs interdits, notamment les sites de restriction utilisés par certains opérateurs biologiques. Ensuite, notre algorithme de chiffrement DNACipher produit plusieurs copies chiffrées de la séquence ADN. Chaque copie est chiffrée avec des opérateurs biologiques différents, et contient donc différents blocs de données désordonnés. Enfin, les séquences ADN sont préparées pour la synthèse : Un même couple d'amorces est ajouté aux extrémités de chaque séquence ADN et les séquences ADN sont vérifiées pour s'assurer qu'elles sont viables pour la synthèse. Enfin, les séquences ADN sont synthétisées et stockées sous forme de molécule d'ADN double brin.

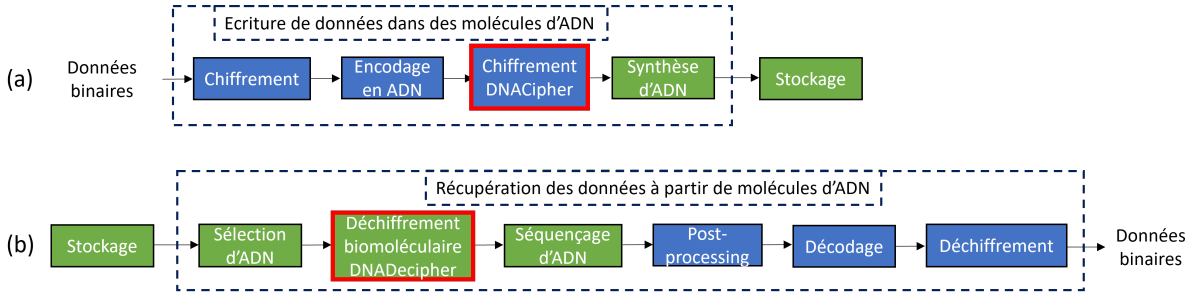



FIGURE 3.25 – Chaîne de stockage incluant le procédé d'écriture de données dans l'ADN (Figure (a)) avec notre chiffrement DNACipher, et le procédé de récupération de données stockées (Figure (b)) avec le déchiffrement biomoléculaire DNADecipher.

La récupération des données à partir des molécules d'ADN est présentée en Figure 3.25 (b). Elle commence par une sélection par PCR des molécules avec les amorces adaptées. Ensuite, le déchiffrement biomoléculaire DNADecipher est appliqué aux molécules d'ADN. Plusieurs fonction biologiques sont

appliquées, elles permettent de déchiffrer les différentes copies de la molécule d'ADN et ainsi d'aligner les blocs de données de chaque copie. Les molécules déchiffrées sont séquencées puis traitées avec un algorithme bio-informatique pour retrouver une séquence consensus. Celle-ci est décodée en séquence binaire, puis déchiffrée avec la clef du crypto-système, ce qui permet de récupérer les données binaires stockées par l'Utilisateur.

Tels que décrits, DNACipher est un encodage numérique et DNADecipher est un processus biologique. Nous les précisons ci-après.

### 3.2.2 Encodage numérique

#### 3.2.2.1 Fonctions numériques inverses des fonctions biologiques

Notre solution de sécurité repose sur une étape de chiffrement numérique DNACipher, qui produit plusieurs séquences chiffrées $D_{enc_i}$ à partir d'une séquence ADN $D$. Pour cela, la séquence $D$ est modifiée avec les fonctions numériques inverses des Rotation et Permutation biologiques présentées en Sections 3.1.7 et 3.1.8. Ces fonctions inverse sont appelées $iRot$ et $iPermut$. Elles reposent sur le changement de positions de blocs dans les séquences ADN, et l'ajout de sites de restrictions qui permettent le déchiffrement biomoléculaire.

La fonction numérique $iRot$ est présentée en Figure 3.26A. La fonction numérique de rotation inverse est donc définie de la façon suivante :

$$D_{enc} = iRot(D, d_2, d_3, R_1, R_2)$$

avec $D$ la séquence ADN telle que $D = d_1d_2d_3d_4$, $d_2$ et $d_3$ les blocs à échanger, $R_i$ des sites de restriction dont la coupe créé des extrémités saillantes compatibles, et $D_{enc}$ la séquence $D$ avec un échange des positions des blocs $d_2$ et $d_3$, encadrés par les sites $R_1$ et $R_2$.

Ainsi, la séquence obtenue $D_{enc}$ pourra être déchiffrée via la fonction biomoléculaire de Rotation $Rot(D_{enc}, R_1, R_2)$ présentée en Figure 3.26B.

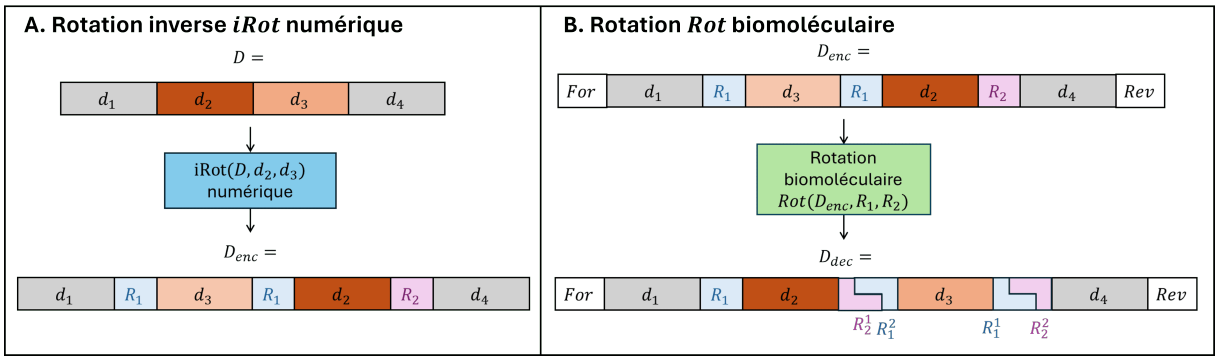



FIGURE 3.26 – Fonction de rotation de deux blocs dans une molécule d'ADN. A. Rotation inverse $iRot$ numérique. B. Rotation $Rot$ biomoléculaire.

De même, la fonction de permutation inverse $iPermut$ est présentée en Figure 3.27A. Elle permet d'échanger des blocs de données entre deux séquences ADN $D_1$ et $D_2$. La fonction numérique de permu-

tation inverse est donc présentée de la façon suivante :

$$(D_{iperm_1}, D_{iperm_2}) = iPermut(D_1, D_2, d_2^1, d_2^2, R_1, R_2)$$

avec $D_1$ et $D_2$ des séquences ADN, $d_2^1$ et $d_2^2$ les derniers blocs de $D_1$ et $D_2$ respectivement, $R_1$ et $R_2$ des sites de restriction dont la coupe créé des extrémités saillantes compatibles, et $(D_{iperm_1}, D_{iperm_2})$ les séquences ADN obtenues par permutation des blocs $d_2^i$ de $D_1$ et $D_2$.

Ces séquences peuvent être déchiffrées par la permutation biologique $Permut(D_{iperm_1}, D_{iperm_2}, R_1, R_2)$ présentée en Figure 3.27B.

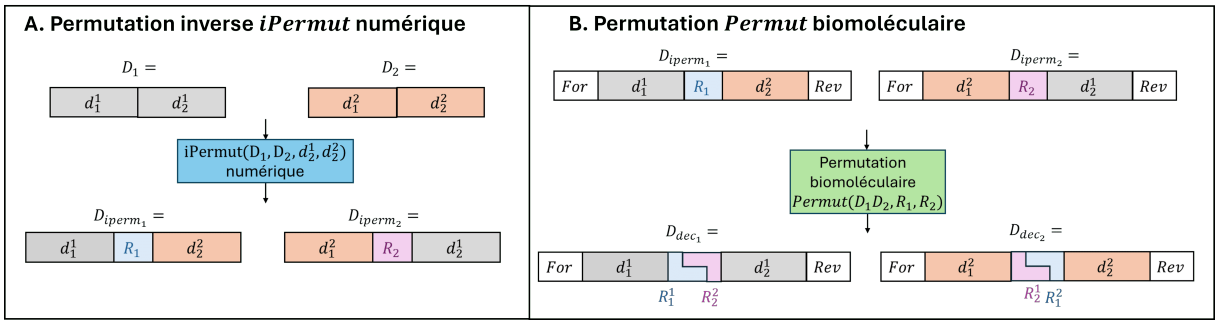



FIGURE 3.27 – Fonction de permutation de deux blocs de deux molécules d'ADN $D_1$ et $D_2$. A. Permutation inverse $iPermut$ numérique. B. Permutation $Permut$ biomoléculaire.

### 3.2.2.2 Processus de chiffrement des données

Le processus de chiffrement de la séquence $D$ repose sur les fonctions numériques $iRot$ et $iPermut$ présentées au paragraphe précédent.

La séquence ADN $D$ en entrée du chiffrement DNACipher a été préalablement encodée avec l'algorithme DSWE présenté dans le Chapitre 2. Ce codage est réalisé en considérant comme motifs interdits l'ensemble des sites de restriction $R_i$ utilisés pour les fonctions biomoléculaires $Rot$, $Permut$, ainsi que les résultats de ligations dans les fonctions $Rot$ et $Permut$ : $R_1^i R_2^j$. Ces motifs particuliers sont retrouvés dans les séquences ADN déchiffrées après Rotation ou Permutation biologiques, comme nous pouvons le voir dans les Figures 3.26B et 3.27B.

A l'issue de l'encodage DSWE, une séquence de bases ADN $D = B_1, ..., B_n$ de $n$ bases ADN est obtenue. Cette séquence est divisée en $m$ blocs de données de longueur variable $\{d_i\}_{i=1..m}$, pour obtenir la séquence $D = d_1 ... d_m$.

Le chiffrement numérique DNACipher est défini de la façon suivante :

$$\{D_{enc_i}\} = DNACipher(D, \{R_i, R_j\})$$

avec : $\{D_{enc_i}\}$ les séquences ADN chiffrées par différentes fonctions numériques $iRot$ et $iPermut$ ; et, $(\{R_i, R_j\})$ des couples de sites de restrictions qui vont venir encadrer certains blocs $d_i$ et qui seront utilisés par les opérateurs biologiques pour par exemple couper une molécule en créant des extrémités saillantes compatibles.

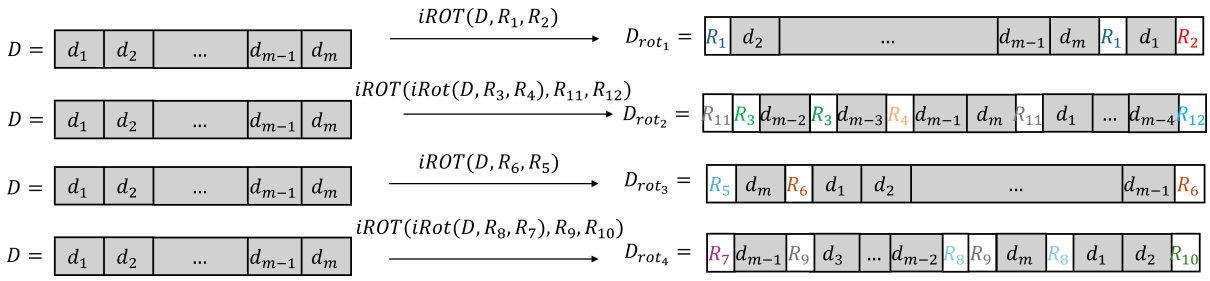



FIGURE 3.28 – Représentation schématique du chiffrement d'une séquence ADN $D$ en multiples séquences avec rotation de certains blocs $d_i$. Dans cet exemple, une rotation inverse $iRot$ différente est appliquée à la séquence $D$ pour obtenir chaque ligne de la matrice de droite.

DNACipher repose sur deux étapes :

1. La première étape de notre protocole consiste à appliquer différentes fonctions de rotation inverse $iRot$ à plusieurs copies de la séquence $D$. Chaque application de $iRot$ permet de modifier la position de deux blocs de données adjacents dans une séquence. Un nombre variable de $iRot$ peut être appliqué à chaque copie de $D$. Les séquences obtenues après les applications de $iRot$ contiennent chacun des blocs $d_i$ de données exactement une fois, dans des ordres différents. Les séquences $D_{rot_i}$ produites sont donc toutes de même taille. Un exemple est présenté en Figure 3.28 où nous avons appliqué une ou plusieurs rotations à différentes copies $D_{rot_i}$ de $D$. On peut voir que pour chaque séquence chiffrée, les positions de deux blocs adjacents pourront être échangées et que pour cela trois sites de restriction ont été ajoutés pour encadrer ces blocs. Par exemple, à la première ligne, la fonction $iRot(D, R_1, R_2)$ est appliquée : les positions des blocs $d_1$ et $(d_2, ..., d_m)$ sont échangées, et des sites de restriction $R_1$ et $R_2$ encadrent ces blocs échangés. A la deuxième ligne, deux rotations successives sont effectuées : la transformation $iRot(iRot(D, R_3, R_4), R_{11}, R_{12})$ est appliquée.

   Nous pouvons considérer cette étape de chiffrement comme une opération sur chaque ligne de la matrice de consensus. Elle modifie l'ordre des blocs de données de la molécule $D$ qui seront réordonnés lors de l'étape de déchiffrement biomoléculaire.

2. La deuxième étape consiste à appliquer différentes permutations inverses $iPermut$ aux séquences $D_{rot_i}$ pour échanger les derniers blocs de données de deux séquences $D_{rot_j}$ et $D_{rot_k}$. Cela est équivalent à effectuer des opérations entre les lignes de la matrice de consensus, comme illustré en Figure 3.29. Dans cet exemple, les séquences des lignes 1 et 2 échangent des blocs de données. On peut voir que ces blocs peuvent être de tailles différentes. A noter que pour que la permutation biologique soit possible, un site de restriction est ajouté avant les blocs permutés. Dans le cas des séquences $D_{enc_1}$ et $D_{enc_2}$, ce sont les sites $R_{13}$ et $R_{14}$. Grâce à ces permutations, les séquences chiffrées ne contiennent pas tous les blocs de données $d_i$. C'est le cas par exemple de la séquence $D_{enc_1}$ qui ne contient pas les blocs $d_1$ et $d_m$, ni le site de restriction $R_2$ nécessaire à sa rotation. Toutefois, elle contient deux fois l'ensemble de blocs $(d_2...d_{m-4})$. Cela rend d'autant plus difficile le processus de consensus si les opérations biomoléculaires ne sont pas faites avant le séquençage. En effet, même si un attaquant réussit à isoler une molécule chiffrée, il ne dispose pas de tous les blocs de données.

Lors du chiffrement DNACipher, chaque fonction numérique de rotation et de permutation ajoute des

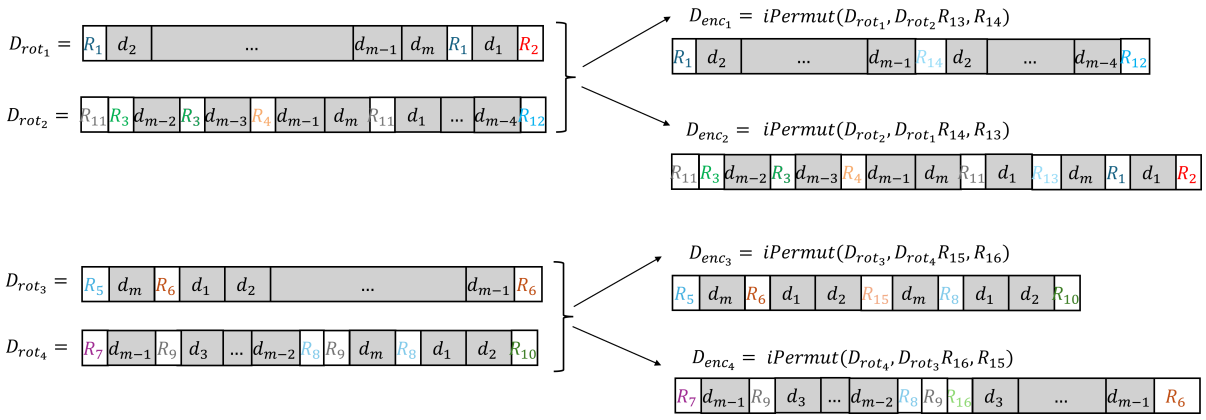



FIGURE 3.29 – Représentation schématique du chiffrement numérique de séquences ADN par permutations inverses $iPermut$ de blocs entre les différentes séquences $D_{rot_i}$ ; séquences qui sont préalablement chiffrées par rotations inverses. Dans cet exemple, les deux premières lignes $D_{rot_1}$ et $D_{rot_2}$ échangent leurs derniers blocs, et des sites de restriction $R_{13}$ et $R_{14}$ sont ajoutés à $D_{enc_1}$ et $D_{enc_2}$ pour permettre le déchiffrement biomoléculaire. De la même façon, les derniers blocs des troisièmes et quatrièmes lignes $D_{rot_3}$ et $D_{rot_4}$ sont échangés, et les sites $R_{15}$ et $R_{16}$ sont ajoutés à $D_{enc_3}$ et $D_{enc_4}$.

sites de restriction dans la séquence qu'elle chiffre. Pour garantir un alignement des séquences déchiffrées et donc un consensus, toutes les séquences déchiffrées doivent contenir les recombinaisons de sites de restrictions obtenues après l'application de toutes les fonctions biologiques de rotation et de permutation. C'est pourquoi nous devons ajouter ces recombinaisons de sites de restriction aux séquences chiffrées. Explicitons quelles sont les recombinaisons de sites ajoutées dans le cas de la rotation. Lors du chiffrement, la fonction numérique $iRot$ ajoute deux sites de restriction $R_1$ et un site $R_2$ à une séquence ADN. La fonction $Rot$ biologique modifie ces sites. On retrouve dans les molécules déchiffrées des recombinaisons de motifs suivantes : $R_1$ ainsi que $R_1^1R_2^2$ et $R_1^2R_2^1$. Ce sont ces recombinaisons qui doivent être ajoutées aux séquences qui ne sont pas chiffrées par la rotation $iRot$, pour permettre l'alignement des séquences déchiffrées. Un cas particulier existe pour le motif résiduel $R_1$. Il n'est pas possible de l'ajouter à toutes les séquences, elles seraient sinon toutes coupées par l'enzyme $E_1$. Nous utilisons donc un motif $R_1'$, très proche de $R_1$, qui permet l'alignement des séquences. Dans le cas de la permutation, la fonction $iPermut$ ajoute un site $R_3$ à une séquence ADN et un site $R_4$ à une autre. La fonction biologique $Permut$ les recombine en motifs $R_1^3R_2^4$ et $R_1^4R_2^3$. Pour permettre l'alignement des séquences déchiffrées, ces recombinaisons sont ajoutées à toutes les séquences non chiffrées par cette permutation.

Notre chiffrement permet donc d'obtenir plusieurs séquences chiffrées de tailles variables, contenant chacune certains des blocs $d_i$ de la séquence originelle $D$ dans le désordre. Cette solution de chiffrement est avant tout exploratoire et, en termes de sécurité, n'est pas au niveau d'un chiffrement AES par exemple. Nous pouvons cependant imaginer son utilisation pour un passage à l'échelle de milliers de molécules. Nous en proposons une première version basée sur des opérateurs biologiques bien connus.

Après cette étape de chiffrement numérique, le couple d'amorces $For_D$ et $Rev_D$ est déterminé, il encadre toutes les séquences chiffrées $D_{enc_i}$. Les amorces sont choisies selon un grand nombre de paramètres, notamment le fait qu'elles ne s'hybrident avec aucunes des séquences qu'elles encadrent. Une fois les amorces placées aux extrémités des séquences, le processus d'encodage numérique est terminé. Les séquences sont alors prêtes à être synthétisées. Dans la section suivante, nous présentons le processus de

**Algorithm 3:** Chiffrement numérique DNACipher de la molécule d'ADN $D$

**Inputs :** Une séquence d'ADN $D$ de $m$ blocs de bases ADN $d_i$ de tailles variables, tel que $D = d_1...d_m$, un ensemble de sites de restriction $\{R_i\}$ et leurs enzymes $\{E_i\}$.

**Output :** Plusieurs séquences d'ADN $D_{enc_i}$ de tailles différentes, avec des blocs $d_i$ (pas forcément tous, pas forcément qu'une fois).

---

1. $\{D_{rot_1}\} = iRot(D, d_1, (d_2..d_m), R_1, R_2)$ ;
% *Rotation inverse de deux blocs adjacents $d_2$ et $(d_2...d_m)$, et ajout des sites $R_1$ et $R_2$ pour encadrer les blocs à échanger.* ;
2. $\{D_{rot_2}\} = iRot(iRot(D, (R_3...d_m), (d_1...d_{m-4}), R_{11}, R_{12}), d_{m-2}, d_{m-3}, R_3, R_4)$ ;
% *Chiffrement d'une séquence ADN par application de deux rotations inverses successives.* ;
3. $\{D_{rot_3}\} = iRot(D, d_m, (d_1...d_{m-1}), R_5, R_6)$ ;
% $4.\{D_{rot_4}\} = iRot(iRot(D, (d_3...d_{m-2}), (d_m...d_2), R_9, R_{10}), d_m, (d_3...d_{m-1}), R_7, R_8)$ ;
% $5.\ \{D_{enc_1}, D_{enc_2}\} = iPermut(D_{rot_1}, D_{rot_2}, d_{m-1}, d_1, R_{13}, R_{14})$ ;
% *Permutation de blocs entre deux molécules $D_{rot_1}$ et $D_{rot_2}$. Tous les blocs situés après $d_{m-1}$ dans $D_{rot_1}$ sont échangés avec les blocs situés après $d_1$ dans $D_{rot_2}$, et les sites de restriction $R_{13}$ et $R_{14}$ sont ajoutés avant ces blocs permutés.* ;
6. $\{D_{enc_3}, D_{enc_4}\} = iPermut(D_{rot_3}, D_{rot_4}, d_2, d_{m-2}, R_{15}, R_{16})$ ;

déchiffrement biomoléculaire, qui repose sur les fonctions biologiques de rotation et de permutation.

### 3.2.3 Déchiffrement biomoléculaire

Pour récupérer les données stockées sur une molécule $D$, le processus commence par une PCR qui permet de multiplier exponentiellement le nombre de différentes molécules $\{D_{enc_i}\}$ ; les différentes copies de $D$ qui ont subie des permutations et des rotations ; identifiées par les mêmes amorces $For_D$ et $Rev_D$. Le déchiffrement biomoléculaire DNADecipher est ensuite appliqué aux molécules $D_{enc_i}$. Celui-ci repose sur les fonctions de Rotation et de Permutations de blocs dans les molécules présentées en Sections 3.1.7 et 3.1.8, respectivement. Pour chaque fonction, il faut connaître les deux enzymes de restriction qui permettent de couper les sites de restriction. Toutefois, il n'est pas nécessaire de connaître les positions ou la taille des blocs échangés.

Le déchiffrement biomoléculaire s'effectue comme suit :

1. Les permutations de blocs entre plusieurs molécules sont effectuées pour recomposer chaque séquence $D_{rot_i}$, qui contient exactement une fois chaque bloc $d_i$ de la séquence à déchiffrer. Ces permutations correspondent à des échanges entre les lignes de notre matrice de concensus et sont effectuées l'une après l'autre dans la solution contenant toutes les molécules $D_{enc_i}$.
2. Les rotations sont effectuées pour obtenir des molécules $D$ à partir des $D_{rot_i}$. L'ordre d'application des rotations $Rot$ est à respecter si une molécule $D_{rot_i}$ est le résultat de plusieurs fonctions $iRot$. En effet, un bloc de données ou une partie de celui-ci peut être la cible de deux ou plus rotations.

Dans ce processus de déchiffrement, les molécules $D_{enc_i}$ sont toutes mélangées dans une même solution. L'application des fonctions biomoléculaires les affectent donc toutes et il est alors important de s'assurer que les opérateurs biologiques ne modifient pas les structures des molécules qui ne les concernent pas. Dans le cas de la fonction de Rotation, les positions de deux blocs adjacents d'une molécule $D_{rot_1}$ sont échangés avec 5 opérateurs biologiques différents. Nous nous sommes assurés lors de l'encodage avec notre algorithme DSWE que $D_{rot_1}$ est la seule molécule qui contient les sites de restriction $R_1$ et $R_2$

de manière à ce que les étapes de coupe selon ces sites n'impactent que $D_{rot_1}$. De même, les étapes de ligation d'extrémités saillantes n'affectent pas les molécules avec des extrémités franches. Toutes les molécules $D_{rot_i}$ sont modifiées par les PCR, qui multiplient le nombre de copies et ajoutent des biotines à certaines extrémités de l'ADN. Ainsi, les molécules $D_{rot_i}$ sont séparées selon leurs extrémités, mais elles ne sont jamais coupées, elles se retrouvent donc toutes dans le tube final en fin de protocole, sans avoir changé de structure. La fonction $Permut$ repose sur un opérateur supplémentaire, la déphosphorylation, qui empêche la ligation. Comme les molécules autres que $D_{rot_1}$ ne sont jamais coupées, elles ne sont pas affectées par cet opérateur.

A l'issue du déchiffrement biomoléculaire, les molécules ADN sont donc toutes des copies de $D_{dec}$ et peuvent être séquencées et exploitées par le processus de consensus. Les blocs de données $d_i$ sont présents dans chaque molécule une unique fois.

Pour la lecture, les molécules sont amplifiées par PCR selon les amorces $For_D$ et $Rev_D$, afin d'obtenir plus de molécules à lire. Pour garantir le bon fonctionnement de la PCR, il est important de connaître la taille de la molécule à amplifier. Le résultat de la PCR est séquencé, et un algorithme bio-informatique permet de retrouver une séquence consensus.

Pour récupérer l'information, la séquence ADN $D_{dec}$ doit encore être traitée numériquement pour supprimer les recombinaisons de motifs n'encodant pas de données. Ce sont les sites de restriction $R_i$ et les concaténations de sites de la forme $R_1^i R_2^j$. Nous donnons un exemple en Figure 3.30, où les molécules sont déchiffrées avec deux Rotations différentes. Après déchiffrement, chaque molécule contient des motifs résiduels $R_i$ de différentes couleurs, qui sont identifiés et retirés numériquement. Les blocs de données $d_i$ représentés en gris sont retrouvés et l'algorithme DSWE peut être appliqué pour récupérer les données. Pour ce faire, celui-ci doit reprendre en entrée les motifs interdits et la taille maximale d'homopolymères. Enfin, la suite binaire est déchiffrée avec le crypto-système choisi et sa clef numérique.

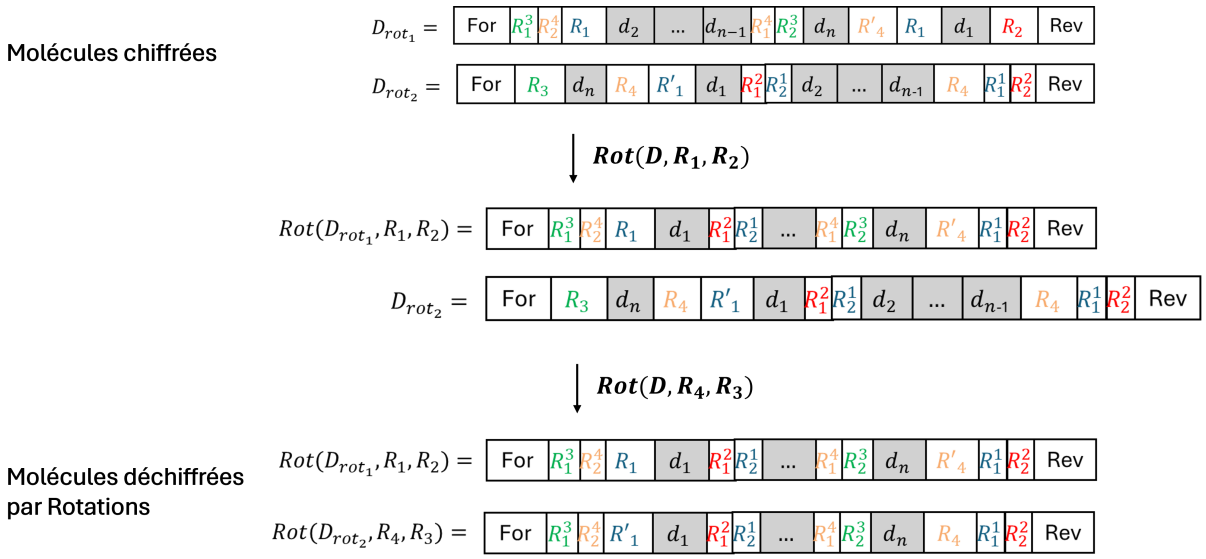



FIGURE 3.30 – Molécules déchiffrées par les rotations $Rot(D, R_1, R_2)$ et $Rot(D, R_4, R_3)$, permettant d'aligner les blocs de données $d_i$.

La clef de sécurité pour retrouver l'information stockée dans les molécules d'ADN est donc autant

biologique que numérique. En effet, côté biologique il faut connaître les enzymes à utiliser pour chaque fonction biomoléculaire de Rotation et de Permutation, les amorces encadrant l'ADN, la taille de la molécule déchiffrée. Pour le déchiffrement numérique, les clefs sont l'ensemble des motifs interdits (sites de restrictions et leurs concaténations), les paramètres d'encodage de DSWE, et la clef du crypto-système numérique.

Nous avons donc présenté comment récupérer l'information stockée dans des molécules d'ADN, en effectuant un déchiffrement biomoléculaire. Celui-ci s'appuie sur deux fonctions que nous avons développé à partir d'opérateurs biologiques élémentaires. Cet algorithme est une première solution exploratoire, dont nous évaluons la complexité dans la section suivante.

### 3.2.4 Complexité et analyse de sécurité

Lors de l'évaluation de la complexité, il est important de prendre en compte le fait que nous manipulons des données biologiques, qui diffèrent grandement des données numériques. Tout d'abord, la confidentialité des données est renforcée par le fait qu'un accès physique est nécessaire pour récupérer des données biologiques. Au contraire, certaines données numériques sont accessibles à distance via le Cloud, par exemple. Ensuite, la lecture d'ADN via le séquençage est une méthode destructive : les données biologiques ne peuvent être lues qu'une fois. Il est important de le savoir, car le nombre de manipulations possibles sur des molécules d'ADN est limité, contrairement au support numérique. Si trop de manipulations sont effectuées, alors les erreurs s'accumulent et l'ADN prend trop de formes différentes. Il devient impossible de lire la molécule souhaitée. Lorsque la sécurité repose sur un déchiffrement biomoléculaire, il n'est donc pas toujours possible de tester en brute force toutes les opérations possibles. En particulier, on ne peut pas copier un nombre infini de fois de l'ADN car le taux d'erreur de la PCR est de l'ordre de $10^{-4}$ à $10^{-7}$ erreurs par base lors d'un cycle [62], sachant qu'une PCR contient généralement 20 à 35 cycles. A partir d'un échantillon d'ADN, un nombre fini de PCR peut être effectué avant de causer trop d'erreurs. De même, il est important de savoir qu'une opération dans une solution affecte toutes les molécules présentes. Il peut donc y avoir des interactions entre différents fichiers stockés dans la même solution. Enfin, les manipulations élémentaires biologiques nécessitent d'importants coûts, temps et expertises en biologie moléculaire. Par exemple, le protocole pour une seule fonction de rotation que nous présentons dans la partie expérimentale nécessite plusieurs jours de manipulations en laboratoire et un coût de plusieurs centaines d'euros. Il n'est donc pas possible d'imiter un algorithme de chiffrement numérique en biologique.

En revanche, nous pouvons utiliser des opérations numériques élémentaires telles que la rotation et la permutation en version biomoléculaire, qui ont une faible complexité numérique mais qui sont sont bien plus complexes dans le domaine biologique. Pour un algorithme de déchiffrement biomoléculaire basé sur les rotations et permutations, la clef de déchiffrement est physique : ce sont des enzymes, des instructions de manipulations biologiques comme des températures et des durées de manipulation, ainsi qu'un savoir-faire. Nous évaluons la complexité de notre solution selon la difficulté à retrouver l'information stockée dans l'ADN sans avoir la clef de chiffrement. Nous suivons ainsi le principe de Kerckhoffs [173] qui stipule que la sécurité d'un crypto-système doit reposer sur le secret de la clef, et non le secret du système lui-même.

Évaluons la complexité de nos fonctions de rotation et de permutation, qui utilisent chacune deux

sites de restriction différents et deux enzymes de restriction associés. Leur complexité repose sur plusieurs critères : Le nombre de couples d'enzymes de restriction possibles, et le nombre de positions possibles où sont placés leurs sites. Concernant la position de tels sites, notre chiffrement a peu de contraintes. Contrairement à un algorithme numérique tel que le chiffrement AES (*cf.* Section 2.2 du Chapitre 2), où les blocs sont de taille connue et fixe, la taille des blocs dans notre chiffrement n'est pas fixée. Les blocs $d_i$ d'une molécule d'ADN peuvent être de tailles variables, tant qu'ils font plus de 50 bases. Cela garantit que les petits blocs ne soient pas jetés avec d'autres contaminants lors d'étapes de purifications de l'ADN. Cette taille variable apporte de la flexibilité au chiffrement : si une fonction telle que la rotation créé des motifs interdits, la taille des blocs peut être modifiée.

Prenons l'exemple de molécules d'ADN commandées chez le fournisseur IDT (Integrated DNA Technologies). Il est possible de commander de l'ADN double brin de 125 à 3000 paires de bases, appelé gBlocks™ [55]. Des rotations peuvent être effectuées avec n'importe quels blocs de plus de 50 bases tant qu'ils sont adjacents, ce qui apporte une beaucoup de possibilités. Considérons la séquence ADN $D = (d_1 d_2 d_3 d_4)$ de 4 blocs de taille variable. La rotation se fait entre les blocs $d_2$ et $d_3$, car les blocs $d_1$ et $d_4$ sont forcément concaténés aux amorces, trop petites pour former un bloc à elles seules. La seule restriction est que chaque bloc doit faire au moins 50 bases. Prenons le cas où la séquence totale est de 3000 bases. Alors chaque bloc peut contenir entre 50 et 2850 bases. Le nombre de rotations possibles entre deux blocs adjacents est de $2800^4$, c'est-à-dire de l'ordre de $10^{11}$ possibilités.

Ainsi, si l'on récupère des séquences ADN chiffrées, étant donnée la flexibilité en terme de taille de blocs de données, il est complexe de retrouver quels blocs ont été déplacés. Toutefois, les fonctions de rotation ne sont pas parfaites, des motifs particuliers encadrent les blocs échangés. C'est pourquoi la complexité de nos fonctions repose aussi sur le nombre de couples de sites de restriction à disposition.

Parmi les 667 enzymes identifiés par le fournisseur New England Biolabs (NEB) [138], 535 créent des extrémités saillantes (de taille 1 à 5 bases). Pour optimiser les ligations d'extrémités saillantes compatibles, nous choisissons des extrémités de 4 bases. Pour les fonctions de rotation et de permutation, nous utilisons en particulier deux enzymes qui créent les mêmes extrémités. Il existe 62 couples d'enzymes qui créent des extrémités de 4 bases compatibles, ils sont détaillés en Annexe C. Par exemple, l'extrémité saillante côté 5' ($CTAG$) peut être créée par les 4 enzymes, AvrII, NheI, SpeI ou XbaI, qui reconnaissent les sites de restriction ($CCTAGG$), ($GCTAGC$), ($ACTAGT$) et ($TCTAGC$), respectivement. L'ensemble des enzymes de restriction qui créent des extrémités saillantes compatibles avec d'autres enzymes est trouvable dans le tableau de sélection [174] du fournisseur NEB, il en existe actuellement 76. Il est aussi possible d'effectuer les fonction de rotation avec des enzymes de restriction de type IIS, qui repèrent un site de restriction et coupent à côté de ce site. Avec ces enzymes, il est possible de choisir l'extrémité saillante créée. En particulier, 32 enzymes de type IIS différents créent des extrémités saillantes de 4 bases, ils peuvent toutes être utilisés pour des fonctions de rotation ou de permutation, ce qui correspond donc à 496 couples d'enzymes possibles.

Il existe donc plusieurs centaines de couples d'enzymes pouvant être utilisés pour chaque fonction de rotation ou de permutation. Pour un algorithme numérique, ce chiffre n'apporte pas une sécurité suffisante. Avec une analyse bio-informatique, il est possible de retrouver quels sites de restriction sont présents dans chaque séquence ADN chiffrée, et de tester les différentes permutations et rotations possibles. Toutefois, la sécurité de notre chiffrement repose sur la difficulté à manipuler l'ADN lorsqu'un attaquant ne connaît

pas les sites de restriction utilisés.

Nous pouvons imaginer plusieurs scénarios où un attaquant souhaite récupérer les données, avec plusieurs niveaux de connaissance et de discrétion :

— Si l'attaquant connaît les amorces, il peut tenter une PCR. Elle multiplie en théorie toutes les molécules encadrées par les primers, toutefois en pratique la PCR amplifie plus les molécules de petite taille. Les longues molécules se retrouvent donc en minorité. Il peut essayer de séquencer avec plusieurs technologies différentes, sachant que certaines technologies séquencent des molécules de quelques centaines de bases maximum. Pour bloquer le séquençage, nous avions envisagé une solution de sécurité reposant par exemple sur l'obstruction des extrémités de l'ADN avec une biotine. Ainsi, le séquençage nanopore serait impossible car l'ADN biotinylé ne peut pas passer par le nanopore. Il est possible de bloquer ainsi l'ADN, toutefois il existe peu de biotines qui pourraient remplir ce rôle. Un tel blocage est donc facilement identifiable, et une biotine peut être séparée de l'ADN par une simple modification chimique. Notre solution de sécurité DNACipher n'empêche donc pas la lecture de l'ADN. Dans la section expérimentale, nous allons montrer que notre solution permet de bloquer la bonne réalisation de l'étape bio-informatique de consensus, qui suit le séquençage.

— Une autre option pour un attaquant est de partitionner la solution et de tester plusieurs manières de déchiffrer les données. C'est une attaque brute force, qui a un coût temporel et financier très important. Dans notre scénario, plusieurs centaines de couples d'enzymes sont possibles pour chaque fonction. Chaque test est coûteux et nécessite plusieurs jours de manipulations. De plus, il y a un risque de corrompre les données biologiques en multipliant les amplifications par PCR pour avoir assez de matériel. En effet, chaque opération de PCR a un taux d'erreur non nul. Dans ce scénario, les données stockées sur quelques kilo-octets ont une valeur très importante, qui justifient de tels efforts, telles qu'une clef cryptographique ou un numéro de transaction blockchain.

  Dans le cas de notre solution, lorsque le protocole de déchiffrement est effectué sur un échantillon des données avec la mauvaise clef, cela modifie de façon inattendue et donc potentiellement irréparable les données. En effet, notre protocole de chiffrement interdit seulement les sites de restriction utilisés par nos fonctions de rotation et de permutation. Il existe beaucoup d'autres sites de restriction, qui peuvent être présents dans les molécules. Cela complique la lecture voir la rend impossible. De plus, une telle manipulation prend du temps et coûte du matériel.

Il est important de noter que la solution de sécurité que nous proposons est partielle, il est possible de la complexifier encore, notamment avec des fonctions qui permutent plus de blocs et utilisent d'autres enzymes.

Dans la deuxième partie de ce chapitre, nous avons présenté une solution de sécurité dont le déchiffrement repose sur des opérateurs biologiques. Celui-ci s'appuie sur deux fonctions que nous avons développé à partir d'opérateurs biologiques élémentaires. La fonction de rotation permet d'échanger la position de deux blocs dans une molécule, tandis que la fonction de permutation échange des blocs entre deux molécules d'ADN. A partir de ces fonctions, nous avons développé une solution de sécurité DNACipher, qui impose des manipulations biologiques pour déchiffrer les données stockées dans l'ADN avant de pouvoir les séquencer. Pour cela, notre solution de chiffrement repose sur des rotations et permutations différentes appliquées à des copies de notre séquence ADN. Son but est de perturber l'étape bio-informatique de

consensus qui suit le séquençage. Pour retrouver la séquence consensus et donc les données chiffrées, il faut déchiffrer toutes les copies chiffrées via un déchiffrement biomoléculaire DNADecipher, qui applique les opérations inverses de rotation et de permutation. Cet algorithme est une première solution exploratoire, dont nous la sécurité peut être améliorée. Dans les sections qui viennent, nous présentons plusieurs résultats expérimentaux qui, à partir de simulations, montrera l'impact de notre solution sur l'algorithme de consensus et , à l'aide d'un protocole biomoléculaire, montrera qu'il est possible de faire une opération de rotation sur une molécule d'ADN.

## 3.3 Résultats expérimentaux

Dans cette section, nous présentons dans une première partie des résultats issues de simulation démontrant comment notre approche perturbe le processus de consensus, comme désiré. Dans une seconde, nous présentons un protocole biologique exploratoire qui implémente la fonction de rotation présentée en Section 3.1.7, et discutons en même temps les résultats de ces manipulations.

### 3.3.1 Impact de DNACipher sur le processus de consensus

L'objectif de cette section est de quantifier la difficulté à retrouver les données stockées dans l'ADN, sans passer par l'étape de déchiffrement biomoléculaire DNADecipher. En effet, si un attaquant connaît les amorces qui encadrent les molécules chiffrées, il peut tenter une PCR et un séquençage, et faire l'analyse bioinformatique des résultats du séquençage.

Ici, les étapes biologiques sont simulées afin de nous permettre de faire un grand nombre de tests. Plus clairement, l'encodage se fait de façon classique. Les données binaires sont encodées en séquences ADN, et des fonctions numériques de rotation et permutation inverses sont appliquées. Les étapes biologiques de synthèse, stockage et séquençage sont simulées par un passage dans le même simulateur [42] déjà utilisé dans les chapitres précédents. En sortie de cette étape, un ensemble de copies ou "reads" d'une molécule d'ADN est traité par un algorithme de consensus bio-informatique pour déterminer une séquence ADN consensus.

Dans cette expérimentation, nous évaluons la difficulté à retrouver l'information sans déchiffrement biomoléculaire. Pour cela, nous testons le consensus seul, ainsi que l'ajout d'une étape de clustering avant consensus.

#### 3.3.1.1 Paramètres de tests

Le benchmark mis en œuvre est le suivant. Un texte aléatoire *lorem ipsum* de 95 Kbytes a d'abord été chiffré avec le cryptosystème AES-256, puis découpé en blocs binaires de 1024 bits, codés séparément en séquences ADN à l'aide de notre codeur DSWE. Les paramètres de DSWE sont les suivants : une taille maximale d'homopolymères $N = 3$ et un dictionnaire de 32 motifs interdits de taille 6 correspondant à des sites de restriction et des recombinaisons de sites.

A l'issue de cet encodage, chaque séquence $S$ est divisée en 10 blocs de 60 à 100 bases. La séquence $S$ est ensuite chiffrée selon l'algorithme 3 présenté en Section 3.2.2. Selon celui-ci, 4 séquences chiffrées $S_{rot_i}$ sont obtenues en appliquant une ou deux rotations inverses $iRot$ à $S$. Puis, deux permutations inverses

*iPermut* sont appliquées aux couples $(S_{rot_1}, S_{rot_2})$ et $(S_{rot_3}, S_{rot_4})$ respectivement. Rappelons que nous ajoutons à chaque séquence chiffrée plusieurs motifs pour permettre leur alignement lors du déchiffrement. En sortie du chiffrement numérique, nous obtenons 4 séquences chiffrées $S_{enc_i}$ de tailles variables (600 à 900 bases). Les sites de restrictions ajoutés pour les 6 rotations et 2 permutations nécessitent que 192 bases ne codent pas d'information, ce qui correspond à 25% des bases d'une séquence encodant 1024 bits. Les bases codant de l'information sont encodées avec DSWE, paramétré pour $N = 3$, $M = 32$ et $m = 6$. Ces 32 motifs sont causés par les 6 rotations et les 2 permutations qui seront appliquées aux séquences ADN. Par exemple, une rotation entraîne 4 motifs interdits de 6 bases : $R_1$, $R_2$, $R_1^1R_2^2$ et $R_1^2R_2^1$. Chaque motif entraîne lors de l'encodage DSWE une perte de $0.4\% = \frac{4}{4^5}$ du taux d'information. Avec 32 motifs interdits de 6 bases, la densité de l'information est conservée avec DSWE, le taux d'information de $1,82$ BPB. Après application de DNACipher, 75% des bases encodent de l'information, ce qui donne un taux d'information des séquences chiffrées de $\frac{75}{100} \times 1,82 = 1,36$ BPB.

Toutes les séquences obtenues sont ensuite encadrées des mêmes amorces, déterminées avec l'algorithme Ithos, un sous-module du logiciel de design d'amorces Genofrag [175], dont les nombreux paramètres sont détaillés en Annexe D.

Nous utilisons le simulateur [42] pour simuler les étapes de synthèse, de stockage et de séquençage. Il prend en entrée les séquences $\{S_{enc_i}\}_{i=1..4}$ et produit $X_i$ copies de $S_{enc_i}$ potentiellement entachées d'erreurs. Ces copies sont post-traitées avec la procédure de consensus de l'algorithme CCSA [78] en une séquence ADN reconstruite. Si trop de copies erronées sont fournies à l'algorithme de consensus, il peut décider de ne pas fournir de séquence ADN en sortie. En outre, l'algorithme peut fournir une séquence consensus erronée.

Dans les tests suivants, nous testons l'algorithme de consensus avec un nombre variable $X_i$ de copies de chaque séquence chiffrée $S_{enc_i}$, avec $X_i \in [20, 100]$. Nous comparons la séquence consensus obtenue aux séquences chiffrées, afin de voir si l'algorithme retrouve une séquence particulière. Nous tenons aussi compte du fait que toute opération biologique a une part d'incertitude, chaque molécule est présente en proportions différentes dans une solution. En particulier, la PCR amplifie plus efficacement les molécules de petite taille.

#### 3.3.1.2 Consensus sur des données chiffrées par rotations

Notre but est d'évaluer si l'algorithme de consensus CCSA peut retrouver une séquence consensus parmi plusieurs séquences chiffrées. Pour cela, nous partons du principe qu'un attaquant connaît les amorces qui encadrent les séquences chiffrées, et la longueur $L$ de la séquence déchiffrée. Si un attaquant retrouve une séquence chiffrée, il peut ensuite identifier les sites de restriction présents dans la séquence et faire une attaque brute force en déchiffrant toutes les rotations possibles en numérique, selon le modèle de la Rotation biomoléculaire présentée en Section 3.1.7.

Dans un premier test, nous avons fait varier le nombre $X_i \in [20, 100]$ de copies de chaque $S_{rot_i}$ issues du simulateur avant d'appliquer l'algorithme CCSA. Nous faisons varier les proportions de chaque séquence chiffrée $S_{rot_i}$, avec une différence maximale de 10 copies entre chaque séquence. Les séquences sont toutes de même longueur $L$, qui est la longueur de la séquence déchiffrée.

Lorsque les copies d'une seule séquence $S_{rot_i}$ de $500-600$ bases sont données en entrée de l'algorithme CCSA, 15 à 25 copies suffisent pour retrouver un consensus sans erreur.

Lorsque 2 séquences chiffrées par rotations $S_{rot_1}$ et $S_{rot_2}$ sont données en entrée de CCSA, l'algorithme retrouve à partir de 75 copies une des séquences chiffrées sans erreur. L'algorithme ne prend en compte que les copies d'une des deux séquences $S_{rot_i}$ et la retrouve sans erreur. Pour moins de 75 copies, un consensus inexact est retrouvé, c'est-à-dire une séquence $S_{rot_i}$ avec quelques substitutions, ou bien une séquence mélangeant des blocs de différentes séquences $S_{rot_i}$, avec des répétitions de $d_i$ et des sites de restriction absents.

Lorsque les copies $X_i$ de trois ou de quatre séquences différentes $S_{rot_i}$ sont prises en entrée de CCSA, il n'est pas garanti de retrouver une séquence consensus correspondant à une des $S_{rot_i}$ même avec 300 copies.

A l'issue de ce premier test, il apparaît que l'algorithme de consensus [78] permet de retrouver une séquence chiffrée parmi les reads de deux séquences de même taille chiffrées par rotation. Toutefois, pour plus de deux séquences chiffrées, il n'est pas garanti de retrouver un consensus.

Une fois une séquence $S_{rot_i}$ retrouvée par consensus, l'ensemble des sites de restriction présents dans la séquence peut être identifié et localisé, par exemple avec l'outil NEBCutter™ [176] comme présenté en Figure 3.31. L'exemple en Figure 3.31 donne les 46 sites présents une fois exactement dans une séquence chiffrée par rotation. Cette séquence contient aussi 10 sites présents 2 fois. Ces sites de restriction peuvent être combinés pour déchiffrer des rotations. En particulier, pour déchiffrer une rotation, nous recherchons les sites de restriction présents deux fois, caractéristiques de la fonction de Rotation, et tentons de leur associer un site présent une unique fois qui a des extrémités saillantes compatibles. Dans l'exemple en Figure 3.31, cela nous permet de retrouver le couple d'enzymes MluI (ACGCGT) et BssHII (GCGCGC), et d'inverser la rotation, qui est ainsi cassée en numérique. Malgré le grand nombre de sites de restriction naturellement présents dans une séquence ADN, il est donc possible de retrouver les sites utiles à une rotation biologique et de l'effectuer en numérique.

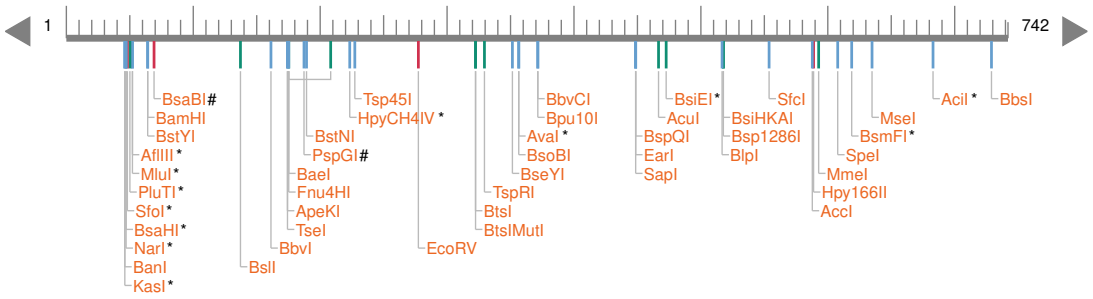



FIGURE 3.31 – Séquence $S_{rot_1}$ retrouvée avec l'algorithme CCSA [78] parmi des reads des séquences $S_{rot_1}$ et $S_{rot_2}$. Représentation avec l'outil NEBCutter™ [176] des sites de restriction présents une fois exactement dans la séquence.

#### 3.3.1.3 Consensus sur les séquences chiffrées par rotations et permutations

La permutation peut modifier la longueur des séquences, puisque le chiffrement peut permuter des blocs de longueurs différentes entre deux séquences. Une séquence chiffrée peut aussi contenir le même bloc à plusieurs positions. Lors de nos tests, nous avons permuté des blocs de 300 à 500 bases entre les séquences, ce qui nous a permis d'obtenir des séquences chiffrées $S_{enc_i}$ avec des longueurs pouvant

diverger de plusieurs centaines de bases. Or, l'algorithme de consensus nécessite de donner en entrée la longueur de la séquence recherchée. Avec notre solution de sécurité, nous ne connaissons pas les longueurs exactes des molécules chiffrées, car ce n'est pas une donnée utile. Notre algorithme de déchiffrement biomoléculaire nécessite seulement de connaître la longueur de la molécule déchiffrée pour récupérer les données. Nous avons donc effectué un test faisant varier la longueur indiquée en entrée de l'algorithme CCSA.

Lorsque les copies d'une seule séquence chiffrée $S_{enc_1}$ sont prises en entrée par l'algorithme CCSA, et que la longueur indiquée diffère de la longueur de la séquence recherchée, l'algorithme ne permet pas de retrouver $S_{enc_1}$, quel que soit le nombre de copies $X_1 \in [10, 200]$. La séquence consensus a quelques erreurs d'effacements ou d'insertion, ou des blocs désordonnés ou absents, et ne correspond pas à $S_{enc_1}$. Pour garantir un consensus correct, il faut connaître la longueur. Celle-ci peut être estimée à 10-20 bases près avec un dépôt sur gel (*cf* Section 3.1.6). Pour ce prochain test, nous partons du principe que nous connaissons la longueur d'une des séquences chiffrées. Nous avons tenu compte du fait que les molécules de petites longueurs sont mieux amplifiées par PCR. Ainsi, nos tests, la proportion des plus petites séquences est toujours au moins égale à celle des plus grandes.

Avec les copies de 2 séquences chiffrées, il est possible de retrouver une des séquences dans un cas : lorsque la longueur de la plus grande séquence est prise en paramètre d'entrée de CCSA, et que la longueur des séquences diffère de plus de 70 bases. En effet, l'algorithme CCSA ne prend pas en compte les copies plus petites que la longueur $L$ indiquée en entrée, et calcule donc le consensus à partir d'une majorité de copies de la longue séquence. Lorsque la différence de longueur entre les séquences est moindre, il est impossible de retrouver à coup sûr une des séquences en sortie du consensus, même avec 200 copies. Il en va de même lorsque l'on teste CCSA avec les copies de 4 séquences chiffrées $S_{enc_i}$, une séquence chiffrée exacte apparaît en sortie de CCSA à coup sûr seulement lorsque le paramètre donné est la taille de la séquence la plus longue, significativement plus longue que les autres (50 à 70 bases). Sinon, il est impossible de retrouver une séquence sans erreur à coup sûr, même avec 300 copies.

| | $S_{rot_i}$ | | | $S_{enc_i}$ | |
|---|---|---|---|---|---|
| Nombre de séquences chiffrées | 2 séquences | 3 séquences | 4 séquences | 2 séquences | 4 séquences |
| Nombre $X$ de copies minimum | 77 | plus de 300 | plus de 300 | 191 | plus de 300 |

TABLE 3.2 – Nombre de copies minimal nécessaire pour retrouver une séquence chiffrée consensus avec l'algorithme CCSA parmi les copies de plusieurs séquences.

L'algorithme CCSA ne permet donc de retrouver qu'une séquence chiffrée $S_{enc_i}$ sur plusieurs dans certaines conditions. Avec une seule séquence il n'est pas possible de déchiffrer une permutation, qui nécessite plusieurs séquences. D'autre part, comme montré dans la présentation du chiffrement, une séquence chiffrée par permutation peut ne contenir que certains blocs de données. Ainsi, déchiffrer une unique séquence $S_{enc_i}$ ne permet pas de retrouver l'information stockée dans les molécules d'ADN.

Il apparaît donc que le fait d'alimenter avec plusieurs séquences chiffrées identifiées comme étant des copies d'une seule et même molécule (les séquences ont les mêmes amorces) ne permet pas de faire aboutir le processus de consensus, comme nous le souhaitions. La rotation joue pour beaucoup mais la permutation est aussi intéressante car, pour mener à bien une attaque numérique, il est nécessaire d'obtenir les deux séquences chiffrées liées par la permutation pour déchiffrer et retrouver l'information.

Une possible contre-mesure est d'ajouter une étape d'analyse bioinformatique pour trier les séquences chiffrées avant de leur appliquer l'algorithme de consensus, ce que nous testons dans la section suivante.

#### 3.3.1.4 Clustering et consensus

S'il est difficile de retrouver les séquences chiffrées avec l'algorithme de consensus, une alternative est de le faire précéder d'une étape de clustering qui permet de grouper les séquences ADN par similarité. De cette manière l'attaquant peut appliquer le consensus sur des séquences très similaires.

Pour nos tests, nous utilisons l'algorithme bioinformatique MeShClust [177], qui repose sur de l'apprentissage automatique supervisé pour déterminer des centres de clusters et les séquences ADN qui sont similaires à ces centres. MeShClust se base sur le décalage moyen ("mean shift"), un algorithme de clustering qui agit par itérations pour regrouper chaque séquence ADN à un sous-ensemble de séquences proches. A l'issue du processus, les séquences suffisamment proches sont assignées à des clusters, déterminées par des séquences dites centres de cluster. La proximité d'une séquence à un centre de cluster est déterminée par son score d'identité, qui doit être supérieur à un certain seuil $I$ pour faire partie du cluster.

Nous utilisons les mêmes séquences que dans la section précédente, et appliquons l'algorithme MeShClust avec un score d'identité $I$ entre 0.7 et 0.8. Selon la similarité des séquences et le nombre de blocs $d_i$ alignés ou non, il est possible, en testant plusieurs scores d'identité, de former des clusters avec une très bonne précision. Avec un score d'identité adapté, à partir de 10 copies de chacune des 4 séquences $D_{enc_i}$, 2 à 4 clusters se forment. Chaque cluster est composé des copies de 1 ou 2 séquences $D_{enc_i}$, avec en moyenne 2 copies mal référencées pour un nombre de copies $X_{total} \in [10, 200]$. Il est alors possible de prendre les copies de chaque cluster séparément afin de leur appliquer une seconde étape de clustering plus fine. Le clustering avec les copies de deux séquences donne des résultats dépendant de la similarité entre les séquences : si elles sont trop similaires, un seul cluster se forme avec $80 - 100\%$ des reads d'une séquence, tandis que toutes les autres copies sont écartés. Si les séquences sont suffisamment différentes, des clusters avec un écart-type très faible se créent dès $10 - 15$ copies.

Une fois les clusters formés, l'algorithme CCSA est appliqué à chaque cluster pour retrouver chaque séquence chiffrée. Si la longueur de cette séquence n'est pas connue, il faut en tester plusieurs. Une séquence exacte n'est retrouvée que lorsque sa longueur $L$ est utilisée comme paramètre de CCSA. Une fois une séquence chiffrée déterminée par consensus, une étape d'analyse permet de déterminer les sites de restrictions présents. Le déchiffrement est alors similaire au déchiffrement moléculaire. Sans connaissance des sites de restriction utilisés, il faut tester toutes les combinaisons possibles. Tout d'abord, les sites de restriction présents une fois exactement sont identifiés dans chaque séquence, afin de déchiffrer les permutations. Il y en a généralement 20 à 50 dans une séquence de 600 bases. Pour déchiffrer une permutation, deux séquences doivent contenir chacune un site $R_1$ ou $R_2$, qui créent les mêmes extrémités lorsqu'ils sont coupés. Chaque permutation possible (généralement 2 à 10) entre des séquences doit être testée. A partir de ces résultats, toutes les rotations possibles dans une séquence doivent être testées. Pour cela, les sites présents exactement deux fois sont identifiés, et des sites présents une fois créant des extrémités complémentaires sont recherchés.

En conclusion, dans nos tests, à partir de 30 copies de chaque séquence chiffrée, si l'on connaît la taille exacte de chaque séquence, il est possible de retrouver un consensus exact dans chaque cluster. Toutefois,

si une taille erronée est donnée en entrée de l'algorithme de consensus, alors la séquence consensus aussi sera erronée.

Nous avons pu voir que notre solution de chiffrement permet donc de bloquer l'étape de consensus, et oblige un attaquant à passer par plusieurs étapes d'analyse bioinformatique des séquences chiffrées qui nécessitent de connaître la taille exactes des séquences chiffrées pour les récupérer.

De telles étapes nécessitent un nombre important de copies de chaque molécule, pour plusieurs raisons : tout d'abord, le clustering n'est pas toujours exact, il faut donc prévoir une marge d'erreur. D'autre part, un consensus doit être effectué pour chaque séquence chiffrée. Or, comme nous allons le voir dans les résultats de notre manipulation biologique, le contenu d'une solution n'est jamais pur. Une solution sensée être composée uniquement d'une séquence ADN contient en vérité beaucoup de formes alternatives composées de quelques blocs. C'est particulièrement vrai pour les molécules d'ADN obtenues par assemblage de petits fragments, une méthode commune chez les fournisseurs d'ADN synthétique pour créer de longues molécules d'ADN double brin. C'est le cas des gBlocks™ Gene Fragments [55] du fournisseur IDT que nous avons utilisé lors de notre manipulation biologique.

Notre solution de sécurité DNACipher, bien qu'exploratoire, permet donc d'empêcher la bonne réalisation de l'étape de consensus, et nécessite qu'un attaquant dispose de nombreuses copies de chaque séquence chiffrée ainsi que de sa taille précise, pour les isoler et les retrouver via plusieurs étapes bioinformatiques.

#### 3.3.1.5 Perspectives

Nous avons présenté une première solution de sécurité qui permet de bloquer l'étape de consensus, et repose sur un déchiffrement biomoléculaire. Il est possible de complexifier cette solution de plusieurs façons, en utilisant d'autres opérateurs présentés dans la Section 3.1 ou en modifiant nos fonctions biomoléculaires de Rotation et de Permutation.

Tout d'abord, nous pouvons ajouter aux séquences chiffrées des blocs de données $\overline{d_i}$ ne codant pas d'information. Le déchiffrement repose alors sur la coupe et suppression de ces blocs, afin d'aligner les blocs codants $d_1$ dans toutes les séquences. Nous pouvons aussi imaginer placer une amorce au milieu d'une séquence. L'amorce doit être déplacée à l'extrémité d'une molécule par une rotation. Ainsi, si une PCR est effectuée avant la rotation, elle n'amplifie qu'une partie de la molécule, et une partie de l'information est perdue.

Une autre solution consiste à agrandir l'alphabet des bases ADN. Cela peut se faire de plusieurs façons. Une première solution est l'utilisation des bases dégénérées : les copies d'une séquence ont des bases différentes à une position donnée. Le pourcentage de bases indique comment décoder. Par exemple, si 50 % des séquences ont une base A à la position $i$, et 50 % ont une base G à cette même position, alors nous déchiffrons en une nouvelle base R. Il est possible de commander de l'ADN avec des bases dégénérées auprès de fournisseurs tels que IDT [178], qui propose un code standardisé de bases dégénérées, ainsi que des bases avec des proportions à customiser. Dans notre solution de sécurité, les bases dégénérées ne sont pas repérables tant qu'elles ne sont pas alignées. Pour les aligner et ainsi les décoder, il faut d'abord effectuer des opérations de rotation et permutation. Ces bases dégénérées reposent sur le déchiffrement de toutes les séquences chiffrées, qui doivent être alignées pour obtenir le niveau de lecture choisi. Un inconvénient des bases dégénérées est leur sensibilité à la PCR, qui peut modifier les proportions de

chaque base à une position.

Une autre façon d'agrandir l'alphabet des bases ADN repose sur les modifications des bases ADN pour obtenir de nouvelles bases. Nous pouvons citer la méthylation, l'ajout d'un groupe méthyle à certaines bases ADN, qui transforme par exemple une base C en base mC. Des modifications de bases ADN sont présentées en Section 3.1.4, elles permettent d'obtenir les bases mC, hmC, mA, et U. La méthylation peut aussi servir à complexifier une fonction de rotation ou de permutation. Certaines enzymes de restriction sont sensibles à la méthylation, ce qui veut dire qu'ils ne reconnaissent que des sites de restriction avec des bases méthylées, ou bien non méthylées. Ainsi, les fonctions de rotation ou de permutations doivent inclure des étapes de méthylation ou de déméthylation pour que les enzymes coupent bien l'ADN.

Les modifications de bases présentées en Section 3.1.4 peuvent aussi s'apparenter à des substitutions. Par exemple, déméthyler une molécule d'ADN transforme toutes les bases mC en bases C. Nous pouvons ajouter une étape de déméthylation dans une rotation, lorsqu'un bloc de données est isolé. Seul ce bloc est déméthylé. A la lecture, certains blocs de données ont des bases mC et d'autres non. Cet opérateur permet d'avoir plusieurs niveaux de lecture dans une molécule d'ADN. Notons que les méthylations ne survivent pas à la PCR, les copies d'une molécule méthylée ne seront pas méthylées.

Plusieurs améliorations de notre solution sont donc possibles. Il est cependant important de pouvoir les tester expérimentalement. Comme nous le verrons dans la section suivante, les simulations ne reflètent pas la complexité des processus biologiques dans leur entièreté.

### 3.3.2 Preuve de concept biologique - Fonction de rotation

Dans cette section, nous présentons le protocole permettant de déchiffrer une molécule d'ADN chiffrée en numérique avec la fonction de rotation. La fonction de rotation est d'une grande complexité expérimentale et repose, comme nous l'avons vu en Section 3.1.7, sur de nombreux opérateurs donc autant de manipulations voire plus. En raison de cette complexité, nous avons choisi d'expérimenter le déchiffrement d'une unique rotation, et non la solution de sécurité DNADecipher globale qui consiste à appliquer différentes rotations et permutations à plusieurs copies chiffrées. Rappelons que ces opérateurs n'ont jusqu'à présent jamais été utilisés à des fins de sécurité. Les expérimentations qui suivent sont donc une toute première preuve de concept montrant qu'il est possible de les utiliser dans cette objectif. Dans ce chapitre, nous présentons la cinquième version du protocole de rotation qui est à ce jour la version la plus optimisée. Nous avons en effet rencontré un certain nombre de difficultés qui résultent de la variabilité de l'efficacité des réactions biochimiques selon leurs paramètres et les conditions expérimentales ; sans parler d'erreurs de manipulations, une seule peut suffire à conduire à l'échec du protocole. Il apparaît évident que plus le nombre d'étapes d'un protocole est important, plus les erreurs s'accumulent. Chaque étape du protocole que nous allons présenter est le fruit de nombreuses réflexions ; *i.e.* de l'analyse des raisons des erreurs et des échecs de certaines manipulations ; qui ne seront pas développées ici.

Cette section est divisée en trois parties. Nous présentons tout d'abord le protocole, avant de donner les résultats expérimentaux. Enfin, nous proposons des pistes d'améliorations.

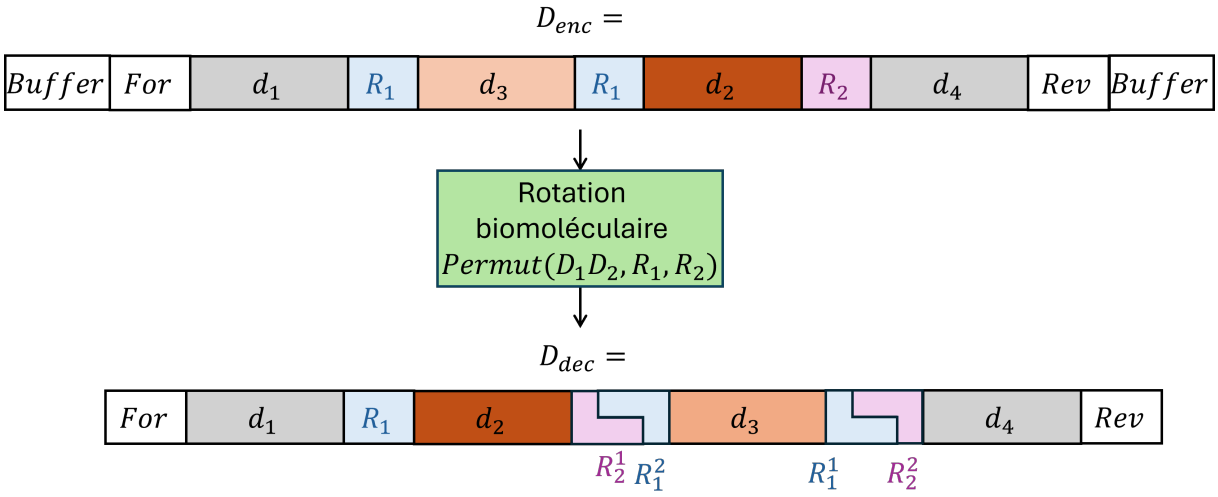



FIGURE 3.32 – Rotation biomoléculaire de la molécule $D_{enc}$ en une molécule déchiffrée $D_{dec}$, par l'échange des positions des blocs de données $d_2$ et $d3$.

#### 3.3.2.1 Objectif et étapes du protocole

L'objectif du protocole proposé est de déchiffrer la molécule d'ADN chiffrée $D_{enc}$ donnée en Figure 3.32. $D_{enc}$ est composée de 4 blocs de données $d_1$, $d_2$, $d_3$ et $d_4$ de tailles équivalentes, séparés par des sites de restriction $R_1 = (GCTAGC)$ et $R_2 = (ACTAGT)$ que des enzymes de restriction NheI et SpeI pourront identifier pour couper la molécule ; coupes qui sont des étapes de la fonction de rotation (cf. Section 3.1.7). Ces enzymes de restriction sont de type II et ont été choisis car elles créent des extrémités saillantes complémentaires, *i.e.* cohésives. Un couple d'amorces de PCR nommées *For* (pour Forward) et *Rev* (pour Reverse) encadrent $D_{enc}$. Des "buffers" sont présents aux extrémités de la séquence. Ce sont des zones tampons de quelques dizaines de paires de bases qui sont utiles lorsque l'ADN est conservé congelé. Des congélations successives tendent à dégrader les extrémités des molécules d'ADN. L'objectif du protocole qui suit est donc de déchiffrer $D_{enc}$ en $D_{dec}$ comme illustré en Figure 3.32.

Le protocole de rotation repose sur des opérations biologiques que l'on peut assimiler à des opérateurs arithmétiques. La digestion enzymatique de l'ADN permet de couper celui-ci en des points précis de la séquence. À l'inverse, la ligation enzymatique permet de lier des séquences d'ADN entre elles. L'amplification de l'ADN par PCR permet de multiplier exponentiellement une séquence en $n$ copies. La nature précise des opérations dépend des propriétés des enzymes utilisés. Ainsi, une même opération conduite par deux enzymes différents ne donnera pas nécessairement le même résultat. Le protocole comporte aussi plusieurs opérations non biologiques qui visent à purifier et/ou sélectionner les produits des réactions enzymatiques et à contrôler de manière qualitative et quantitative le bon déroulé de celles-ci. Ainsi, la purification de l'ADN sur colonne nous permet d'éliminer la plupart des contaminants protéiques et les tampons de réactions, qui sont les solutions spécifiques dans lesquelles sont effectuées les réactions. La sélection par billes magnétiques permet d'isoler des fragments d'ADN en fonction de certaines propriétés.

Concernant les méthodes de contrôle pour quantifier l'efficacité des étapes intermédiaires, nous avons considéré plusieurs outils. Le dosage de l'ADN est une méthode quantitative qui renseigne sur la concentration en ADN d'un échantillon de 0,1 à 120 ng ADN à l'aide d'un fluorimètre tel que le Qubit 4

Fluorometer de ThermoFisher. Il permet d'évaluer le rendement de l'étape précédente. Un deuxième outil est l'électrophorèse sur gel d'agarose, une méthode qualitative qui nous renseigne sur la taille des molécules présentes dans l'échantillon. Cette méthode est aussi semi-quantitative, elle permet d'avoir une idée de la concentration des molécules de certaines tailles. Néanmoins, il s'agit d'une méthode destructive car à chaque fois il nécessite d'utiliser une partie de la solution, qui ne pourra plus être manipulée ensuite. Quelques nanogrammes d'échantillons suffisent généralement. Cette observation est utile pour vérifier si une coupe ou une ligation de molécules a bien fonctionné, par exemple.

Enfin, nous disposons d'un dernier outil à la fois qualitatif et quantitatif : le séquençage d'ADN. Ce dernier permet de connaître précisément les molécules présentes dans une solution, en utilisant au minimum quelques femtomoles ($10^{-15}$ moles) d'ADN, ce qui correspond généralement à quelques microlitres de solution pour les concentrations avec lesquelles nous travaillons. Il s'agit aussi d'une méthode destructive, qui nécessite d'utiliser une partie de la solution.

Après plusieurs expérimentations, nous avons ainsi développé un protocole qui correspond à l'application des opérateurs élémentaires impliqués dans la rotation (*i.e.* Figure 3.21 de la Section 3.1.7). Il est constitué des 14 étapes suivantes :

1. (a) Amplification par PCR des molécules $D_{enc}$ selon les amorces $For$ et $Rev$ biotinylées, dans 5 tubes à essai séparés.
   (b) Purification et évaluation de la concentration, dans les 5 tubes.
2. (a) Digestion par l'enzyme NheI, dans les 5 tubes. Séparation de la molécule chiffrée en 3 molécules : $(Ford_1)$, $(d_3)$ et $(d_2d_4Rev)$.
   (b) Purification et évaluation de la concentration, dans les 5 tubes.
3. Mélange des 5 tubes.
4. Capture de l'ADN double brin avec des extrémités biotinylées.
   (a) Préparation des billes magnétiques
   (b) Capture de l'ADN double brin biotinylé $(Ford_1)$ et $(d_2d_4Rev)$ dans le tube A1, et des molécules $(d_3)$ dans le tube B1
   (c) Purification et évaluation de la concentration dans les tubes A1 et B1.
5. Tube A1 : Phosphorylation de l'ADN et ligation des molécules $(Ford_1)$ et $(d_2d_4Rev)$ dans le but de former la molécule $(Ford_1d_2d_4Rev)$.
6. (a) Tube A1 : Amplification par PCR avec un primer $For$ standard et un primer $Rev$ biotinylé.
   (b) Tube A1 : Purification et évaluation de la concentration
7. Tube A1 : Digestion par l'enzyme SpeI pour séparer $(Ford_1d_2)$ et $(d_4Rev)$
8. Tube A1 : Capture de l'ADN double brin avec une extrémité biotinylée.
   (a) Préparation des billes
   (b) Capture de l'ADN double brin biotinylé $(d_4Rev)$ dans le tube C, et des molécules $(Ford_1d_2)$ dans le tube A2
   (c) Tubes C et A2 : Purification et évaluation de la concentration
9. (a) Mélange des tubes A2 $(Ford_1d_2)$ et B1 $(d_3)$ et ligation dans le but d'obtenir des molécules de la forme $(Ford_1d_2d_3)$

(b) Tube A2B1 : Purification et évaluation de la concentration

10. (a) Mélange des tubes A2B1 ($For d_1 d_2 d_3$) et C ($d_4 Rev$), phosphorylation et ligation pour obtenir la molécule déchiffrée ($For d_1 d_2 d_3 d_4 Rev$)
    (b) Tube final : Purification et évaluation de la concentration
11. (a) Tube final : Digestion par l'enzyme SpeI pour couper des molécules avec des formes non désirées (formes intermédiaires)
    (b) Tube final : Purification et évaluation de la concentration
12. PCR avec des amorces standards pour amplifier les molécules déchiffrées ($For d_1 d_2 d_3 d_4 Rev$)
13. Purification avec une extraction par gel d'agarose pour sélectionner les molécules de la taille attendue.
14. PCR avec des amorces standards et purification

Nous allons maintenant détailler chaque étape, et justifier les choix effectués dans ce protocole.

**Étape 1 : Séparation en 5 tubes et PCR** Les molécules d'ADN sont séparées en 5 tubes pour permettre de maximiser la quantité d'ADN manipulée. En effet, les colonnes que nous utilisons pour les étapes de purification peuvent traiter au maximum 5 microgrammes d'ADN, et nous souhaitons démarrer le protocole avec une quantité plus importante que cela.

Dans chaque tube, une PCR de 30 cycles est effectuée avec une ADN polymérase Q5. Cet enzyme est choisi car il amplifie l'ADN avec une haute fidélité (enzyme de type HF). Il est aussi de catégorie Hot start, ce qui veut dire qu'il est possible de contrôler l'activation de l'enzyme et d'éviter les phénomènes d'amplifications non spécifiques en début de réaction. Ici, nous faisons une PCR avec des amorces *For* et *Rev* biotinylées, ce qui permet d'ajouter des biotines aux extrémités de toutes les molécules synthétisées lors de la PCR. Ces biotines sont des protéines qui ont la propriété de se lier de manière spontanée et covalente a la streptavidine. Cette interaction protéine/protéine sera utilisée pour séparer l'ADN biotinylé du reste des molécules lors de l'étape de capture par les billes magnétiques. Les biotines sont greffées aux amorces de PCR par les extrémités 5' et une petite séquence d'acides aminés permet d'espacer la biotine de l'amorce. Cette modification des amorces de PCR est appelée "Biotine-TEG". La distance accrue entre la biotine et l'amorce permet d'optimiser l'interaction avec la streptavidine lors de la capture. Les amorces de PCR biotinylées et non biotinylées (standard) sont commandées chez Integrated DNA Technologies (IDT), avec l'option "HPLC grade" qui garantit une grande pureté. Le fait que ces amorces soient biotinylées n'affecte pas les conditions de PCR selon le fournisseur.

Comme beaucoup d'étapes à venir, celle-ci se termine par un processus de purification et de contrôle qualité. La purification permet de se débarrasser des amorces et autres molécules contaminantes. Sans elle, l'enzyme NheI que l'on va utiliser à l'étape suivante fonctionnera mal. En particulier, son rendement est estimé à seulement 25% [179] en présence de l'ADN polymérase Q5 utilisée pour la PCR. Lors de la purification, la solution d'ADN passe dans une colonne qui laisse passer les petites molécules (*i.e.* de moins de 50 paires de base) tandis que les grandes restent coincées dans une membrane. Ces grandes molécules sont ensuite récupérées par élution avec de l'eau ultra-purifiée. Plus précisément, l'élution a lieu avec 20 µL d'eau purifiée (eau Milliq) à 50°C. Pour toutes les purifications de ce protocole, un kit commercial "DNA Clean Up and Concentration NEB" et le protocole associé de NEB (New England Biolabs) sont

utilisés, permettant d'optimiser cette étape. Deux contrôles qualité sont effectués. Un dosage Qubit permet d'estimer la concentration d'ADN dans les solutions. Toutefois, même si la solution contient de l'ADN synthétique très pur, plusieurs brins simples et doubles de plusieurs tailles différentes sont probablement présents. Il n'est donc pas possible de connaître la quantité réelle de molécules d'ADN présentes. Un dosage peut être utile pour pouvoir mélanger l'ADN de différentes solutions de façon équimolaire dans une unique solution. Le dosage nécessite d'utiliser 1 $\mu L$ de solution maximum, et est effectué avec un fluorimètre Qubit. Notons qu'un volume moindre peut être suffisant si l'ADN est très concentré. Le deuxième contrôle est l'analyse des échantillons par électrophorèse sur gel d'agarose (avec une forte concentration de 2%) pourvérifier que la PCR a bien fonctionné, et que les amorces de la PCR ont bien été éliminées de la solution.

**Étape 2 : Coupe $CutS$ dans les 5 tubes avec l'enzyme NheI** La coupe, ou digestion enzymatique, se fait séparément dans les 5 tubes pour maximiser la quantité d'ADN utilisée, car la purification la suivant est limitée à 5 µg d'ADN par colonne. Les molécules chiffrées sont coupées par NheI pour obtenir trois fragments d'ADN ($Ford_1$), ($d_3$) et ($d_2d_4Rev$). L'enzyme NheI choisi est de type HF (High-Fidelity) et assure une fidélité de coupe au niveau de son site de restriction. Les enzymes commandés chez New England Biolabs (NEB) sont de grande qualité. En temps normal, *i.e.* en dehors de notre contexte, la réaction s'effectue typiquement pendant 1h, avec 10 unités d'enzyme par µg d'ADN. Avec ces paramètres, nous avons cependant pu voir expérimentalement une proportion importante de molécules non coupées. Une hypothèse est que les recommandations du fournisseur ne sont pas adaptées à notre réaction. En effet, contrairement aux conditions standards, il y a énormément de molécules d'ADN courtes à couper par microgramme d'ADN. C'est pourquoi dans notre protocole, nous avons choisi ne pas respecter les recommandations et d'augmenter la quantité d'enzymes et de les mettre "en excès" dans la solution. Plus précisément, nous avons laissé plus de 120 unités d'enzymes par µg d'ADN agir 24 heures à 16°C pour obtenir potentiellement plus de découpes. Le risque de découpes non spécifiques augmente avec le temps de réaction, mais reste normalement très rare compte tenu que ce sont des enzymes dits de haute fidélité ("HF"). Nous le verrons cela a fonctionné.

À l'issue de la digestion, une étape de chauffe de la solution à 80°C pendant 20 minutes permet de dénaturer l'enzyme NheI, pour qu'il n'interfère pas avec les étapes suivantes. Cette étape de chauffe n'affecte pas l'ADN biotinylé .

Une purification est effectuée à la fin de cette étape, comme après chaque digestion enzymatique, pour éliminer les enzymes de restriction, et changer l'ADN de solution. Cela permet notamment de se débarrasser de la protéine "Recombinant Albumin", à laquelle l'enzyme de restriction s'est accroché pour faciliter la digestion. En effet, si cette protéine empêche l'adhésion de l'enzyme aux tubes et aux surfaces des pipettes, elle peut interagir avec la streptavidine et ainsi causer des problèmes lors de l'étape suivante : la capture. Une purification est donc nécessaire pour se débarrasser de ce type d'interférence. Pour remettre l'ADN dans une nouvelle solution, l'élution permettant de décrocher l'ADN de la membrane se fait avec de l'eau mQ à 50°C pour optimiser le rendement de la récupération d'ADN. À l'issue de cette étape, des contrôles qualité sont effectués : un dosage Qubit et une électrophorèse sur gel d'agarose à 2%, qui permettent respectivement d'évaluer la concentration et la taille des molécules en solution.

**Étape 3 : Mélange $Mix$ des 5 tubes** Cette étape consiste à rassembler les solutions contenues dans les 5 tubes pour former une seule solution.

**Étape 4 : Capture $SepBiotin$ de l'ADN double brin avec des extrémités biotinylées** La capture d'ADN se fait avec des billes magnétiques micrométriques, nous parlons alors de couplage. Elles sont recouvertes d'une couche de streptavidine qui va se lier aux biotines présentes aux extrémités des molécules d'ADN. Ces billes sont commandées chez Invitrogen France et ont pour référence "Streptavidine Dynabeads™ MyOne™ C1".

Avant de faire la capture, les billes sont préparées pour optimiser le rendement de la capture. Cette étape vise à éliminer l'excès de streptavidine non couplés aux billes présentes dans la solution. Le processus d'élimination comporte plusieurs étapes de lavage successives à l'aide d'une solution contenant un détergent puissant nommé Tween™ 20 (Invitrogen France). Le détergent permet de réduire les interactions entre les protéines. Après chaque lavage, les billes sont placées sur un portoir magnétique pour séparer les billes du reste de la solution. Sans cette étape de préparation, le rendement de capture d'ADN biotinylé par les billes serait moindre en raison de la liaison d'une partie de l'ADN aux streptavidines libres. Enfin, la quantité de couples billes-streptavidine à préparer dépend de la quantité d'ADN biotinylé théorique à capturer. Dans notre cas, 1 mg de billes permet de capturer 20 µg d'ADN biotinylé.

La capture est la mise en contact des molécules d'ADN biotinylées avec les billes couplées aux streptavidines. Les solutions contenant les billes et l'ADN sont mélangées, avec une solution tampon contenant du détergent Tween™ 20 pour éviter les captures d'ADN non biotinylé. La solution finale d'un millilitre est placée pendant 40 minutes à 16°C sur un agitateur rotatif qui permet d'homogénéiser la solution et donc la rencontre entre la biotines et la streptavidine. Cette agitation douce évite aussi que les billes sédimentent. Plus la dilution de la solution est importante, plus il faut de temps pour que les biotines et les streptavidines se rencontrent. Le temps est donc ajusté à 40 minutes, contre 5 minutes pour des solutions plus concentrées. Ensuite, un aimant collé à la paroi du tube permet de séparer les billes recouvertes d'ADN biotinylé du reste de la solution. Nous avons optimisé cette étape en allongeant sa durée de 2 à 5 minutes. Les molécules d'ADN sont alors divisées en deux phases : la phase stationnaire, immobilisée sur la paroi du tube par les aimants, et la phase mobile en solution, qui ne s'est pas liée aux billes. La phase mobile est récupérée par pipetage dans le tube à essai, et placée dans un nouveau tube B1. Le tube B1 doit donc contenir les blocs $d(d_3)$.

Les billes sur la phase stationnaire sont rincées plusieurs fois avec une solution tampon contenant du Tween™ 20, puis remises en suspension dans cette même solution. Pour casser la liaison très forte entre la streptavidine des billes et la biotine de l'ADN, une hydrolyse chimique est effectuée avec de l'eau ultra pure, chauffée à 70°C. En chimie organique, l'hydrolyse est une réaction dans laquelle une molécule organique et de l'eau réagissent en rompant une liaison covalente pour former deux molécules organiques avec des groupes fonctionnels comprenant les atomes de la molécule d'eau. Ici, l'eau ultra pure permet de rompre les liaisons covalentes biotine-streptavidine. Le rendement [180] de cette opération avec l'eau ultra pure est supérieur à 95% pour une température de 70°C. Cette opération permet de récupérer les billes, par ailleurs coûteuses, pour de nouvelles utilisations. Un aimant sur la paroi du tube à essai permet de séparer les billes des molécules d'ADN, qui sont placées dans un tube A1. Ce tube contient donc les molécules $(Ford_1)$ et $(d_2d_4Rev)$.

A l'issue de cette étape, nous effectuons les purifications nécessaires. Dans le Tube A1, l'ADN se trouve déjà dans une solution d'eau très pure, nous faisons donc une purification uniquement pour augmenter la concentration d'ADN. Dans le Tube B1, l'ADN se trouve dans la solution avec du détergent Tween™ 20, une purification est donc requise pour l'éliminer et ne pas affecter les étapes suivantes.

**Étape 5 dans le tube A1 : Phosphorylation *Phos* et ligation *LigS* d'extrémités saillantes** Le but de cette étape est de faire la ligation des deux molécules ($Ford_1$) et ($d_2d_4Rev$) pour former la molécule partiellement déchiffrée ($Ford_1d_2d_4Rev$). La ligation est précédée d'une étape de phosphorylation : Grâce à l'action de l'enzyme T4 Polynucleotide Kinase (T4 PNK), un groupement phosphate est ajouté aux extrémités 5' de chaque molécule d'ADN. Ce groupement est essentiel pour la réaction de ligation. Comme les opérations précédentes ont pu endommager l'ADN, et en particulier entraîner des pertes de groupement phosphate, la phosphorylation enzymatique permet de s'assurer que les groupements sont bien présents. Ensuite, l'enzyme de ligation T7 DNA ligase lie les extrémités cohésives d'ADN double brin présentes dans la solution du tube A1. La T7 ligase ne lie que des extrémités cohésives, et non les extrémités franches. En théorie, cela permet de lier ($Ford_1$) à ($d_2d_4Rev$), pour ainsi former la molécule ($Ford_1d_2d_4Rev$).

**Étape 6 dans le tube A1 : PCR avec un primer For standard et un primer Rev biotinylé** Cette étape de PCR a deux objectifs. Le premier est d'amplifier la molécule partiellement déchiffrée ($Ford_1d_2d_4Rev$). En effet, seules les molécules encadrées par les amorces For et Rev seront amplifiées exponentiellement. Une molécule avec une seule amorce sera amplifiée linéairement. Le second objectif est d'obtenir une molécule partiellement déchiffrée ($Ford_1d_2d_4Rev$) avec seulement une extrémité biotinylée. Nous devons donc utiliser une seule amorce biotinylée dans la réaction de PCR pour pouvoir séparer les molécules. Cette séparation sera effectuée à l'étape 8, après la seconde opération de coupe enzymatique. Nous choisissons ici de biotinyler *Rev* car, après la seconde opération de coupe, c'est cette portion de la molécule qui sera la plus courte, comparée à la molécule contenant $For$. Elle sera donc plus facile à capturer si l'on considère les problématiques d'encombrement stérique au niveau moléculaire.

Il est essentiel d'effectuer une purification et donc de changer de solution à l'issue de cette étape car l'enzyme de restriction SpeI utilisé à l'étape suivante est actif à 0% [179] dans une solution de PCR. De plus, l'enzyme utilisé pour la PCR est toujours actif et pourrait combler les extrémités cohésives qui seront créées à l'étape suivante. Il est donc nécessaire de s'en débarrasser. L'élution est réalisée dans 20 µL d'eau ultra pure à 50°C.

**Étape 7 dans le tube A1 : Digestion *CutS* par l'enzyme SpeI** L'objectif de cette étape est de couper la molécule ($Ford_1d_2d_4Rev$) en deux molécules ($Ford_1d_2$) et ($d_4Rev$). Pour cela, la digestion est effectuée à 16°C durant 16 heures. Comme lors de l'étape 2, cette longue durée ainsi que la quantité d'enzymes en excès permettent de maximiser la proportion de molécules qui vont être coupées.

**Étape 8 dans le tube A1 : Capture *SepBiotin* de l'ADN avec une extrémité biotinylée** Le processus est similaire à l'étape 4. Les billes sont tout d'abord préparées pour supprimer tout excès de streptavidine dans la solution. À partir de la concentration d'ADN évaluée à l'étape 10, le volume de billes magnétiques à utiliser est déterminé. Les solutions sont mélangées avec une solution tampon Tween™ 20,

puis placées sur un agitateur rotatif pour faciliter les liaisons entre les billes recouvertes de streptavidines et les extrémités biotinylées de l'ADN. Ensuite, les molécules ($Ford_1d_2$) non immobilisées par les billes sont récupérées dans le tube par pipetage et placées dans le tube A2. Les molécules ($d_4Rev$) qui ont une extrémité *Rev* biotinylée sont capturées via un aimant, puis séparées des billes par hydrolyse, pour être placées dans le tube C.

Une purification est effectuée dans les tubes A2 et C. L'élution est effectuée dans 20 µL d'eau ultra pure mQ à 50°C.

**Étape 9 : Mélange $Mix$ des tubes A2 et B1, phosphorylation $Phos$ et ligation $Lig$** L'objectif de cette étape est de faire la ligation des molécules ($Ford_1d_2$) présentes dans le tube A2 avec les molécules ($d_3$) du tube B1 qui ont été écartées lors de la première capture (Étape 4). Pour cela, des volumes similaires provenant du tube A2 et du tube B1 sont mélangés. Ensuite, une phosphorylation est effectuée avec l'enzyme T4 PNK. Il garantit que chaque molécule à un groupe phosphate à ses extrémités, et peut donc se lier à d'autres extrémités de molécules. Enfin, l'enzyme de ligase T7 est utilisé pour lier les molécules ($Ford_1d_2$) et ($d_3$) par leurs extrémités cohésives.

Dans le tube A2B1, une purification est effectuée pour réduire le volume de solution dans lequel se trouve l'ADN. Une concentration plus importante augmente les probabilités de rencontre entre les molécules d'ADN, ce qui favorisera l'efficacité de la ligation à l'étape suivante. Des contrôles qualité sont aussi effectués, sous forme d'un dosage et d'un dépôt sur gel.

**Étape 10 : Mélange $Mix$ des tubes A2B1 et C, phosphorylation $Phos$ et ligation $LigS$** Cette étape est similaire à l'étape 9. À l'aide des concentrations obtenues à l'étape précédente, des volumes équivalents provenant des tubes A2B1 et C sont mélangés. Le tube A2B1 contient les molécules ($Ford_1d_2d_3$) tandis que le tube C contient les molécules ($d_4Rev$). Une phosphorylation permet de préparer les extrémités de l'ADN, puis la ligation est effectuée avec l'enzyme T7 ligase. L'objectif est alors d'obtenir la molécule déchiffrée ($Ford_1d_2d_3d_4Rev$).

Une purification est requise dans le tube final pour réduire le volume et remplacer le tampon de la solution.

**Étape 11 : Coupe $CutS$ par l'enzyme SpeI** Cette étape permet d'optimiser les résultats, et de retrouver plus facilement la molécule déchiffrée parmi les molécules partiellement déchiffrées que l'on retrouve en fin de protocole. En effet, à la fin de celui-ci et comme présenté en Figure 3.33, la molécule déchiffrée ne contient plus le site de restriction $R_2$ de l'enzyme SpeI, mais seulement le site $R_1$ de NheI et les concaténations des moitiés des sites de NheI et SpeI, $R_1^1R_2^2$ et $R_2^1R_2^1$. Au contraire, la molécule chiffrée $D_{enc}$ contient $R_2$, et la molécule partiellement déchiffrée obtenue à l'étape 5 aussi. Cette étape permet donc de couper ces molécules indésirables, ce qui donnera des séquences ADN bien plus petites que la taille de molécule déchiffrée attendue. Ainsi, lors de l'étape de purification par électrophorèse sur gel d'agarose, la bande d'intérêt correspondant aux molécules déchiffrées ne sera pas coexistante avec la bande correspondant aux molécules chiffrées. En d'autres termes, la différence de tailles entre les deux types de molécules nous permettra de plus facilement les discriminer.

La digestion par l'enzyme SpeI a lieu pendant une heure à 37°C, avec une quantité d'enzymes en excès (120 unités d'enzymes par microgramme d'ADN à digérer) afin de maximiser le nombre de coupes.

Il est nécessaire de se débarrasser de l'enzyme et de la solution tampon pour pouvoir faire une PCR sans interférences. De nouveau, nous faisons une purification sur colonne en éluant l'ADN dans 20 µL d'eau ultra-pure préalablement chauffée à 50°C. Le dosage est effectué selon la méthode décrite précédemment.

**Étape 12 : PCR avec des amorces standards** Une PCR avec des amorces standards permet d'augmenter la proportion de molécules encadrées par les amorces For et Rev. 15 cycles d'amplification par PCR sont faits à partir de 10 ng de matrice de l'étape 19.

**Étape 13 : Séparation *SepGel* électrophorétique des molécules selon leurs tailles et extraction** Lors de cette étape, l'ADN de la taille souhaitée est extrait de la solution. Pour cela, l'ADN est déposé sur un gel d'agarose, et un courant électrique est appliqué. L'ADN étant chargé négativement, il va migrer dans le gel d'agarose, à une vitesse dépendant de sa taille. Plus elle est grande, plus la migration est lente, et inversement. Ainsi, à l'issue de la migration, nous avons un profil composé de différentes bandes où chaque bande contient des molécules d'ADN de même taille. Finalement, l'ADN de la taille souhaitée est extrait : sur une table à UV, avec un scalpel, la bande d'ADN de la taille correspondante est découpée. Pour récupérer l'ADN contenu dans le gel d'agarose, nous utilisons un kit commercial de NEB : "Monarch® DNA Gel Extraction Kit" dont le principe est similaire à celui utilisé pour les purifications sur colonne employé jusqu'ici. De nouveau, un dosage de l'ADN ainsi purifié est réalisé. À l'issue de cette étape, nous avons 50 µL d'ADN à environ 0.7 ng/µL.

**Étape 14 : PCR avec des amorces standards et purification** Une ultime PCR de 25 cycles est effectuée suivant le même protocole qu'à l'étape 20 avec 2 ng d'ADN, pour amplifier l'ADN obtenu à l'étape précédente. Enfin, une étape de purification permet de préparer les échantillons pour le séquençage d'ADN par la technique développée par Oxford Nanopore Technologies (ONT).

Pour résumer, ce protocole résulte de nombreuses expérimentations et de plusieurs échecs. Il est complexe de part le nombre de manipulations à réaliser et, comme nous le verrons au travers son expérimentation, il peut être encore amélioré.

#### 3.3.2.2 Résultats expérimentaux

Dans cette partie, nous présentons les données que nous avons encodées et les résultats obtenus à plusieurs étapes du protocole. Les manipulations ont été effectuées par Julien Leblanc, ingénieur de recherche en biotechnologies au CNRS.

Lors de la manipulation, nous avons observé l'évolution de la quantité d'ADN et le type de molécules créées à différentes étapes grâce à plusieurs outils :

— Évaluation de la quantité d'ADN à disposition, présentée dans le Tableau 3.4.

— Évaluation de la taille des molécules lors de dépôts sur gel, en figures 3.35 et 3.37.

— Séquençages, en milieu et fins de protocole, dont les résultats quantitatifs sont présentés en Tableaux 3.3, 3.5, et 3.6.

**Paramètres**

Pour cette expérimentation nous avons considéré le texte "Déclaration des Droits de l'Homme et du Citoyen. Les hommes naissent et demeurent libres et égaux en droits.". Celui-ci a été chiffré avec l'AES-256 puis encodé en une séquence ADN $D$ avec notre algorithme $DSWE$, paramétré de manière à limiter la taille des homopolymères à 3 bases et interdit 4 motifs de 6 bases. Sont bien entendu pris en compte les sites de restriction $R_1 = (GCTAGC)$ et $R_2 = (ACTAGT)$ des enzymes NheI et SpeI, et les motifs $R_1^1 R_2^2 = (GCTAGT)$ et $R_2^1 R_2^1 = (ACTAGC)$, qui résultent du mélange des deux restrictions précédentes. La séquence ADN obtenue $D$ fait 424 bases. Elle est divisée en 4 blocs $d_1$, $d_2$, $d_3$, $d_4$ de tailles équivalentes. Pour les chiffrer, la fonction numérique $iRot$ est appliquée à $D$, comme présenté en Figure 3.33. Pour cela, les positions des blocs $d_2$ et $d_3$ sont échangées, le site de restriction $R_1$ de NheI est ajouté avant et après $d_3$, tandis que le site $R_2$ de SpeI est ajouté après $d_3$. Puis, des amorces de PCR 20 bases sont ajoutés aux extrémités de la séquence. Elles sont déterminées avec l'algorithme Ithos, un sous-module du logiciel de design d'amorces Genofrag [175], dont les nombreux paramètres sont détaillés en Annexe D. Enfin, des "buffers" de 25 bases chacun encadrent la séquence.

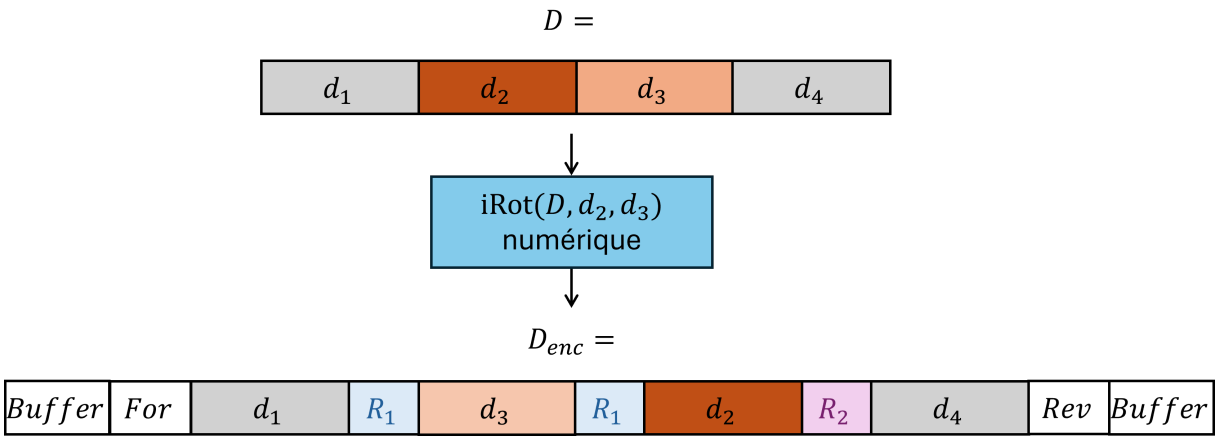



FIGURE 3.33 – La séquence ADN $D$ est chiffrée par rotation en une molécule $D_{enc}$ avec échange des deux blocs de données $d_2$ et $d_3$.

La synthèse d'une molécule de 514 paires de bases a ainsi été commandée chez le fournisseur d'ADN synthétique IDT™, ils sont appelés gBlocks™ Gene Fragments [55]. Ces molécules sont choisies pour plusieurs raisons : Le taux d'erreur médian lors de leur synthèse est très faible (1/5000), et elles bénéficient d'un contrôle qualité accru. En effet, si le fournisseur remarque un problème, les molécules sont synthétisées de nouveau.

Lors de la première version du protocole, nous avions travaillé en duplicata, c'est-à-dire en appliquant le même protocole à deux 2 gBlocks™ différents en parallèle. Toutefois, face aux nombres d'optimisation à conduire et face aux coûts que représente chaque réaction, nous avons fait le choix de nous concentrer sur un seul gBlock™ jusqu'à obtenir un résultat satisfaisant. Pour donner un ordre de grandeur, chaque protocole de rotation représente un coût de plusieurs centaines d'euros et plusieurs jours de travail au laboratoire. La dernière version du protocole (V5), ici décrite, a été réalisée deux fois avec le même gBlock™. À ce jour, il reste des pistes d'optimisations à explorer. Une version définitive du protocole

mériterait d'être testée en duplicata. Cela permettrait de vérifier la reproductibilité des résultats pour un même gBlock™ et pour des gblocks™ différents.

**Contrôle qualité de la solution synthétisée avant le protocole**

A la réception des tubes d'IDT™, plusieurs vérifications ont été effectuées. La première est la vérification par dosage de la concentration d'ADN dans les tubes. Le dosage ne nous renseigne pas sur la quantité réelle de molécules chiffrées car la solution peut contenir des brins simples et doubles de plusieurs tailles. Nous avons donc fait l'hypothèse, qu'à cette étape, nous disposons d'un mélange parfait d'ADN double brin de la taille attendue. Le dosage nécessite d'utiliser 0,1 à 120 ng d'ADN, et a été effectué avec un fluorimètre Qubit® Fluorometer (Invitrogen™ Q32851). Le kit de dosage utilisé est "AccuGreen™ High Sensitivity dsDNA Quantitation Kit (Invitrogen™). L'ADN fourni par IDT est contenu dans un petit volume de 50 µL (microlitres), avec une concentration théorique de 10 ng par microlitre. En pratique, la concentration réelle mesurée au Qubit est de 17 ng/µL. Un dépôt sur gel d'agarose a aussi été effectué afin de vérifier la pureté du produit, en observant la taille des molécules. Nous obtenons en Figure 3.34 à gauche une bande nette principale, montrant que les molécules d'ADN sont en grande majorité de tailles similaires et proches de celle attendue (514 paires de bases).

Une partie du produit a aussi été utilisée pour un séquençage nanopore afin de vérifier les données en entrée de notre algorithme. Chaque read lu par séquençage est ensuite comparé aux blocs de données $d_i$, à l'aide de l'algorithme bioinformatique d'alignement de Smith-Waterman [181]. Il permet d'identifier quels blocs contiennent les séquences lues, et de donner un score d'alignement entre 0 (aucun alignement) et 10 (alignement parfait) à chaque bloc. Un bloc est considéré comme présent dans une séquence ADN lorsque le score est égal ou supérieur à 7, ce qui correspond à une tolérance d'erreur d'environ 20% des bases.

Comme présenté dans le Tableau 3.3, nous retrouvons la molécule souhaitée en majorité, pour 45,60% des 126521 reads. Les molécules synthétisées par IDT™ n'atteindront jamais un taux de pureté de 100% ; un taux inatteignable en biologie et en chimie quel que soit le processus considéré. De plus, les gBlock™ d'IDT™ sont construits par l'assemblage ordonné de molécules plus courtes, car il n'existe aucune méthode de synthèse chimique ou enzymatique de l'ADN qui permette de fabriquer de telles molécules en une fois. Cela explique la présence de $32,92\%$ de blocs $d_4$ dans la solution (*cf* Figure 3.3) : Il existe des sous-produits qui ont persisté malgré les procédés de purification du fournisseur IDT™.

Toutefois, nous remarquons que la proportion de molécules attendues que l'on arrive à lire est plus faible que la proportion réelle. Lorsque l'on observe le dépôt sur gel, en Figure 3.34 à gauche, nous ne voyons qu'une seule bande nette. Nous pouvions donc nous attendre à une pureté d'environ 90%, sachant en plus que les gBlocks™ disposent d'un fort contrôle qualité de la part du fournisseur IDT™. Ce décalage entre le résultat sur gel et par séquençage peut être expliqué en raison de biais du séquençage et du traitement des données. En effet, la préparation pour le séquençage repose sur plusieurs étapes enzymatiques et de sélection similaires à celles présentées dans notre protocole. Ces étapes sont donc sujettes aux mêmes problèmes de rendement entre la quantité d'ADN en début et fin de processus. Le séquençage en lui-même introduit aussi un biais, avec le séquençage potentiel de molécules incomplètes, ainsi que le logiciel de "basecalling" choisi et ses paramètres. Par exemple, le pourcentage d'occurrence

des molécules souhaitées est de 45,6 % en utilisant le logiciel Guppy v6.3.9, contre 39,8% pour Dorado V0.4.1, le nouveau basecaller pour séquençage Nanopore de ONT (Oxford Nanopore Technologies). Enfin, il existe un biais lié au décodage et à la recherche des molécules chiffrées dans les résultats du basecalling. En effet, nous cherchons lors du séquençage des blocs précis, avec un taux d'erreur maximal fixé. C'est pourquoi nous avons 11,68% de molécules "U" ou "Unknown" que nous n'arrivons pas à identifier. Des paramètres plus flexibles, par exemple aux extrémités souvent mal lues des molécules, pourraient augmenter le pourcentage d'occurrences de la molécule chiffrée. Notons que toutes les molécules partielles telles que $d_4$ et $d_1d_3$ pourront interférer avec le protocole, puisqu'elles ne sont pas de formes attendues.

TABLE 3.3 – Résultat du séquençage nanopore des molécules synthétisées par IDT (gBlock01). Nombre et pourcentage d'occurrence de chaque molécule parmi 90847 reads. Seuls les molécules présentes à plus de $0,5\%$ sont affichées.

| Molécules | Nombre d'occurrences | Pourcentages d'occurrences |
|---|---|---|
| $d_1d_3d_2d_4$ | **57777** | **45,67%** |
| $d_4$ | 41652 | 32,92% |
| U | 14782 | 11,68% |
| $d_1d_3$ | 3580 | 2,83% |
| $d_3d_2d_4$ | 3048 | 2,41% |
| $d_1d_3d_2$ | 2472 | 1,95% |
| $d_1$ | 880 | 0,70% |

### Résultats expérimentaux intermédiaires du protocole

Ici, nous présentons et discutons les résultats de dosage, de dépôt sur gel et de séquençage à plusieurs étapes intermédiaires de notre protocole pour évaluer les performances des opérations biomoléculaires impliquées dans notre protocole.

TABLE 3.4 – Évolution de la quantité de matière aux différentes étapes du protocole, estimée à partir des concentrations obtenues par dosage au fluorimètre Qubit® Fluorometer (Invitrogen™ Q32851), avec le kit de dosage "AccuGreen™ High Sensitivity dsDNA Quantitation Kit (Invitrogen™).

| Étape 0 | Étape 1 | Étape 3 | Étape 4 | Étape 5 | Étape 8 | Étape 9 | Étape 10 | Étape 11 |
|---|---|---|---|---|---|---|---|---|
| 850 ng | 5662 ng | 4780 ng | 2696 ng | 992 ng | 508 ng | 304 ng | 254 ng | 146 ng |

**Étape 1** Les 5 PCRs prennent en entrée environ 2 ng d'ADN chacune. Après l'étape de PCR et de purification, les concentrations dans les 5 tubes varient de 55,2 ng/µL à 64,0 ng/µL, dans 19 µL de solution chacun. Cette étape a donc permis de multiplier la quantité d'ADN par un facteur 570, comme nous le voyons dans la Table 3.4. Notons que ce gain est diminué par les purifications, dont le rendement est estimé à environ 70%. La Figure 3.34 présente le dépôt sur gel avant et après la PCR. Avant la PCR, la bande est nette et à une taille qui correspond à celles des molécules avec les "buffers" aux extrémités, *i.e.* 514 paires de bases. Après la PCR, les molécules sont plus petites car les "buffers" aux extrémités sont supprimés. La PCR n'amplifie que ce qui est encadré par les amorces. Nous voyons que la bande

se situe à la taille attendue (environ 464 paires de bases). De par le principe de la PCR, nous avons aussi augmenté la proportion de molécules déchiffrées. Le taux de pureté est maintenant probablement supérieur à 90%. Ce chiffre n'est pas vérifié via un séquençage puisque tout le matériel est utilisé pour la manipulation.

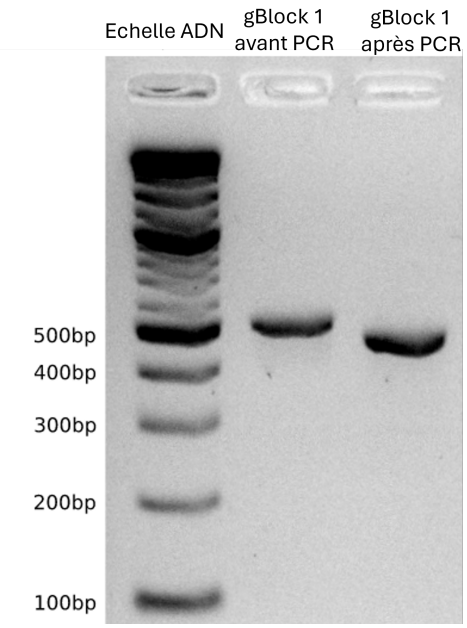



FIGURE 3.34 – Résultat de l'électrophorèse sur gel d'agarose de la molécule chiffrée commandée chez IDT. La première colonne contient un marqueur de taille (1Kb plus DNA ladder NEB) et sert d'échelle, la deuxième colonne contient les gBlocks avant la PCR (514 paires de bases), tandis que la 3ème colonne contient les gBlocks après la PCR (464 pb). Note : le gel est formé à partir d'une solution TBE (Tris/Borate/EDTA) avec 3 % d'agarose finale.

**Étape 3** Après avoir mélangé les 5 tubes, nous obtenons une concentration de 47.8 ng/µL dans un volume d'environ 100 µL. Cela correspond donc à une quantité d'ADN de 4780 ng. Environ 2 µL ont été utilisés pour le dépôt sur gel 1 de la Figure 3.35, pour apprécier la pureté de l'ADN et l'efficacité de la digestion par l'enzyme de restriction NheI. Nous pouvons nous attendre à un rendement de la coupe supérieur à 90%, car l'enzyme est de haute fidélité et utilisé en grande quantité. Sur le gel, nous observons 3 bandes nettes dont les tailles correspondent à celles obtenues par la coupe de l'étape 3, *i.e.* 101, 120 et 231 paires de bases. Nous pouvons donc estimer que la digestion enzymatique avec l'enzyme NheI a donc bien fonctionné. Pour rappel, l'électrophorèse sur gel d'agarose est une méthode de contrôle qualitative et semi-quantitative qui présente des limites dans la sensibilité de détection des molécules d'ADN. Autrement dit, ce n'est pas parce que nous ne distinguons pas de bandes qu'il n'y a pas d'ADN. Avec l'expérience, nous pensons que nous avons aussi des molécules d'ADN non digérées et partiellement digérées, mais en concentration si faible que nous ne pouvons pas les détecter avec cette méthode. Cette hypothèse est d'autant plus probable qu'avoir une réaction biologique totale (de 100%) est excessivement difficile en biologie moléculaire.

**Étape 4** Après la séparation des molécules par capture, nous avons évaluer les concentrations et les tailles des molécules dans les tubes. À cette étape, le Tube A1 devrait contenir les molécules d'ADN capturées ($Ford_1$) de 120 paires de bases et ($d_2d_4Rev$) de 231 paires de bases. La concentration est de 98,0 ng/µL dans ce tube. Nous pouvons observer sur le Gel 2 de la Figure 3.35 que les bandes encadrées en vert correspondant aux molécules de 120 et 231 paires de bases sont nettes. Toutefois, nous distinguons des traces, appelées "smears", d'ADN de tailles inattendues. Nous pensons que ces molécules ont pu s'hybrider provisoirement aux extrémités cohésives de l'ADN capturé par les billes. Nous avons aussi très probablement capturé des molécules d'ADN partiellement digérées et non digérées résultant de l'étape 2 de coupe.

Dans le Tube B1, les molécules non capturées par les billes devraient être de la forme ($d_3$) de 101 paires de bases. La concentration dans B1 est de 36,8 ng/µL. Cette différence avec A1 peut s'expliquer par le fait, qu'en théorie, deux tiers des molécules sont capturées. Sur le Gel 2 de la Figure 3.35, nous observons une bande nette à la taille attendue pour ($d_3$). La plupart des molécules récupérées sont donc de la bonne taille. A partir des concentrations obtenues par dosage, nous estimons que les 2 tubes contiennent environ 2696 ng d'ADN, c'est à dire un rendement de 56% par rapport à l'étape précédente (*cf* Tableau 3.4), ce qui est satisfaisant compte tenu de la complexité de l'étape de capture. De par la quantité de matière et l'aspect sur le gel, nous déduisons donc que la capture a bien fonctionné.

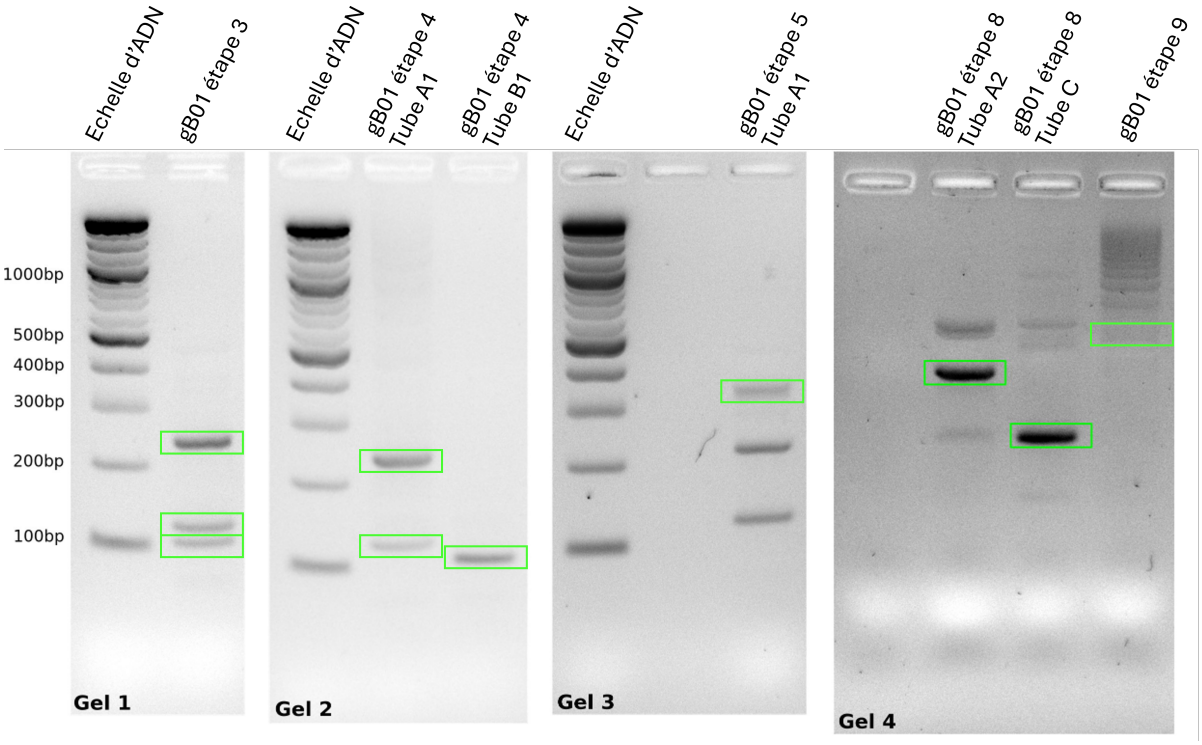



FIGURE 3.35 – Résultat de l'électrophorèse sur gel d'agarose des molécules d'ADN dites "gBlock01" à différentes étapes de notre protocole. Dépôt de 50 à 100 ng d'ADN, comparé à une échelle "1 Kb plus DNA ladder NEB" qui sert à indiquer la taille des blocs apparaissant sur les pistes. Les bandes encadrées en vert représentent les molécules de tailles attendues. Note : le gel est formé à partir d'une solution TBE (Tris/Borate/EDTA) avec 3 % d'agarose finale.

**Étape 5 : Molécule partiellement déchiffrée** A cette étape, nous devrions retrouver la molécule partiellement déchiffrée ($Ford_1d_2d_4Rev$) obtenue par ligation des molécules ($Ford_1$) et ($d_2d_4Rev$). La ligation suppose que les extrémités saillantes cohésives soient toujours présentes, et qu'il n'y a pas d'ADN contaminant (par exemple ($d_3$)) qui se soit concaténé. Les précédentes étapes comportent des réactions imparfaites (réactions enzymatiques et capture des molécules d'ADN), qui peuvent amener à des erreurs et créer des formes inattendues. Il faut donc s'attendre à ce qu'il n'y ait pas que la molécule partiellement déchiffrée dans la solution. Pour voir quelles molécules ont été créées par les étapes précédentes, nus avons fait un dépôt sur gel et un séquençage.

Le séquençage produit 75% de "reads" (ou lectures) de bonne qualité ('phred' score supérieur ou égal à 10). Cela veut dire qu'ils sont d'une longueur supérieure à 20 pb (paires de bases) et ont une précision d'identification de chaque base supérieure ou égale à 90%. Ces paramètres de seuillage sont ceux proposés par défaut par le fournisseur de produits de séquençage nanopore : ONT ("Oxford Nanopore Technologies"). Une telle proportion de lecture de bonne qualité indique que le séquençage s'est bien déroulé. En effet, d'expérience et avec le matériel que nous utilisons, il est normal d'avoir 25% de lecture jugée de mauvaise qualité. Chaque "read" lu est ensuite comparé aux blocs de données $d_i$, à l'aide de l'algorithme bioinformatique d'alignement de Smith-Waterman. Un bloc $d_i$ est considéré comme présent dans un 'read' lorsque le score d'alignement est égal ou supérieur à 7, ce qui correspond à une tolérance d'erreur d'environ 20% des bases.

Dans le Tube A1, la concentration d'ADN est satisfaisante, avec 49.6 ng/µL. Le Gel 3 de la Figure 3.35 présente une bande (encadrée en vert) de taille correspondant au résultat de la ligation, *i.e.* 363 paires de bases. D'après les résultats du séquençage présentés dans le Tableau 3.5, la forme attendue de molécules ($d_1d_2d_4$) est majoritaire à $38,39\%$, représentant 14767 des 38461 reads totaux. Notre protocole fonctionne donc bien. Mais il y a aussi d'autres molécules de formes différentes, dont nous expliquons les origines probables. Lorsqu'une seule des deux coupes à l'étape 3 a fonctionné, nous obtenons des molécules ($Ford_1d_3$) (moins de 2% des molécules) de 221 paires de bases ou ($d_3d_2d_4Rev$) (moins de 1%) de 332 pb. D'autre part, la ligation ne fonctionne pas à 100%, c'est pourquoi nous observons des bandes aux environs des tailles de 120 et 231 paires de bases. Ce sont probablement des molécules non assemblées : ($Ford_1$) de 120 paires de bases et ($d_2d_4Rev$) de 231 paires. Nous observons aussi sur ce gel quelques molécules d'environ 460 paires de bases qui sont les molécules chiffrées du début ($Ford_1d_3d_2d_4Rev$), présentes à $9,58\%$. Les étapes précédentes ne sont pas totales. C'est notamment le cas de la coupe avec les enzymes de restriction. Nous nous attendons donc toujours à avoir des molécules chiffrées. D'autre part, de manière surprenante, nous obtenons 17% de molécules ($d_4$). Nous supposons que ces molécules proviennent du tube de départ, où ($d_4$) constituait $32,92\%$ des occurrences (*cf.* Tableau 3.3). En effet, lors de la PCR, cette molécule peut avoir été amplifiée linéairement puisqu'elle contient le primer $For$. Par la suite, nous aurions co-purifié ces molécules lors de l'étape 4 de capture par billes magnétiques. $11,68\%$ des molécules sont de forme inconnue $U$, ce qui veut dire que l'algorithme d'alignement n'arrive pas à identifier les blocs composant ces molécules. Après analyse, il s'avère que ces molécules sont de petites tailles (50 pb) et hétéroclites. Elles correspondent à des portions des blocs $d_1$ et $d_2$. Nous pensons que ces molécules sont des sous-produits de PCR issus de problèmes d'appariement des amorces de PCR. Finalement, l'analyse de la taille des reads montre que la grande majorité des molécules est de la taille attendue d'environ 363 paires de bases.

L'analyse des résultats du séquençage permet aussi de montrer que la séparation par capture de l'étape 4 fonctionne bien. Pour cela, nous observons le nombre d'occurrences de chaque bloc $d_i$ dans les molécules. L'étape 4 a pour but d'isoler les molécules $d_3$ et nous observons que seulement 6% des 86058 blocs lus sont des blocs $d_3$, présents dans des molécules de formes intermédiaires. Nous n'observons donc pas de blocs $d_3$ seuls, l'étape 4 de capture a bien permis de les isoler.

TABLE 3.5 – Résultat du séquençage nanopore à l'étape 5 du protocole. Nombre et pourcentage d'occurrence de chaque molécule parmi 38461 reads. Seuls les molécules présentes à plus de 1% sont affichées.

| Molécules | Nombre d'occurrences | Pourcentage d'occurrences |
|---|---|---|
| $d_1d_2d_4$ | **14767** | **38,39%** |
| $d_4$ | 6880 | 17,89% |
| $d_1d_3d_2d_4$ | 3683 | 9,58% |
| $d_1$U | 2929 | 7,62% |
| U | 2878 | 7,48% |
| U$d_2d_4$ | 1717 | 4,46% |
| U$d_4$ | 1177 | 3,06% |
| $d_1d_2$ | 1023 | 2,66% |
| $d_1d_3$U | 560 | 1,46% |
| $d_1d_2d_4$U | 471 | 1,22% |

**Étape 8** A cette étape, nous devons retrouver dans le Tube C les molécules $d_4Rev$ de 125 pb et les molécules $Ford_1d_2$ de 220 pb dans le Tube A2 et A3. Nous avons eu un problème de dépôt de marqueur de taille lors du dépôt du Gel 4 de la Figure 3.35. Il ne nous a pas été possible de faire une deuxième fois, car tout l'ADN restant a été utilisé pour continuer la manipulation. Malgré tout, nous pouvons analyser les résultats.

Dans le Tube C, nous obtenons une bande très nette (encadrée en vert) à une taille qui semble correspondre à la molécule attendue ($d_4$) de 125 pb. Nous retrouvons aussi des bandes moins nettes de molécules plus longues, qui pourraient correspondre à des fragments non digérés ($d_1d_2d_4$) de 351 pb et d'autres molécules inattendues. La molécule majoritaire est de la taille attendue ce qui veut dire que la capture aurait bien fonctionné. Les molécules non attendues pourraient être des molécules ($d_1d_2d_4$) non coupées, potentiellement biotinylées et des sous-produits déjà identifiés dans le tube A1 de l'étape 5 (Gel 2). Ce sont les résidus des étapes précédentes. La concentration d'ADN est de 10.0 ng/µL dans 20 µL, soit une quantité suffisante pour être exploitable.

Le Tube A2 contient les molécules récupérées par pipetage après la capture de l'ADN avec extrémités biotinylées. La concentration dans ce tube est de 15.4 ng/µL dans 20 µL, nous avons donc bien réussi à récupérer de l'ADN. Sur le Gel 4 de la Figure 3.35, la bande la plus nette encadrée en vert semble correspondre à la taille attendue (220 pb). Nous retrouvons aussi des bandes plus grandes qui pourraient être des molécules partiellement déchiffrées de l'étape 5 de 363 pb ($d_1d_2d_4$) et des petites bandes de taille similaire aux blocs ($d_4$) non capturés. Le Tube A3 devrait contenir l'ADN ($Ford_1d_2$) récupéré lors des trois lavages de billes. La concentration d'ADN dans ce tube est très faible, 100 à 1000 fois moins importante que dans A2, le contenu de A3 n'est donc pas exploité.

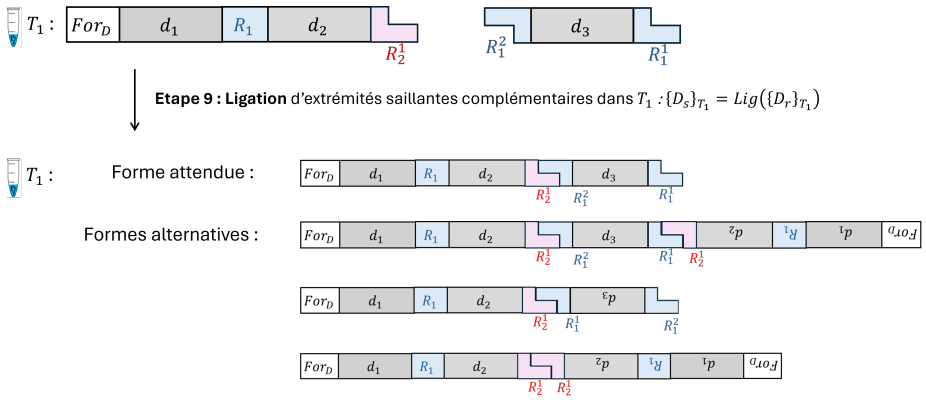



FIGURE 3.36 – Ligation de l'étape 9 créant des molécules de la forme attendue ainsi que des molécules aux formes alternatives, car toutes les extrémités saillantes créées par NheI et SpeI sont cohésives, c'est-à-dire compatibles entre elles.

Globalement, le dépôt sur gel montre que le protocole semble bien fonctionner jusqu'ici. Nous remarquons de plus en plus de produits non désirés au fur et à mesure des étapes. La molécule $(d_4)$ est toujours présente, ainsi que le reliquat de la PCR avec extrémités biotinylées. Nous voyons aussi des molécules $(d_1d_2d_4)$ non digérées par l'enzyme SpeI à l'étape 7 avec des extrémités biotinylées ou non. Ces résultats étaient attendus. En effet, plus on multiplie le nombre d'étapes, plus on accumule les sous-produits non désirés et le risque d'erreurs. Dans le Tableau 3.4, nous observons un rendement de 51% comparé à l'étape 5, ce qui est raisonnable car nous avons effectué une capture et plusieurs purifications.

**Étape 9** Dans le Tube A2B1, la concentration d'ADN est de 15.2 ng/µL dans 20 µL. Le Gel 4 de la Figure 3.35 montre que l'on obtient des molécules de nombreuses tailles. Il y a bien une bande qui semble être de la taille attendue de 333 paires de bases pour $(Ford_1d_2d_3)$. Les autres molécules sont probablement des ligations de multiples molécules $(Ford_1d_2)$ et $(d_3)$ telles que présentées en Figure 3.36. Cela est possible car les extrémités cohésives créées par les enzymes NheI et Spei sont symétriques : deux extrémités identiques peuvent se lier entre elles. En conséquence, $d_2$ et $d_3$ peuvent se lier pour former $(d_2d_3)$, et aussi $(d_3d_2)$. Ce problème est à l'avenir à prendre en considération. Nous présenterons une idée de solution pour éviter ce problème dans la Section 3.3.3.1.

**Étape 10** A cette étape, la dernière ligation a été effectuée. Les modifications de la structure des molécules d'ADN sont terminées. Après purification, la concentration dans le tube final est de 12.7 ng/µL dans 20.0 µL de solution d'eau ultra-pure. L'évolution de la quantité de matière présentée au Tableau 3.4 est logique, avec par exemple un rendement de 83% entre les étapes 9 et 10, étant donnée la purification effectuée à la fin de chaque étape de mélange.

**Étape 11** Après une étape de coupe avec l'enzyme SpeI, les molécules chiffrées et partiellement déchiffrées sont fractionnées en plusieurs molécules, tandis que la molécule déchiffrée n'est pas affectée. Après purification, la concentration dans le tube est de 7,3 ng/µL dans 20 µL d'eau ultra-pure. Un séquençage à cette étape montre que nous avons 53 reads de la forme attendue parmi 63298, *i.e.* $0,08\%$ de molécules déchiffrées. C'est pourquoi nous avons effectuer plusieurs étapes d'optimisation, pour isoler et amplifier

ces molécules. Notons qu'à cette étape, la molécule $d_4$ est largement majoritaire avec $43,24\%$ des occurrences, suivie de formes partielles ou complètes de la molécule $(d_1d_2)$, à $29,64\%$. Une hypothèse pour la présence de ces molécules est le fait que les ligations ne se font pas avec une quantité équimolaire de chaque molécule. Les molécules en excès restent donc dans la solution. À l'avenir, mieux maîtriser ce paramètre permettrait d'améliorer les résultats.

**Étape 12** Trois PCR de 15 cycles avec amorces standards ont été effectuées. Appelées par la suite PCR A, B et C, elles utilisent respectivement 2, 5 et 10 ng d'ADN. Utiliser des quantités de matière différentes pour la PCR a pour but d'observer l'effet de la quantité sur la spécificité de la réaction. En effet, la PCR peut échouer à cause de la faible quantité de molécule déchiffrée à amplifier. Mais il existe aussi un risque de suramplification de molécules de formes non désirées lorsqu'il y a trop de matière. Finalement, comme nous le voyons sur le Gel 1 de la Figure 3.37, la quantité d'ADN n'a eu aucun effet sur la spécificité. Nous remarquons uniquement un effet sur la quantité de matière produite à la fin des réactions, comme en témoigne l'augmentation de l'intensité des bandes avec l'augmentation de la quantité de matrice.

Les bandes représentées en vert en Figure 3.37, qui semblent correspondre à la taille attendue (464 pb), ont été découpées au scalpel. La concentration d'ADN récupéré après découpe a ensuite été évaluée par dosage. Elle est pour la PCR A de 0.14 ng/µL, pour la PCR B de 0,68 ng/µL, et pour la PCR C de 0,74 ng/µL. Nous observons donc une importante perte d'ADN (31,2 ng en tout). Une partie de cette perte est logique. Elle est due au fait que l'on ne récupère que l'ADN d'une taille particulière. Mais cette faible concentration est aussi causée par le rendement de l'extraction sur gel, qui est très faible. Nous obtenons après la purification une concentration de moins de 1 ng/µL dans 20 µL, alors que la concentration normale d'ADN après PCR est de 10 à 100 ng/µL dans 20 µL.

Après avoir récupéré les bandes par découpe sur le gel, l'ADN obtenu après la PCR C est séquencé, car c'est la PCR qui a permis d'obtenir le plus de matière. Notons que nous n'avons pas effectué de dépôt sur gel à cette étape, afin de pouvoir utiliser tout l'ADN pour le séquençage, sans perte supplémentaire. Le résultat du séquençage est présenté dans le Tableau 3.6, avec les pourcentages d'occurrence de chaque molécule parmi les 11145 reads retenus. $14,38\%$ des molécules sont des molécules déchiffrées $(d_1d_2d_3d_4)$. La rotation a donc bien fonctionné. De plus, cette molécule a le deuxième pourcentage d'occurrence derrière les molécules $(d_4)$ qui constituent 19% des reads. $(d_4)$ n'est pas de la taille attendue, elle peut donc être écartée lors du traitement informatique. La présence de $(d_4)$ en majorité est due à plusieurs facteurs. Tout d'abord, des molécules $(d_4)$ sont déjà présentes dans la solution de départ à $32,92\%$. Ensuite, cette molécule est amplifiée linéairement lors de chaque PCR car elle est concaténée à l'amorce $For$. D'autre part, lors de la réaction d'assemblage de l'étape 10, le nombre de molécules $(d_4)$ était probablement en large excès par rapport à $(d_1d_2d_3)$ et aux autres molécules. En effet, si l'on considère que chaque étape précédente s'est parfaitement déroulée, alors il y avait 273,6 ng de $(d_1d_2d_3)$ et donc une quantité d'ADN de 1,384 pmol, comparé à 1,843 pmol de $(d_4)$ (142,0 ng). Ainsi, la molécule $(d_4)$ était en excès d'au moins 25%. Du fait de cette différence, beaucoup de molécules $(d_4)$ n'ont pas trouvé de $(d_1d_2d_3)$ auxquelles se lier et sont restées en excès dans la solution.

Néanmoins, cela n'explique pas comment la molécule $(d_4)$ peut se retrouver dans la solution après la séparation par taille faite au scalpel sur gel. Deux hypothèses sont possibles. La première est la mauvaise séparation physique des molécules lors de la migration sur gel. La forte concentration d'ADN

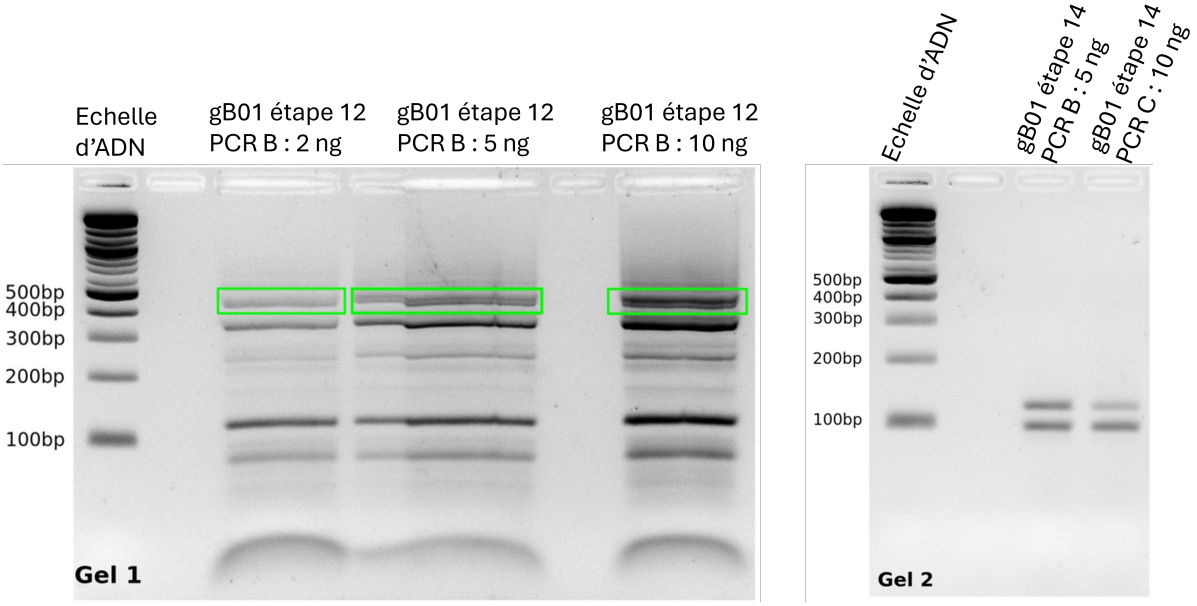



FIGURE 3.37 – Résultat de l'électrophorèse sur gel d'agarose des molécules d'ADN dites "gBlock01" aux étapes 12 et 14 du protocole. La première colonne "DNA ladder" d'échelle. Les bandes encadrées en vert représentent les molécules de la taille souhaitée. 50 µL de solution contenant l'ADN utilisée pour le Gel 1, et 1 µL pour chaque solution dans le Gel 2. Note : le gel est formé à partir d'une solution TBE (Tris/Borate/EDTA) avec 3 % d'agarose finale et la migration.

déposé sur gel pourrait créer des agglutinations de molécules de tailles hétérogènes. En effet, les longues molécules peuvent être des obstacles physiques dans le gel, ralentissant ainsi la migration des molécules plus petites qui tentent de les traverser. Des molécules $(d_4)$ ont donc pu être ralenties et se trouver sur la bande correspondant normalement aux molécules de 460 bases. Un gel différent du gel d'agarose pourrait permettre de mieux séparer les petites molécules. La deuxième hypothèse est la formation de molécules non attendues. Par exemple, l'hybridation de molécules $(d_4)$ avec d'autres via les extrémités cohésives. Ce sont des hybridations fragiles et réversibles par complémentarité des bases. Il n'y a pas de ligation enzymatiques entre les extrémités. D'autre part, des hétérodimères ont pu se former à l'étape 12 lors d'une PCR. Un hétérodimère est une molécule d'ADN où les deux brins ne sont pas alignés, *eg* Figure 3.38. Ces structures secondaires au niveau de l'ADN modifient considérablement la cinétique de migration de la molécule concernée. Elles sont alors ralenties. De par sa structure "tige-boucle", un hétérodimère peut aussi freiner ou s'emmêler avec d'autres molécules. C'est aussi le cas pour les molécules circulaires, formées par la ligation des deux extrémités de la même molécule d'ADN. Ainsi, lors de la découpe dans le gel, nous pouvons retrouver des petites molécules linéaires qui se sont emmêlées à de longues molécules avec des structures tridimensionnelles de type tige-boucle et circulaire.

Ces hypothèses sont à prendre en compte à l'avenir. En ce sens, nous proposerons un protocole amélioré de l'étape 13 de séparation des molécules par taille dans la Section 3.3.3.1.

Enfin, il est possible de relativiser la présence de 19% de molécules $(d_4)$ lorsque l'on considère l'espace occupé par chaque type de molécule dans la solution finale. Pour cela, nous prenons en compte le nombre de bases de chaque molécule. En effet, la molécule déchiffrée représente $14,38\%$ des molécules lues, et prend $27,33\%$ de l'espace total dans la solution finale. C'est la molécule qui prend le plus de place, devant

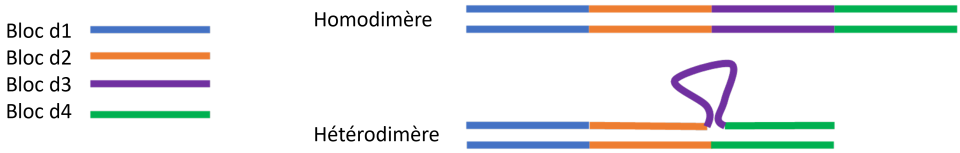



FIGURE 3.38 – Exemple d'hétérodimère : molécule d'ADN où les deux brins ne sont pas alignés.

la molécule chiffrée (de départ) avec $20,20\%$. La molécule $(d_4)$ ne prend que $10,12\%$ de l'espace total. Cette manière de présenter les résultats en nombre total de bases est pertinente dès lors que l'on compare des molécules de taille différentes.

Il est important de noter que le résultat du séquençage montre aussi d'autres molécules que la molécule déchiffrée et $(d_4)$, *cf* Tableau 3.6. $12,88\%$ sont des molécules partiellement déchiffrées $(d_1d_2d_4)$, obtenues à l'étape 5 après la première ligation. Nous pouvons là aussi faire plusieurs hypothèses. Ces molécules n'ont pas été coupées à l'étape 7 ou bien, les fragments coupés ont été mal séparés par capture à l'étape 8 et se sont reformés à l'étape 9. Le séquençage montre aussi que l'on retrouve la molécule chiffrée de départ $(d_1d_3d_2d_4)$ à $10,63\%$. Il est inévitable de retrouver cette molécule car aucune opération n'est totale. De plus, cette molécule est encadrée des deux amorces *For* et *Rev* et a été exponentiellement amplifiée à chaque étape de PCR. Nous retrouvons aussi des fragments inconnus "U" dans les molécules identifiées. Ils correspondent aux 45 à 50 premières ou dernières bases des fragments $d_1$ ou $d_2$. Ainsi, les $7,7\%$ de molécules $d_1U$ correspondent à $d_1$ et les 50 premières bases de $d_2$, tandis que dans 2.39% de $(Ud_2d_4)$, $U$ est composé des 45 dernières bases de $d_1$. Enfin, les 6.53% de fragment inconnu U correspondent aux 50 premières bases de $d_1$. Plusieurs hypothèses peuvent expliquer la présence de ces fragments de blocs. Le primer *Rev* a pu s'hybrider partiellement avec le fragment de $d_2$ et ainsi amplifier seulement une partie du bloc $d_2$ lors des PCR. Ou, un nouveau bloc $d_2$ a pu être créé à la suite d'erreurs introduites par les PCR. À ce jour, ces hypothèses ne sont pas vérifiées, mais elles expliqueraient toutes les formes intermédiaires mineures présentes dans la solution finale.

**Étape 14** A cette étape, nous souhaitons amplifier les molécules déchiffrées récupérées par extraction sur gel, et augmenter la concentration d'ADN, qui est très faible. Une PCR standard de 20 cycles est donc effectuée pour chaque solution PCR A, B et C qui contenaient à l'origine 2 ng, 5 ng et 10 ng d'ADN, respectivement. Après ces PCR, la concentration de la PCR A est trop basse pour être mesurée (inférieur à 0,1 ng), synonyme que la PCR n'a pas fonctionné. Pour les PCR sur B et C nous obtenons respectivement 9.4 ng/µL et 6.2 ng/µL dans 20 µL d'eau ultra-pure. Malheureusement, à cette étape, l'amplification par PCR ne permet pas de décupler la proportion de molécules déchiffrées. En effet, nous avons fait un séquençage pour la PCR C, et nous observons que $56,06\%$ des 232881 reads sont de forme inconnue U ; correspondant aux 55 dernières bases de $d_4$ ; tandis que $38,5\%$ sont des molécules $(d_4)$. La PCR amplifie donc seulement une forme tronquée ou complète de $d_4$ et très peu notre molécule déchiffrée, présente à $0,1\%$. Notre hypothèse est que l'on a un mélange de trop de formes de molécules différentes à l'étape 13, *cf.* Tableau 3.6. En conséquence, il n'est pas possible d'amplifier proprement $(d_1d_2d_3d_4)$ parmi toutes les formes intermédiaires qui s'hybrident mal lors des cycles de PCR. Un prochain protocole consisterait alors à vérifier le contenu de la solution par un dépôt sur gel avant d'envisager une PCR.

TABLE 3.6 – Résultat du séquençage à l'étape 13 du protocole. Nombre et pourcentage d'occurrence de chaque molécule parmi 11145 reads, ainsi que l'espace occupé par chaque type de molécule dans la solution, exprimé en pourcentage de bases ADN totales.

| Molécules | Nombre d'occurrences | Pourcentage d'occurrences | Bases totales | Pourcentage d'espace |
|---|---|---|---|---|
| $d_4$ | 2118 | 19,00% | 264750 | 10.12% |
| $d_1d_2d_3d_4$ | **1603** | **14,38%** | **714938** | **27.33%** |
| $d_1d_2d_4$ | 1435 | 12,88% | 460635 | 17.61% |
| $d_1d_3d_2d_4$ | 1185 | 10,63% | 528510 | 20.20% |
| $d_1$U | 858 | 7,70% | 145860 | 5.58 % |
| U | 728 | 6,53% | 50960 | 1.95 % |
| $d_1d_3$U | 296 | 2,66% | 80216 | 3.07 % |
| U$d_2d_4$ | 266 | 2,39% | 71820 | 2.75 % |
| U$d_4$ | 249 | 2,23% | 42330 | 1.62 % |
| $d_1d_4$ | 232 | 2,08% | 56840 | 2.17 % |
| U$d_2d_3d_4$ | 202 | 1,81% | 74942 | 2.86 % |
| U$d_3d_2d_4$ | 159 | 1,43% | 58989 | 2.25 % |
| $d_1d_2$ | 157 | 1,41% | 34540 | 1.32 % |
| $d_4d_3$ | 137 | 1,23% | 30962 | 1.18 % |

**Résultats finaux**

Dans cette partie de Chapitre, nous avons montré qu'il était possible de faire un rotation de blocs de données stockés sur une molécule. Malgré les nombreuses étapes, nous sommes parvenus à obtenir un pourcentage important de molécules déchiffrées en fin de protocole, avec $14,38\%$ à l'étape 13. Nous avons plus de molécules déchiffrées que de molécules chiffrées, ce qui n'était pas évident. De plus, la molécule déchiffrée est la molécule majoritaire derrière les petites molécules ($d_4$). L'important travail de recherche et développement que nous avons mené a permis plusieurs optimisations pour parvenir à ce protocole (le 5ème en fait). La première version ne permettait pas du tout de retrouver de molécules déchiffrées, tandis que les versions suivantes ne permettaient de retrouver qu'un petit pourcentage de molécule partiellement déchiffrée en milieu de protocole. Le résultat obtenu à l'étape 13 de $14,38\%$ de molécules déchiffrées est donc très positif et permet d'isoler cette molécule déchiffrée, et cela malgré le fait qu'elle soit mélangée à de nombreuses formes intermédiaires.

L'analyse des résultats a aussi mis en évidence plusieurs pistes d'améliorations pour augmenter la proportion de molécules déchiffrées en fin de protocole et diminuer la proportion de molécules contaminantes. Nous revenons sur celles-ci dans la section suivante.

## 3.3.3 Discussion

### 3.3.3.1 Améliorations possibles

**Mélanges équimolaires fins**

Une amélioration possible est d'ajuster plus finement les quantités de molécules dans les réactions d'assemblage, en étant pessimiste sur le nombre de molécules complètes et correctes, de manière à avoir un mélange équimolaire. En particulier, si une molécule partielle est particulièrement présente, telle que

($d_4$) dans notre expérimentation, il faut en tenir compte.

**Extraction sur gel en passant par une dénaturation**

Comme nous l'avons vu, l'extraction sur gel de l'étape 13 ne permet pas de retrouver uniquement des molécules de la taille attendue. Pour améliorer la précision de cette étape, les changements suivants seront à mettre en place. L'ADN doit être dénaturé avant la migration sur gel, afin d'éviter les formes inattendues telles que l'ADN circulaire et les hétérodimères, qui prennent de l'espace et peuvent obstruer la migration de petites molécules. L'électrophorèse se fera sur un gel de polyacrylamide à la place d'un gel d'agarose. Ce type de gel à un maillage plus serré et donc mieux adapté à la séparation de molécules de petites tailles. Enfin, une solution plus diluée est déposée sur le gel pour faciliter la séparation physique des molécules.

Le protocole de l'étape 13 serait alors le suivant :

— Chauffer les échantillons à 98°C dans un tampon avec de l'urée ou du $\beta$-mercaptoéthanol afin de dénaturer ADN et l'empêcher de se renaturer à température ambiante.

— Migration sur gel polyacrylamide SDS-PAGE 8 %. Ce choix de gel sépare plus finement que l'agarose.

— Découpe de la bande d'intérêt au scalpel.

— Renaturation de l'ADN très doucement : 98°C 1' puis 98°C à 20°C avec -1°C par minute.

— Dépôt sur gel d'agarose pour vérifier ce que l'on a maintenant comme molécule, avant d'envisager une PCR.

**Optimisation biologique par le choix des enzymes**

Les enzymes choisis dans notre protocole créent des extrémités saillantes qui sont toutes cohésives. En conséquence, si un tel enzyme coupe une molécule en deux fragments $d_1$ et $d_2$, alors l'extrémité saillante de $d_1$ peut s'hybrider avec une autre copie de $d_1$. Il en est de même pour $d_2$. Cela entraîne des molécules de formes non désirées lors des ligations d'extrémités saillantes. Une optimisation est alors le choix des enzymes pour qu'ils ne créent pas de telles extrémités.

**Optimisation de la séparation par capture d'extrémités biotinylées** La problématique de la séparation d'ADN est une question ouverte en biologie moléculaire. Il existe une nouvelle méthode [182] de séparation de molécules d'ADN par taille, utilisant la microfluidique. Celle-ci permet de séparer les molécules de plusieurs milliers de paires de bases. La preuve de concept effectuée avec des molécules de 48,5k et 166k paires de bases fonctionne très bien. Mais, elle demande aussi une expertise en microfluidique et un laboratoire spécifique. Il y a donc des perspectives pour la séparation de molécules selon leur taille, en plus des méthodes actuelles. Cela est intéressant car, au-delà de notre manipulation, un objectif est d'utiliser de longues molécules d'ADN double brin, afin de pouvoir faire de multiples opérations et d'optimiser la densité des molécules.

Il est aussi possible d'utiliser une technique basée sur des lasers pour libérer l'ADN biotinylé après séparation par capture. En effet, dans le protocole actuel, nous séparons par hydrolyse la biotine de la

streptavidine qui est couplée à une bille magnétique. Or, cette liaison biotine-streptavidine est très forte et stable sur un large spectre de pH et de températures. L'hydrolyse n'est peut-être pas la meilleure méthode et nous soupçonnons qu'un nombre inquantifiable de liaisons restent intactes. Une solution reposerait alors sur des biotines avec un espace clivable les reliant aux amorces. Il est possible de commander des amorces de PCR avec des biotines particulières, les "5' PC Biotin" du fournisseur IDT. Celles-ci ont un bras espaceur photoclivable entre la biotine et les bases ADN. Comme son nom l'indique, ce bras peut être clivé lorsqu'il est exposé à un laser ou une LED d'une longueur d'onde spécifique (300-350 nm). Ainsi, une fois l'ADN capturé avec les billes magnétiques, l'ADN peut être séparé des biotines par exposition à cette longueur d'onde. De plus, l'extrémité clivée dispose d'un groupe phosphate libre et peut donc être utilisée pour des opérations de ligation.

#### 3.3.3.2 Anciennes versions du protocole

Avant de conclure, il nous semble pertinent de revenir sur quelques difficultés rencontrées dans nos premiers protocoles, qui ont constitué nos premières tentatives de détourner des outils biologiques dans un but de sécurité.

Dans une première version de ce protocole, nous utilisions une séparation de molécules d'ADN par hybridation. Cet opérateur est présenté en Section 3.1.6. Pour rappel, des sondes ADN contenant les amorces des brins d'ADN à capturer sont utilisées. Les sondes s'hybrident par complémentarité de séquence à l'ADN ciblé. Ces sondes ADN sont aussi biotinylées et peuvent donc être couplées à un ensemble streptavidine-bille magnétique pour être capturées avec un aimant. Cette étape s'est avérée très complexe à mettre en oeuvre et a démontré un très mauvais rendement. Pour cause, nous souhaitons capturer des molécules d'ADN double brin. Pour qu'un brin puisse s'hybrider à une sonde, il est nécessaire de dénaturer l'ADN double brin. En pratique, l'ADN dénaturé se renature très rapidement, et n'a pas le temps de s'hybrider aux sondes. En effet, la dénaturation avec la température utilisée en V1 est adaptée pour une utilisation dans une étape transitoire de la PCR, mais pas pour une dénaturation qui dure dans le temps. Une publication [183] montre qu'une proportion importante de l'ADN dénaturé se renature au bout de quelques minutes. Dans notre protocole, malgré la dénaturation à très haute température et l'utilisation de glace pour empêcher la renaturation, nous pouvions avoir beaucoup de renaturation de la molécule chiffrée (plus de 70 %). De plus, la capture repose sur l'hybridation entre les sondes ADN et les brins simples. Rien n'empêche les brins simples complémentaires de reformer des brins doubles au lieu de s'attacher aux sondes. Notons que le protocole repose sur la capture simultanée de quatre brins simples ou de deux brins doubles, comme à l'étape 4 de notre protocole. Cela multiplie donc les risques de capturer des mauvais brins par des hybridations de séquences partiellement complémentaires. Ainsi, puisque beaucoup de brins se sont renaturés entre eux et que l'on a des captures non spécifiques, le rendement de l'étape de capture est très mauvais. Cette première version de protocole nous a permis d'obtenir un pourcentage d'occurrence de molécules déchiffrées de $0,29\%$. Nous savions qu'un protocole sans développement et non optimisé ne nous permettrait pas d'obtenir une majorité de molécules déchiffrées, c'est pourquoi nous avons réfléchi à plusieurs optimisations de cette première version.

La deuxième version de notre protocole se proposait d'utiliser une solution chimique à base de soude (NaOH) pour améliorer le rendement de l'étape de capture. La soude attaque les liaisons hydrogènes et est très agressive, elle permet donc de dénaturer l'ADN. Toutefois, la quantité de soude utilisée entraîne

un pH de 14 (basique) dans la solution (1 mol/L). Pour pouvoir renaturer l'ADN, il faut cependant compenser l'effet basique apporté par la soude. Pour cela, une solution tampon phosphate ( 0.1 mol/L pH 7.4 ) a été ajoutée : 40 µL de ce tampon pour 4 µl d'échantillon d'ADN dénaturé. Cette nouvelle solution implique donc de faire varier le pH de la solution, ce qui peut poser plusieurs problèmes lors du protocole. Par exemple, les biotines sont sensibles à l'acide et pourraient être abimées ou dégradées. De plus, équilibrer le pH d'une solution de quelques centaines de microlitres est extrêmement sensible en plus de nécessiter un appareillage coûteux et parfaitement calibré. Après plusieurs tentatives, nous en avons conclu que l'étape de capture passant par une dénaturation posait trop de problème. Il faut utiliser une autre méthode.

La troisième version de ce protocole présentait un changement de stratégie pour rester sur de l'ADN double brin et ainsi éviter les étapes de dénaturation et renaturation. Nous n'utilisons plus de sondes ADN composées de biotines et d'amorces. À la place, les biotines sont ajoutées aux extrémités des molécules d'ADN double brin par PCR. Une autre optimisation est l'accroissement de la quantité d'ADN utilisée en séparant le produit PCR en 5 tubes, puisque nous savons que nous allons perdre beaucoup de produit au cours des étapes. D'autre part, nous ajoutons une étape de PCR au milieu du protocole pour amplifier et sélectionner la molécule partiellement déchiffrée attendue. Enfin, nous ajoutons à ce protocole une étape de purification avant chaque digestion enzymatique, pour se débarrasser de la solution tampon dans laquelle la PCR a été faite et optimiser le rendement de cette étape. En effet, d'après le fournisseur NEB, certains enzymes ont un très mauvais rendement lorsqu'ils sont utilisés dans la même solution que celle utilisée pour faire une PCR. Pour optimiser les étapes de coupe avec les enzymes de restriction SpeI et NheI, nous utilisons un excès d'enzymes dans les solutions par rapport aux quantités recommandées (une centaine d'unités par microgramme d'ADN au lieu de 10).

Lors de la V4, nous avons eu un problème lors du séquençage des échantillons. Tout d'abord, le séquençage est une méthode destructive. Autrement dit,les échantillons sont perdus durant l'opération. Ensuite, compte tenu des faibles quantités de matière dont nous disposions, l'ensemble des produits à analyser ont été séquencés ne laissant aucun reliquat. Nous n'avions pas le droit à un deuxième essai. Malheureusement, ce jour-là, nous avons rencontré un problème technique lors de la préparation des échantillons au séquençage. Les résultats produits par celui-ci ont été décevants avec très peu de lectures de bonne qualité à analyser. Malgré tout, en tendance, ces résultats nous ont confirmé que nous étions sur la bonne voie d'optimisation du protocole.

Pour la V5, nous avons testé deux fins alternatives, à partir de l'étape 11, *i.e.* après la fin des étapes de rotation. La stratégie retenue dans le protocole était de faire une séparation après un dépôt sur gel pour découper la bande d'ADN de bonne taille. L'alternative consistait à faire 2 PCR, une purification, une capture avec des amorces biotinylées *For* et *Rev*, une nouvelle purification, et enfin une PCR et une purification finale. L'objectif de ce protocole est d'amplifier et d'isoler les molécules avec les amorces *For* et *Rev*. La multiplication d'étapes de PCR permet d'amplifier exponentiellement les molécules qui ont les deux amorces, tandis que les molécules avec une seule amorce sont amplifiées linéairement. La première PCR nous a bien permis d'amplifier des molécules de la taille attendue, tout en amplifiant aussi des molécules d'autres tailles. La seconde PCR ne permet pas d'améliorer les résultats, nous obtenons $2,16\%$ de molécules déchiffrées à cette étape. Après la capture, les résultats ne sont pas bons. Nous n'avons pas eu de meilleurs résultats qu'aux étapes précédentes avec des molécules de beaucoup de formes toujours

présentes. À la fin du protocole, c'est-à-dire après la PCR finale, nous avons obtenu un "smear" sur le dépôt sur gel, ce qui correspond à beaucoup de molécules de tailles variées. Les résultats du séquençage montrent que l'on obtient $1,46\%$ de molécules déchiffrées. La multiplication des PCR dans cette fin alternative de version 5 n'a donc pas permis d'isoler les molécules déchiffrées.

Pour conclure, nous avons développé un protocole permettant d'effectuer une rotation biomoléculaire dans une molécule d'ADN. La rotation est la fonction la plus complexe que nous avons développé : elle repose sur deux cycles de coupe, séparation et ligation, tandis que la permutation repose sur un unique cycle. Ainsi, une preuve de concept de la rotation peut raisonnablement nous indiquer que la permutation fonctionnerait aussi. Ce protocole expérimental est le résultat de nombreuses réflexions et optimisations d'opérateurs biologiques qui ne sont pas typiquement voués à cet usage. Dans la partie expérimentale, nous avons prouvé que l'on peut déchiffrer une molécule et la retrouver. En particulier, la molécule déchiffrée représente $14,38\%$ des occurrences en fin de protocole, c'est la molécule majoritaire après une molécule de petite taille. Ce pourcentage représente une forte amélioration lorsque nous le comparons à nos premiers résultats, où nous ne retrouvions la molécule qu'à $0,29\%$. Malgré le nombre important d'étapes de notre protocole, qui apportent chacune de l'incertitude et un taux d'erreur, notre travail d'optimisation du protocole a permis d'obtenir un résultat satisfaisant. D'autre part, nous avons aussi présenté plusieurs pistes d'améliorations basées sur notre analyse des résultats, qui pourront encore augmenter le rendement de notre protocole.

## 3.4 Conclusion

Dans ce chapitre, nous avons présenté une solution exploratoire de sécurité basée sur des opérateurs biologiques, et montré qu'il est possible de déchiffrer une molécule d'ADN avec les fonctions biologiques avancées que nous avons développé.

Dans la première partie de ce chapitre, nous avons identifié des opérateurs biologiques élémentaires afin de manipuler de l'ADN, et exprimé ceux-ci sous forme de fonctions. Plusieurs classes d'opérateurs ont été présentées. Un opérateur élémentaire peut modifier les extrémités d'une molécule d'ADN double brin, couper une telle molécule en plusieurs fragments selon des motifs précis, lier des brins doubles dès lors qu'elles ont des extrémités franches, ou bien lorsque leurs extrémités saillantes sont compatibles. Un opérateur peut aussi modifier certaines bases de l'alphabet $\{A, C, T, G\}$ des séquences ADN, amplifier certaines molécules ADN par PCR, ou encore séparer des molécules spécifiques selon leur taille, la présence de sous-séquences, ou bien de biotines à leurs extrémités. A partir de certains de ces opérateurs biologiques, nous avons développé des fonctions plus avancées de rotation et de permutation. Celles-ci permettent respectivement d'échanger les positions de deux blocs adjacents dans une molécule d'ADN, et d'échanger deux blocs de deux molécules d'ADN différentes. Elles sont donc similaires à des fonctions numériques de rotation et de permutation de blocs de données, mais chaque opérateur élémentaire qui les compose a une paramétrabilité et une complexité spécifiques au support ADN.

Dans une deuxième partie, nous avons présenté une solution de sécurité DNACipher, qui impose des manipulations biologiques pour déchiffrer les données stockées dans l'ADN avant de pouvoir les séquencer. Pour cela, notre solution de chiffrement repose sur des rotations et permutations différentes

appliquées à des copies de notre séquence ADN. Son but est de perturber l'étape bio-informatique de consensus qui suit le séquençage. Pour retrouver la séquence consensus et donc les données chiffrées, il faut déchiffrer toutes les copies chiffrées via les opérations inverses de rotation et de permutation. Ainsi, le déchiffrement biomoléculaire DNADecipher doit être effectué avant le séquençage. Il repose sur les deux fonctions avancées de rotation et de permutation que nous avons développé. Ces fonctions nécessitent de maîtriser la structure des séquences ADN qui encodent de l'information, en raison des opérateurs de coupe qui identifient des motifs particuliers. Nous avons donc encodé de l'information avec notre algorithme DSWE, présenté en Section 2.3 du Chapitre 2. Les séquences ADN encodées avec DSWE sont ensuite chiffrées avec notre algorithme DNACipher, qui applique différentes permutations et rotations à des copies des séquences ADN. Celles-ci sont ensuite écrites dans des molécules d'ADN par synthèse, et stockées. Pour récupérer les données stockées dans l'ADN, il faut effectuer le déchiffrement biomoléculaire DNADecipher avec des rotations et permutations biologiques. Cela permet d'aligner les blocs de données de chaque molécule, qui peuvent ensuite être séquencées et traitées par l'algorithme bio-informatique de consensus. Ce dernier permet de retrouver les séquences ADN encodant l'information, qui sont décodées avec DSWE. Cette solution de sécurité reste une première ébauche, qui pourrait être complexifiée.

Dans une troisième partie, nous avons présenté plusieurs résultats expérimentaux. Tout d'abord, nous avons testé notre algorithme de sécurité DNACipher à l'aide d'un simulateur des étapes biologiques de la chaîne. Nous avons montré que notre chiffrement DNACipher bloque l'étape de consensus, qui ne peut retrouver de séquence parmi les données chiffrées. Une possible contre-mesure pour un utilisateur est une étape supplémentaire de clustering des séquences chiffrées avant consensus, elle requiert toutefois de disposer d'un grand nombre de copies de chaque séquence et n'est pas garantie. Dans un second temps, nous avons testé la viabilité de notre solution de sécurité par une preuve de concept biologique. Pour cela, nous avons encodé de l'information avec DSWE puis dans une molécule d'ADN chiffrée avec une rotation, la plus complexe des deux fonctions que nous avons développé, et nous avons montré qu'il est possible de déchiffrer cette molécule par des manipulations biologiques. Nous avons séquencé le résultat et retrouvé la molécule déchiffrée pour 14% des occurrences, c'est la molécule majoritaire dans la solution finale, après un bloc isolé de données. Cette manipulation est une première preuve de concept de sécurité de données dans l'ADN à l'aide de manipulations biologiques, ce qui à notre connaissance, est inédit. Ce résultat est une première ébauche qui peut être améliorée. Concernant la rotation, nous avons déjà identifié et détaillons plusieurs pistes d'améliorations. Nous avons aussi donné des pistes d'amélioration du protocole biologique, afin d'augmenter le rendement de cette fonction. En effet, en partant de rien et en détournant l'usage normal de plusieurs opérateurs biologiques, nous avons réussi avec un long travail d'optimisation à effectuer une rotation dans une molécule d'ADN. Le résultat de notre cinquième expérimentation est encourageant, comparé à nos premières tentatives.

Ainsi, nous avons montré qu'il est possible d'effectuer une fonction numérique de rotation avec des opérateurs biologiques, dans le but de sécuriser les données stockées dans l'ADN, ce qui, à notre connaissance, n'a jamais été effectué auparavant.

# CONCLUSION GÉNÉRALE ET PERSPECTIVES

Les travaux de cette thèse ont eu pour but de sécuriser les données stockées dans des molécules d'ADN. En particulier, nous avons exploré comment assurer la confidentialité des données stockées dans l'ADN avec une solution intégrée nativement à la technologie de stockage. Dans ce cadre, nous avons apporté plusieurs contributions.

Dans le premier chapitre de cette thèse, nous avons fourni des définitions générales des procédés numériques et biologiques de la chaîne de stockage de données dans l'ADN, et les contraintes qu'ils imposent sur la structure de l'ADN et l'encodage de l'information. Ce cadre nous a permis d'identifier la menace et les risques qui pèsent sur la chaîne de stockage, sur la base d'un scénario d'usage que nous pensons représentatif du déploiement d'une telle chaîne. L'analyse de risques STRIDE que nous avons effectuée présente un premier résultat, inédit à notre connaissance. Il apparaît que la chaîne de stockage présente des vulnérabilités à toutes ses étapes, avec des enjeux en termes de confidentialité, de disponibilité, d'intégrité et de traçabilité ; enjeux qui pèsent non seulement sur les données, mais aussi sur les molécules et les manipulations biologiques. Nous avons effectué un état des lieux des solutions de traitement du risque, en présentant les méthodes de sécurité existantes, tant numériques que biologiques. Après étude des méthodes spécifiques aux données biologiques, nous avons conclu qu'elles présentent des solutions de sécurité partielles, théoriques, ou avec un intérêt limité pour le domaine étudié, *i.e.* le stockage de données dans l'ADN. En conséquence,il nous a semblé pertinent d'approfondir les travaux existants et de développer une solution assurant la confidentialité des données stockées dans l'ADN à long terme, à l'aide de manipulations biologiques et en prenant en compte les contraintes de ce support.

Dans le Chapitre 2, nous avons présenté un algorithme d'encodage de données dans l'ADN que nous avons développé afin de pouvoir maîtriser la structure de l'ADN et le manipuler pour garantir la sécurité. En effet, certaines manipulations biologiques reposent sur l'utilisation de motifs dans une séquence ADN, afin notamment de couper et lier des molécules à des positions précises. Dans ce chapitre, après avoir détaillé les contraintes biologiques à respecter pour encoder des données en séquences ADN, nous avons présenté un état de l'art des méthodes de codage existantes. Nous avons distingué plusieurs classes, selon le nombre et la variabilité des contraintes prises en compte, qui sont : une taille maximale d'homopolymères de $N$ bases, un G-C content équilibré, et un nombre $M$ de motifs interdits de $m$ bases. Nous avons présenté notre solution, qui repose sur un nouvel encodage dynamique par fenêtre glissante (DSWE) permettant : i) d'éviter les homopolymères de plus de $N$ bases en encodant des données binaires de taille fixe dans des mots code ADN de longueur variable, et ii) d'empêcher l'apparition des motifs interdits pendant l'encodage de la séquence ADN à l'aide d'une fenêtre glissante. Après avoir exposé notre méthode, nous avons donné plusieurs résultats expérimentaux, dont une comparaison aux méthodes de l'état de l'art et tout particulièrement mCGR, la seule méthode qui prend en compte les mêmes contraintes que DSWE.

Il s'avère que pour tout $N > 2$, notre méthode DSWE présente un meilleur taux d'information. Enfin, en utilisant un récent simulateur de chaîne de stockage d'ADN, nous montrons que la récupération des données n'est pas affectée par notre DSWE.

Dans le troisième chapitre, nous avons proposé une solution qui intègre nativement la sécurité à la technologie de stockage dans l'ADN. Pour cela, nous avons développé deux fonctions biologiques de rotation et de permutation, qui modifient la structure de molécules d'ADN encodant de l'information. Ces fonctions sont basées sur des opérateurs biologiques élémentaires que nous avons identifiés dans la première partie du Chapitre 3. Les fonctions de rotation et de permutation permettent respectivement d'échanger les positions de deux blocs adjacents dans une molécule d'ADN, et d'échanger deux blocs de deux molécules d'ADN différentes. Sur la base de ces fonctions, nous avons développé une solution de sécurité DNACipher, qui impose des manipulations biologiques pour déchiffrer les données stockées dans l'ADN avant de les séquencer. Ainsi, nous avons présenté une chaîne sécurisée de stockage dans l'ADN, avec un encodage numérique DSWE qui prend en compte les contraintes du support ADN et un chiffrement biomoléculaire intégré nativement à la technologie de stockage, qui s'inscrit dans une perspective de sécurité à long terme. Nous avons testé notre solution de sécurité à l'aide d'un simulateur des étapes biologiques de la chaîne, et montré que notre chiffrement DNACipher bloque la bonne réalisation de l'étape de consensus qui suit le séquençage. Nous avons aussi démontré la viabilité de notre solution de sécurité par une preuve de concept biologique. Pour cela, nous avons encodé de l'information avec DSWE et chiffré la séquence avec une rotation numérique, avant de la synthétiser. Au terme d'un long travail d'optimisation d'opérateurs biologiques qui ne sont pas typiquement voués à cet usage, nous avons montré qu'il est possible de déchiffrer cette molécule par une rotation biologique ; rotation qui est la plus complexe des deux fonctions que nous avons développées. Cette sécurisation de données dans l'ADN à l'aide de manipulations biologiques n'a, à notre connaissance, jamais été réalisée.

Comme nous l'avons vu, le stockage de données dans l'ADN est en plein développement. Pour l'instant, les manipulations sont coûteuses en temps et en matériel, et demandent une expertise en biologie moléculaire. Cependant, compte tenu des investissements dans le stockage de données sur ADN, les coûts de synthèse sont en forte baisse, avec un objectif fixé à un dollar pour un tera-octet de données d'ici 2030 par la DNA Data Storage Alliance. D'autre part, l'ADN est un support de stockage qui présente plusieurs avantages spécifiques. Tout d'abord, comparé à d'autres polymères qui pourraient permettre de disposer de plus de 4 bases, l'ADN est un support non obsolescent. Il y aura toujours de l'ADN partout, et donc toujours des moyens de lire cet ADN, *i.e.* des séquenceurs. De plus, l'ADN est particulièrement stable, et ne se dégrade pas dans le temps si certaines conditions basiques sont réunies. Enfin, des robots commencent à être développés afin d'automatiser certaines étapes biologiques. Microsoft dispose par exemple d'un système [184] entièrement automatisé permettant de stocker et de récupérer des données dans de l'ADN synthétique. Cette évolution permet un gain de temps, d'argent et de main d'oeuvre, et démocratise l'utilisation de machines de synthèse et séquençage pour des non-spécialistes et donc l'accès à la technologie de stockage dans l'ADN.

A l'issue de cette thèse, le constat est le suivant : La sécurisation des données stockées dans les molécules d'ADN étant un sujet nouveau, plusieurs de nos contributions sont de premiers travaux qui peuvent être améliorés. Voici donc des éléments et des perspectives qui pourraient permettre de poursuivre la réflexion engagée.

Tout d'abord, la première analyse de risques présentée au Chapitre 1 pose les bases en matière d'identification et de traitement du risque. Celle-ci reste une première ébauche, qui pourra être complétée en affinant ou modifiant le scénario d'usage. De plus, cette analyse de risque devra être mise à jour, car l'identification de menaces est un processus qui doit être régulièrement effectué, et adapté aux technologies spécifiques utilisées. En effet, de nouvelles menaces ou failles peuvent survenir, le traitement du risque doit alors s'adapter aux évolutions du Système d'Information. Nous pouvons citer un exemple d'évolution du système : les machines de synthèse pourront dans le futur être manipulées plus facilement, en théorie sans avoir besoin d'un expert en biologie de synthèse [184]. Cela entraîne une démocratisation de l'accès aux technologies et donc au stockage de données dans l'ADN, mais cela implique aussi de nouvelles menaces sur le système. En effet, les technologies sont alors moins maîtrisées sur site, et plus facile d'accès. D'autre part, il est toujours plus facile de prévenir le risque que de gérer les conséquences d'attaques sur un système. Dans les réalisations de cette thèse nous nous sommes concentrés principalement sur la confidentialité des données, un des risques principaux identifiés dans le chapitre 1. Certains risques ne sont donc pas pris en compte, et restent à explorer. Nous pouvons citer la traçabilité comme problème de sécurité ouvert, notamment en raison de la facilité à copier des molécules d'ADN à l'identique.

Dans le deuxième chapitre, nous avons présenté une solution d'encodage dynamique, qui génère à partir de données binaires chiffrées des séquences ADN avec un G-C content équilibré, une taille d'homopolymères de $N$ bases maximum, et interdit $M$ motifs de $m$ bases. Nous avons testé notre encodage avec une taille variable $N$ et pour au maximum $M = 76$ motifs interdits de $m = 6$ bases. Une généralisation possible de notre encodage est la prise en compte des motifs interdits de tailles distinctes, chacun avec une fenêtre glissante adaptée, au prix toutefois d'une augmentation de la complexité. Nous avons aussi présenté quelques résultats d'implémentation de notre encodage dynamique avec des codes correcteurs. Toutefois, l'utilisation de codes correcteurs avec un codage dynamique reste une problématique ouverte.

Dans le chapitre 3, nous avons présenté une chaîne sécurisée de stockage dans l'ADN, reposant sur un déchiffrement biomoléculaire. Ce déchiffrement repose sur plusieurs opérateurs élémentaires, bien connus des biologistes, mais nous avons souhaité les utiliser dans un nouveau cadre, dans un but de sécurité. Ce nouvel usage est donc à développer, afin d'optimiser le rendement de ces opérateurs, qui restent complexes en pratique. D'autre part, n'avons pas utilisé tous les opérateurs présentés dans ce chapitre. Afin de complexifier notre solution de sécurité qui reste une première ébauche, nous pourrions utiliser des opérateurs tels que les modifications de bases ADN. D'une part, cela permettrait de complexifier l'alphabet classique de 4 bases A, C, G, et T utilisé classiquement pour encoder de l'information. D'autre part, certains opérateurs comme la coupe d'ADN avec des enzymes de restriction sont sensibles aux modifications de bases. Ainsi, pour pouvoir effectuer une rotation dans une molécule d'ADN, on pourrait imaginer de modifier certaines bases de l'ADN et donc le niveau de lecture de la molécule d'ADN, avant certaines coupes. Nous pouvons aussi citer les modifications des extrémités de l'ADN, qui permettent d'empêcher ou de faciliter une ligation de l'ADN. De tels opérateurs permettent d'ajouter de la complexité à un algorithme de chiffrement. Toutefois, il faut noter que chaque opérateur apporte une complexité propre, par exemple certaines modifications de bases ne survivent pas à l'amplification par PCR. Enfin, dans la perspective de sécuriser les données stockées dans l'ADN avec une solution biologique, nous pourrions nous intéresser à d'autres propriétés que la confidentialité. Les opérateurs identifiés dans le Chapitre 3 pourraient aussi être utiles pour la traçabilité, en modifiant certaines bases pour tatouer des

molécules d'ADN, par exemple.

# BIBLIOGRAPHIE


1. Taylor, P., *Amount of data created, consumed, and stored 2010-2020, with forecasts to 2025* en, 2023, https://www.statista.com/statistics/871513/worldwide-data-created/ (2024).
2. Castillo, M., From hard drives to flash drives to DNA drives, *American Journal of Neuroradiology* **35,** 1-2 (2014).
3. Schwyter, A., *Fini les data centers, place aux données stockées sur l'ADN* fr-FR, 2021, https://www.challenges.fr/high-tech/fini-les-data-centers-place-aux-donnees-stockees-sur-l-adn_744391 (2024).
4. Siddik, M. A. B. *et al.,* The environmental footprint of data centers in the United States, *Environmental Research Letters* **16,** 064017 (2021).
5. De Silva, P. Y. & Ganegoda, G. U., New trends of digital data storage in DNA, *BioMed research international* **2016** (2016).
6. Organick, L. *et al.,* Probing the physical limits of reliable DNA data retrieval, *Nature communications* **11,** 616 (2020).
7. Willerslev, E. *et al.,* Long-term persistence of bacterial DNA, *Current Biology* **14,** R9-R10 (2004).
8. Orlando, L. *et al.,* Revisiting Neandertal diversity with a 100,000 year old mtDNA sequence, *Current Biology* **16,** R400-R402 (2006).
9. Orlando, L. *et al.,* Recalibrating Equus evolution using the genome sequence of an early Middle Pleistocene horse, *Nature* **499,** 74-78 (2013).
10. Kruglyak, K. M. *et al.,* Next-generation sequencing and applications to the diagnosis and treatment of lung cancer, *Lung cancer and personalized medicine : Novel therapies and clinical management,* 123-136 (2016).
11. *DNA Storage Alliance Website* en-US, https://dnastoragealliance.org/ (2023).
12. Neiman, M., On the molecular memory systems and the directed mutations, *Radiotekhnika* **6,** 8 (1965).
13. Davis, J., Microvenus, *Art Journal* **55,** 70-74 (1996).
14. Wong, P. C. *et al.,* Organic data memory using the DNA approach, *Communications of the ACM* **46,** 95-98 (2003).
15. Ailenberg, M. & Rotstein, O. D., An improved Huffman coding method for archiving text, images, and music characters in DNA, *Biotechniques* **47,** 747-754 (2009).
16. Gibson, D. G. *et al.,* Creation of a bacterial cell controlled by a chemically synthesized genome, *Science* **329,** 52-56 (2010).

17. Church, G. M. *et al.,* Next-generation digital information storage in DNA, *Science* **337,** 1628-1628 (2012).

18. Goldman, N. *et al.,* Towards practical, high-capacity, low-maintenance information storage in synthesized DNA, *Nature* **494,** 77-80 (2013).

19. Clermont, D. *et al.,* Assessment of DNA Encapsulation, a New Room-Temperature DNA Storage Method, *Biopreservation and Biobanking* **12,** 176-183 (2014).

20. Paunescu, D. *et al.,* Reversible DNA encapsulation in silica to produce ROS-resistant and heat-resistant synthetic DNA'fossils', *Nature protocols* **8,** 2440-2448 (2013).

21. Bonnet, J. *et al.,* Chain and conformation stability of solid-state DNA : implications for room temperature storage, *Nucleic Acids Research* **38,** 1531-1546 (2010).

22. Organick, L. *et al.,* Random access in large-scale DNA data storage, *Nature biotechnology* **36,** 242-248 (2018).

23. Yu, M. *et al.,* High-throughput DNA synthesis for data storage, *Chemical Society Reviews* (2024).

24. Singh, J., Cyber-attacks in cloud computing : A case study, *International Journal of Electronics and Information Engineering* **1,** 78-87 (2014).

25. ANSSI, *Panorama de la cybermenace 2022 – CERT-FR* 2022, `https://www.cert.ssi.gouv.fr/cti/CERTFR-2023-CTI-001/` (2023).

26. *Business Insights and Trends 2023* en, `https://www.gartner.com/en/insights` (2023).

27. Security, I., *Cost of a data breach 2023 | IBM* en-us, 2023, `https://www.ibm.com/reports/data-breach` (2023).

28. *Technologies de l'information — Techniques de sécurité — Systèmes de management de la sécurité de l'information — Vue d'ensemble et vocabulaire* Standard (International Organization for Standardization (ISO), fév. 2018).

29. Kosuri, S. & Church, G. M., Large-scale de novo DNA synthesis : technologies and applications, *Nature methods* **11,** 499-507 (2014).

30. Yazdi, S. H. T. *et al.,* Portable and error-free DNA-based data storage, *Scientific reports* **7,** 5011 (2017).

31. Bornholt, J. *et al., A DNA-based archival storage system* in *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems* (2016), 637-649.

32. Grass, R. N. *et al.,* Robust chemical preservation of digital information on DNA in silica with error-correcting codes, *Angewandte Chemie International Edition* **54,** 2552-2555 (2015).

33. Ross, M. G. *et al.,* Characterizing and measuring bias in sequence data, *Genome biology* **14,** 1-20 (2013).

34. Kadri, K., Polymerase chain reaction (PCR) : principle and applications, *Synthetic Biology-New Interdisciplinary Science* (2019).

35. Erlich, Y. & Zielinski, D., DNA Fountain enables a robust and efficient storage architecture, *science* **355,** 950-954 (2017).

36. Wang, Y. *et al.,* Construction of bio-constrained code for DNA data storage, *IEEE Communications Letters* **23,** 963-966 (2019).

37. Ping, Z. *et al.,* Towards practical and robust DNA-based data archiving using the yin–yang codec system, *Nature Computational Science* **2,** 234-242 (2022).

38. Löchel, H. F. *et al.,* Fractal construction of constrained code words for DNA storage systems, *Nucleic acids research* **50,** e30-e30 (2022).

39. Grass, R. N. *et al.,* Genomic encryption of digital data stored in synthetic DNA, *Angewandte Chemie* **132,** 8554-8558 (2020).

40. Deng, L. *et al.,* Optimized code design for constrained DNA data storage with asymmetric errors, *IEEE Access* **7,** 84107-84121 (2019).

41. Gabrys, R. *et al.,* Asymmetric Lee distance codes for DNA-based storage, *IEEE Transactions on Information Theory* **63,** 4982-4995 (2017).

42. Hamoum, B. & Dupraz, E., Channel Model and Decoder with Memory for DNA Data Storage with Nanopore Sequencing, *IEEE Access* (2023).

43. Berrou, C. & Glavieux, A., Near optimum error correcting coding and decoding : Turbo-codes, *IEEE Transactions on communications* **44,** 1261-1271 (1996).

44. MacKay, D. J. & Neal, R. M., Near Shannon limit performance of low density parity check codes, *Electronics letters* **33,** 457-458 (1997).

45. Yan, Z. *et al.,* A Segmented-Edit Error-Correcting Code With Re-Synchronization Function for DNA-Based Storage Systems, *IEEE Transactions on Emerging Topics in Computing* (2022).

46. Chandak, S. *et al., Improved read/write cost tradeoff in DNA-based data storage using LDPC codes* in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)* (2019), 147-156.

47. Hamoum, B. *et al., Channel Model with Memory for DNA Data Storage with Nanopore Sequencing* in *2021 11th International Symposium on Topics in Coding (ISTC)* (2021), 1-5.

48. Masaki, Y. *et al.,* Quantification of synthetic errors during chemical synthesis of DNA and its suppression by non-canonical nucleosides, *Scientific Reports* **12,** 12095 (2022).

49. Coatrieux, G. *et al., Storing the declaration of human rights on one data DNA molecule* in *CominLabs day 2022* (2022), 1-1.

50. Lee, H. H. *et al.,* Terminator-free template-independent enzymatic DNA synthesis for digital information storage, *Nature communications* **10,** 2383 (2019).

51. *Resources* en-US, `https://dnascript.com/enzymatic-dna-synthesis-eds-technology/` (2021).

52. Wang, Y. *et al.,* Nanopore sequencing technology, bioinformatics and applications, *Nature biotechnology* **39,** 1348-1365 (2021).

53. Jain, M. *et al.,* The Oxford Nanopore MinION : delivery of nanopore sequencing to the genomics community, *Genome biology* **17,** 1-11 (2016).

54. Ceze, L. *et al.,* Molecular digital data storage using DNA, *Nature Reviews Genetics* **20,** 456-466 (2019).

55. *gBlocks and gBlocks HiFi Gene Fragments | IDT* en, `https://eu.idtdna.com/pages/products/genes-and-gene-fragments/double-stranded-dna-fragments/gblocks-gene-fragments` (2023).

56. Stemmer, W. P. *et al.,* Single-step assembly of a gene and entire plasmid from large numbers of oligodeoxyribonucleotides, *Gene* **164,** 49-53 (1995).

57. *GeneArt™ Instant™ Designer* `https://www.thermofisher.com/order/gene-dna-fragments-design/index.html` (2024).

58. Lee, S. B. *et al.,* Optimizing storage and handling of DNA extracts, *Forensic DNA analysis : Current practices and emerging technologies, ed. Jaiprakash G. Shewale, and Ray H. Liu,* 19-38 (2013).

59. Ping, Z. *et al.,* Carbon-based archiving : current progress and future prospects of DNA-based data storage, *GigaScience* **8,** giz075 (2019).

60. Tanaka, N. *et al.,* Sequencing artifacts derived from a library preparation method using enzymatic fragmentation, *PLoS One* **15,** e0227427 (2020).

61. Mertes, F. *et al.,* Targeted enrichment of genomic DNA regions for next-generation sequencing, *Briefings in functional genomics* **10,** 374-386 (2011).

62. Potapov, V. & Ong, J. L., Examining sources of error in PCR by single-molecule sequencing, *PloS one* **12,** e0169774 (2017).

63. Wozniak, A. V. A., *A systematic and extensible approach to DNA primer design for whole gene synthesis* thèse de doct. (Massachusetts Institute of Technology, 2005).

64. Chuang, L.-Y. *et al.,* Specific primer design for the polymerase chain reaction, *Biotechnology letters* **35,** 1541-1549 (2013).

65. Biolabs, N. E., *A Typical DNA Tailing Reaction Protocol* en, 2012, `https://www.neb.com/en/protocols/0001/01/01/a-typical-dna-tailing-reaction` (2024).

66. Valencia, C. A. *et al.,* Sanger sequencing principles, history, and landmarks, *Next Generation Sequencing Technologies in Medical Genetics,* 3-11 (2013).

67. Shendure, J. & Ji, H., Next-generation DNA sequencing, *Nature biotechnology* **26,** 1135-1145 (2008).

68. Esposito, A. M. *et al.,* Phylogenetic Diversity of Animal Oral and Gastrointestinal Viromes Useful in Surveillance of Zoonoses, *Microorganisms* **10,** 1815 (2022).

69. Wick, R. R. *et al.,* Performance of neural network basecalling tools for Oxford Nanopore sequencing, *Genome biology* **20,** 1-10 (2019).

70. Ahsan, M. U. *et al.,* A signal processing and deep learning framework for methylation detection using Oxford Nanopore sequencing, *Nature Communications* **15,** 1448 (2024).

71. Lee, H. *et al.,* Third-generation sequencing and the future of genomics, *BioRxiv,* 048603 (2016).

72. *The power of Q20+ chemistry* en, Section : Flexible, déc. 2021, `https://nanoporetech.com/q20plus-chemistry` (2023).

73. Xu, Z. *et al.,* Fast-bonito : a faster deep learning based basecaller for nanopore sequencing, *Artificial Intelligence in the Life Sciences* **1,** 100011 (2021).

74. Delahaye, C. & Nicolas, J., Sequencing DNA with nanopores : Troubles and biases, *PloS one* **16,** e0257521 (2021).

75. Notredame, C. *et al.,* T-Coffee : A novel method for fast and accurate multiple sequence alignment, *Journal of molecular biology* **302,** 205-217 (2000).

76. Edgar, R. C., MUSCLE : multiple sequence alignment with high accuracy and high throughput, *Nucleic acids research* **32,** 1792-1797 (2004).

77. Lassmann, T., *Kalign 3 : multiple sequence alignment of large datasets* 2020.

78. Lavenier, D., Constrained Consensus Sequence Algorithm for DNA Archiving, *arXiv preprint arXiv :2105.04993* (2021).

79. Baker, E. A. G. *et al.,* Silico : a simulator of long read sequencing in PacBio and Oxford Nanopore, *BioRxiv,* 076901 (2016).

80. Wick, R. R., Badread : simulation of error-prone long reads, *Journal of Open Source Software* **4,** 1316 (2019).

81. Li, Y. *et al.,* DeepSimulator : a deep simulator for Nanopore sequencing, *Bioinformatics* **34,** 2899-2908 (2018).

82. *Mission 4 - Contrôler et sanctionner* fr, `https://www.cnil.fr/fr/mission-4-controler-et-sanctionner` (2023).

83. *Note scientifique de l'office - Le stockage de données sous la forme d'ADN* Scientific note (Office parlementaire d'évaluation des choix scientifiques et technologiques, 2021), `https://www.senat.fr/rap/r21-285/r21-2851.pdf`.

84. 14 :00-17 :00, *Norme ISO/IEC 27001 – Systèmes de management de la sécurité de l'information* fr, `https://www.iso.org/fr/standard/27001` (2023).

85. Générale de la Sécurité intérieure, D., *L'état de la menace cyber en France* fr, `https://www.dgsi.interieur.gouv.fr/la-dgsi-a-vos-cotes/cyberdefense/letat-de-la-menace-cyber-en-france` (2023).

86. Katz, J., *Digital signatures* (Springer, 2010).

87. Hellman, M. E., An overview of public key cryptography, *IEEE Communications Magazine* **40,** 42-49 (2002).

88. CyberEdu, *Sensibilisation et initiation à la cybersécurité* Educational document (ANSSI, 2015).

89. De la Sécurité des Systèmes d'Information (ANSSI), A. N., *Guide de la méthode Ebios Risk Manager* fr, 2023, https://cyber.gouv.fr/la-methode-ebios-risk-manager (2024).

90. INTERPOL, *Définition du bioterrorisme* fr, 2019, https://www.interpol.int/fr/Infractions/Terrorisme/Bioterrorisme (2024).

91. Clusif, *Les fondamentaux de MEHARI (Méthode harmonisée d'analyse des risques)* fr, 2020, https://clusif.fr/services/management-des-risques/les-fondamentaux-de-mehari/ (2024).

92. Shostack, A., *Threat Modeling : Designing for Security* (John Wiley & Sons, 2014).

93. Faezi, S. *et al., Acoustic Side Channel Attack Against DNA Synthesis Machines : Poster Abstract* in *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)* ISSN : 2642-9500 (avr. 2020), 186-187.

94. Ney, P. *et al., Computer security, privacy, and DNA sequencing : compromising computers with synthesized DNA, privacy leaks, and more* in *26th USENIX Security Symposium (USENIX Security 17)* (2017), 765-779.

95. Ney, P. *et al.,* DNA sequencing flow cells and the security of the molecular-digital Interface, *Proceedings on Privacy Enhancing Technologies* (2021).

96. Kadian, P. *et al.,* Robust digital watermarking techniques for copyright protection of digital data : A survey, *Wireless Personal Communications* **118,** 3225-3249 (2021).

97. Heider, D. & Barnekow, A., DNA watermarks : A proof of concept, *BMC molecular biology* **9,** 1-10 (2008).

98. Haughton, D. & Balado, F., BioCode : Two biologically compatible Algorithms for embedding data in non-coding and coding regions of DNA, *BMC bioinformatics* **14,** 1-16 (2013).

99. Liss, M. *et al.,* Embedding permanent watermarks in synthetic genes (2012).

100. Hamad, S. *et al.,* DNA watermarking using Codon Postfix technique, *IEEE/ACM transactions on computational biology and bioinformatics* **15,** 1605-1610 (2017).

101. Security, M., *What is identity and access management (IAM) ?* en, https://www.microsoft.com/en-us/security/business/security-101/what-is-identity-access-management-iam (2024).

102. Ometov, A. *et al.,* Multi-factor authentication : A survey, *Cryptography* **2,** 1 (2018).

103. Boneh, D. & Zhandry, M., *Quantum-secure message authentication codes* in *Advances in Cryptology–EUROCRYPT 2013 : 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings 32* (2013), 592-608.

104. Of Standards, N. I. & Center, T. .-. C. S. R., *Hash Functions* en, https://csrc.nist.gov/projects/hash-functions (2024).

105. Martino, R. & Cilardo, A., SHA-2 acceleration meeting the needs of emerging applications : A comparative survey, *IEEE Access* **8,** 28415-28436 (2020).

106. Dworkin, M. J., SHA-3 standard : Permutation-based hash and extendable-output functions (2015).

107. Of Standards, N. I. & Center, T. .-. C. S. R., *Message Authentication Codes* en, `https://csrc.nist.gov/projects/message-authentication-codes` (2024).

108. Krawczyk, H. *et al., RFC2104 : HMAC : Keyed-hashing for message authentication* 1997.

109. Kelsey, J. *et al.,* SHA-3 derived functions : cSHAKE, KMAC, TupleHash, and ParallelHash, *NIST special publication* **800,** 185 (2016).

110. Pirzada, S. J. H. *et al., The parallel CMAC authentication algorithm* in *2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN)* (2019), 800-804.

111. Gilbert, H. & Handschuh, H., *Security analysis of SHA-256 and sisters* in *International workshop on selected areas in cryptography* (2003), 175-193.

112. Puzis, R. *et al.,* Increased cyber-biosecurity for DNA synthesis, *Nature Biotechnology* **38,** 1379-1381 (2020).

113. Dworkin, M. J., *Advanced Encryption Standard (AES)* en, rapp. tech. NIST FIPS 197-upd1 (National Institute of Standards et Technology, Gaithersburg, MD, 2023), NIST FIPS 197-upd1, `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf` (2023).

114. Mosca, M., Cybersecurity in an era with quantum computers : Will we be ready ?, *IEEE Security & Privacy* **16,** 38-41 (2018).

115. Bernstein, D. J. & Lange, T., Post-quantum cryptography, *Nature* **549,** 188-194 (2017).

116. Chen, L. *et al., Report on post-quantum cryptography* (US Department of Commerce, National Institute of Standards et Technology ..., 2016).

117. Clelland, C. T. *et al.,* Hiding messages in DNA microdots, en, *Nature* **399,** 533-534, issn : 1476-4687, `https://www.nature.com/articles/21092` (2020) (juin 1999).

118. Gao, Q., *A few DNA-based security techniques* in *2011 IEEE Long Island Systems, Applications and Technology Conference* (2011), 1-5.

119. Cui, G. *et al., An encryption scheme using DNA technology* in *2008 3rd International Conference on Bio-Inspired Computing : Theories and Applications* (2008), 37-42.

120. Leier, A. *et al.,* Cryptography with DNA binary strands, *Biosystems* **57,** 13-22 (2000).

121. Magdeldin, S., *Gel electrophoresis : Principles and basics* (BoD–Books on Demand, 2012).

122. Ning, K., A pseudo DNA cryptography method, *arXiv preprint arXiv :0903.2693* (2009).

123. Amin, S. T. *et al., A DNA-based implementation of YAEA encryption algorithm.* in *Computational intelligence* (2006), 120-125.

124. Sabry, M. *et al., Design of DNA-based advanced encryption standard (AES)* in *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)* (2015), 390-397.

125. Satyanarayana, C. *et al.,* A novel level-based DNA security algorithm using DNA codons, *Computational Intelligence and Big Data Analytics : Applications in Bioinformatics,* 1-13 (2019).

126. Siddaramappa, V. & Ramesh, K., DNA-Based XOR operation (DNAX) for data security using DNA as a storage medium, *Integrated Intelligent Computing, Communication and Security,* 343-351 (2019).

127. Sreeja, C. *et al., DNA for information security : A Survey on DNA computing and a pseudo DNA method based on central dogma of molecular biology* in *International conference on computing and communication technologies* (2014), 1-6.

128. UbaidurRahman, N. H. *et al.,* A novel DNA computing based encryption and decryption algorithm, *Procedia Computer Science* **46,** 463-475 (2015).

129. Gehani, A. *et al.,* in *Aspects of Molecular Computing* 167-188 (Springer, 2003).

130. Lu, M. *et al.,* Symmetric-key cryptosystem with DNA technology, *Science in China Series F : Information Sciences* **50,** 324-333 (2007).

131. Lai, X. *et al.,* Asymmetric encryption and signature method with DNA technology, *Science China Information Sciences* **53,** 506-514 (2010).

132. Zhang, Y. *et al.,* A DNA-Based Encryption Method Based on Two Biological Axioms of DNA Chip and Polymerase Chain Reaction (PCR) Amplification Techniques, *Chemistry–A European Journal* **23,** 13387-13403 (2017).

133. Zhang, Y. *et al., DNA cryptography based on DNA Fragment assembly* in *2012 8th International Conference on Information Science and Digital Content Technology (ICIDT)* **1** (2012), 179-182.

134. Chao, R. *et al.,* Recent advances in DNA assembly technologies, *FEMS yeast research* **15,** 1 (2015).

135. Zhang, Y. *et al.,* DNA based random key generation and management for OTP encryption, *Biosystems* **159,** 51-63 (2017).

136. Mohsin, A. H. *et al.,* Based blockchain-PSO-AES techniques in finger vein biometrics : A novel verification secure framework for patient authentication, *Computer Standards & Interfaces* **66,** 103343 (2019).

137. Faezi, S. *et al., Acoustic Side Channel Attack Against DNA Synthesis Machines* in *IEEE International Conference on Cyber-Physical Systems (ICCPS)* (2020), 186-187.

138. Biotechnologies, N. E., *Enzyme Finder* en, `https://enzymefinder.neb.com/` (2023).

139. Cai, K. *et al., Coding for segmented edits with local weight constraints* in *2021 IEEE International Symposium on Information Theory (ISIT)* (2021), 1694-1699.

140. Pic, X. & Antonini, M., *A constrained Shannon-Fano entropy coder for image storage in synthetic DNA* in *2022 30th European Signal Processing Conference (EUSIPCO)* (2022), 1367-1371.

141. Bibak, K. *et al.,* Unweighted linear congruences with distinct coordinates and the Varshamov–Tenengolts codes, *Designs, Codes and Cryptography* **86,** 1893-1904 (2018).

142. Song, W. *et al.,* Codes with run-length and GC-content constraints for DNA-based data storage, *IEEE Communications Letters* **22,** 2004-2007 (2018).

143. Li, X. *et al.,* Multiple errors correction for position-limited DNA sequences with GC balance and no homopolymer for DNA-based data storage, *Briefings in Bioinformatics* **24,** bbac484 (2023).

144. Mishra, P. *et al.,* Compressed DNA coding using minimum variance Huffman tree, *IEEE Communications Letters* **24,** 1602-1606 (2020).

145. Dimopoulou, M. *et al., A JPEG-based image coding solution for data storage on DNA* in *2021 29th European Signal Processing Conference (EUSIPCO)* (2021), 786-790.

146. Dimopoulou, M. *et al., A biologically constrained encoding solution for long-term storage of images onto synthetic DNA* in *2019 27th European Signal Processing Conference (EUSIPCO)* (2019), 1-5.

147. Benerjee, K. G. & Banerjee, A., On DNA codes with multiple constraints, *IEEE Communications Letters* **25,** 365-368 (2020).

148. Nguyen, T. T. *et al.,* Capacity-approaching constrained codes with error correction for DNA-based data storage, *IEEE Transactions on Information Theory* **67,** 5602-5613 (2021).

149. Van Wijngaarden, A. J. & Immink, K. A. S., Construction of maximum run-length limited codes using sequence replacement techniques, *IEEE Journal on Selected Areas in Communications* **28,** 200-207 (2010).

150. Weber, J. H. & Immink, K. A. S., Knuth's balanced codes revisited, *IEEE Transactions on Information Theory* **56,** 1673-1679 (2010).

151. Press, W. H. *et al.,* HEDGES error-correcting code for DNA storage corrects indels and allows sequence constraints, *Proceedings of the National Academy of Sciences* **117,** 18489-18496 (2020).

152. Katz, J. & Lindell, Y., *Introduction to modern cryptography : principles and protocols* (Chapman et hall/CRC, 2007).

153. Vaidehi, M. & Rabi, B. J., *Design and analysis of AES-CBC mode for high security applications* in *2014 2nd International Conference on Current Trends In Engineering and Technology (ICCTET)* (2014), 499-502.

154. Shokrollahi, A., *LDPC codes : An introduction* in *Coding, cryptography and combinatorics* (2004), 85-110.

155. Westall, J. & Martin, J., An introduction to Galois fields and Reed-Solomon coding, *School of Computing Clemson University Clemson,* 29634-1906 (2010).

156. Lenz, A. *et al., Concatenated codes for recovery from multiple reads of DNA sequences* in *2020 IEEE Information Theory Workshop (ITW)* (2021), 1-5.

157. La Guardia, G. G., Asymmetric quantum reed-solomon and generalized reed-solomon codes, *Quantum Information Processing* **11,** 591-604 (2012).

158. Yazdi, S. H. T. *et al.,* DNA-based storage : Trends and methods, *IEEE Transactions on Molecular, Biological and Multi-Scale Communications* **1,** 230-248 (2015).

159. Tagore, R., *Gitanjali* (Tredition Classics, 2012).

160. Biolabs, N. E., *Protocol for blunting ends by 3' overhang removal and 3' recessed (5' overhang) end fill-in using T4 DNA Polymerase (M0203)* en, 2015, `https://www.neb.com/en/protocols/2014/01/13/protocol-for-blunting-ends-by-3-overhang-removal-and-fill-in-of-3-recessed-5-overhang-ends-using1` (2024).

161. Biolabs, N. E., *Product Information for Bst DNA Polymerase, Large Fragment* en, https://www.neb.com/en/products/m0275-bst-dna-polymerase-large-fragment#Product%20Information (2023).

162. Biolabs, N. E., *Product Information for Mung Bean Nuclease* en, https://www.neb.com/en/products/m0250-mung-bean-nuclease#Product%20Information (2023).

163. Xue, L. *et al.,* Sensitive and homogeneous protein detection based on target-triggered aptamer hairpin switch and nicking enzyme assisted fluorescence signal amplification, *Analytical chemistry* **84,** 3507-3513 (2012).

164. Horspool, D. R. *et al.,* Efficient assembly of very short oligonucleotides using T4 DNA Ligase, *BMC research notes* **3,** 1-9 (2010).

165. Bilotti, K. *et al.,* Mismatch discrimination and sequence bias during end-joining by DNA ligases, *Nucleic Acids Research* **50,** 4647-4658 (2022).

166. Cartron, P.-F. *et al.,* Méthylation/déméthylation de l'ADN et expression du génome, *Revue Francophone des Laboratoires* **2015,** 37-48 (2015).

167. Richa, R. & Sinha, R. P., Hydroxymethylation of DNA : an epigenetic marker, *EXCLI journal* **13,** 592 (2014).

168. Shagin, D. A. *et al.,* Regulation of average length of complex PCR product, *Nucleic acids research* **27,** e23-i (1999).

169. Scientific, T., *PCR Primer Design Tips* en, 2019, https://www.thermofisher.com/blog/behindthebench/pcr-primer-design-tips (2024).

170. ThermoFisher, *Dynabeads Streptavidin for Target Enrichment User Guide* en, https://assets.thermofisher.com/TFS-Assets%2FLSG%2Fmanuals%2FMAN0028561-DynabeadsStreptavidinforTargetEnrichment-UG.pdf (2023).

171. Edvotek, t. b. e. c., *Principes et pratique de l'électrophorèse sur gel d'agarose* fr, 2015, https://www.edvotek.com/site/pdf/101fr.pdf (2024).

172. ThermoFisher, *General Recommendations for DNA Electrophoresis* en, 2012, https://assets.thermofisher.com/TFS-Assets/LSG/manuals/MAN0012614_Gen_Recommend_DNA_Electrophoresis_UG.pdf (2024).

173. Petitcolas, F. A., in *Encyclopedia of Cryptography, Security and Privacy* 1-2 (Springer, 2023).

174. Biotechnologies, N. E., *Selection chart for Compatible Cohesive Ends of restriction enzymes* en, https://www.neb.com/en/tools-and-resources/selection-charts/compatible-cohesive-ends-and-generation-of-new-restriction-sites (2024).

175. Ben Zakour, N. *et al.,* GenoFrag : software to design primers optimized for whole genome scanning by long-range PCR amplification, *Nucleic Acids Research* **32,** 17-24 (2004).

176. Vincze, T. *et al.,* NEBcutter : a program to cleave DNA with restriction enzymes, *Nucleic acids research* **31,** 3688-3691 (2003).

177. James, B. T. *et al.,* MeShClust : an intelligent tool for clustering DNA sequences, *Nucleic acids research* **46,** e83-e83 (2018).

178. (IDT), I. D. T., *Mixed bases offered by IDT* en, https://eu.idtdna.com/pages/products/custom-dna-rna/mixed-bases (2024).

179. Biolabs, N. E., *Activity of Restriction Enzymes in PCR Buffers* en, 2024, https://www.neb.com/en/tools-and-resources/usage-guidelines/activity-of-restriction-enzymes-in-pcr-buffers (2024).

180. Holmberg, A. *et al.,* The biotin-streptavidin interaction can be reversibly broken using water at elevated temperatures, *Electrophoresis* **26,** 501-510 (2005).

181. Xia, Z. *et al.,* A review of parallel implementations for the Smith–Waterman algorithm, *Interdisciplinary Sciences : Computational Life Sciences,* 1-14 (2021).

182. Ström, O. E. *et al.,* High-Throughput Separation of Long DNA in Deterministic Lateral Displacement Arrays, *Micromachines* **13,** 1754 (2022).

183. Wang, X. *et al.,* Characterization of denaturation and renaturation of DNA for DNA hybridization, *Environmental health and toxicology* **29** (2014).

184. Takahashi, C. N. *et al.,* Demonstration of end-to-end automation of DNA data storage, *Scientific reports* **9,** 4998 (2019).

185. Leblanc, J. *et al.,* In vitro construction and long read sequencing analysis of a 24 kb long artificial DNA sequence encoding the Universal Declaration of the Rights of Man and of the Citizen, *bioRxiv,* 2023-06 (2023).

Annexe A

# Fonction de rotation biologique - Algorithme

L'algorithme 4 présenté en page suivante correspond à la fonction de rotation biologique, présentée en Section 3.1.7 qui échange deux blocs adjacents dans une molécule d'ADN double brin. L'algorithme présente la rotation des deux blocs adjacents $d_2$ et $d_3$. Pour effectuer cette opération, il faut connaître les amorces $For_D$ et $Rev_D$ de la molécule $D$, ainsi que les sites de restrictions $R_1$ et $R_2$, et les enzymes $E_1$ et $E_2$ associés.

**Algorithm 4:** Définition d'une fonction de rotation de 2 blocs dans une molécule d'ADN $D$

**Inputs :** Un ensemble de copies $D_h$ de molécules d'ADN $D$ de $d$ paires de bases de la forme $\left(For_D d_1 R_1 d_2 R_1 d_3 R_2 d_4 Rev_D\right)$, les sites de restriction $R_1$ et $R_2$ de $r$ bases et leurs enzymes $E_1$ et $E_2$, les amorces $For_D$ et $Rev_D$.

**Output :** La molécule $Rot(D, R_1, R_2)$ avec les blocs $d_2$ et $d_3$ échangés.

---

1. $\{D_i\} = PCR(D_h, For_D^{biot}, Rev_D^{biot})$ ;
% *PCR avec des amorces biotinylées pour ajouter des biotines aux extrémités des molécules* $D_h$ ;
2. $\{D_j\} = CutS(D_i, E_1)$ ;
% *Coupe saillante avec l'enzyme* $E_1$ *selon les sites de restriction* $R_1$*, créant des blocs avec extrémités saillantes* ;
3. $(\{D_k\}_{T_1}, \{D_l\}_{T_2}) = SepBiotin(\{D_j\}, For_D^{biot}, Rev_D^{biot})$ ;
% *Séparation par capture avec les amorces. Les blocs capturés* $\left(For_D^{biot} d_1\right)$ *et* $\left(d_3 d_4 Rev_D^{biot}\right)$ *sont dans le tube* $T_1$*, tandis que le bloc* $\left(d_2\right)$ *se retrouve dans le tube* $T_2$. ;
4. $\{D_m\}_{T_1} = Lig(\{D_k\}_{T_1})$ ;
% *Ligation des extrémités saillantes compatibles dans le tube* $T_1$*, pour obtenir les séquences* $\left(For_D^{biot} d_1 d_3 d_4 Rev_D^{biot}\right)$ ;
5. $\{D_n\}_{T_1} = CutS(\{D_m\}_{T_1}, E_2)$ ;
% *Coupe saillante avec* $E_2$ *dans* $T_1$*, créant deux blocs* $\left(For_D d_1 d_3\right)$ *et* $\left(d_4 Rev_D\right)$ ;
6. $\{D_p\} = PCR(D_n, For_D^{biot}, Rev_D)$ ;
% *PCR avec des amorces* $For_D^{biot}$ *biotinylées et* $Rev_D$ *standards* ;
7. $(\{D_r\}_{T_1}, \{D_q\}_{T_3}) = SepBiotin(\{D_p\}_{T_1}, For_D^{biot})$;
% *Séparation par capture, pour isoler les blocs* $\left(For_D d_1 d_3\right)$ *dans* $T_1$*, et* $\left(d_4 Rev_D\right)$ *dans* $T_3$. ;
8. $\{D_s\}_{T_1} = Mix(\{D_r\}_{T_1}, \{D_l\}_{T_2})$ ;
% *Mélange des tubes à essai* $T_1$ *et* $T_2$ ;
9. $\{D_t\}_{T_1} = LigS(\{D_s\}_{T_1})$ ;
% *Ligation des extrémités saillantes compatibles dans le tube* $T_1$*, pour obtenir les séquences* $\left(For_D d_1 d_3 d_2\right)$ ;
10. $\{D_v\}_{T_1} = Mix(\{D_t\}_{T_1}, \{D_q\}_{T_3})$ ;
% *Mélange des tubes à essai* $T_1$ *et* $T_3$ ;
11. $\{D_w\}_{T_1} = LigS(\{D_v\}_{T_1})$ ;
% *Ligation des extrémités saillantes compatibles dans le tube* $T_1$*, pour obtenir les séquences* $\left(For_D d_1 d_3 d_2 d_4 Rev_D\right)$ ;

Annexe B

# Fonction de permutation biologique - Algorithme

L'algorithme 5 présenté en page suivante correspond à la fonction de permutation biologique, présentée en Section 3.1.8 qui échange deux blocs de deux molécules d'ADN double brin.

La fonction détaillée est $Permut(D_1, D_2, R_1, R_2)$, où les molécules $D_1$ et $D_2$ encadrées des mêmes primers, échangent les blocs contenus après les sites de restrictions $R_1$ et $R_2$, respectivement. Pour effectuer cette fonction, il faut connaître les amorces $For_D$ et $Rev_D$ des molécules, ainsi que les sites de restriction $R_1$ et $R_2$ et les enzymes associés $E_1$ et $E_2$.

**Algorithm 5:** Permutation des blocs $d_2^1$ et $d_2^2$ des molécules d'ADN $D_1$ et $D_2$

**Inputs :** Un ensemble de copies $D_h$ de molécules d'ADN $D_1$ et $D_2$, avec $D_1$ de la forme $\left(For_D d_1^1 R_1 d_2^1 Rev_D\right)$ et $D_2$ de la forme $\left(For_D d_1^2 R_2 d_2^2 Rev_D\right)$, les sites de restriction $R_1$ et $R_2$ de $r$ bases et leurs enzymes $E_1$ et $E_2$, les amorces $For_D$ et $Rev_D$.

**Output :** Les molécules $Permut(D_1, D_2, R_1, R_2)$ et $Permut(D_2, D_1, R_2, R_1)$ avec les blocs $d_2^1$ et $d_2^2$ échangés.

---

1. $\{D_i\} = PCR(D_h, For_D, Rev_D^{biot})$ ;
% *PCR avec des amorces* $For_D$ *standards et* $Rev_D^{biot}$ *biotinylées, qui ajoutent des biotines aux extrémités* $Rev_D$ *des molécules* ;
2. $\{D_j\} = CutS(D_i, E_1)$ ;
% *Coupe saillante avec l'enzyme* $E_1$ *selon les sites de restriction* $R_1$*, créant deux blocs avec extrémités saillantes* ;
3. $(\{D_k\}_{T_1}, \{D_l\}_{T_2}) = SepBiotin(\{D_j\}, Rev_D^{biot})$ ;
% *Séparation par capture avec les amorces* $Rev_D^{biot}$ *biotinylées. Les blocs capturés* $\left(R_1^2 d_2^1 Rev_D^{biot}\right)$ *et* $\left(For_D d_1^2 R_2 d_2^2 Rev_D^{biot}\right)$ *sont dans le tube* $T_2$*, tandis que le bloc* $\left(For_D d_1^1 R_1^1\right)$ *se trouve dans le tube* $T_1$. ;
4. $\{D_k\}_{T_1} = Dephos(\{D_k\}_{T_1})$ et $\{D_l\}_{T_2} = Dephos(\{D_l\}_{T_2})$ ;
% *Déphosphorylation des extrémités saillantes dans les tubes* $T_1$ *et* $T_2$ ;
5. $\{D_m\}_{T_2} = CutS(\{D_l\}_{T_2}, E_2)$ ;
% *Coupe saillante avec* $E_2$ *dans* $T_2$*, créant deux blocs* $\left(For_D d_1^2 R_2^1\right)$ *et* $\left(R_2^2 d_2^2 Rev_D^{biot}\right)$ ;
6. $(\{D_n\}_{T_2}, \{D_p\}_{T_3}) = SepBiotin(\{D_m\}_{T_2}, Rev_D^{biot})$;
% *Séparation dans* $T_2$ *avec les amorces* $Rev_D^{biot}$ *biotinylées, pour isoler les blocs* $\left(R_1^2 d_2^1 Rev_D^{biot}\right)$ *et* $\left(R_2^2 d_2^2 Rev_D^{biot}\right)$ *dans* $T_2$*, et* $\left(For_D d_1^2 R_2^1\right)$ *dans* $T_3$. ;
7. $\{D_r\}_{T_1} = Mix(\{D_k\}_{T_1}, \{D_n\}_{T_2})$ ;
% *Mélange des tubes à essai* $T_1$ *et* $T_2$ *dans* $T_1$ ;
8. $\{D_r\}_{T_1} = LigS(\{D_q\}_{T_1})$ ;
% *Ligation des extrémités saillantes compatibles dans le tube* $T_1$*, pour obtenir les molécules* $\left(For_D d_1^1 R_1^1 R_2^2 d_2^2 Rev_D^{biot}\right)$ ;
9. $\{D_s\}_{T_1} = Mix(\{D_r\}_{T_1}, \{D_p\}_{T_3})$ ;
% *Mélange des tubes à essai* $T_1$ *et* $T_3$ *dans* $T_1$ ;
10. $\{D_v\}_{T_1} = LigS(\{D_s\}_{T_1})$ ;
% *Ligation des extrémités saillantes compatibles dans le tube* $T_1$*, pour obtenir les séquences* $\left(For_D d_2^1 R_1^2 R_1^2 d_2^1 Rev_D^{biot}\right)$ ;

Annexe C

# Liste d'extrémités saillantes de taille 4 créées par plusieurs enzymes

Nous présentons ici la liste d'extrémités saillantes de taille 4 pouvant être créées par plusieurs enzymes différents. Les extrémités saillantes sont lues dans le sens de lecture 5' vers 3', les sites de restriction exacts sont indiqués entre parenthèses ainsi que la présence de sites dégénérés. Si il existe une version High Fidelity (H-F) de l'enzyme, elle est précisée par les initiales HF.

- **CATG 5' saillant :**
  - BspHI (5'-GATCGC-3') avec site de restriction de 6 paires de bases
  - NcoI-HF® (5'-CCATGG-3') avec site de restriction de 6 paires de bases
  - PciI (5'-ACATTG-3') avec site de restriction de 6 paires de bases
- **CCGG 5' saillant :**
  - AgeI-HF® (5'-ACCGGT-3') avec site de restriction de 6 paires de bases
  - XmaI (5'-CCCGGG-3') avec site de restriction de 6 paires de bases
  - BspEI (5'-TCCGGA-3') avec site de restriction de 6 paires de bases
  - SgrAI (5'-CCCGGY-3') avec taille de site (degenerated) :8 paires de bases
  - NgoMIV (5'-GCCGGC-3') avec site de restriction de 6 paires de bases
- **CGCG 5' saillant :**
  - AscI (5'-GGCGCG-3') avec site de restriction de 8 paires de bases
  - AflIII (5'-ACATGT-3') avec taille de site (degenerated) : 6 paires de bases
  - MluI-HF® (5'-ACGCGT-3') avec site de restriction de 6 paires de bases
  - BssHII (5'-GCGCGC-3') avec site de restriction de 6 paires de bases
  - MauBI (5'-CCGCGG-3') (thermofisher.com) 8 paires de bases
- **CTAG 5' saillant :**
  - AvrII (5'-CCTAGG-3') avec site de restriction de 6 paires de bases
  - NheI-HF® (5'-GCTAGC-3') avec site de restriction de 6 paires de bases
  - SpeI-HF® (5'-ACTAGT-3') avec site de restriction de 6 paires de bases
  - XbaI (5'-TCTAGA-3') avec site de restriction de 6 paires de bases

— **GATC 5’ saillant :**
  — BamHI-HF® (5’-GGATCC-3’) avec site de restriction de 6 paires de bases
  — BclI-HF (5’-TGATCA-3’) avec site de restriction de 6 paires de bases
  — BglII (5’-AGATCT-3’) avec site de restriction de 6 paires de bases
— **GCGC 3’ saillant :**
  — HaeII (5’-RGCGCY-3’) avec taille de site (degenerated) : 6 paires de bases
  — PluTI (5’-GGCGCC-3’) avec site de restriction de 6 paires de bases
— **GGCC 5’ saillant :**
  — EaeI (5’-YGGCCR-3’) avec taille de site (degenerated) : 6 paires de bases
  — NotI-HF® (5’-GCGGCCGC-3’) avec site de restriction de 8 paires de bases
  — PspOMI (5’-GGCCGCC-3’) avec site de restriction de 6 paires de bases
  — EagI-HF® (5’-CGGCCG-3’) avec site de restriction de 6 paires de bases
— **GTAC 5’ saillant :**
  — Acc65I (5’-GTACCA-3’) avec site de restriction de 6 paires de bases
  — BsiWI-HF® (5’-GTACGT-3’) avec site de restriction de 6 paires de bases
  — BsrGI-HF® (5’-GTACAT-3’) avec site de restriction de 6 paires de bases
  — RJR REBASE Enz 2746 (5’-WGTACW-3’) avec taille de site (degenerated) : 6 paires de bases
— **TCGA 5’ saillant :**
  — XhoI (5’-CTCGAG-3’) avec site de restriction de 6 paires de bases
  — SalI (5’-GTCGAC-3’) avec site de restriction de 6 paires de bases
  — PspXI (5’-PTCGT’-3’) avec taille de site (degenerated) : (10)
  — SgrDI (5’-CGTCGACG-3’) avec site de restriction de 8 paires de bases
  — AbsI (5’-CCTCGAGG-3’) avec site de restriction de 8 paires de bases
— **TGCA 5’ saillant :**
  — ApaLI (5’-GGCC-3’) avec site de restriction de 6 paires de bases
  — SfcI (5’-CCGCG-3’) avec taille de site (degenerated) : 6 paires de bases
— **TGCA 3’ saillant :**
  — NsiI-HF® (5’-AATGC-3’) avec site de restriction de 6 paires de bases
  — BsiHKAI (5’-GRCYTG-3’) avec taille de site (degenerated) : 6 paires de bases
  — PstI-HF® (5’-CTGCAG-3’) avec site de restriction de 6 paires de bases
  — SbfI-HF® (5’-CCTGCAGG-3’) avec site de restriction de 10 paires de bases

Annexe D

# Paramètres ITHOS de choix d'amorces

Nous présentons ici la liste de paramètres pris en entrée de l'algorithme Ithos, un sous-module du logiciel de design d'amorces Genofrag [175], afin de déterminer un couple d'amorces pour un ensemble de séquences ADN. Les paramètres sont choisis d'après l'article [185].

- Longueur de primers $sizePrimer = 20$
- G-C content compris entre $pcGCMin = 45$ et $pcGCMax = 50$
- Température d'hybridation $Tm$ comprise entre $oligoTmMin = 50$ et $oligoTmMax = 61$, avec $dnaConc = 500$ et $saltConc = 50$
- Taille maximale de "hairpin", ou épingle à cheveu, de $maxHpDup = 4$ et $maxHpLoop = 4$
- Taille maximale d'homopolymère : nbRepeat 4
- Paramètres pour l'autocomplémentarité : $maxDeltaGAuto = -10000$, $maxDeltaGAuto3 = -7000$, $sizeDeltaGAuto = 6$, et $sizeDeltaGAuto3 = 8$
- Stabilité interne aux extrémités 3' et 5 :$sizeExt5 = 5$, $sizeExt3 = 5$, $deltaG5 = -4000$, $deltaG3min = -6000$, et $deltaG3max = -4000$
- Non-hybridation des amorces avec les séquences : $sizeDeltaGHybrid3 = 8$, $maxDeltaGHybrid3 = -9000$, et $maxDeltaGHybrid = -16000$

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

**Titre :** Sécurité des données stockées sur molécules d'ADN

**Mot clés :** Stockage de données dans l'ADN, sécurité, confidentialité, encodage dynamique, chiffrement biomoléculaire, analyse de risques

**Résumé :** La quantité de données numériques produites dans le monde chaque année augmente exponentiellement et les supports actuels de stockage atteignent leurs limites. Dans ce contexte, le stockage de données sur molécules d'ADN est très prometteur. Stockant jusqu'à $10^{18}$ octets par gramme d'ADN pour une consommation d'énergie quasi nulle, il a une durée de vie 100 fois plus longue que les disques durs. Cette technologie de stockage étant en développement, il est opportun d'y intégrer nativement des mécanismes pour sécuriser les données. C'est l'objet de cette thèse. Notre première contribution est une analyse des risques de l'ensemble de la chaîne de stockage, qui nous a permis d'identifier des vulnérabilités des procédés numériques et biologiques, en termes de confidentialité, d'intégrité, de disponibilité et de traçabilité. Une seconde contribution est l'identification d'opérateurs élémentaires permettant des manipulations simples de l'ADN. Avec ceux-ci, nous avons développé notre troisième contribution, une solution de chiffrement DNACipher qui impose un déchiffrement biomoléculaire des molécules avant de pouvoir lire les données correctement. Cette solution, qui repose sur des enzymes, a nécessité le développement d'un codage des données numériques en séquences ADN appelée DSWE ; notre quatrième contribution. Cet algorithme respecte les contraintes liées aux procédés biologiques (*e.g.* homopolymères) et à notre DNACipher. Enfin, notre dernière contribution est une validation expérimentale de notre chaîne de stockage sécurisée. C'est la première preuve de concept montrant qu'il est possible de sécuriser ce nouveau support de stockage sur la base de manipulations biomoléculaires.

**Title:** Security of data stored into DNA molecules

**Keywords:** Data storage into DNA molecules, security, confidentiality, dynamic encoding, biomolecular encryption, risk analysis

**Abstract:** The volume of digital data produced worldwide every year is increasing exponentially, and current storage solutions are reaching their limits. In this context, data storage on DNA molecules holds great promise. Storing up to $10^{18}$ bytes per gram of DNA for almost no energy consumption, it has a lifespan 100 times longer than hard disks. As this storage technology is still under development, the opportunity presents itself to natively integrate data security mechanisms. This is the aim of this thesis. Our first contribution is a risk analysis of the entire storage chain, which has enabled us to identify vulnerabilities in digital and biological processes, particularly in terms of confidentiality, integrity, availability and traceability. A second contribution is the identification of elementary biological operators for simple manipulations of DNA. Using these operators, we have developed a DNACipher encryption solution that requires biomolecular decryption (DNADecipher) of the molecules before the data can be read correctly. This third contribution, based on enzymes, required the development of a coding algorithm for digital data into DNA sequences, a contribution called DSWE. This algorithm respects the constraints of biological processes (*e.g.* homopolymers) and our encryption solution. Our final contribution is an experimental validation of our secure storage chain. This is the first proof of concept showing that it is possible to secure this new storage medium using biomolecular manipulations.