Adaptation d'Outils Cryptographiques pour un Contexte Post-Quantique

Cyrius Nugier

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue le *04/07/23* **par :**

Cyrius Nugier

Adaptation d'Outils Cryptographiques pour un Contexte Post-Quantique

---

JURY

Olivier Blazy	Professeur des universités	Président du jury
Fabien Laguillaumie	Professeur des universités	Rapporteur
Cédric Lauradoux	Chargé de recherche	Examinateur
J.-C. Deneuville	Enseignant chercheur	Examinateur
Vincent Nicomette	Professeur des universités	Directeur de thèse
Vincent Migliore	Maître de conférences	Directeur de thèse

École doctorale et spécialité :

MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture

Unité de Recherche :

Laboratoire d'analyse et d'architecture des systèmes – LAAS-CNRS

Directeur(s) de Thèse :

Vincent Nicomette et *Vincent Migliore*

Rapporteurs :

Fabien Laguillaumie et *Olivier Blazy*

'T ain't what you do
It's the way that you do it
That's what gets results

Jimmy Lunceford

Remerciements

Les travaux présentés dans ce manuscrit ont été effectués dans l'équipe *Tolérance aux fautes et sécurité de fonctionnement* (TSF) du *Laboratoire l'analyse et d'architecture des systèmes* (LAAS) du *Centre national de la recherche scientifique* (CNRS). Je remercie les directeurs du laboratoire M. Liviu Nicu et M. Mohamed Kaâniche, ainsi que la responsable d'équipe Mme Hélène Waeselynck de m'avoir accueilli dans cette structure, et m'avoir permi de mener mes travaux dans de bonnes conditions.

Je remercie sincèrement mes directeurs de thèse, « *les vincents* » : M. Vincent Nicomette pour son soutien, ses encouragements, l'écoute dont il a fait preuve, et l'énergie qu'il a déployé pour m'aider dans mes recherches. M. Vincent Migliore pour la précision de ces connaissances techniques et la volonté qu'il a de mettre ses compétences au service des autres, sa capacité à toujours me faire prendre du recul sur les recherches.

Bien que trop nombreux pour les nommer ici, je tiens à remercier tous les doctorants de l'équipe que j'ai eu la chance de côtoyer, pour toutes les discussions scientifiques, techniques philosophiques enrichies par nos expériences de vie différentes, et toujours dans un esprit de recherche de la vérité passionné. Je remercie en particulier M. Florent Galtier et M. Rémi Adelin, qui ont toujours su être à l'écoute et dont le soutien à été précieux.

J'ai eu l'occasion de collaborer avec des membres de la communauté de la cryptographie, ou de me tourner vers certains de ses membres pour obtenir de l'aide. Je les remercie d'avoir été toujours ouverts et bienveillants.

Je tiens finalement à remercier tous ceux qui m'ont soutenu mentalement pendant ce parcours d'obstacles qui est la thèse. Je pense spécialement à ma famille, ma seconde famille de danseurs du TRAC, et par dessus tout le support précieux et indéfectible de ma femme.

Table des matières

Liste des Abréviations

Abréviations

VPN	Virtual Private Network / réseau privé virtuel
TOR	The Onion Router / le routeur en oignon
KEM	Key Encapsulation Mechanism / mécanisme d'encapsulation de clés
ABE	Attribute Based Encryption / chiffrement basé attributs
KP-ABE	Key-Policy ABE / ABE avec politique dans la clé
CP-ABE	Ciphertext-Policy ABE / ABE avec politique dans le chiffré
PQ-ABE	Post-Quantum ABE / ABE post-quantique
IBE	Identity Based Encryption / chiffrement basé identité
FIBE	Fuzzy Identity Based Encryption / chiffrement basé identité floue
HABE	Hyerarchical ABE / chiffrement hiérarchique basé attributs
SHA	Secure Hash Algorithm / algorithme de hachage sécurité
NIST	National Institute of Standards and Technology/ institut américain des standards technologiques
DDoS	Distributed Denial of Service / déni de service distribué
QoS	Quality of Service / qualité de service
AES	Advanced Encryption Standard / standard de chiffrement avancé
AES-GCM	AES-Galois Counter Mode / AES en mode compteur de Galois
IND-CCA2	Indistinguishability against adaptative chosen ciphertext attack/ indistinguabilité contre les attaques a chiffrés choisis adaptativement
IND-CPA	Indistinguishability against chosen plaintext attack/ indistinguabilité contre les attaques à clair choisi
LWE	Learning With Errors / apprentissage avec erreurs
R-LWE	Ring LWE / LWE dans un anneau
RSA	Rivest Shamir Adleman cryptosystem/ cryptosystème Rivest Shamir Adleman

Chapitre 1 : Notations utilisées dans le protocole QASTOR

$\mathcal{U}$	Distribution Uniforme
LG	Liste des Groupes
$\tilde{G}_i$	Groupe initialement prévu à la profondeur i, balisé « Guard »

G_i	Groupe initialement prévu à la profondeur i, non balisé « Guard »
$\widehat{G}_i$	Groupe final à la profondeur i
A_i	Attributs du groupe G_i
ALGO	Nom d'un algorithme du protocole
PAQUET	Type d'un paquet échangé dans le protocole
$[\mathrm{m}]_{\mathrm{k}}$	Chiffrement Symétrique de m avec la clé k
PK_i,SK_i	Un couple clé publique / clé secrète
$[\mathrm{m}]_{PK_i}$	Chiffrement Asymétrique de m avec la clé PK_i
$[\![\mathrm{m}]\!]_{\mathrm{att}}$	Chiffrement ABE de m tel que les attributs att permettent de déchiffrer
`src`	Client qui souhaite établir une communication
`dst`	Serveur avec qui le client veut communiquer
`load balancer`	Algorithme de répartition des charges
`step`	Balise : Instruction de passer à un groupe suivant
`end`	Balise : Instruction de se connecter au serveur
`cmd`	Balise : L'une des deux instructions de transmission
`new`	Balise : La communication désirée est nouvelle
`old`	Balise : La communication désirée est déjà établie avec un autre client
`confirm`	Balise : Le client a pu se reconnecter
`fail`	Balise : Le client a échoué à se reconnecter
`payload`	Données échangées entre le client et le serveur
R_i	Relai membre du groupe i
LB_i	`load balancer` du groupe i
C_i	Connexion avec un membre du groupe i
rt_i	Clé temporaire du groupe i
rl_i	Clé locale aux groupes $i-1$ et i
$\widehat{rl}_{i+1}$	Clé locale aux groupes i et $i+1$ si une communication est deja établie
K	Clé de la communication
S	Token Temporaire
T	Token Temporaire pour reconnexion
D	Valeur de défi
B_i	Bloc i d'un paquets

Chapitre 2 : Notations relatives au chiffrement commutatif

CE	Commutative Encryption / chiffrement commutatif
RGPD	Réglement Général sur la Protection des Données
VANET	Vehicular Ad Hoc NETwork / réseau véhiculaire ad hoc
CEF	Commutative Encryption Family / famille de chiffrements commutatifs
SE	Super Encryption / sur-chiffrement
GHE	Group Homomorphic Encryption / chiffrement homomorphe de groupe
SEF	Super Encryption Family / famille de sur-chiffrements
PHS	Principal Homogeneous Spaces / espaces principaux homogènes
FNAA	Finite Non-commutative Associative Algebras / algèbres finies associatives non-commutatives
HHS	Hard Homogeneous Spaces / espaces principaux résistants
CES	Commutative Encryption Scheme / schéma de chiffrement commutatif
SES	Super Encryption Scheme / schéma de sur-chiffrement
VS	Valid Set / Ensemble de listes d'utilisateurs valides
PRE	Proxy Re-Encryption / re-chiffrement par proxy

Chapitre 3 : Nouvelles architectures et primitives post-quantiques

PIM	Processing In Memory / calcul en mémoire
ALU	Arithmetical and Logical Unit / unité arithmétique et logique
BP	Bank Partition / partition d'une banque mémoire liée physiquement
RAM	Random Acces Mémory / mémoire vive
SIMD	Single Instruction Multiple Data / instruction unique données multiples

Introduction

La cryptographie, essentielle à la protection de la vie privée de chacun et utilisée presque par tous quotidiennement lors de la navigation sur Internet, est assujettie en permanence à des changements sur de multiples plans. Le changement principal, qui est au centre de l'attention de la communauté cryptographique ces dernières années, résulte de l'arrivée possible des ordinateurs quantiques. Comme ceux-ci peuvent théoriquement mettre en danger nos systèmes de chiffrements à clé publique actuels, un effort de standardisation de primitives cryptographiques dites *post-quantiques* est en cours. Si ces primitives n'étaient pas en place avant l'arrivée d'un ordinateur quantique suffisamment puissant, son propriétaire pourrait déchiffrer une partie conséquente du trafic mondial actuel.

Le changement de primitives n'est cependant pas le seul défi auquel font face les cryptographes actuellement. Les protocoles doivent eux aussi être adaptés pour ne pas constituer une surface d'attaque supplémentaire à un ordinateur quantique, ce qui peut être une tâche ardue pour les plus complexes d'entre eux, au premier rang desquels les protocoles multipartites ou ceux basés sur des chiffrements de groupes.

De plus, les utilisations de la cryptographie changent elles aussi : l'amélioration des performances de nos ordinateurs et des infrastructures réseaux permettent une consommation de vidéos bien plus forte et parmi celles-ci beaucoup de vidéos en direct. Les utilisateurs sont aussi plus conscients de leur intérêt à protéger leur vie privée, et à maîtriser pour qui leurs données peuvent être accessibles.

Par ailleurs, les ordinateurs sur lesquels s'exécutent les outils de protection de la vie privée eux aussi évoluent. Plusieurs utilisations sont améliorées matériellement, comme les réseaux de neurones ou les calculs graphiques pour l'audiovisuel et le jeu vidéo. La cryptographie fait partie de ces cibles que les architectures modernes cherchent à accélérer. Ces améliorations des architectures matérielles peuvent avoir en retour des effets sur la cryptographie, même si elles n'ont pas été conçues dans cette optique. Par exemple, des informations fuitant par des mesures physiques peuvent mettre en danger la sécurité des primitives.

Les évolutions de chacun de ces thèmes constituent de multiples sujets de recherche, et pour minimiser les effets de ces changements sur nos systèmes à l'aube d'un éventuel ordinateur quantique, les réponses proposées se veulent isolées et le moins impactantes possible sur les autres sujets. Cependant, ceux-ci sont interdépendants et s'influencent nécessairement les uns les autres.

Premièrement, l'arrivée éventuelle de la cryptanalyse quantique induit un remplacement des primitives par des équivalents post-quantiques. Ce changement pourra être quasiment transparent pour la plupart des protocoles. Ceux-ci s'exécuteront comme prévu sur nos machines actuelles. Certains changements se feront tout de même ressentir : les nouvelles primitives n'ont pas les mêmes performances, tailles de clés, tailles de chiffrés... Une concordance entre un protocole et les primitives qu'il utilise est nécessaire. Des adaptations des deux côtés peuvent alors être requises,

d'autant plus que certaines propriétés des protocoles sont elles aussi menacées par l'ordinateur quantique.

Ces protocoles doivent aussi être adaptés au regard des évolutions observées sur les utilisations dont ils font l'objet. Les utilisateurs semblent parfois accepter une baisse de vie privée en échange d'un service qui leur semble meilleur. Cependant, à service équivalent, ils souhaitent malgré tout davantage de protection de leur vie privée. Les changements de comportement des utilisateurs dans leur consommation de données chiffrées doivent être intégrés à la conception des protocoles, d'autant plus que ceux-ci peuvent aussi être sujets à d'autres contraintes extérieures complexes, légales par exemple.

On peut aussi mettre en avant les synergies entre l'étude des primitives et de leurs supports d'exécution. D'un côté, chaque nouvelle instruction du processeur, chaque opération vectorielle, sont autant d'opportunités d'accélérer une primitive. De l'autre côté, certaines peuvent être ajoutées spécialement pour cette raison. Des co-processeurs sont développés afin d'obtenir des performances maximales d'exécution. En outre, les moyens d'exploiter les fuites physiques d'informations s'améliorent également. Bien que les technologies mémoire cherchent à mitiger celles-ci, les algorithmes doivent aussi s'adapter en fonction de ce modèle de vulnérabilité.

Nous postulons donc qu'il est nécessaire dans cette période de changements, de proposer des solutions transversales et ainsi plus poussées. Une telle vision globale permet de se préparer à l'arrivée d'un ordinateur quantique et d'être proactif sur les changements qui s'opèreront dans les années à venir. Nous développons dans cette thèse des propositions sur quatre problématiques, dont une vision transversale apporte de meilleurs résultats.

Cette thèse commence par une section de rappels, permettant de s'accorder sur une base sémantique minimale sur les termes de cryptographie classiques utilisés dans celle-ci. Ensuite le chapitre 1 présente plus précisément le contexte cryptographique à l'approche d'un ordinateur quantique.

Dans le chapitre 2, nous présenterons un nouveau protocole, nommé QasTor, inspiré du célèbre protocole Tor, et qui répond à une consommation numérique grandissante de live-streaming de masse, tout en préservant l'anonymat de ses utilisateurs, même face à un ordinateur quantique. Nous discuterons des compromis anonymat/efficacité que ce protocole propose.

Dans le chapitre 3, nous étudierons l'ensemble des protocoles multipartite impliquant des chiffrements et/ou déchiffrements successifs et dont la sécurité est mise en danger à cause de l'arrivée potentielle des ordinateurs quantiques. Parmi ceux-ci se trouvent les protocoles d'interception réglementée, dont nous proposons une adaptation basée sur des primitives post-quantiques, et illustrée par une adaptation au contexte des voitures connectées.

Enfin, dans le chapitre 4, nous étudierons l'impact que les architectures émergentes ont sur les primitives post-quantiques candidates au processus de standardisation, car celui-ci s'est concentré sur des architectures existantes. L'exemple qui illustre le mieux cet impact est l'utilisation de Classic McEliece sur une architecture incluant du calcul en cache, qui pourrait parfaitement être diffusée durant l'ère post-

quantique. Nous proposons des pistes d'évaluation de la quantité d'information que peut récupérer un attaquant grâce à des fuites par canaux auxiliaires et montrons comment profiter des nouvelles architectures pour diminuer celle-ci.

Finalement, nous présenterons les conclusions apportées par ces travaux et les perspectives à court et moyen terme de ceux-ci. Il est précisé que la liste des publications ayant eu lieu au cours de cette thèse se trouve avant la bibliographie.

Rappels

Afin de faire de ce manuscrit un document plus compréhensible pour des personnes n'ayant pas de compétences avancées en cryptographie, cette section « rappels » est incluse afin de présenter des notions basiques de la cryptographie classique. Les lecteurs familiers avec le sujet sont invités à démarrer par la lecture du chapitre Contexte et Évolutions, qui présente l'impact de l'ordinateur quantique sur le domaine de la cryptographie et en déduit des pistes de recherche à suivre.

> Ces rappels ont pour objectif de rendre le manuscrit intelligible sans pour autant rentrer dans des détails trop techniques. Je recommande très fortement à un lecteur curieux souhaitant avoir une compréhension plus rigoureuse des sujets évoqués de suivre le cours en ligne de D.Boneh [Boneh 2020a], et pour aller encore plus loin de s'aider du livre incroyablement complet (et incroyablement long) qui y est associé [Boneh 2020b].

Cryptologie. Le nom vient du grec ancien κρυπτός, kruptós (« couvert, caché ») et de λόγος, lógos (« parole, discours ») qui en tant que suffixe définit une science. La cryptologie est ainsi la science de cacher, de protéger des messages. Le domaine est historiquement associé à l'étude de la langue et pratiquée par des littéraires, et des érudits aux connaissances encyclopédiques. Cependant, dès les prémices de la seconde guerre mondiale, ce sont des mathématiciens et des statisticiens qui de plus en plus s'approprient cette science [Singh 1999]. A la suite de la seconde guerre mondiale, c'est l'avènement de l'ordinateur qui va finir d'inscrire la cryptologie comme intersection des mathématiques et de l'informatique.

Cryptographie et Cryptanalyse. La cryptologie est divisée en deux domaines concurrents : d'un coté la Cryptographie, de γραφία, graphía (« écriture ») et de l'autre la Cryptanalyse, de ἀναλύω, analuô (« délier »). On peut les imaginer comme « la défense » et « l'attaque ». Le premier a pour objectif de transmettre un message caché (d'empêcher sa lecture par des personnes non désirées) et le second cherche à retrouver le sens de ce message caché (lorsque l'on n'est pas l'un des destinataires du message). Les deux sont incroyablement liés, le défenseur devant sans cesse se mettre dans la peau d'un attaquant pour anticiper toutes les faiblesses de sa méthode de protection du message.

Stéganographie. Une méthode très simple pour empêcher la lecture d'un message consiste à faire en sorte qu'il soit difficile à identifier. Une technique efficace pour cela consiste à cacher un message dans un autre, ce dernier paraissant anodin. L'exemple le plus parlant connu reste l'acrostiche grivoise associée à George Sand et Musset [Cryptage 2006], se lisant de manière romantique par une personne non-avertie mais de manière érotique par le lecteur sachant qu'il ne faut lire qu'une ligne sur deux. De la même manière, des informations peuvent être cachées dans des images, du son ou des vidéos.

Cryptographie Symétrique. On peut aussi transformer le message afin de le rendre illisible (et pas invisible). Si cette transformation était prévisible, il serait

aisé de comprendre le message même une fois chiffré. Il faut donc se servir d'une valeur secrète qu'on appelle clé (K pour *key* en anglais) qui pourra changer entre les chiffrements. Si Alice et Bob veulent s'échanger des messages en toute sécurité, ils commencent par se mettre d'accord sur une valeur de clé. Ensuite, Alice écrit son message (m) et le chiffre avec la clé grâce à un algorithme nommé *Encrypt*, et envoie le résultat, le chiffré (CT pour *ciphertext* en anglais) à Bob. Celui-ci reçoit ce chiffré, et grâce à K et l'algorithme inverse de déchiffrement nommé *Decrypt*, il peut retrouver le message.

Cryptographie Asymétrique. Il est aussi possible que la clé de chiffrement et la clé de déchiffrement ne soient pas les mêmes. Auquel cas on utilisera les termes de *clé publique* (PK) pour la clé de chiffrement et de *clé secrète* (SK) pour la clé de déchiffrement. Ces clés sont créées par un algorithme de génération de clé nommé $KeyGen$. Si la clé secrète ne peut pas se déduire de la clé publique, cette dernière peut alors être diffusée sans risque (d'où son nom). On peut imaginer la clé publique comme un cadenas à code, que Bob initialise à une certaine valeur qu'il garde secrète. Alice peut alors utiliser ce cadenas pour fermer une boîte contenant son message secret et seul Bob pourra l'ouvrir.

Le meilleur des deux mondes. Dans la pratique, les primitives de cryptographie asymétrique sont beaucoup plus lentes que celles de cryptographie symétrique. On profite alors du meilleur des deux mondes grâce à un mécanisme d'échange de clé (KEM pour *Key Exchange Mechanism*). La cryptographie asymétrique est alors utilisée pour permettre à Alice d'échanger une clé avec Bob. Celle-ci sert ensuite de clé symétrique, ce qui permet d'échanger des messages chiffrés plus longs avec des meilleures performances.

Kerckhoffs. Un des plus grands retours sur l'expérience « Enigma » de la seconde guerre mondiale, en cryptographie, est désormais énoncé sous le nom de principe de Kerckhoffs. Bien que connu dès la fin du XIX^{e} siècle, Claude Shannon (père de la théorie de l'information) lui donne son importance en le reformulant : « L'adversaire connaît le système ». Cela signifie que la sécurité d'une primitive cryptographique ne doit pas se baser sur le fait que celle-ci soit méconnue. Les algorithmes doivent être considérés comme publics et les clés doivent être les seuls éléments considérés cachés. Ce principe veut aussi que l'on considère la stéganographie comme inefficace car elle se base justement sur l'ignorance de l'attaquant de la présence du message.

Standards de sécurité. L'application du principe de Kerckhoffs et des autres mesures de précaution apprises mène à deux exigences de sécurité notables. La première est que l'attaquant ne doit pas pouvoir mener d'attaque lui permettant de distinguer un chiffré dont il ne connaît pas la clé d'un message aléatoire, même si l'attaquant a accès à une machine qui permet de chiffrer et déchiffrer des messages (sécurité dite IND-CCA2). La seconde est que l'attaquant ne doit pas avoir d'attaque qui lui permet de distinguer le chiffré du message m_0 du chiffré du message m_1 s'il ne connaît pas la clé de chiffrement (sécurité IND-CPA). Toutes les primitives de chiffrement modernes et futures doivent satisfaire au moins ces exigences.

Bits de Sécurité. Bien que travaillant avec des concepts mathématiques, la cryptographie reste ancrée dans le monde réel. Ainsi, si une attaque existe en théorie, mais que son temps de calcul est tellement long qu'aucun attaquant réel

ne pourrait réalistement arriver à bout, on estime qu'elle est infaisable et donc qu'il n'y a pas d'attaque. On appelle le nombre de bits de sécurité d'une primitive cryptographique le logarithme du nombre d'opérations nécessaires à la meilleure attaque connue. Si on considère qu'un attaquant peut de manière réaliste posséder 1 million d'ordinateurs ($\approx 2^{20}$), que ceux-ci ont une fréquence de 1 GHz (1 milliard de calculs par seconde $\approx 2^{30}$) et qu'ils peuvent calculer pendant un an (environ 30 million de secondes $\approx 2^{25}$ on arrive à $2^{20+30+25} = 2^{75}$ cycles de calcul disponibles. En considérant que les ordinateurs se perfectionnent avec les années et deviennent moins coûteux et plus performants, il convient de prendre quelques puissances de 2 de sécurité pour trouver le standard de 2^{80}, soit 80 bits de sécurité. C'est la valeur qui a été recommandée pendant longtemps par des institutions officielles nationales et internationales. C'était vrai jusqu'en 2020 en France, avant que le standard passe à 100 [ANSSI 2014]. Il est intéressant de noter que dans le monde réel, à un niveau d'attaque correspond un prix : celui des ordinateurs et de l'énergie pour les faire fonctionner. On peut considérer un système de chiffrement comme résistant si les attaques sur celui-ci sont trop coûteuses pour être intéressantes au regard de ce qu'elles apprennent à l'attaquant.

Jeux de paramètres. La plupart des systèmes cryptographiques peuvent se décliner avec plusieurs jeux (ou sets) de paramètres. Ceux-ci indiquent la taille des variables du système. De manière générale, en faisant augmenter la taille de ces variables, on augmente aussi la difficulté de l'attaque. Cela peut sembler intéressant, mais en contrepartie le temps de calcul du chiffrement et du déchiffrement aux aussi augmentent. Il s'agit alors de trouver le jeu de paramètres minimal permettant aux utilisateurs légitimes de communiquer avec le niveau de sécurité dont ils ont besoin, de la manière la plus efficace possible. En augmentant la taille des variables, l'augmentation du temps de calcul est bien plus faible pour les utilisateurs légitimes que pour les attaquants, on préfère donc par précaution avoir des niveaux de sécurité légèrement surdimensionnés. Par exemple, on privilégierait 128 bits de sécurité plutôt que 80 ou 100, sachant que les ordinateurs classiques ne pourront pas atteindre ce niveau d'attaque d'ici longtemps [1]. Des applications critiques (banques, armées) peuvent même choisir d'avoir 256 bits de sécurité ou plus compte tenu de leurs enjeux.

Signature, Hachage. Le fait que le message ne tombe pas dans de mauvaises mains n'est que le minimum nécessaire pour établir une communication sécurisée entre Alice et Bob. Il leur faut aussi s'assurer que 1) ils sont bien en communication l'un avec l'autre et pas avec une tierce personne et 2) les messages qu'ils reçoivent n'ont pas été modifiés et qu'ils sont bien ceux générés par l'émetteur. On parle alors de propriétés comme l'authentification et l'intégrité. Celles-ci peuvent être assurées grâce à des algorithmes indispensables : une signature (et l'algorithme associé de vérification de la signature) et un algorithme de hachage. On peut comprendre un algorithme de hachage comme un « mixeur » de messages : la sortie semble aléatoire, et il n'est pas possible à partir de la sortie de deviner l'entrée correspondante. Grâce à cela, deux personnes peuvent se prouver qu'elles connaissent la même valeur en

1. Si la loi de Moore continue (doublage de la puissance de calcul tous les 18 mois) il faudrait 30 ans pour atteindre 100 bits de sécurité, et 72 ans pour atteindre 128.

échangeant le condensat (résultat du hachage) de cette valeur. La signature, telle qu'elle est le plus communément pratiquée, peut être comprise comme un chiffrement inversé : le signataire utilise sa clé privée pour signer le message et la clé publique appliquée à la signature doit retourner le message uniquement si celui-ci est inaltéré.

Complexité. On s'intéresse dans ce document à deux types d'algorithmes que l'on distingue : les algorithmes *polynomiaux* dont le temps de calcul est un polynôme selon la taille de l'entrée (n^3, n^2...) et ceux dont le temps est *exponentiel* selon la taille de l'entrée ($2^n, n^n$...). Afin de permettre que le chiffrement et le déchiffrement soient faciles et les attaques difficiles, on utilise des fonctions dites *à trappe*. Ces fonctions ont la particularité d'être polynomiales à calculer, mais exponentielles à inverser. Ces fonctions sont très rares et fortement étudiées en cryptographie. On parle de complexité asymptotique pour parler du comportement du temps de calcul lorsque n tend vers l'infini, car il n'est pas impossible que pour n proche de 0 un algorithme polynomial soit plus long qu'un algorithme exponentiel. Certaines complexités polynomiales sont notables : celles *linéaires* (de degré 1) et celles *constantes* (de degré 0).

Déterminisme. On dit des algorithmes qui renvoient toujours la même sortie pour une entrée donnée qu'ils sont déterministes. À l'inverse, d'autres algorithmes peuvent aussi se servir de valeurs aléatoires, ce qui sera susceptible de faire varier les résultats selon ces valeurs aléatoires. Certains algorithmes de cryptographie tels des chiffrement peuvent alors avoir des probabilités non-nulles de retourner un résultat incorrect. Si on fait en sorte que cette probabilité, certes non-nulle, soit tellement faible que le cas ne se présentera jamais en pratique, on dit que l'algorithme est correct « avec une probabilité écrasante ».

Quantique. Depuis l'émergence d'un nouveau type d'ordinateur, l'ordinateur quantique, il a fallu approfondir la nomenclature de la cryptographie. Ainsi, on parlera de systèmes cryptographiques *Classiques* s'ils s'exécutent sur des ordinateurs classiques et résistent à un attaquant qui utilise des ordinateurs classiques, de cryptographie *Quantique* s'ils s'exécutent sur des ordinateurs quantiques et résistent à un attaquant qui utilise des ordinateurs quantiques, ou finalement de cryptographie *Post-quantique* s'ils s'exécutent sur des ordinateurs classiques mais résistent à un attaquant qui utilise des ordinateurs quantiques.

> Certains « *ordinateurs quantiques* » sont en réalité des « *calculateurs quantiques* » : ils ne sont pas programmables et ne peuvent effectuer qu'un jeu fini d'opérations. N'ayant pas besoin de cette nuance, et pour suivre l'appellation la plus usitée, nous parlerons uniquement d'« *ordinateur quantique* » tout au long de cette thèse.

Attribute Based Encryption (ABE). Des systèmes de chiffrement avancés ont aussi vu le jour. Ceux-ci permettent d'adresser des messages à des destinataires non plus uniques, mais définis par des attributs. On peut par exemple envoyer un message avec une condition « Chercheur et LAAS » qui sera lisible par des personnes ayant des clés correspondant à la fois aux attributs « Chercheur » et « LAAS ». On appelle ces ABE dans lesquels la condition est portée par le chiffré des CP-ABE (pour Ciphertext Policy). A l'inverse, on peut baliser le chiffré comme « Budget »,

« Top secret », « 2022 », et une personne dont la clé est par exemple « Top Secret ou Urgent » pourra le déchiffrer. Ces ABE dans lesquels la condition est dans la clé sont appelés KP-ABE (pour Key Policy). Parmi les propriétés très importantes des systèmes ABE, on trouve le fait que toutes les clés soient différentes, mais aussi la non-collusion, c'est-à-dire que deux personnes malveillantes mettant en commun leur clé ne peuvent pas déchiffrer des messages que l'une d'entre elles n'aurait pas pu deviner seule.

Cette liste de rappels n'a couvert que des points préliminaires. Les sujets qui font l'objet de discussions plus approfondies au cours de cette thèse sont présentés dans le chapitre Contexte et Évolutions, plus technique, qui suit.

Chapitre 1

Contexte et Évolutions

1.1 Topologie en couches de la cryptographie

Historiquement, la cryptographie était pratiquée par des érudits et des fins connaisseurs du langage et utilisée principalement à des fins militaires, parfois commerciales, ou même récréatives. C'est autour de la seconde guerre mondiale que la cryptographie est progressivement vue de moins en moins comme une discipline littéraire et de plus en plus comme une discipline scientifique associée aux mathématiques [Singh 1999].

Avec l'apparition et la diffusion progressive des ordinateurs, la cryptographie s'est retrouvée transformée. L'exécution des algorithmes de chiffrement par des machines, incroyablement plus rapide que calculée à la main, permet d'envisager des chiffrements d'une plus grande complexité. Aussi, il n'y a plus besoin de savoir faire soi-même les calculs, ce qui permet de se servir de cryptographie sans aucune connaissance dans le domaine. L'utilisation de la cryptographie a pu se diffuser à la population générale et la notion de vie privée commence à prendre un sens concret. Aujourd'hui, grâce à cet accès à la cryptographie, la vie privée peut être de plus en plus considérée comme un droit humain et intégrée aux systèmes et aux lois [Eur-Lex 2016].

En parallèle, la cryptanalyse aussi devient dépendante de l'informatique. Le principe de Kerckhoffs permet de définir la mesure de la sécurité d'un cryptosystème uniquement selon la capacité de l'attaquant. Celle-ci est alors exprimée en cycles de calcul sur un ordinateur, ce qui illustre bien qu'une cryptanalyse à la main est devenue virtuellement impossible.

Les cryptographes se concentrent ainsi sur l'étude des fonctions à trappe dont l'inversion est aussi difficile qu'un problème réputé dur à résoudre, comme celui de la factorisation de grands nombres. L'objectif étant que plus une communauté bienveillante de chercheurs butte sur la recherche d'un meilleur algorithme de résolution de ce problème, moins une communauté malveillante plus petite a des chances de trouver un tel algorithme.

Cependant, la diversification des outils utilisant la cryptographie et des supports physiques sur lesquels s'exécutent les calculs a augmenté la surface d'attaque. La protection d'un message n'est plus uniquement garantie par l'utilisation d'une fonction à trappe résistante.

On propose dans la figure 1.1 une segmentation sémantique des différentes couches désormais impliquées dans la protection de données secrètes, et dont les faiblesses peuvent mener à des fuites d'informations secrètes, pour mieux comprendre les évolutions de celles-ci et les risques associés.

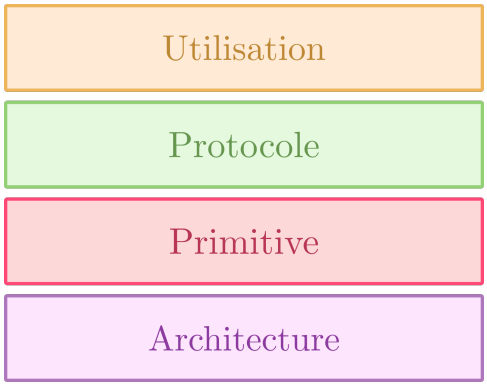


FIGURE 1.1 – Segmentation en couches des outils de protection de données

> La problématique de cette thèse s'articulant autour des relations entre ces couches et leurs évolutions. On utilisera un soulignage de la couleur d'une couche pour rappeler l'appartenance d'une information à la couche concernée.

1. La couche la plus haute est naturellement celle que pilote l'utilisateur (et donc la Couche Utilisation). On peut considérer dans cette couche toutes les applications possibles des outils cryptographiques : la plus couramment utilisée est certainement la transmission de données sur l'Internet, mais on peut aussi penser à d'autres utilisations comme l'Internet des objets (IoT), la création d'identités uniques ou temporaires (UIds), les cryptomonnaies, la génération procédurale dans les jeux vidéos, l'anonymat et la dissimulation de transactions, le traitor-tracing pour trouver qui diffuse un contenu censé être restreint...
2. La couche juste en dessous est celle qui gère les transmissions entre les utilisateurs. On peut l'appeler la Couche Protocole. Il s'agit de définir les données transmises et donc les modes d'utilisation des outils cryptographiques, sans pour autant spécifier ceux-ci. On peut penser à des protocoles tels que le three-pass-protocol de Shamir [Shamir 1981], le proxy-reencryption [Ateniese 2006], le routage en oignon TOR [Dingledine 2004], voire même les modes d'utilisations bien connus d'AES (AES-GCM) ou de RSA (RSA-OAEP). On choisit de ne pas inscrire les couches télécommunications et réseaux dans cette topographie, en considérant que tous les échanges de données sont « parfaits » : pas de perte de données, les acquittements de réception ne seront pas mentionnés.
3. En dessous, la Couche Primitive contient les primitives cryptographiques. Elle s'interface avec la couche précédente en proposant des fonctions abstraites (en boîte noire), telles que *hash*, *encrypt*, *decrypt*... Cette couche contient le détail des algorithmes qui calculent le résultat de ces fonctions (AES, RSA, DH, ECDH, SHA2, Chacha20...). De ces algorithmes dépendent donc la complexité asymptotique. C'est dans cette couche que s'inscrit l'étude des propriétés de sécurité des primitives et de leurs performances.
4. Finalement, ces algorithmes, bien qu'ils soient des objets mathématiques, ont besoin d'un support physique afin de produire un résultat. Ce support est

classiquement un ordinateur personnel, téléphone ou serveur, dont les caractéristiques et les choix de conception jouent sur les performances des algorithmes. On peut donc l'appeler Couche Architecture. Les attaques par canaux auxiliaires et leurs contre-mesures sont des sujets typiques de cette couche.

On peut illustrer le fonctionnement de ces différentes couches en imaginant un utilisateur voulant envoyer un message sur un forum (utilisation), son navigateur va utiliser HTTPS (protocole), dont le fonctionnement repose sur l'authentification RSA2048 (primitive), grâce aux calculs effectués sur son ordinateur personnel (architecture).

La couche historiquement la plus associée à la cryptographie est la Couche Primitive, les autres étant historiquement de moindre intérêt : les messages étaient calculés à la main ou mécaniquement, les protocoles limités à l'envoi du chiffré et avaient pour seule utilisation des discussions privées typiquement militaires. Cependant, les autres couches s'étant progressivement agrandies, elles font partie intégrante de la sécurité et la protection des systèmes. On peut montrer que des failles sur n'importe quelle couche peuvent avoir un impact sur la sécurité du système dans son ensemble :

1. Sur la couche utilisation, si l'utilisateur a partagé naïvement son mot de passe dans le forum, aucune autre couche n'a d'utilité. Souvent, sur cette couche, c'est l'utilisateur lui-même le problème. Malheureusement, la seule solution efficace pour le moment semble être les actions de sensibilisation, qui sont mises en place de l'école jusqu'en entreprise, mais qui peuvent tout de même être appuyées de quelques garde-fous logiciels.
2. Dans la même logique, le choix d'un protocole non-sécurisé (par exemple http plutôt que https) introduit aussi des vulnérabilités. Les failles des protocoles sont souvent dues à leur conception.

   > Le travail de F.Galtier et R.Cayre dans notre équipe de recherche met en valeur un grand nombre de failles dans des protocoles Bluetooth Low Energy [Cayre 2021]. Celles-ci semblent être souvent le résultat d'une négligence partielle ou totale des aspects sécurité et d'un focus sur la fonctionnalité et la rentabilité, comme aussi illustré dans [Newlin 2016].

   Ici, ce sont les concepteurs des protocoles qui peuvent laisser des failles. Contre celles-ci, des outils de preuves formelles tels que ProVerif peuvent aider à éviter quelques erreurs connues.
3. Les cryptographes ne connaissent que trop d'exemples d'attaques sur la couche primitive, dont la complexité devient inférieure au seuil du nombre de bits de sécurité. On peut citer les plus connues : les attaques linéaires, différentielles, par rencontre au milieu, les compromis temps-mémoire... Les réductions de sécurité permettent de prouver qu'un système cryptographique est au moins aussi sûr qu'un problème mathématique connu est dur à résoudre. Or, la difficulté de résolution de ce problème se mesure avec la complexité du meilleur algorithme qui résout celui-ci. Les attaques viennent donc du fait que les mathématiciens et cryptographes découvrent petit à petit de meilleurs algorithmes.

4. Les ordinateurs en tant qu'objets physiques sont eux aussi vulnérables. Lors du déroulement de l'algorithme, on peut réaliser des mesures physiques qui renseignent sur les variables et paramètres de celui-ci. Par exemple, on peut mesurer le temps de calcul, la consommation de courant du processeur, ou insérer des sondes de mesure du champ électromagnétique produit. La conception de l'architecture de l'ordinateur peut aussi créer des failles de sécurité, comme les fameuses attaques Spectre, Meltdown, et récement HertzBleed [Abu-Ghazaleh 2019, Wang 2022].

Si une seule des couches présente une vulnérabilité, le caractère privé du message est compromis. Il est donc nécessaire non seulement d'étudier les outils et méthodes disponibles à l'intérieur de chacune de ces couches, mais aussi d'étudier comment celles-ci fonctionnent de manière transversale, une modification dans une couche pouvant avoir des répercussions sur celles adjacentes.

1.2 Évolutions du contexte cryptographique à l'approche de l'ordinateur quantique

Avant de nous concentrer sur les relations entre les différentes couches, nous allons présenter les évolutions qu'elles subissent individuellement avec le temps. Nous nous concentrerons principalement sur les changements dûs à l'apparition d'ordinateurs quantiques, d'abord théoriques, mais maintenant de plus en plus concrets.

Afin de mieux comprendre à quel point l'impact sur certaines couches est fort, nous commencerons par une présentation simplifiée du fonctionnement et des capacités des ordinateurs quantiques. Ensuite, nous regarderons quels changements ont lieu sur nos ordinateurs classiques en parallèle du développement de l'ordinateur quantique et comment ces évolutions vont s'articuler.

> Il me serait impossible de présenter ici tout un cours de calcul quantique. Je recommande fortement à un lecteur curieux souhaitant avoir une compréhension plus poussée du calcul quantique de suivre le cours donné par Ryan O'Donnell, disponible en ligne [O'donnell 2018]. Pour une lecture amusante et pédagogique sur le sujet je conseille *Bananaworld* de Bub Jeffrey [Bub 2016].

Qbits. Dans un ordinateur classique, l'information est représentée sous forme de bits. Il s'agit physiquement d'un courant (voltage) qui sera interprété comme un 0 s'il est faible ou comme un 1 s'il est élevé. Ce courant est la mesure d'un nombre gigantesque d'électrons se déplaçant dans un matériau conducteur. Les physiciens sont capables de passer à une échelle beaucoup plus petite, « quantique », pour représenter l'information associée à l'état d'une seule particule. Plusieurs méthodes sont possibles pour utiliser des particules (qbits) dans un ordinateur quantique, dont : le spin d'un électron (0 pour « Up », 1 pour « Down ») [Loss 1998], la polarisation d'un photon (0 si horizontale, 1 si verticale) [Bub 2016], mais d'autres bien plus complexes existent [Nayak 2008].

La différence principale entre les bits et qbits est que ces derniers ne sont pas restreints aux états 0 et 1. Ils peuvent prendre une infinité de valeurs différentes qui

peuvent être soit $|0\rangle$, l'équivalent du 0 classique, soit $|1\rangle$ l'équivalent du 1 classique, soit une *superposition* des deux : $\alpha|0\rangle + \beta|1\rangle$ avec $\alpha, \beta \in \mathbb{C}$ et $|\alpha|^2 + |\beta|^2 = 1$. La proximité de cette définition avec celle des probabilités (dont la somme doit faire 1) peut justifier l'appellation de « probabilités complexes ». Cependant, cette appellation est à prendre avec des pincettes, la différence entre les capacités du calcul probabiliste et quantique ayant été prouvée théoriquement et expérimentalement (inégalité CHSH) [Clauser 1969]. Le terme d'*amplitude* est alors utilisé pour décrire α et β.

Opérations sur les qbits. A l'image des portes logiques classiques (NOT, AND, OR, XOR...), il est possible d'effectuer des opérations sur un ou plusieurs qbits. Sur un qbit, on trouvera par exemple la porte de Pauli (similaire à une porte NOT), qui à $\alpha|0\rangle + \beta|1\rangle$ associera $\beta|0\rangle + \alpha|1\rangle$, ou encore la porte de Hadamard qui y associe $\frac{1}{\sqrt{2}}(\alpha+\beta)|0\rangle + \frac{1}{\sqrt{2}}(\alpha-\beta)\alpha|1\rangle$. Il est démontré possible de recréer toutes les portes classiques à l'aide de portes quantiques, qui auront alors sur $|0\rangle$ et $|1\rangle$ le même fonctionnement que les classiques avec 0 et 1). Cela a notamment pour implication que tous les circuits classiques sont reproductibles sur des ordinateurs quantiques sans augmentation de complexité. Cependant, l'inverse n'est pas vrai : par exemple, la porte de Hadamard ne peut pas être reproduite classiquement. Il faut noter que la difficulté et le coût de la création des différentes portes quantiques peuvent varier selon le support physique choisi pour les qbits.

Mesure d'un qbit. Afin de pouvoir interpréter le résultat d'un calcul quantique, il faut observer l'état de nos qbits à la fin de celui-ci. Cependant, contrairement aux ordinateurs classiques, ce résultat n'est pas déterministe : par exemple, le qbit $\alpha|0\rangle + \beta|1\rangle$ s'il est observé (ou mesuré), sera dans l'état $|0\rangle$ avec une probabilité de $|\alpha|^2$ (l'appareil de mesure indiquant alors 0) et dans l'état $|1\rangle$ avec une probabilité de $|\beta|^2$ (l'appareil de mesure indiquant alors 1).

Théorème de non-clonage. Première différence notable avec les bits classiques : on peut facilement copier la valeur d'un bit dans un autre bit, mais il n'existe pas de façon de créer une copie d'un qbit correcte pour tous les états possibles. La preuve est mathématique [1], et seul un changement de notre modèle de la physique pourrait justifier l'inverse. Cette propriété peut permettre de s'assurer de « l'intégrité » d'un qbit.

BB84. Ce protocole est un exemple de ce qui peut être fait grâce à ce théorème de non-clonage. En se basant sur de la polarisation selon deux bases différentes, Alice et Bob peuvent échanger des qbits, et leurs mesures leur permet de partager des bits d'information. Parmi les bits partagés, certains peuvent être « consommés » (rendus publics) ce qui permet de s'assurer qu'une tierce personne n'a pas déjà mesuré un qbit. Cet attaquant ayant une chance sur 4 d'être détecté à chaque bit échangé, Alice et Bob peuvent créer une clé avec une sécurité arbitrairement haute en complexité linéaire. Si ce protocole était universellement accessible, il stopperait net le jeu du chat et de la souris qui oppose la cryptographie et la cryptanalyse par une victoire de la cryptographie. Cependant, même si celui-ci a été réalisé dans des

1. Les transformations sur l'état des qbits sont représentées par des matrices unitaires, or les fonctions permettant une copie ne peuvent pas être représentées par celles-ci.

dimensions impressionnantes (240km) [Fröhlich 2017], nous en sommes encore trop loin scientifiquement pour le rendre accessible à tous.

> Je recommande à ceux qui n'aurait jamais entendu parlé de ce protocole la lecture de l'article Wikipédia « BB84 » en français, qui est assez accessible et comme c'est rarement le cas, meilleur que sa contrepartie anglophone.

Avantage quantique. Les différences notables de complexité entre classique et quantique s'expliquent par les propriétés des systèmes à plusieurs qbits. Les qbits formant ces systèmes ont la capacité de s'intriquer, c'est-à-dire avoir des états dépendants les uns des autres peu importe la distance qui les sépare. La mesure de l'un de ces qbits sera alors déterminée par la mesure de l'autre, peu importe le temps qui sépare les mesures.

Associée à une autre propriété, cette capacité donne son efficacité au calcul quantique : il s'avère qu'un système de n qbits est représenté par 2^n variables. Il est commun d'en profiter lors des algorithmes de calcul quantique, en commençant par créer un état qui représente les 2^n valeurs de n bits, avant de tous les soumettre « simultanément » à une fonction. (c sur la figure 1.2)

Cependant, ce procédé a une limite : un seul résultat peut être lu, et on ne connaît pas l'entrée ayant donné ce résultat. On résout ce dernier problème en ajoutant assez de qbits à l'entrée de la fonction et on l'adapte pour avoir en sortie la valeur d'entrée intriquée au résultat. La lecture du résultat conditionne donc les qbits d'entrée à ne pouvoir prendre que des valeurs associées au résultat lu. (e sur la figure 1.2)

Transformée de Fourier quantique. La transformée de Fourier est une pierre angulaire de l'analyse du signal, car elle a pour effet de remplacer une représentation temporelle des données par une représentation fréquentielle et vice-versa. Cette transformation permet de mettre en valeur des fréquences fortes, ou des éléments de périodicité d'une liste par exemple.

Grâce à des portes quantiques de « rotation » (dont la porte de Hadamard fait partie, et dont aucun équivalent classique n'existe), on arrive à déterminer que la complexité en nombre de portes est de n^2 alors qu'elle est de $n2^n$ pour la transformée de Fourier rapide, voire 2^{2n} pour l'algorithme naïf de la transformée de Fourier discrète. La transformée de Fourier quantique permet alors de transformer efficacement une superposition d'états en superposition de fréquences. (f sur la figure 1.2) Combiné aux autres éléments avantageux des circuits quantiques, cela permet de trouver des indices sur la périodicité d'une fonction (circuit complet de la figure 1.2).

Algorithme de Shor. Découvert en 1994 et incroyablement impactant, cet algorithme permet d'effectuer une factorisation ou de résoudre le problème du logarithme discret en temps polynomial (n^3), grâce aux informations récoltées avec le circuit précédemment présenté. Or, l'hypothèse de sécurité des primitives les plus connues et répandues de la cryptographie classique est que ces problèmes soient durs à résoudre. La complexité de factorisation et du logarithme discret étant passée d'exponentielle à polynomiale, c'est une majorité écrasante des primitives cryptogra-

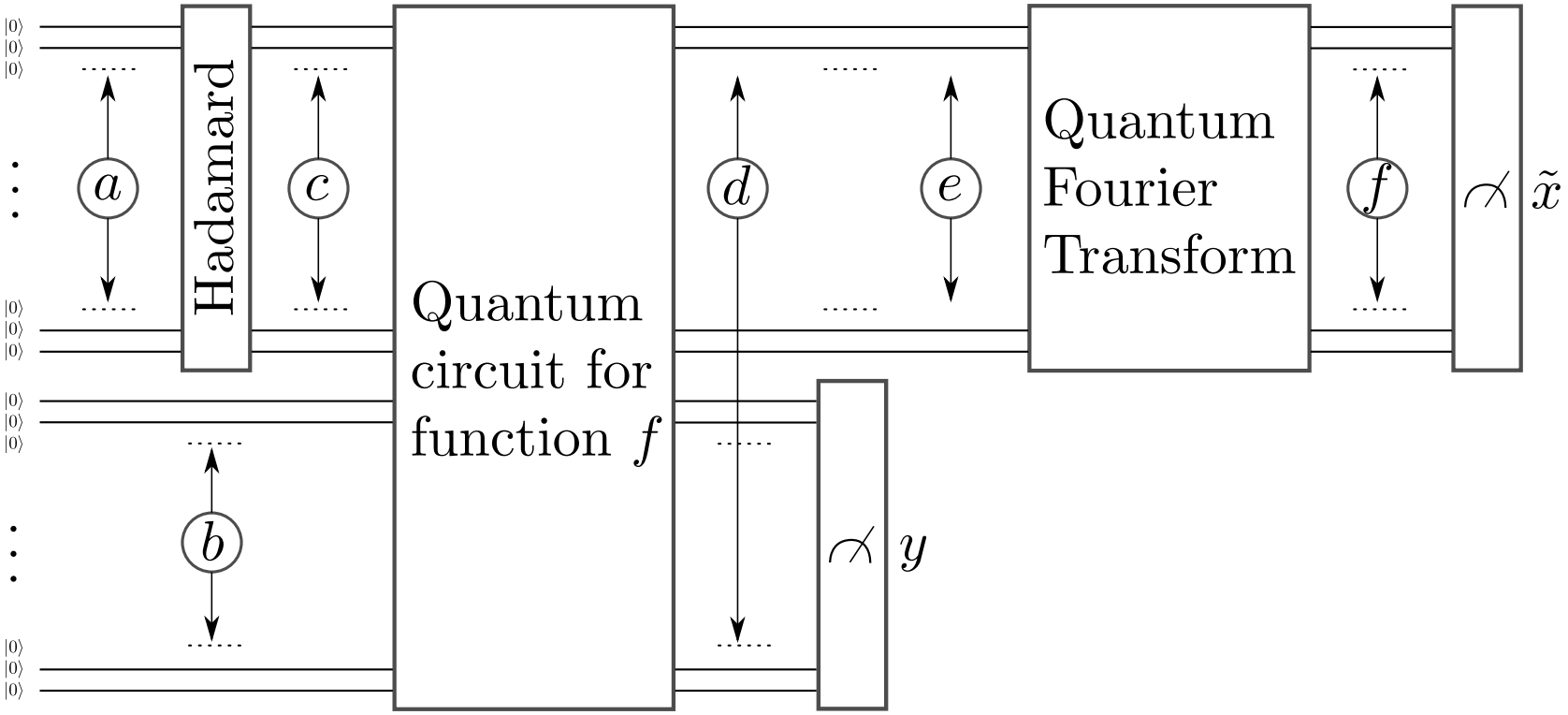


FIGURE 1.2 – Circuit quantique aidant à trouver des caractéristiques de périodicité de la fonction f
a) n qbits « entrée » initialisés à $|0\rangle$
b) m qbits « résultat » initialisés à $|0\rangle$
c) Superposition de toutes les 2^n entrées possibles avec amplitudes égales
d) Superposition de toutes les 2^n paires entrée-résultat possibles avec amplitudes égales
e) Superposition de toutes les entrées x telles que $f(x) = y$
f) Superposition de toutes fréquences $\tilde{x}$ avec amplitude selon leur force
Mesures : $\tilde{x}$, une fréquence des entrées dont l'image est y

phiques utilisées (RSA/DH) qui se retrouvent en danger, de même pour les systèmes qui se reposent sur celles-ci.

> Il s'agit peut-être la découverte la plus impactante en cryptographie moderne depuis le décryptage des codes Enigma. Cela se justifie par le fait que depuis peu, *« Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer »* [Shor 1999] est devenu l'article le plus cité en lien avec la cryptographie (12000 citations) devant le cryptosystème ElGamal [ElGamal 1985] (11750 citations), et le chiffrement basé identité [Boneh 2001] (10450 citations) [Rieck 2022].

Algorithme de Grover. Une autre utilisation de la capacité des ordinateurs quantiques à « faire des calculs sur toutes les entrées possibles en même temps » permet de simplifier la résolution d'un autre problème très connu. Le problème SAT est le suivant : étant donné f qui prend n bits en entrée, y a-t-il une entrée x pour laquelle $f(x) \neq 0$? Sans connaître f, il ne semble n'y avoir qu'à tenter tous les x et répondre oui si un résultat est non nul. Comme il y a 2^n possibilités, cela arrivera en moyenne au bout de 2^{n-1} calculs de f. Ce problème permet d'encapsuler tous les problèmes visant à trouver un antécédent de y par une fonction g : il suffit de définir $f(x) = 1$ si $g(x) = y$ et 0 sinon. C'est exactement le problème que souhaite résoudre un attaquant, qui à partir du chiffré, veut inverser le chiffrement pour retrouver le message.

Grâce à l'algorithme quantique de Grover, le problème SAT est désormais d'une complexité de $2^{n/2}$, ce qui peut basiquement se traduire par une division par deux de tous les niveaux de sécurité des primitives cryptographiques, même symétriques (AES) et hachages (SHA).

Every cloud has a silver lining. D'après la lecture des articles de recherche publiquement disponibles, personne ne dispose d'ordinateur quantique ayant réussi à factoriser des nombres plus grands que 1 005 973 ($\approx 2^{20}$) [Peng 2019]. Il reste donc un chemin technique et scientifique considérable à parcourir avant de considérer les systèmes que nous utilisons actuellement comme vulnérables (cela pourrait même ne jamais être le cas...). La cause notable des limitations des ordinateurs quantiques actuels est la difficulté de maintenir un qbit stable dans le temps. Ceux-ci ont tendance à changer d'état avant que l'on ait pu effectuer toutes les étapes de calcul que l'on souhaite. C'est pour cela qu'un des problèmes les plus « en vogue » dans le calcul quantique est celui de la correction d'erreurs car si une porte de correction d'erreurs ou un algorithme assez rapide est trouvé, cela ouvrirait la porte à des calculs plus longs et donc de meilleurs résultats.

> On voit que la métrique de coût est plus adaptée que celle de bits de sécurité, car elle met sur une même échelle la sécurité contre les deux types d'ordinateurs. Le coût des attaques est bien trop grand sur ordinateur quantique, même si la complexité est moindre.

Effort de standardisation post-quantique du NIST. La chronologie de la transition vers le quantique en cryptographie peut donc démarrer à l'apparition du calcul quantique et pourrait se terminer par une utilisation universelle de distribution quantique de clés (QKD).

Cette transition sera certainement très longue. Elle a commencé dans un contexte purement scientifique et se poursuit principalement pour des intérêts stratégiques par les banques, armées, qui de plus en plus s'emparent de l'utilisation de ces protocoles. À présent, certaines commercialisations commencent à voir le jour. L'infrastructure nécessaire n'en permet pas encore une utilisation portable ou universelle.

Durant cette même période de temps pourrait se trouver aussi le jour où par progrès scientifique, les ordinateurs quantiques pourront rendre réalisables des attaques sur les schémas de cryptographie classique. Si ce jour arrivait alors que l'usage de QKD soit généralisé, il n'y aurait aucun souci à se faire. Par prudence, il faut anticiper le cas où il arriverait avant.

Cette problématique est identifiée comme sérieuse en 2016, où la communauté scientifique se réunit autour du NIST pour se lancer dans une procédure de détermination de primitives post-quantiques permettant de résister aux ordinateurs quantiques via des calculs classiques en attendant la cryptographie quantique.

Heureusement, tout n'est pas à refaire : l'algorithme de Grover a baissé les niveaux de sécurité, mais il n'est pas polynomial. Pour les primitives qui ne sont touchées que par cet algorithme (cryptographie symétrique et hachages), il n'y a qu'à prendre des jeux de paramètres plus grands pour compenser. Certes, cela veut dire un peu plus de temps de calcul pour les utilisateurs légitimes, mais ces outils sont tellement rapides et optimisés que cela peut être considéré comme un surcoût insignifiant.

Cependant, c'est la cryptographie asymétrique qui se retrouve sans primitives sécurisées. Il faut alors trouver des concepts mathématiques qui garantissent une sécurité contre les algorithmes quantiques, en faire des primitives cryptographiques, et parmi celles-ci standardiser les meilleures, ce que cherche à faire le NIST.

Cette standardisation n'est pas totalement terminée, elle s'est réalisée en plusieurs rounds :

— **Round 1** (12/2017) : Le NIST accepte 69 primitives candidates. Celles-ci sont évaluées selon leur niveau de sécurité, leur performance, voire leur élégance mathématique. Des attaques, ou des performances trop basses vont éliminer une primitive.

— **Round 2** (01/2019) : Le NIST retient 26 primitives, en fusionnant éventuellement celles trop similaires. Cette fois-ci, la sélection se concentre davantage sur les tailles des clés, des chiffrés et le temps d'exécution de KeyGen, Encrypt et Decrypt.

— **Round 3** (07/2020) : Sont retenus 7 finalistes (4 KEMs, 3 signatures) et 8 remplaçants (5 KEMs, 3 signatures) au cas où des attaques seraient découvertes.

— **Round 4** (07/2022) : Un premier standard est nommé parmi les primitives finalistes basées sur les Réseaux Euclidiens. Celles basées sur les codes correcteurs d'erreur restent à l'étude pour peut-être une nomination supplémentaire.

Cette standardisation étant bientôt finie et avec déjà une primitive nommée (CRYSTALS-KYBER), les outils cryptographiques actuels vont pouvoir être mis à jour, avec du post-quantique en « roue de secours », qui pourra être utilisé à tout moment si la menace de la cryptanalyse quantique semble se concrétiser.

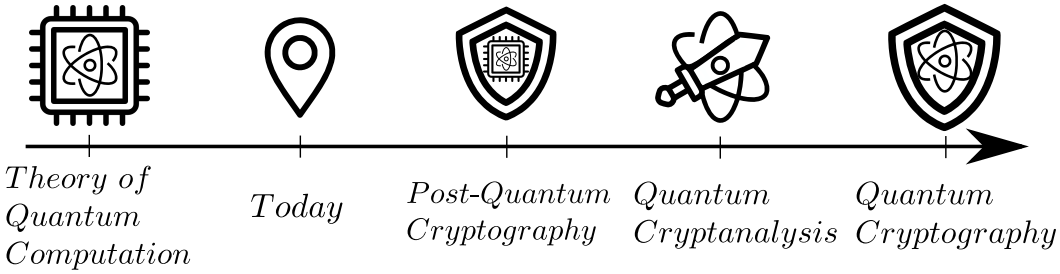


FIGURE 1.3 – Chronologie idéale de la transition cryptographique

> D'autres chronologies sont possibles, bien que moins réjouissantes. 1) La cryptanalyse quantique peut déjà exister (c'est très peu probable). 2) La cryptanalyse quantique peut arriver avant l'adoption de la cryptographie post-quantique, les systèmes non mis à jour seront vulnérables. 3) On découvre des attaques sur l'ensemble des primitives post-quantiques et ne pourrions plus que nous reposer sur de la stéganographie pour préserver notre vie privée.

Hidden Subgroup Problem. Il est important, bien que plus technique, de présenter ce pourquoi certaines primitives sont résistantes aux ordinateurs quantiques et pas d'autres. En réalité la différence est plus faible qu'il n'y paraît : les primitives classiques basées sur la factorisation (ex : RSA) ou le logarithme discret (DH), et des primitives post-quantiques basées sur les réseaux euclidiens (CRYSTALS-KYBER), ou sur les codes correcteurs d'erreur (BIKE) ont en fait un point commun. Ces primitives sont toutes des variations d'un même problème mathématique : le Hidden Subgroup Problem.

Dans ce problème, on définit comme paramètre public un groupe G. Le secret à deviner va être un sous-groupe H de G. Pour deviner H, l'attaquant peut choisir des éléments $g \in G$. Cet élément subit une projection sur G/H, ce qui signifie que deux éléments $g_1, g_2 \in G$ tels que $g_1 - g_2 \in H$ auront le même résultat. Pour compliquer les choses, le résultat n'est pas affiché tel quel, mais sous la forme d'une chaîne de caractères qui n'a rien à voir avec sa valeur. Ces chaînes de caractères pouvant par exemple être des couleurs, technique appelée la colorisation. Cette notion de couleur est utilisée dans la littérature pour illustrer qu'il n'y a pas d'opération possible entre les couleurs, à part vérifier leur égalité.

Ce que permet l'algorithme de Shor c'est de résoudre ce problème, mais à une condition, c'est que le groupe G soit Abélien, autrement dit, commutatif. C'est de là que vient toute la différence entre la cryptographie classique (qui utilise donc des groupes Abéliens) et post-quantique qui utilise des groupes non-Abéliens. Parmi ces derniers, deux types de groupes particuliers ont toute leur importance : les groupes dihédraux D_{2n} des rotations et symétries des polygones réguliers à n côtés, et les groupes symétriques S_n des permutations d'ensembles à n éléments.

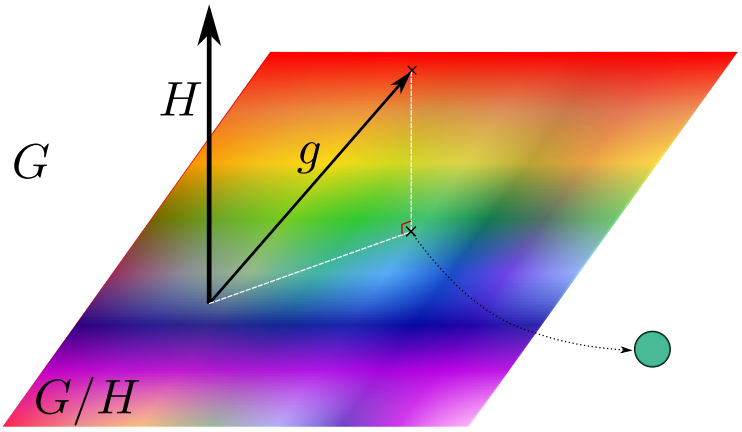


FIGURE 1.4 – Exemple de Hidden Subgroup Problem dans un espace vectoriel à trois dimensions

Il a été prouvé que résoudre le HSP dans ces groupes correspond respectivement à résoudre le problème du vecteur le plus court (SVP - Shortest Vector Problem) [Regev 2003] et le problème des isométries de graphes [Ettinger 1999]. La difficulté à résoudre ces deux problèmes est le fondement de la sécurité de primitives post-quantiques. Les systèmes basés Réseaux euclidiens et Codes Correcteurs d'erreurs dépendent de variantes du SVP (respectivement Learning With Error et Syndrome Decoding). Le problème des isométries de graphes, moins courant, a des utilisations dans certaines preuves Zero-Knowledge.

> C'est pour cette raison qu'il existe un standard déjà nommé CRYSTALS-KYBER, qui a été choisi comme la meilleure primitive dont la sécurité dépend de LWE. Les primitives encore restantes sont celles basées sur le Syndrome Decoding.

Dans les groupes dihédraux, plusieurs algorithmes inspirés de près ou de loin par l'algorithme de Shor existent qui visent à résoudre le HSP. Au fur et à mesure des inventions, la complexité diminue, ou des compromis temps-mémoire sont trouvés, mais elle ne cesse d'être exponentielle, ce qui justifie la sécurité des primitives post-quantiques actuelles [Regev 2004a, Kuperberg 2005a, Kuperberg 2011a].

Chiffrements commutatifs. Les primitives de chiffrement ne sont pas les seules qui se servaient des groupes Abéliens. Certains protocoles profitaient aussi de la commutativité des chiffrements entre eux pour créer des propriétés de sécurité. Ces protocoles dits de chiffrement commutatif sont aussi sources de vulnérabilités face à un ordinateur quantique. Notablement parmi ceux-ci, les protocoles d'interception réglementée, permettant de concilier sécurité et vie privée sont touchés par ces vulnérabilités.

Calcul en mémoire. Pendant que les ordinateurs quantiques deviennent progressivement une réalité, les ordinateurs classiques ne cessent eux aussi de s'améliorer. Certes, il semble que les efforts d'augmentation de la densité de transistors commencent à s'essouffler, à cause de barrières techniques comme la dissipation de

température... La loi de Moore, qui prédit le doublement des capacités de calcul tous les deux ans montre ces premiers signes d'essoufflement. Cependant, la miniaturisation, l'augmentation du nombre de transistors et de la fréquence d'horloge ne sont pas les seules solutions pour augmenter les performance des ordinateurs.

La conception fonctionnelle de l'ordinateur, sa micro-architecture, est aussi un des facteurs de performance. Par exemple, sur une architecture 32 bits, bien qu'on sache qu'il est possible de créer toutes les fonctions binaires de 2×32 bits en entrée vers 32 bits en sortie avec un nombre limité de portes logiques, on implémente que les plus communes (addition, multiplication...) et pour effectuer des fonctions plus rares, on utilise un algorithme basé sur ces opérations.

Ce choix de fonctions (Instruction Set) codées « en dur » dans la machine se répercute sur la conception d'algorithmes, car on préférera utiliser ces fonctions plus rapides à calculer. Par conséquent certains calculs se retrouvent désavantagés, si trop complexes à réaliser grâce à ces fonctions. Des jeux d'instructions étendus aident à résoudre ce problème.

Une autre limite est bien entendu la taille des opérandes : pour faire une comparaison de deux nombres écrits sur 128 bits, il faudra au minimum quatre comparaisons de 32 bits successives. Ce qui est dommage, c'est que sur une architecture simple, ces opérations se feront toutes dans l'Unité Arithmétique et Logique les unes à la suite des autres, alors qu'étant indépendantes, elles auraient pu se faire toutes en même temps. La réponse à ce problème dit de parallélisation des opérations est soit du calcul réparti (entre plusieurs processeurs), soit une augmentation de la taille des opérandes. Celle-ci est notamment possible à l'aide du calcul dit vectoriel (ex : AVX2), qui peut effectuer des opérations ordinaires sur 256 bits en simultané par exemple.

Ce niveau de parallélisme, bien que très pratique, limite encore des applications manipulant des variables de dimensions supérieures, ou plus nombreuses, comme c'est le cas des réseaux neuronaux. Pour s'adapter à ces besoins, les chercheurs proposent des façons de transformer des éléments de stockage en éléments de calcul (Processing In Memory) [Siegl 2016]. Il s'agit d'adapter des zones de la mémoire (bien plus grandes que le coeur de processeur) afin de pouvoir faire des calculs sur des grandes plages de données en simultané.

Attaques par canaux auxiliaires. Les primitives cryptographiques peuvent avoir des preuves de sécurité basées sur des problèmes théoriques difficiles à résoudre, mais ces preuves sont bien inutiles si les ordinateurs sur lequel les primitives s'exécutent permettent à tous d'accéder aux valeurs protégées. Comme on n'observe pas que le canal principal (le chiffré), mais que l'on se sert d'informations physiques, on parle d'attaques par canaux auxiliaires. La plus célèbre d'entre elles est l'attaque par analyse de consommation sur RSA.

> Le chiffrement RSA va effectuer l'une de deux opérations pour chaque bit de la clé : *square* si c'est un 0, *square and multiply* si c'est un 1. *Square* est plus rapide que *square and multiply*, on peut donc savoir en mesurant la consommation énergétique du processeur si les bits de la clé sont des 1 ou des 0.

Pour éviter des attaques par simple analyse du temps de calcul, les primitives cryptographiques ont des implémentations dites en temps constant : le temps de calcul est indépendant du secret (pas de `if`, `for`, `while` basé sur un secret). Cela ne suffit pas à empêcher les attaques par timing, car toutes les données de l'ordinateur n'ont pas les mêmes temps d'accès : une donnée dans un registre du processeur est accessible instantanément, mais une donnée en mémoire centrale prendra plus de temps à être disponible. Il faut s'assurer que l'implémentation est aussi en adresses constantes : les adresses mémoire des variables sont indépendantes des secrets.

Mais ce sont des contremesures limitées car l'observation du système peut se faire selon d'autres critères que le temps. On peut comparer la quantité d'énergie consommée lors d'un même chiffrement en changeant juste la clé ou le message. Les variations de cette consommation peuvent donner des indices sur ceux-ci. On peut même aller jusqu'à utiliser des sondes afin de mesurer les émissions électromagnétiques à des endroits très précis du processeur.

Modèle du poids de Hamming. Les bascules de transistors des mémoires de nos ordinateurs actuels ont une particularité : leur consommation est plus faible lors des cycles où elles conservent leurs valeurs (0 vers 0 ou 1 vers 1) que lorsqu'elles changent de valeur (0 vers 1 ou 1 vers 0). La différence de consommation du processeur va alors être affine au nombre de changements de valeurs (appelés « bitflips »). Si une valeur a est remplacée par une valeur b, les bits ayant changé de valeur sont les bits à 1 de $a \oplus b$. Leur nombre, appelé poids de Hamming (HW), est donc $HW(a \oplus b)$. On voit que si a valait 0, c'est le poids de Hamming de b qui est mesurable, ce qui est problématique si b est secret.

Ce modèle, associé à une connaissance fine de l'architecture de l'ordinateur peut aider à trouver des vulnérabilités dans des calculs et des moyens de mieux protéger des données d'attaques toujours plus évoluées et performantes.

Consommation de données. En devenant plus performants, nos ordinateurs classiques permettent de supporter une consommation d'informations toujours plus importante, autrement dit « l'offre crée la demande ». Toutes les statistiques d'utilisation des plus célèbres plateformes de streaming mènent à la même conclusion : le temps d'écran s'allonge, une proportion de plus en plus grande de celui-ci est accordée à la vidéo, dont une part grandissante est attribuée à la vidéo en direct [Viméo 2020]. Les plateformes de diffusion et les réseaux sociaux se sont adaptés à cette demande.

Protection et anonymat. En contrepartie, la conscience des problématiques de vie privée dans la navigation en ligne n'a jamais été aussi grande [Galov 2022]. La demande pour des services sécurisés et protecteurs de la vie privée tels que des messageries chiffrées de bout-en-bout ou des services de VPN n'a jamais été aussi forte. Cependant, ces outils d'anonymisation sont rarement conciliables avec la demande grandissante de diffusion massive de vidéos en direct car ils demandent de transiter par plus de noeuds du réseau, ce qui crée un supplément de charge de travail. Et les utilisateurs d'internet sont souvent prêts à accepter un compromis entre une partie de leur vie privée et la capacité d'accéder à de meilleures fonctionnalités.

Vue d'ensemble. On peut regrouper l'ensemble des évolutions et des avancées des différentes couches dans un schéma-bilan présenté figure 1.5.

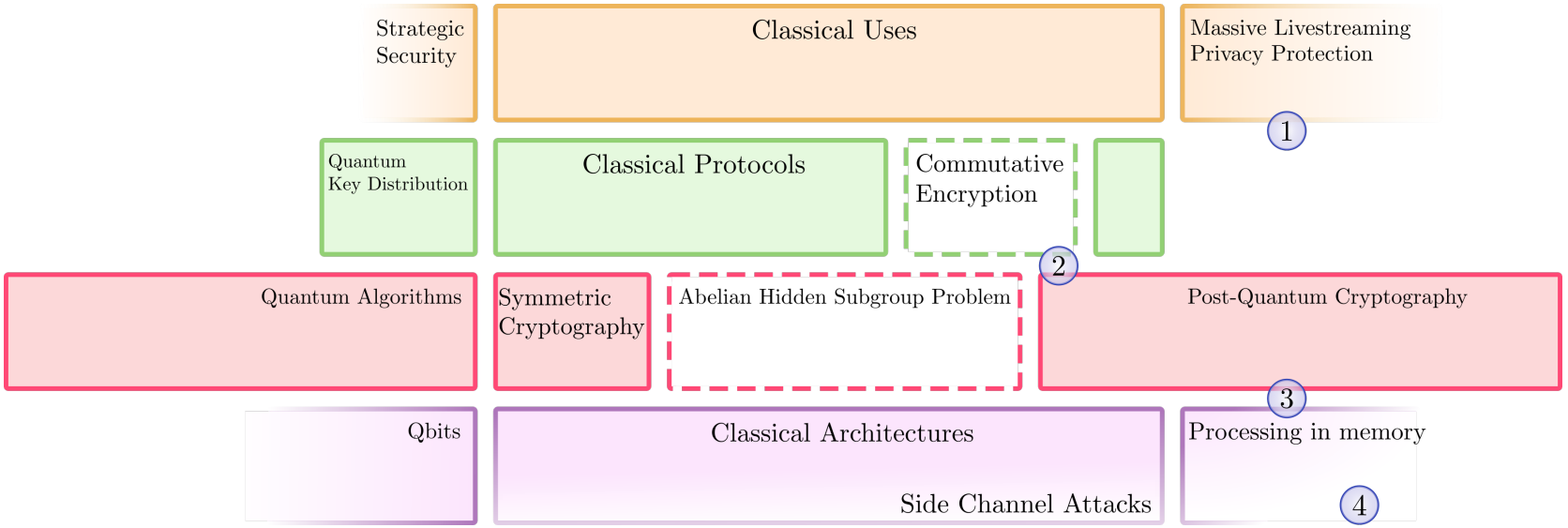


FIGURE 1.5 – Évolution des différentes couches de la protection des données.
1-4 : Contributions présentées dans cette thèse.
Pointillés : Sécurité compromise
Dégradé Horizontal : Domaine en cours d'expansion
Dégradé Vertical : Domaine menacé par des avancées scientifiques

1.3 Problématique

Les évolutions de chacune des couches se font de manière isolée, mais elles impactent nécessairement celles supérieures et inférieures. Pour minimiser les effets de ces changements, des outils sont proposés dans chacune des couches afin d'assurer que les interfaces avec les couches adjacentes restent autant que possible intactes.

Premièrement, les conséquences de l'arrivée éventuelle de la cryptanalyse quantique et le retrait des primitives basées sur le Abelian Hidden Subgroup Problem et les adaptations nécessaires à celles-ci sont amoindries par l'arrivée des primitives post-quantiques. Ce changement pourra être quasiment transparent pour la plupart des protocoles les plus simples (les appels à KeyGen, Encrypt, Decrypt, ne changeant pas) qui s'exécuteront comme prévu sur nos machines actuelles. Cependant, le changement n'est pas parfaitement transparent : les nouvelles primitives n'ont pas les mêmes performances, tailles de clés, tailles de chiffrés... Des réflexions dans les deux directions doivent être engagées : selon le protocole, le choix de la primitive à utiliser n'est plus évident, les primitives post-quantiques présentant de grandes différences de performance. En retour cela peut motiver des changements dans les protocoles, pour limiter le nombre d'échanges, la quantité de clés à stocker, sans compter que des fonctionnalités de protocoles spécifiques (ex : chiffrement commutatif), si on changeait uniquement de primitives, viendraient à disparaître.

Un autre moteur d'évolution des protocoles cryptographiques est bien sûr l'évolution de leur utilisation. Des protocoles proposant aux utilisateurs des meilleurs compromis entre vie privée, leur sécurité, et leur capacité à pouvoir échanger leurs données contre des services font évoluer la demande de ces utilisateurs. En retour, les protocoles doivent s'adapter à ces variations dans les demandes. Que privilégient les utilisateurs ? Un faible temps de connexion ? Une forte bande passante ? Il faut adapter les protocoles pour optimiser ces métriques et garantir au mieux la vie privée des utilisateurs, tout en considérant les contraintes légales qui pèsent sur de tels systèmes.

On peut constater aussi que les évolutions très fortes de primitives cryptographiques vont impacter et être impactées par la couche physique. D'un côté, certaines primitives peuvent très bien s'exécuter plus rapidement sur d'autres architectures, ainsi que de manière plus sécurisée contre les attaques physiques. De l'autre côté, les architectures s'adaptent pour intégrer des outils d'amélioration de ces nouvelles primitives : échantillonnage du bruit pour les systèmes LWE, ajout de nouvelles instructions... De plus en plus de recherches adoptent une vision d'ensemble de ces problématiques pour proposer des solutions globales, comme l'effet sur les performances d'adaptation de primitives les rendant plus résilientes aux attaques par canaux auxiliaires [Migliore 2019].

On peut observer une forte similarité entre ces précédentes remarques : le point commun est que l'observation de plusieurs couches permet une résolution de problèmes de manière plus poussée. La problématique qui émerge globalement en cryptographie dans cette période de préparation à l'arrivée des ordinateurs quantiques est donc le besoin de synergie transversale entre les différentes couches. Il ne s'agit plus seulement de s'assurer de l'existence d'empilements de blocs sécurisés pour garantir la protection des données pour des utilisations simples. Il s'agit aussi de

proactivement étudier les liens entre ces couches et les conséquences des évolutions attendues sur chacune des couches pour trouver des solutions meilleures aux besoins anticipés.

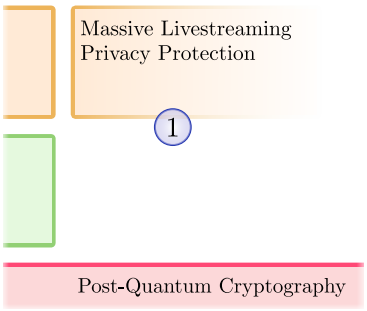
Pour illustrer ce besoin et proposer de telles solutions, nous avons ciblé des problématiques spécifiques aux interfaces entre différentes couches, et dont la résolution est plus poussée par une étude transversale que de manière segmentée. Nous traiterons celles-ci des couches les plus hautes aux plus basses. La numérotation est similaire à la figure 1.5.

1. La consommation numérique évolue parallèlement vers une quantité grandissante de livestreaming de masse, et vers des outils de protection des utilisateurs. Des protocoles savent satisfaire chaque besoin individuellement, mais peut-on concevoir une solution qui satisfasse les deux simultanément, tout en n'utilisant que des primitives post-quantiques ?
2. Les utilisations que permettaient le chiffrement commutatif sont-elles toutes perdues à cause de l'algorithme de Shor ? Il faut étudier quelles propriétés sont transférables en utilisant des primitives post-quantiques et lesquelles ne le sont pas, pour préserver le plus d'utilisations possibles. Les mécanismes d'interception réglementée, actuellement tous basés sur du chiffrement commutatif peuvent-ils être post-quantiques ?
3. L'effort de standardisation du NIST s'est concentré sur des architectures existantes. Les primitives sont donc conçues pour obtenir leurs meilleures performances sur celles-ci. Cependant, elles risquent d'être exécutées sur de nouvelles architectures au cours de leur durée de vie. Quelles sont les conséquences de ce changement sur les performances des algorithmes et la manière de les concevoir ?
4. Les fuites d'informations par canaux auxiliaires sont de plus en plus maîtrisées sur les architectures classiques. Peut-on transférer ces connaissances aux architectures de calcul en mémoire ? Quelles sont les répercussions sur les primitives ?

Les chapitres de cette thèse répondent à ces points de notre problématique. Chacun d'entre eux contiendra une présentation du contexte et un état de l'art qui lui sera plus spécifique.

Chapitre 2

QasTor - Protocole de Routage Anonymisé pour le Livestreaming Massif



Ce chapitre propose un protocole adapté aux besoins des utilisateurs modernes et basé sur des primitives post-quantiques.

Sommaire

Souvent la sécurité des personnes et leur vie privée sont présentées comme deux objectifs inconciliables. C'est d'autant plus vrai en sécurité informatique et en cryptographie. Dans un monde où la protection des données personnelles serait accessible à tous, il est en effet alors possible pour des personnes mal intentionnées de communiquer à l'abri de toute écoute. Le terrorisme, le cyber-crime, les trafics illégaux et autres actes immoraux sont ainsi de réelles menaces, mais sont parfois surtout utilisées de manière alarmiste, afin de faire pencher la balance du côté de la surveillance permanente.

Cette tendance se retrouve tout aussi bien chez nous en France, qu'au Royaume-Uni [Tue 2022], voire plus généralement dans l'Union Européenne [Cattafesta 2021, Fassinou 2021, Ruiz 2020]. Si le débat se retrouve régulièrement sur le devant de la scène médiatique, c'est surtout à cause de la situation aux États-Unis [Lovejoy 2020], où les débats ont été particulièrement houleux. Pourtant, même si tous ces pays ont le plus souvent une forte culture du respect des droits de l'Homme et de la défense fervente des libertés individuelles, ces derniers ne résistent pas à l'attrait de l'argument choc : « seuls les criminels ont besoin de chiffrement ». Et il est tellement aisé pour une personne honnête de venir confirmer « moi de toute façon je n'ai rien à cacher », en renonçant ainsi à sa liberté de vie privée pour montrer sa bonne foi. L'équilibre liberté/sécurité s'érode alors et glisse lentement et progressivement en faveur de la surveillance généralisée.

Cette tendance se développe paradoxalement en symbiose avec l'accessibilité et les performances grandissantes des services de messageries individuelles sécurisées. Les plus connus tels que Telegram ou Signal (application portable, détenue par un organisme non-lucratif et dont le code est open-source), connaissent un essor certain ces dernières années [Agrawal 2021]. L'histoire racontée par la progression en escalier de leur croissance [Gaudiaut 2021] est généralement la suivante (illustré en figure 2.1) :

— Une grande fuite d'informations personnelles est annoncée publiquement et prend beaucoup d'ampleur dans les médias. Les événements les plus vifs dans les mémoires sont encore ceux liés à Facebook - nouvellement Meta - à cause de leur impact : le scandale Cambridge Analytica [Untersinger 2018], couplé peu après avec l'annonce de la mise à jour des conditions d'utilisation de Whatsapp revoyant à la baisse la protection des données.

— Ces annonces ont un effet sur les utilisateurs car le nombre d'inscriptions sur des plateformes de communication plus respectueuses de la vie privée augmente dans les jours qui suivent.

— Le temps d'écran de ces utilisateurs ne se démultiplie pas infiniment, et les plateformes qui travaillent à la valorisation commerciale de données privées ont donc intérêt à limiter l'accès aux plateformes respectueuses de la vie privée.

— Cela donne alors des raisons d'invoquer l'argument du risque d'atteinte à la sécurité nationale, et à juste titre, comme le démontre l'utilisation de Signal dans la tentative de Coup du 6 Janvier au capitole de Washington DC [Bump 2022]. Il peut en émerger de nouvelles propositions de lois sécurité qui veulent limiter le chiffrement, ce qui peut être perçu en soi comme une mise en danger de la vie privée et redémarre le cycle.

— C'est alors la différence de « confort » qui peut enrayer le cycle, les plateformes sécurisées gardant peu leurs utilisateurs sur le long terme.

Ce qui ressort de ce fonctionnement est qu'une partie des utilisateurs ne fait pas confiance aux organismes d'État pour la gestion du compromis vie privée/sécurité. De toute évidence, il semble impossible de détecter les utilisateurs malveillants sans passer par le déchiffrement des messages d'utilisateurs légitimes au cours de la recherche. Pour cette raison, le public semble être pro-sécurité en tant que groupe, mais en composé d'individus pro-vie privée, dans la limite d'un confort d'utilisation.

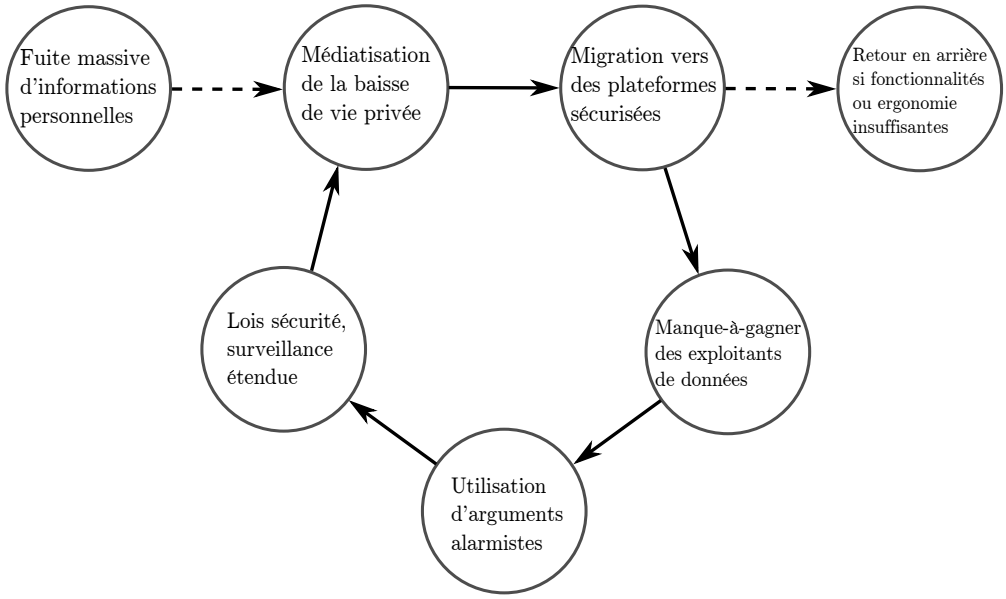


FIGURE 2.1 – Étapes du cycle des débats vie privée/sécurité. En pointillé les éléments qui peuvent renforcer ou affaiblir le cycle.

La gestion totale de la vie privée par l'État ne peut pas être envisagée, même si on suppose l'utilisation d'algorithme « oracle » de détection des utilisateurs malveillants qui ne se trompe jamais, et ne déchiffre aucun message d'utilisateur légitime. Il demeure dans ces conditions des raisons de vouloir se soustraire à ce contrôle des messages privés.

La plus importante est la possibilité d'assurer un contre-pouvoir, rôle censé être assumé par les médias, aussi appelés « quatrième pouvoir ». Cette nécessité est causée par le delta qui peut exister entre ce qui constitue l'action légale et l'action morale (problème très connu des bacheliers 2022 qui ont eu à disserter sur la question « Revient-il à l'État de décider de ce qui est juste ? »). Les dérives possibles de l'absence de ce contre-pouvoir se sont illustrées plusieurs fois dans l'Histoire et dans la fiction comme c'est le cas dans le roman de Georges Orwell « 1984 », ou l'État possède à la fois un ministère de la vérité décidant des limites de la pensée autorisée, et un outil de surveillance pour appliquer ces limites : Big Brother. Ce n'est qu'un exemple des divers questionnements éthiques sur la surveillance et ses contre-mesures [Macnish 2023]. Les outils de surveillance et de protection semblent pouvoir être vus comme des éléments éthiquement neutres, et c'est leur utilisation qui leur confère un caractère éthique ou non.

Afin de protéger les utilisateurs de cas de surveillance non éthiques, comme justement le travail journalistique du contrôle de l'État par ses constituants, ou la transmission de secrets commerciaux ou personnels, il convient d'avoir à disposition des utilisateurs des outils de discussion privée et sécurisée adaptés aux moyens de surveillance.

Cette capacité à transmettre des messages dont la lecture est protégée, bien que nécessaire, n'est pas le seul prérequis pour assurer le travail de ce contre-pouvoir.

Les métadonnées d'une communication peuvent souvent être tout autant signifiantes que leur contenu. La date, l'heure, la quantité de messages transmis, leur volume, souvent révélateurs du format des données, sont des éléments cruciaux qui sont à protéger, mais plus que tout, il faut protéger les identités des participants à la conversation. Cela est délicat à réaliser car les adresses IP de ceux-ci peuvent si on ne prend pas de précautions être reliées par déduction aux utilisateurs eux-même.

C'est dans cet objectif de protéger l'identité de leurs utilisateurs que des services d'anonymisation tels que des VPN ou TOR se mettent en place. Les VPN permettent de créer un « tunnel » vers un tiers de confiance qui se chargera de transmettre des informations potentiellement sensibles. Alors que l'identité de ce tiers de confiance est exposée, celle du client ne l'est pas. Le problème majeur de ce fonctionnement est qu'un noeud intermédiaire (le tiers de confiance) connaît à la fois l'identité de l'émetteur et celle du destinataire. Pour palier à ce problème, TOR propose un fonctionnement avec plusieurs noeuds intermédiaires.

Cependant, TOR est basé sur le cryptosystème RSA. Or, celui-ci est vulnérable face à un ordinateur quantique. De plus, il est probable que les premiers ordinateurs quantiques soient accessibles uniquement aux institutions avec le plus de moyens, à savoir les États (armées) et éventuellement des entreprises (banques ou hautes technologies). Face à un système de surveillance quantique, il est nécessaire d'adapter le protocole TOR. Une telle adaptation ne sera néanmoins pas facilement adoptée par les utilisateurs si celle-ci n'est pas adaptée aux utilisations qu'ils désirent.

Dans ce chapitre, nous présenterons le fonctionnement de TOR, rappellerons ses limites les plus connues et proposerons un nouveau protocole plus adapté aux nouveaux modes de communication.

2.1 Le protocole TOR Classique

Aujourd'hui, TOR est connu du grand public sous la forme d'un navigateur, codé en C, Python et Rust. Celui-ci permet à la fois l'anonymisation des communications (masquer le lien entre la source et la destination) et un contournement qui permet l'accès à des ressources qui peuvent être bloquées dans certains pays. Ces propriétés donnent au navigateur TOR à la fois une presse très négative en tant que plate-forme d'accès au dark web et aux activités illicites qu'il permet [Truelist 2022], mais aussi une presse positive en tant que parangon de la protection de la liberté de la presse et d'expression [Gallagher 2014].

Cependant, TOR n'est pas qu'un navigateur, c'est aussi un réseau. Afin de répondre aux demandes des 2 à 3 millions d'utilisateurs connectés à chaque instant, le réseau TOR contient en permanence environ 7 000 serveurs, ou relais. Ces derniers sont classifiés selon leurs performances (« Fast », « Stable ») et selon leur utilisation recommandée (« Guard », « Exit »). Le réseau propose une bande passante maximale d'environ 750 Gbits/s, dont seulement 250 sont réellement consommés. De ceux-ci, environ 2 Gbits/s sont utilisés uniquement pour maintenir à jour la liste des noeuds du réseau (annuaire) et distribuer celle-ci [Project 2022].

Ce réseau est construit autour d'un protocole central : celui dit du routage en oignon. Bien que toutes les informations soient amplement disponibles sur ce

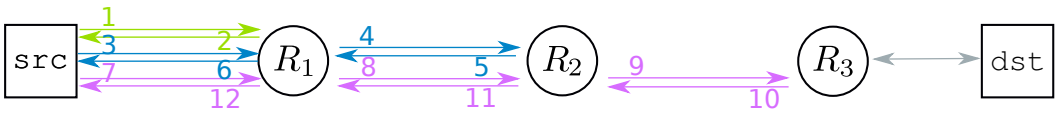


FIGURE 2.2 – Étapes de la distribution des clés dans le protocole TOR classique.

mécanisme [Dingledine 2004], nous rappelons ici les points principaux nécessaires pour la discussion.

— Un client `src` veut établir une communication avec un serveur `dst`, qui peut être hors du réseau TOR.

— Le client obtient un annuaire des relais du réseau auprès d'un des noeuds stables chargés de la diffusion de l'annuaire. Celui-ci contient les adresses des relais et leur classification.

— Dans cet annuaire, il choisit une liste de relais de taille n (souvent $n = 3$) dont le relai d'entrée R_1 est classifié « Guard » et le relai de sortie R_3 est classifié « Exit ».

— Il crée une connexion avec R_1 et grâce à un mécanisme d'échange de clés (KEM) basé sur le logarithme discret, il crée en un seul aller-retour de données une clé partagée notée K_1. Désormais, R_1 déchiffrera tous les messages provenant de `src` avec cette clé.

— Une fois des clés partagées entre `src` les i premiers relais, `src` envoie au relai R_i via tous les relais R_1, ..., R_{i-1} les instructions pour transmettre les informations nécessaires à un échange de clés avec R_{i+1}, chiffrées successivement avec toutes les clés K_i à K_1. On crée ainsi en un seul aller-retour de données une clé partagée notée $K_{\mathrm{i+1}}$. Désormais, R_{i+1} déchiffrera tous les messages provenant de R_i avec cette clé, et R_i chiffrera tous les messages provenant de R_{i+1} avec sa clé K_i.

— Une fois des clés partagées entre `src` et tous les relais choisis, `src` va indiquer à R_n à quel serveur `dst` il souhaite se connecter en chiffrant des instructions de connexion chiffrées successivement avec toutes les clés de K_n à K_1.

Ce processus est représenté par le schéma 2.2.

2.2 Les faiblesses et limites de TOR

Un article de synthèse des recherches [Saleh 2018] contient un aperçu assez global des sujets qui orbitent autour de TOR. Celles-ci contiennent tout autant de moyens de limiter l'utilité de TOR, que de voies d'amélioration permettant de compenser ces points faibles. Les points les plus sensibles stratégiquement sont :

— L'anonymat, en tant qu'argument premier de l'utilisation de TOR.

— La sécurité du réseau en tant que mesure de résistance à des comportements malveillants (relais corrompus, déni de service, flooding...)

— La qualité de service en tant que mesure de l'ergonomie d'utilisation, qu'on peut répartir en plusieurs métriques :
  — Temps de création de communication,
  — Ping entre `src` et `dst`,
  — Bande passante,
  — Impacts d'une déconnexion.
— La capacité du réseau à attirer de nouveaux utilisateurs. La protection offerte par TOR reste liée à son nombre d'utilisateurs et de relais. Plus ceux-ci sont nombreux, plus leur contribution individuelle est indiscernable et plus ils sont anonymisés.

2.2.1 Anonymat

Il existe un grand nombre de métriques d'anonymat envisageables et autant d'attaques visant à les diminuer. Celles-ci sont généralement probabilistes, comme par exemple les très intuitives métriques de *linkability* : dans un réseau de $n+2$ relais dont un `src` et un `dst` qui sont en communication, quelle est la probabilité moyenne/maximale que les n autres relais ont de deviner la paire (`src`, `dst`) parmi toutes les autres possibles ? Pour un bon anonymat, il faut que ces valeurs soient les plus basses possibles. On étend aussi généralement cette métrique à une collusion de k parmi n relais.

Pour illustrer, dans une communication protégée par un VPN, sans collusion, tous les noeuds sauf le relai ont une probabilité de $\frac{1}{n(n+1)}$ de trouver la bonne paire parmi les noeuds restants. En effet, il y a $n+2$ noeuds au total avec `src` et `dst`, et, hormis le noeud relai, les autres noeuds savent qu'ils ne sont ni `src` ni `dst` et que `src` et `dst` sont distincts. En revanche, le noeud relai à une probabilité de 1 de trouver correctement. Ainsi, la métrique « moyenne » peut être bonne (ici, $\frac{n-1}{n(n+1)} + \frac{1}{n}$), mais on désire que la métrique « max » le soit aussi.

Dans TOR avec trois relais, sans collusion, tous les noeuds sauf les relais ont une probabilité de $\frac{1}{n(n+1)}$ comme avant. Le R_3 connaît `dst`, il a donc une probabilité de $\frac{1}{n}$, tout comme le relai R_1 qui peut reconnaître `src` comme n'étant pas un relai connu [1]. En revanche, R_1 ne connaît pas `dst`, R_3 ne connaît pas `src` et R_2 n'a pas d'information. On a donc un compromis qui a fait légèrement augmenter la métrique « moyenne » $\left(\text{ici, } \frac{n-2}{n(n+1)} + \frac{2}{n}\right)$ mais qui a réduit la métrique « maximum ».

Aussi utile que cette métrique soit en tant que condition nécessaire à un bon réseau d'anonymisation, elle est loin d'être suffisante. C'est ce qui a été démontré depuis des années en Chine [Winter 2012], où c'est le simple fait d'être repéré par son fournisseur d'accès comme voulant se connecter à un réseau anonymisant qui permet de bloquer cet accès. En effet, « pas d'accès à un réseau anonymisant = pas d'anonymat ». Il a donc fallu inclure une métrique de la détectabilité d'une connexion à un sous-réseau anonymisant.

Cette précédente métrique a pour condition une autre, encore plus élémentaire : celle de la détectabilité du sous-réseau anonymisant lui-même. Si on ne distingue

1. Idéalement le protocole TOR fait en sorte que ce ne soit pas le cas, mais en pratique c'est souvent faisable. On prend donc l'hypothèse que le noeud d'entrée connaît `src`.

pas quels noeuds ou quel trafic appartiennent au sous-réseau, il est impossible de savoir si un client espionné se connecte à un noeud du sous-réseau ou un noeud extérieur. Mais malheureusement pour le réseau TOR, celui-ci abrite un trafic et des communications repérables, et est facilement différenciable du réseau général, comme démontré expérimentalement dans [Barker 2011] et [Houmansadr 2013] par exemple.

La désanonymisation peut aussi se faire de façon active : si des relais d'entrée ou de sortie sont compromis, ils peuvent adopter des comportements spécifiques qui, en restant invisibles aux yeux du client, permettent de faire ressortir le chemin d'une communication dans le réseau, et ainsi détériorer directement la métrique de linkability. Cela peut se faire en utilisant des ports inhabituels [Sulaiman 2013], ou en rejouant des paquets par exemple [Pries 2008].

2.2.2 Dénis de service

Réduire l'anonymat est le moyen le plus simple de dissuader des utilisateurs de rejoindre le réseau, en sapant sa principale raison d'être. Mais il est parfois plus simple de saboter la qualité de service jusqu'à ce que l'utilisation du réseau soit tellement peu pratique qu'elle soit abandonnée. C'est une dissuasion par la pénibilité.

Ces attaques ont malheureusement prouvé leur efficacité : [Evans 2009] et [Barbera 2013] sont toutes basées sur une exploitation maximale de mécanismes légitimes. L'attaque la plus simple et la moins coûteuse est encore de créer le plus de communications simultanées, chacune avec le chemin le plus long possible, et lancer sur chacune le plus grand téléchargement possible avant l'expiration des clés (environ 4 GB). Ces attaques peuvent être très efficaces et surtout très peu coûteuses, comme montré dans [Jansen 2019]. D'après les auteurs, il suffirait de 20k$ par mois pour une dégradation de plus de 30% du réseau.

L'accessibilité des attaques DDoS est inquiétante : on trouve même du « DDoS as a Service » facilement et peu coûteux sur Internet [Makrushin 2017]. Heureusement ces attaques sont souvent génériques et n'ont pas les mêmes « performances » de dégradation du réseau. On peut tout de même imaginer que ce genre d'attaque puisse être utilisé pour causer des perturbations temporaires, par exemple pour un système autoritaire qui cherche à échapper ponctuellement à la vigilance des médias et de l'opinion publique.

Les contre-mesures ne sont malheureusement pas encore à la hauteur : le mécanisme naturel à mettre en place est la gestion de confiance. Lorsqu'un noeud commence à avoir un comportement identifiable comme malveillant, les noeuds l'ayant détecté peuvent éviter la propagation de dégâts. [Evans 2009] propose par exemple des balises pour éviter que trop d'informations de ping soient transmises. Ces contre-mesures ont un effet limité car la gestion de confiance ne se diffuse pas entre les relais. Il n'y a pas d'accord collectif sur la confiance accordable à un relai.

2.2.3 Qualité de service

Une mauvaise qualité de service est peut être le facteur qui peut le plus augmenter la pénibilité d'utilisation de TOR. Une mauvaise QoS ne dissuadera pas un utilisateur « actif » cherchant activement de l'anonymat (journalistes, activistes, etc.) de se servir de TOR. Cependant, elle dissuadera un utilisateur « passif » qui aurait bien volontairement rejoint le réseau TOR pour plus de vie privée si son expérience de navigation avait été équivalente par ailleurs. Or, l'anonymat fournie par le réseau TOR est directement liée au nombre de noeuds qu'elle contient et à son nombre de clients. Ainsi, une mauvaise QoS fait fuir les utilisateurs « passifs », qui en quittant le réseau font perdre en anonymat aux utilisateurs actifs. Il est donc crucial pour tout le réseau de fournir la meilleure QoS possible.

La QoS se décompose en plusieurs métriques qui doivent être prises en compte :

— Le temps d'établissement d'une communication,

— Le ping une fois la communication établie,

— La bande passante disponible,

— La fréquence et les conséquences des déconnexions.

L'idée est, bien entendu, pour le réseau anonymisant de minimiser toutes les métriques à part la troisième qui est à maximiser. Un attaquant aurait alors pour objectif de faire l'opposé.

De tout le protocole, le choix impactant le plus toutes ces métriques est le nombre de relais. En augmentant n, on fait naturellement augmenter le nombre d'allers-retours d'échanges de clé et le ping, les déconnexions sont plus fréquentes, et la bande passante de bout en bout est celle minimale parmi tous les relais. A l'inverse, si on diminue trop n, on augmente la probabilité que des utilisateurs légitimes choisissent n noeuds malveillants et soient donc désanomymisés, d'où le compromis habituel de 3.

Pour améliorer le protocole, la méthode du choix des relais peut être modifiée. Plutôt que de choisir des relais aléatoirement, on peut les choisir selon la latence qu'il y a entre eux, ainsi que selon leur charge et leur capacité [Akhoondi 2012, Panchenko 2012]. Ce choix des relais prend déjà en compte les bandes passantes, depuis l'ajout de la balise « Fast » aux relais qui permet de garantir une bande passante minimale élevée en évitant de choisir un relai qui fasse goulot d'étranglement de la communication.

Les déconnexions sont moins étudiées que les autres métriques. Certains relais ont une balise « Stable » indiquant qu'ils ont peu de chances d'être déconnectés, mais les conséquences d'une déconnexion restent les mêmes : la nécessité de re-créer toutes les communications qui passaient par ce relai. Ce qui est autant coûteux que le mécanisme de création de communication. Le fait d'être ainsi virtuellement contraint à n'utiliser qu'une sous-partie des relais (établis par des personnes souhaitant aider activement le réseau) est encore un facteur de réduction de l'anonymat. Il est dommage de ne pas pouvoir utiliser les ressources des utilisateurs « passifs », qui certes seront plus fréquemment déconnectés, mais représentent une manne pour augmenter l'anonymat et les performances du réseau.

Souvent suite à des évènements de prise de conscience de l'importance de la vie privée, on observe un pic dans le nombre d'utilisateurs, mais ceux-ci ne semblent pas rester sur le réseau, certainement à cause d'une différence de « confort » avec leur navigation habituelle. En améliorant la QoS, ces utilisateurs seraient certainement plus longtemps membres du réseau, ce qui peut être fortement impactant car souvent les plateformes numériques sont sujettes à des effets boule de neige, comme c'était le cas lors du gain en popularité de Facebook. Il est classique que de disposer d'un grand nombre d'utilisateurs est un argument pour que de nouveaux utilisateurs rejoignent le réseau.

2.2.4 Évolution des utilisations

Comme évoqué à l'instant, il est dans l'intérêt de tous les utilisateurs « actifs » et « passifs » que ces derniers se sentent bien sur le réseau et qu'ils y restent. Pour cela, il est important de leur fournir sur le réseau anonymisant tout ce qu'ils auraient pu avoir sur le réseau classique, notamment des vidéos, du streaming et surtout du live-streaming. La preuve en est faite par les chiffres dans [streamfizz 2020].

Il y a deux chiffres à remarquer plus précisément : *« La vidéo en direct représentera 17% du trafic vidéo Internet d'ici 2022.[...] La Coupe du monde 2018 a culminé à 7,9 millions de téléspectateurs simultanés, ce qui en fait l'un des flux en direct les plus populaires de tous les temps. »*.

> Et si toutes ces personnes avaient voulu passer par TOR ? Ce n'est pas un exemple très réaliste, mais on peut en imaginer un plus plausible, en se concentrant sur une communauté plus proche du milieu informatique, et souvent plus sensible aux problématiques de vie privée. Je pense particulièrement au public de Twitch. Il s'agit de la première plateforme mondiale de live-streaming, le plus souvent connue pour les jeux vidéos. Le flux présenté en première page de Twitch atteint presque en permanence 10 000 visionnages simultanés et les tournois des jeux vidéos les plus connus rassemblent souvent plus de 100 000 voire un million de personnes.

Si autant d'utilisateurs étaient passés par TOR, le réseau aurait été encore plus encombré que par les attaques de déni de service présentées plus tôt tant le volume demandé est grand. La raison est simple, pour chaque communication, les ressources utilisées sont augmentées d'un facteur multiplicatif n. Le réseau TOR gère les diffusions « one-to-many » comme il gérerait autant de communications « one-to-one ».

Pour récapituler, si on souhaite améliorer TOR, il lui faut plus d'utilisateurs. Pour en avoir plus, il faut arrêter la fuite des utilisateurs « passifs ». Pour cela, il faut que la navigation anonyme soit aussi pratique que leur navigation habituelle. Il faut donc d'un côté améliorer les métriques de qualité de service, et de l'autre adapter le réseau aux nouvelles utilisations, notamment celle du live-streaming massif. Pour tout cela, on peut

— diminuer le temps d'établissement de la communication en réduisant le nombre d'allers-retours nécessaires,

— distribuer les charges dans le réseau,

— avoir un mécanisme pour diminuer les effets des déconnexions,

— avoir un mécanisme de gestion de confiance pour limiter les comportements malveillants,

— et surtout, diminuer le facteur multiplicatif des ressources.

2.2.5 Vulnérabilité aux algorithmes quantiques

L'établissement de la communication entre `src` et `dst` est entièrement basé sur des primitives connues pour être vulnérables face aux ordinateurs quantiques, à savoir RSA et Diffie Hellman. Le protocole TOR pourrait être assez facilement adapté en remplaçant ces mécanismes par des chiffrements et KEMs post-quantiques, sans changements majeurs hormis les tailles de clés et de chiffrés.

Cependant, il est indispensable dans la recherche d'un protocole de substitution à TOR d'inclure dans la conception l'intégration de primitives post-quantiques pour s'assurer de sa sécurité à long terme. De plus, il faut favoriser les primitives reconnues et standardisées. Celles-ci auront plus de chances d'être implémentées sur les machines des utilisateurs et même peut-être d'être accélérées matériellement, ce qui en accélérant le protocole permettra une meilleure expérience utilisateur. Or, ce point a déjà été établi comme étant crucial.

Nous proposons ci-après un nouveau protocole de distribution de clés qui complète les objectifs d'amélioration de la qualité de service, de résilience aux dénis de service, tout en étant en capacité à répondre à une forte demande de live-streaming simultané, et à résister à un ordinateur quantique.

2.3 Proposition de protocole

Ce protocole ayant pour but d'améliorer TOR, vise à l'adapter au streaming, en se servant de chiffrement par attributs et est adaptable à des primitives de cryptographie post-quantiques. Il a donc fait l'objet d'un dépôt de brevet sous le nom de « QasTor - post-Quantum Attribute-based Streaming Through Onion Routing » [Nugier 2022].

2.3.1 Vue d'ensemble

Il est plus aisé pour comprendre la construction de notre protocole de commencer par présenter ce en quoi il diffère ou est semblable au TOR classique.

— La circulation des données entre `src` et `dst` ne change pas. En montant, `src` va toujours chiffrer n fois, avec n clés symétriques, et chacun des n relais effectuera un seul déchiffrement. Le dernier relai transmettra les données en clair à `dst`, qui n'appartient pas au réseau. En descendant, on procède dans le sens inverse.

— En revanche, toute la méthode de distribution de ces clés symétriques change : tout d'abord, les relais ne sont plus des noeuds choisis individuellement. Ceux-

ci sont désormais répartis en groupes et choisis en tant que membres de leur groupe.

— Du chiffrement par attributs est utilisé pour que n'importe quel noeud d'un groupe puisse intervenir.

— Chaque groupe est muni d'un algorithme de distribution des charges de travail `load balancer`. Celui-ci peut représenter un fonctionnement centralisé ou distribué, qui peut aller du simple jeton circulant jusqu'à une blockchain.

— Comme dans TOR classique, il est important que les noeuds d'entrée dans le réseau soient balisés comme tels (« Guard »).

— Contrairement à TOR, comme il y a plusieurs noeuds dans un même groupe, un noeud ayant prévu de se déconnecter peut demander un transfert de sa charge de travail à d'autres noeuds du même groupe. Plus important encore, ce transfert peut aussi se faire après une mise hors ligne imprévue sans besoin de créer une nouvelle communication.

— Afin de réduire la charge totale et la bande passante du réseau, des flux identiques (provenant donc du même serveur `dst`) peuvent être mis en commun pour plusieurs clients `src` différents.

— Comme dans TOR, nous conservons la propriété que les connexions entrantes et sortantes d'un relai ne peuvent pas être liées par leur contenu. S'il y a très peu de trafic on peut éventuellement faire une association par timing, mais ce n'est plus faisable pour des noeuds fortement connectés.

Les différentes phases du protocole sont articulées autour de trois axes d'amélioration de TOR. Les algorithmes et variables impliquées pour assurer ces fonctionnalités sont détaillés dans la section suivante.

1. *Un seul aller-retour de données vers le serveur pour établir une communication.* Le fonctionnement n'est plus itéré par rebonds vers le client (allers-retours « ping-pong »). Il n'y a pas besoin d'une interaction entre chaque relai et `src` avant de passer au relai suivant. Pour cela, quand il veut démarrer une connexion, `src` choisit n groupes et envoie un paquet au `load balancer` du premier groupe contenant toutes les informations nécessaires à distribuer les clés pour la mise en place d'une nouvelle connexion. Le `load balancer` choisit un relai de son groupe et lui transmet. Ce relai envoie au `load balancer` du groupe suivant. Le dernier relai se connecte à `dst`. Pour finir les échanges de clés, des informations redescendent de relai en relai vers `src` sans passer par les `load balancer` de leur groupe respectif. Des validations leur sont cependant aussi envoyées. Un routage en oignon classique est déjà possible à ce point.
2. *Mise en commun de bande passante.* Afin de profiter d'un effet de factorisation de la bande passante et du calcul, lorsque l'un des relais reconnaît une trace du flux, il pourra se servir des données reçues sur le chemin déjà existant. Cependant, pour pouvoir donner les clés au nouveau client, il faut des phases supplémentaires afin de s'assurer de ne pas les partager avec des clients non légitimes. Des défis sont donc aussi envoyés lors de la phase descendante. Le client doit faire remonter aux relais les réponses à ces défis, tout en s'assurant

aussi de son côté de l'intégrité des relais en leur donnant eux aussi des défis. Lorsque les défis sont validés, les clés redescendent vers `src` qui est donc ainsi connecté à un flux déjà existant. La figure 2.3 représente tous les paquets échangés pour ces deux premières spécifications selon leur type. Le détail du contenu des paquets est présenté dans la prochaine section.

3. *Rétablissement de communication.* Le protocole permet aux noeuds d'un groupe de prendre en charge les communications qui devaient être gérées par un autre membre du groupe qui va se déconnecter, ou qui vient de se déconnecter. Pour cela, une déconnexion est détectée par un relai en aval, celui-ci envoie une demande de reconstruction de la communication au `load balancer` du groupe du relai déconnecté, qui choisira un nouveau relai responsable et celui-ci devra s'identifier auprès des relais en amont et en aval dans la communication. Ce mécanisme est présenté par la figure 2.4 un peu plus loin.

> La présentation détaillée du protocole et des algorithmes qui le composent peut être difficile à suivre, notamment à cause de la quantité de variables impliquées. Nous proposons pour faciliter la lecture de regarder en parallèle la présentation disponible ici : [*cloud.laas.fr*]. Celle-ci trace toutes les variables impliquées dans le protocole sur un exemple d'utilisation de chaque fonctionnalité présentée juste au dessus. Les algorithmes du protocole détaillé sont présentés dans leur ordre d'utilisation dans cette présentation. La page de la présentation correspondant à leur première utilisation est indiquée, par exemple avec (p. 1).

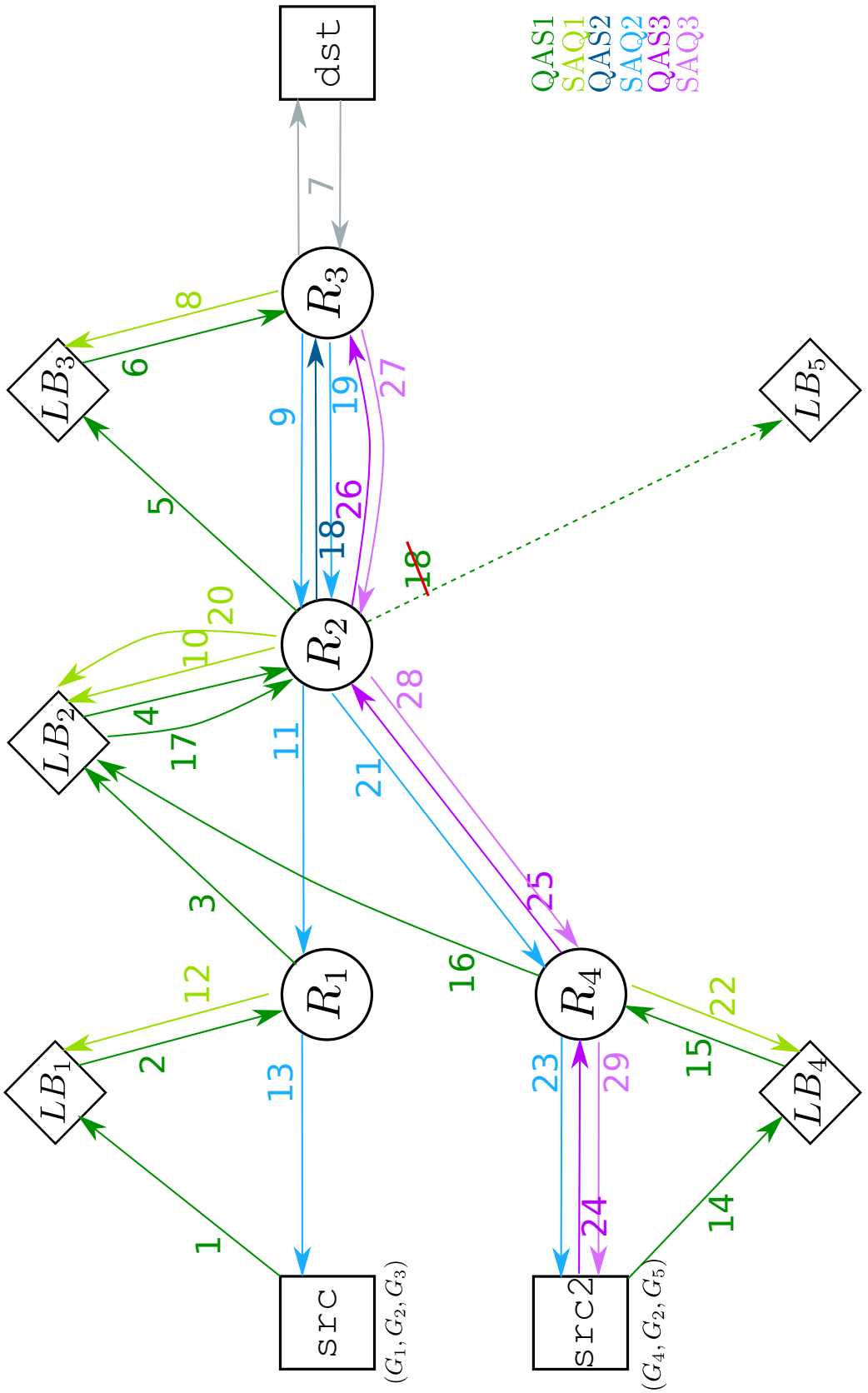


FIGURE 2.3 – Paquets échangés lors de la construction de deux communications vers le même serveur.

2.3.2 Protocole détaillé

2.3.2.1 Éléments publics

Trois primitives de chiffrement sont choisies et rendues publiques. Le choix de ces primitives sera discuté après la présentation du protocole.

— Un chiffrement symétrique à flot (ex : AES-GCM, Chacha20-poly1305...) dont la notation utilise des crochets fins : $[\mathrm{m}]_{\mathrm{k}}$

— Un chiffrement asymétrique (ex : RSA, CRYSTALS-KYBER...) dont la notation utilise des crochets épais : $\lfloor \mathrm{m} \rceil_{\mathrm{k}}$

— Un chiffrement par attributs KP-ABE (ex : OpenAbe [Zeutro 2018], R-LWE ABE [Liu 2022]...) dont la notation utilise des crochets doubles : $[\![\mathrm{m}]\!]_{\mathrm{att}}$

Une méthode de hachage est aussi choisie et rendue publique $h(x)$ dont le condensat est de la taille des clés du chiffrement symétrique choisi.

Un annuaire des groupes est établi. Pour chacun des groupes, il y est précisé quel est son `load balancer` (LB), ainsi que s'il est susceptible d'être utilisé comme premier groupe relai grâce à une balise « Guard », notée par $\sim$. Les attributs associés à ces groupes sont publics ou calculables à partir des informations publiques. Le `load balancer` est chargé d'avoir des secrets partagés (notés rt) avec les membres de son groupe.

Pour garantir la sécurité cryptographique, les clés doivent êtres désignées aléatoirement de manière uniforme. Nous noterons $x \in \mathcal{U}$, une variable x déterminée ainsi. Pour chaque format de paquet utilisé, pour chaque variable les composant, une taille fixe est choisie et rendue publique.

Afin de garantir l'uniformité des formats au cours du protocole, nous utiliserons du padding. Nous noterons $Pad(x)$ une quantité de padding généré aléatoirement en suivant la distribution uniforme parmi les chaînes de bits de même longueur que la variable x. Grâce à la propriété IND-CPA qui est un pré-requis du choix des chiffrements, les chiffrés et les paddings sont indistinguables, ainsi que les clés si elles sont piochées de façon indistinguable de l'uniforme.

Un nombre de relais intermédiaires n est fixé. Par défaut, nous proposons de garder le même que pour TOR classique : $n = 3$. Seuls les groupes « Guard » (notés $\tilde{G}$) savent que leur distance au client peut être 1. Seuls les noeuds en connexion avec un serveur savent que leur distance au client est n. Pour que cela reste vrai pour toutes les valeurs de n, le protocole doit faire en sorte que les algorithmes et variables utilisés par les différents noeuds soient indépendants de leur distance à `src` afin de ne pas leur donner d'information sur celle-ci. Pour cette raison, nous aurons recours à des fonctions utilitaires :

$Lshift$(elem, list) : (list(2), list(3), $\cdots$, list(n), elem)

$LshiftPad$(list) : (list(2), list(3), $\cdots$, list(n), Pad(list(1))

Par la suite, nous allons présenter chaque algorithme du protocole, dans un ordre cohérent avec le scénario représenté dans la figure 2.3, et discuter des choix de conception. De nombreux types de variables ayant à être manipulés, nous leur avons associé une couleur.

2.3.2.2 Client : initialisation de la communication

Cet algorithme est appelé Qas1c et s'exécute spontanément à l'initiative du client `src` lorsque celui-ci désire communiquer avec `dst` ou s'abonner à un de ses flux (p. 3). On utilisera la notation `dst` dans les deux cas pour plus de clarté.

Le client choisit n groupes parmi LG et établit une liste $\mathcal{G} = (\tilde{G}_1, G_2, \cdots, G_n)$, puis récupère ou calcule leurs attributs respectifs $(A_1, A_2, \cdots, A_n)$. Sauf dans le cas où aucun de ces groupes ne reconnaît le flux demandé vers `dst`, ces groupes ne seront pas tous ceux via lesquels la communication sera finalement établie. On notera la composition finale des groupes $\widehat{\mathcal{G}} = (\widehat{G}_1, \widehat{G}_2, \cdots, \widehat{G}_n)$. On considérera aussi que $\texttt{src} = R_0 \in G_0$.

Pour chaque niveau de profondeur i, le client désigne une clé « locale » $rl_i \in \mathcal{U}$ qui permettra les échanges entre les relais désignés des groupes G_{i-1} et G_i. Il génère une paire de clés publique/secrète (PK_i,SK_i) à partir de $seed \in \mathcal{U}$ qui lui permet d'anticiper des échanges avec des groupes n'appartenant pas à $\mathcal{G}$, ainsi qu'une valeur $D_i \in \mathcal{U}$ qui permettra de créer un secret partagé $K_i \in \mathcal{U}$.

Pour que le client puisse créer plusieurs communications en simultané sans risquer de confusion entre les retours de plusieurs relais, il lui faut attribuer à chacune de celles-ci un identifiant. Il choisit donc $S_0 \in \mathcal{U}$ pour celle-ci.

Il construit enfin un paquet QAS1 et l'envoie à LB_1. Ce paquet est composé des blocs $(B_1, B_2, \cdots, B_n, B_{n+1})$, dont chacun est construit de la manière suivante :

$B_{i<n} = [\![\cdots [\![[\![\texttt{step}, h(\texttt{dst}, n-i), LB_{i+1}, rl_i, rl_{i+1}\ ]\!]_{A_i}]\!]_{A_{i-1}}\ ]\!] \cdots]\!]_{A_1}$

$B_n = [\![[\![\cdots [\![[\![\texttt{end}, h(\texttt{dst}, 0), \texttt{dst}, rl_n, Pad(rl)\ ]\!]_{A_n}]\!]_{A_{n-1}}]\!] \cdots]\!]_{A_1}$

$B_{n+1} = [\ R_0, S_0, D_{list}, PK_{list}]_{rl_1}$, avec :

$D_{list} = (D_1, \cdots, D_n)$

$PK_{list} = (PK_1, \cdots, PK_n)$

Note : pour de nombreux systèmes ABE, les chiffrements et déchiffrements qui succèdent au premier sont fortement plus rapides que celui-ci. Le padding de B_n est présent car il n'y a pas de destinataire suivant.

2.3.2.3 Load Balancer : initialisation de la communication

Cet algorithme, appelé QAS1lb, est exécuté lorsqu'un `load balancer` LB_i reçoit un paquet QAS1 d'un relai R_{i-1} (p. 7).

Ayant une clé secrète ABE valide pour le dernier chiffrement, LB_i peut alors déchiffrer correctement les blocs $B_{1 \leq i \leq n}$ pour retrouver dans le premier bloc (`cmd`, $h(\texttt{dst}, n-i)$, LB_{i+1}, rl_i, rl_{i+1}). Ayant obtenu ainsi rl_i, il peut alors déchiffrer le bloc B_{n+1} et retrouver ainsi les valeurs (R_{i-1}, S_{i-1}, D_{list}, PK_{list}).

Un relai R_i du groupe est désigné, selon les charges et capacités des membres et éventuellement selon $h(\texttt{dst}, n-i)$ pour être responsable de cette communication. Par exemple, si un relai gère déjà une communication avec l'empreinte $h(\texttt{dst}, n-i)$, il est idéal de lui confier cette nouvelle communication, mais ce n'est pas requis par le protocole. Si le relai qui gère déjà la communication est proche d'une surcharge, il vaudra mieux en choisir un autre. Le relai désigné pour la nouvelle communication

possède un secret en commun avec le `load balancer`, noté rt_i et devra plus tard confirmer la prise en charge de la connexion. Afin d'éviter la confusion en cas de plusieurs créations de communications simultanées, une connexion vers ce relai C_{lb_i} identifie uniquement cette tentative de création de communication.

Le bloc B'_{n+1} est construit de la manière suivante : $[R_{i-1}, S_{i-1}, D_{list}, PK_{list}]_{rt_i}$

Un nouveau paquet QAS1 est constitué et envoyé à R_i via C_{lb_i}. Il est composé de $[B_1, \cdots B_{n+1}]_{rt_i}$.

2.3.2.4 Relai : Initialisation de la communication

Cet algorithme, appelé Qas1r, est exécuté lorsqu'un relai R_i reçoit un paquet QAS1 du `load balancer` LB_i via une connexion C_{lb_i} (p. 12).

Possédant une clé secrète rt_i, R_i peut alors déchiffrer le paquet. Le premier bloc B_i est désormais en clair, ce qui permet de retrouver (`cmd`, $h(\texttt{dst}, n-i)$, LB_{i+1}, rl_i, rl_{i+1}). Il peut accéder aussi au bloc B_{n+1} et retrouver ainsi (R_{i-1}, S_{i-1}, D_{list}, PK_{list}), et notamment PK_i, le premier élément de PK_{list}. Il n'est pas grave que les autres clés publiques lui soient accessibles grâce à la sécurité du schéma asymétrique choisi.

Il désigne une valeur $I_i \in \mathcal{U}$ et définit $K_{R_{i-1},h(\texttt{dst},n-i)} = h(I_i, D_i)$. Ce sera la clé du routage par oignon pour les communications entre R_i et les clients vers `dst` à distance $n-i$ qui transitent par R_{i-1}. Afin d'éviter la confusion en cas de plusieurs créations de communication simultanées, un token temporaire $S_i \in \mathcal{U}$ est associé à celle-ci.

Si `cmd` est `step` :

Si, de plus, l'empreinte est connue :

Si, de plus, l'empreinte identifie une communication non établie :

Il existe alors déjà une communication en cours d'établissement. La nouvelle demande qui vient d'arriver est alors mise en attente jusqu'à ce que la communication soit établie ou échoue à se créer. Si elle est établie, la prochaine section est exécutée, sinon, elle a échoué à s'établir et il faudra considérer l'empreinte comme nouvelle.

Sinon, l'empreinte identifie une communication déjà établie :

Cette communication passe par le relai R_{i+1} d'un autre groupe $\widehat{G}_{i+1}$ via une connexion C_{i+1}, qui utilise une clé locale $\widehat{rl}_{i+1}$. Le relai construit alors un paquet QAS2 pour l'envoyer via la connexion C_{i+1}. Il est construit de la façon suivante :

$[S_i, LshiftPad(D_{list}), LshiftPad(PK_{list})]_{\widehat{rl}_{i+1}}$

Si au contraire, c'est une nouvelle empreinte :

On considère donc désormais cette empreinte connue. Le relai crée un paquet QAS1 pour l'envoyer à LB_{i+1}. Il est construit avec :

$B_{i \leq n} = LshiftPad(B_{i \leq n})$

$B_{n+1} = [R_i, S_i, LshiftPad(D_{list}), LshiftPad(PK_{list})]_{rl_{i+1}}$

Si, au contraire, `cmd` n'est pas `step`, c'est donc `end`,

Le relai connaît alors `dst`. Il tente de se connecter à `dst` s'il ne l'est pas déjà. Il essaye aussi d'établir une connexion C_{n-1} avec R_{n-1}. S'il échoue à établir l'une des deux, il notifie son `load balancer` via la connexion C_{lb_n}. Il lui envoie un paquet SAQ1 composé ainsi : [`fail`, $Pad(S_{n-1})$, $Pad(T_{n-1})$, $Pad(T_n)$, `dst`, $Pad(K_{R_{n-1},h(\texttt{dst},0)})$ $]_{rt_n}$.

S'il y arrive, il va confirmer qu'il gère la nouvelle communication, en notifiant son `load balancer` via la connexion C_{lb_n}. Il lui envoie un paquet SAQ1 composé ainsi : [`confirm`, S_{n-1}, T_{n-1}, $Pad(T_n)$, `dst`, $K_{R_{n-1},h(\texttt{dst},0)}]_{rt_n}$. Ces informations sont nécessaires au `load balancer` pour désigner un successeur à R_n si celui-ci venait à être déconnecté.

Comme tous les identifiants de connexion S vont paraître en clair dans la phase descendante pour que les relais puissent savoir à quel création de communication associer ces retours, on ne pourra pas les réutiliser ensuite. Le relai génère donc un nouvel identifiant T_n qui permettra au remplaçant de R_{n-1} de se faire connaître si ce dernier se déconnectait.

Il crée ensuite un paquet SAQ2 et l'envoie à R_{n-1} via C_{n-1}. Le paquet est construit de la façon suivante :

$(S_{n-1}, [T_{n-1}, (n-1) \times Pad(B), B_n]_{rl_n})$

$B_n = [\texttt{new}, Pad(S), Pad(D), I_n]_{PK_n}$

2.3.2.5 Load Balancer : acquittement de la prise en charge d'une communication par un relai

Cet algorithme, appelé Saq1lb, est exécuté lorsqu'un `load balancer` LB_i reçoit un paquet SAQ1 d'un relai R_i de son groupe via une connexion C_{lb_i} (p. 30). Cela lui permet de retrouver la clé rt_i et déchiffrer les informations correspondantes.

Le paquet est composé ainsi : [balise, S_{i-1}, T_{i-1}, T_i, R_i, $K_{R_{i-1},h(\texttt{dst},n-i)}]_{rt_i}$. Si la balise est bien `confirm`, il enregistre ces nouvelles valeurs (qui permettront de rétablir la communication si le relai est déconnecté). Si en revanche elle ne parvient pas ou est `fail`, il peut tenter de relancer la phase montante (Qas1lb) ou la reconnexion (RepairLb) en choisissant un nouveau relai du groupe, ou d'abandonner cette communication. Suite à une reconnexion réussie, la balise `update` signalera de mettre à jour des informations déjà existantes.

2.3.2.6 Relai : phase descendante d'une création de communication

Cet algorithme, appelé Saq2r, est exécuté lorsqu'un relai R_i reçoit un paquet SAQ2 d'un relai R_{i+1} (qui peut être jusque là inconnu) via une connexion C_{i+1} (p. 32). Le paquet est composé de la façon suivante : $(S_i, [T_i, B_{list}]_{rl_{i+1}})$. Grâce à S_i, le relai retrouve la clé rl_{i+1} dont il se sert pour retrouver T_i et B_{list}. Il génère un token T_{i-1} qui permettra d'identifier le remplaçant de R_{n-1} si celui-ci se déconnecte.

Si S_i n'est pas déjà associée à une connexion C_{i-1},

La connaissance du relai R_{i-1} obtenue lors de l'algorithme Qas1r lui permet d'essayer d'établir une connexion C_{i-1}. S'il réussit il envoie le paquet SAQ2 suivant :

$B_i = [\texttt{new}, Pad(S), Pad(D), I_i]_{PK_i}$

$(S_{i-1}, [T_{i-1}\ Lshift(B_i, B_{list})]_{rl_i})$

Sinon, S_i était déjà associée à une connexion C_{i-1},

Il est possible que R_{i-1} ait eu connaissance de h(`dst`, $n-i$) s'il est lui-même un relai à cette profondeur et qu'il ait décidé de feindre le rôle de client pour réclamer des clés de déchiffrement du flux. Il convient alors de ne pas lui confier les vraies clés de communication et d'en utiliser une temporaire J_i jusqu'à ce que `src` ait prouvé qu'il connaît bien `dst`, ce qu'il ne peut faire que s'il est légitime. Afin d'éviter la confusion entre les validations de plusieurs clients, le relai choisit un token $S'_i \in \mathcal{U}$ pour cette validation.

Le bloc B_i, désormais en première position dans la liste, est déchiffrable grâce à SK'_i créée précédemment pour retrouver $h(D'_i,\texttt{dst})$. Le relai envoie à R_{i-1} via la connexion C_{i-1} le paquet SAQ2 suivant :

$B_i = [\texttt{old}, S'_i, D'_i, J_i\ ]_{PK_i}$

$(S_{i-1}, [T_{i-1}, Lshift(B_i, B_{list})]_{rl_i})$

Dans les deux cas, s'il a réussi, le relai va pouvoir confirmer qu'il gère la nouvelle communication, en notifiant son `load balancer` via la connexion C_{lb_i}. Il lui envoie un paquet SAQ1 composé ainsi : $[\texttt{confirm}, S_{i-1}, T_{i-1}, T_i, R_{i+1}, K_{R_{i-1},h(\texttt{dst},n-i)}]_{rt_i}$. Ces informations sont nécessaires au `load balancer` pour désigner un successeur à R_i si celui-ci venait à être déconnecté. S'il a échoué, la balise est remplacée par `fail`.

2.3.2.7 Client : réception du paquet retour

Cet algorithme, appelé Saq2c, est exécuté lorsqu'un client `src` reçoit un paquet SAQ2 d'un relai (jusque là inconnu) R_1 via une connexion C_1 (p. 41).

Le paquet est composé de la façon suivante : $(S_0, [T_0,B_{list}]_{rl_1})$. Grâce à S_0, le client retrouve la clé rl_1 dont il se sert pour retrouver B_{list}. Connaissant toutes les clés secrètes SK_i ; il peut déchiffrer chacun des blocs avec la clé correspondante (ordre inverse des groupes).

Pour chaque B_i, si la balise est `new`, il définit $K_i = K_{R_{i-1},h(\texttt{dst},n-i)} = h(I_i,D_i)$.

Si toutes les balises sont `new`,

Le client est désormais capable d'échanger des données avec `dst` et peut exécuter l'algorithme DATAc.

Sinon, au moins une des balises est `old`,

Il doit prouver à des relais déjà responsables de la gestion d'une communication avec `dst` que lui aussi connaît bien son identité. Par sécurité, il peut vérifier qu'il n'y a plus que des balises `old` après la première qu'il trouve, notons son index j. Il vérifie aussi que la valeur de retour des défis est correcte. Pour identifier les valeurs de retour, il crée des identifiants $S'_{i\neq n} \in \mathcal{U}$ et $S'_n = S_n$.

Il va envoyer un paquet QAS3 via la connexion C_1. Ce paquet est composé de (S'_1, $B_1, \cdots, B_n$), dont les blocs sont construits de la manière suivante :
$B_{1 \leq i < j} = [\ \cdots\ [\ [\ S'_{i+1}, H_i\ ]_{K_i}\ ] \ldots\ ]_{K_1}$
$B_{j \leq i < n} = [\ \cdots\ [\ [\ [\ \cdots\ [\ [\ S'_{i+1}, H_i\ ]_{J_{i-1}}] \ldots\ ]_{J_j}\ ]_{K_{j-1}}\ ] \ldots\ ]_{K_1}$
$B_n = [\ \cdots\ [\ [\ [\ \cdots\ [\ [\ Pad(S'_n), H_n\ ]_{J_n}] \ldots\ ]_{J_j}\ ]_{K_{j-1}}\ ] \ldots\ ]_{K_1}$
Avec $H_i = h(\texttt{dst}, D'_i)$ si $i \geq j$ (balise `old`), ou $H = Pad(h(\texttt{dst}, D'_i))$ sinon.

2.3.2.8 Client : mode DATA

Cet algorithme est appelé DATAc et est exécuté lorsqu'un client a toutes les clés d'un chemin vers un serveur `dst`(p. 44).

S'il veut envoyer un message à `dst`,

Le client envoie à R_1 via la connexion C_1 le paquet DATA suivant :

$[[\cdots[\texttt{payload}]_{K_n}] \ldots]_{K_1}$

S'il reçoit un paquet DATA de R_1 via la connexion C_1,

Il déchiffre le paquet avec toutes les clés de K_1 à K_n pour retrouver `payload`.

2.3.2.9 Relai : mode DATA

Cet algorithme est appelé DATAr et est exécuté lorsqu'un relai R_i reçoit un paquet DATA (p. 45).

Si le paquet vient d'un relai en aval R_{i-1},

Il le déchiffre avec la clé associée $K_{R_{i-1}, h(\texttt{dst}, n-i)}$ et envoie le résultat au relai associé R_{i+1} via C_{i+1}.

Si le paquet vient d'un relai en amont R_{i+1} ou d'un serveur `dst`,

Il peut y avoir plusieurs relais et clés associés. Pour chaque association, il chiffre le paquet entrant avec $K_{R_{i-1}, h(\texttt{dst}, n-i)}$ et l'envoie à R_{i-1} via la connexion C_{i-1}.

2.3.2.10 Relai : phase montante d'un ajout à une diffusion existante

Cet algorithme, appelé Qas2r, est exécuté lorsqu'un relai R_i reçoit un paquet QAS2 d'un relai R_{i-1} via une connexion déjà établie C_{i-1} (p. 80).

Le relai peut ainsi retrouver la clé correspondante rl_i et déchiffrer le paquet reçu, qui est de la forme $[S_{i-1}, D_{list}, PK_{list}]_{rl_i}$. Un token temporaire $S_i \in \mathcal{U}$ est associé à cette création de communication. Il crée aussi une paire de clés asymétriques (SK'_i, PK'_i) pour protéger le retour de la valeur de comparaison du défi qui sera fournie par R_n.

Si le relai n'est pas le dernier

Il ne connaît pas `dst` mais il connaît R_{i+1}, avec lequel il partage une connexion C_{i+1}, et une clé locale rl_{i+1}. Il crée un défi $D'_i \in \mathcal{U}$ qui sera envoyé à `src` pour qu'il puisse prouver qu'il connaît bien `dst` et pas uniquement une empreinte. Il crée une paire de clés asymétriques (SK'_i, PK'_i) pour protéger le retour de la valeur de comparaison du défi qui sera fournie par R_n. Il lui envoie donc le paquet QAS2 suivant :

$[S_i, Lshift(D'_i,D_{list}), Lshift(PK'_i,PK_{list})]_{rl_{i+1}}$

Sinon, il est le dernier

Il connaît donc `dst`. Il génère un défi $D'_n \in \mathcal{U}$, qui sera envoyé à `src` pour qu'il puisse prouver qu'il connaît bien `dst` et pas uniquement une empreinte. Il répond aussi aux défis de D_{list} en calculant $h(D_i,\texttt{dst})$ ou $h(D'_i,\texttt{dst})$, bien qu'il ne puisse pas faire la différence entre les deux cas. Il définit $J_n = h(D_n,\texttt{dst})$. Il génère un token T_{n-1} qui permettra d'identifier le remplaçant de R_{n-1} si celui-ci se déconnecte. De plus, il faudra identifier les retours des défis pour éviter une confusion, le relai crée un token S_n.

Il envoie à R_{n-1} via C_{n-1} le paquet SAQ2 suivant :

$(S_{n-1},[T_n, B_{n-1}, \cdots, B_1, B_n]_{rl_n})$, avec :

$B_n = [\texttt{old}, S_n, D'_n, h(D_n,\texttt{dst})]_{PK_n}$

$B_{i<n} = [Pad(\texttt{old}), Pad(S), Pad(D), h(D'_i,\texttt{dst})]_{PK'_i}$

2.3.2.11 Relai : validation des défis phase montante

Cet algorithme, appelé Qas3r, est démarré lorsqu'un relai R_i reçoit un paquet QAS3 d'un relai R_{i-1} via une connexion C_{i-1} (p. 97). Ce paquet contient S'_i, et B_{list}. S'_i en clair permet de savoir quelle clé utiliser. Le relai déchiffre avec cette clé tous les blocs.

S'il s'agit d'une clé temporaire (J_i), alors un défi D'_i et une valeur de comparaison $h(\texttt{dst},D'_i)$ fournie par R_n y sont associés. Il vérifie que la seconde valeur du premier bloc de la liste (B_i) est bien égale à cette valeur de comparaison. Il ne poursuit que si c'est le cas. Il n'y a pas cette vérification dans le cas d'une clé permanente.

S'il n'est pas le dernier relai,

Il y a donc un relai suivant R_{i+1} auquel il envoie via la connexion C_{i+1} le paquet QAS3 suivant : $(S'_{i+1}, LshiftPad(B_{list}))$.

Sinon, il est le dernier relai et connaît `dst` ,

Le relai (qui est donc R_n) va envoyer au relai R_{n-1} via la connexion C_{n-1} le paquet SAQ3 suivant :

$([S'_n]_{rl_{n-1}}, (n-i) \times Pad(B_n), B_n)$

Avec $B_n = [K_{R_{i-1},h(\texttt{dst},n-i)}]_{J_n}$

2.3.2.12 Relai : validation des défis phase descendante

Cet algorithme, appelé Saq3r, est exécuté lorsqu'un relai R_i reçoit un paquet SAQ3 d'un relai R_{i+1} via une connexion C_{i+1} (p. 106). Ceux-ci sont associés à un autre relai R_{i-1} avec une connexion C_{i-1} et une clé locale rl_{i+1}. Le paquet déchiffré avec celle-ci contient l'identifiant S'_i, et B_{list}.

Si S'_i, identifie un ajout à une communication existante,

Le relai effectue $B_{list} = Lshift(K_{R_{i-1},h(\texttt{dst},n-i)}, B_{list})$, puis chiffre tous les éléments de B_{list} avec J_i.

Sinon S'_i, identifie une nouvelle communication,

Le relai effectue $B_{list} = LshiftPad(B_{list})$, puis chiffre tous les éléments de B_{list} avec $K_{R_{i-1},h(\texttt{dst},n-i)}$.

Ensuite, il envoie à R_{i-1} le paquet SAQ3 suivant :

$([S'_i]_{rl_{i-1}}, B_{list})$

2.3.2.13 Client : validation des défis phase descendante

Cet algorithme, appelé Saq3c, est exécuté lorsqu'un client `src` reçoit un paquet SAQ3 d'un relai R_1 via une connexion C_1 (p. 110). Le paquet déchiffré avec rl_1 contient B_{list}. Le client inverse la liste et déchiffre grâce aux SK_i correspondantes. Il trouve toutes les clés nécessaires K_1, ..., K_n pour exécuter l'algorithme DATAc.

2.3.2.14 Relai et Client : détection de déconnexion

Cet algorithme nommé Deco est exécuté par un client ou un relai lorsque le relai en amont d'une communication déjà établie R_{i+1} est déconnecté (p. 115). La déconnexion des relais en aval est normale : elle a lieu soit quand le client `src` ne veut plus communiquer avec `dst`, soit quand un relai a vu tous les relais en aval se déconnecter. Seule la déconnexion des relais en amont est anormale.

Le relai (ou client) choisit deux tokens temporaires pour cette reconnexion T'_i et T''_i. Le premier servira à identifier le retour, le second permettra de remplacer T_i qui va être envoyé en clair. Il envoie à LB_{i+1} un paquet DECO construit de la manière suivante :

$(T_i \ [\![T'_i, T''_i]\!]_{rl_i})$

2.3.2.15 Load Balancer : demande de rétablissement de communication

Cet algorithme nommé RepairLb est exécuté par un `load balancer` lorsqu'il reçoit un paquet DECO (p. 117). Il retrouve l'identifiant du relai en amont. Il retrouve la clé locale associée et récupère les nouveaux identifiants. Il vérifie que ses connexions avec le relai précédement responsable sont bien rompues.

Il désigne un nouveau relai R'_i responsable de la communication avec lequel il partage un secret rt'_i. Il lui envoie via une connexion C_{lb_i} un paquet REP avec toutes les informations qui étaient liées à T_i, et un nouvel identifiant T'_i.

$[R_{i-1}, h(\texttt{dst},n-i), LB_{i+1}, rl_{i-1}, rl_i, T'_{i-1}, T''_{i-1}, R_{i+1}, K_{R_{i-1},h(\texttt{dst},n-i)}, T_i, T'_i \ ]_{rt_{i'}}$

S'il n'y a pas de relais suivant, mais `dst`, il enverra plutôt :

$[R_{n-1}, \ h(\texttt{dst},0), \ Pad(LB_n), \ rl_{n-1}, \ Pad(rl_i), \ T'_{n-1}, \ T''_{n-1}, \ \texttt{dst}, \ K_{R_{n-1},h(\texttt{dst},0)}, Pad(T_n), Pad(T'_n) \ ]_{rt'_n}$

Pour les autres communications qui passaient par le relai déconnecté, si aucun paquet DECO n'est reçu pour signaler une déconnexion (et donc réclamer une reconnexion), il faut comprendre qu'elles avaient été déconnectées précédemment et donc il ne faut rien faire.

2.3.2.16 Relai : demande de rétablissement de communication

Cet algorithme est nommé RepairR, et est exécuté par un relai R'_i lorsqu'il reçoit de LB_i via C_{lb_i} un paquet REP (p. 121). Ayant la clé ABE correspondant aux attributs requis, il le déchiffre pour retrouver (R_{i-1}, $h(\texttt{dst}, n-i)$, LB_{i+1}, rl_{i-1}, rl_i, T'_{i-1}, T''_{i-1}, R_{i+1}, $K_{R_{i-1},h(\texttt{dst},n-i)}$, T_i, T'_i)
Il tente de créer une connexion C_{i-1} vers R_{i-1} et une connexion C_{i+1} vers R_{i+1} (ou vers `dst`).

S'il échoue à créer l'une des deux,

Il envoie à LB_i le paquet SAQ1 suivant :

$[\texttt{fail}, S_{i-1}, T_{i+1}, R_{i+1}\ K_{R_{i-1},h(\texttt{dst},n-i)}]_{rt_i}$

Sinon,

Il envoie à R_{i-1} le paquet RECO suivant :

$(T_{i-1}, [R'_i, T''_{i-1}]_{rl_i})$

Puis il envoie à R_{i+1} (s'il n'est pas `dst`) le paquet RECO suivant :

$(T_i, [R'_i, T'_i]_{rl_i})$

Finalement, il envoie à LB_i via C_{lb_i} le paquet SAQ1 suivant :

$[\texttt{confirm}, T_{i-1}, T''_{i-1}, T'_i, R_{i+1}\ K_{R_{i-1},h(\texttt{dst},n-i)}]_{rt_i}$

2.3.2.17 Relai et Client : rétablissement de communication

Cet algorithme nommé Reco, est exécuté par un relai ou client qui reçoit un paquet RECO (p. 125). Le paquet contient un identifiant en clair.

Si l'identifiant correspond à une déconnection signalée précédement,

On peut considérer la communication rétablie sans changement de clés. On pourra passer en mode DATA.

Si l'algorithme est exécuté par un relai, celui-ci doit informer LB_i des changements d'identifiants en lui envoyant un paquet SAQ1 composé ainsi :

$[\texttt{update}, S_{i-1}, T_{i-1}, T''_i, R'_{i+1}, K_{R_{i-1},h(\texttt{dst},n-i)}]_{rt_i}$

Sinon, l'identifiant est celui d'un relai en aval,

On se sert alors de rl_{i-1} pour déchiffrer l'autre partie du paquet et retrouver T'_i qui vient remplacer l'ancien T_i désormais dévoilé. On utilisera les mêmes clés avec ce relai qu'avec son prédécesseur.

Le relai doit informer LB_i des changements d'identifiants en lui envoyant un paquet SAQ1 composé ainsi s'il n'est pas le dernier relai :

$[\texttt{update}, T_{i-1}, T_{i-1}, T'_i, R'_{i+1}, K_{R_{i-1},h(\texttt{dst},n-i)}]_{rt_i}$

Ou s'il est le dernier relai :

$[\texttt{update}, T_{n-1}, T_{n-1}, Pad(T'_n), \texttt{dst}, K_{R_{n-1},h(\texttt{dst},0)}]_{rt_n}$

2.4 Discussions

Dans cette section, nous allons prendre petit à petit du recul et étudier le protocole dans ses aspects plus techniques et chiffrés, puis sur ses aspects plus généraux.

Delay. Le plus simple pour commencer est de commenter le temps d'établissement de la communication. Celui-ci peut se décomposer en temps de transmission et temps de calcul. Du côté du nombre de transmissions, TOR se servait de $2 + 4 + \cdots + 2n$ transmissions. Pour QasTor, si on considère les transmissions du chemin critique (c'est-à-dire sans SAQ1), on ne se sert que de $3n$ pour une première communication et entre $4n + 1$ et $5n - 1$ pour une communication partagée. Pour $n = 3$, on a un gain (9 au lieu de 12) pour les premières communications et une légère perte (entre 13 et 14 au lieu de 12) pour celles existantes. Augmenter n va en faveur de notre protocole.

Ce compromis va dans le bon sens : des nouvelles communications sont normales lors de navigations qui ne sont pas en direct et donc il peut y avoir une fréquence d'établissement de communication plus élevée. Au contraire, la durée de visionnage moyenne d'un direct étant de 28 minutes, il y a moins besoin d'échanges de clés. De manière générale, ce profil correspond mieux aux utilisations modernes.

Par ailleurs, ce temps prend en compte qu'il n'y a pas de relations géographiques entre les noeuds d'un groupe. Si les groupes venaient à être construits de façon à imposer une borne supérieure au ping entre les membres (ce qui semble naturel pour optimiser les algorithmes distribués dans le groupe), alors l'établissement des nouvelles communications serait d'autant plus rapide.

Du côté du temps de calcul, on a un clair surcoût dû à l'utilisation de multiples ABE et chiffrements asymétriques. Ce surcoût peut être grandement diminué par du précalcul. Par exemple, toutes les paires de clés PK_i/SK_i peuvent être préparées avant l'instruction de l'utilisateur d'établir une nouvelle communication, car elles ne dépendent pas de l'instruction. Cela peut aussi être fait pour les clés PK'/SK' des relais. Pour gagner du temps sur ABE, on peut souvent selon les primitives étudiées, précalculer une partie des variables permettant le déchiffrement. Pour les primitives ABE basées sur les couplages de courbes elliptiques, [Costello 2010, Scott 2011] proposent des méthodes de réduction des temps de déchiffrement.

> R. Adelin et moi-même avons travaillé sur ces méthodes d'optimisations basées sur le précalcul de variables (multiples de points de courbes elliptiques). Nous avions aussi commencé une piste de recherches sur la capacité de pousser plus loin ces optimisations lorsque les attributs utilisés sont prévisibles (comme c'est le cas dans QasTor). Un pre-print de ces travaux préliminaires est disponible [Nugier 2019]. Depuis, la partie expérimentale de ces travaux a été beaucoup plus poussée et ils ont vocation à être publiés prochainement.

Workload. Du côté du facteur multiplicatif, celui-ci ne change pas si on ne considère que des communications non partagées. Dès qu'un flux est partagé vers plusieurs clients, celui-ci diminue. Au fur et à mesure que des clients se connectent au même flux, de plus en plus d'entre eux se connectent via un groupe relai connaissant

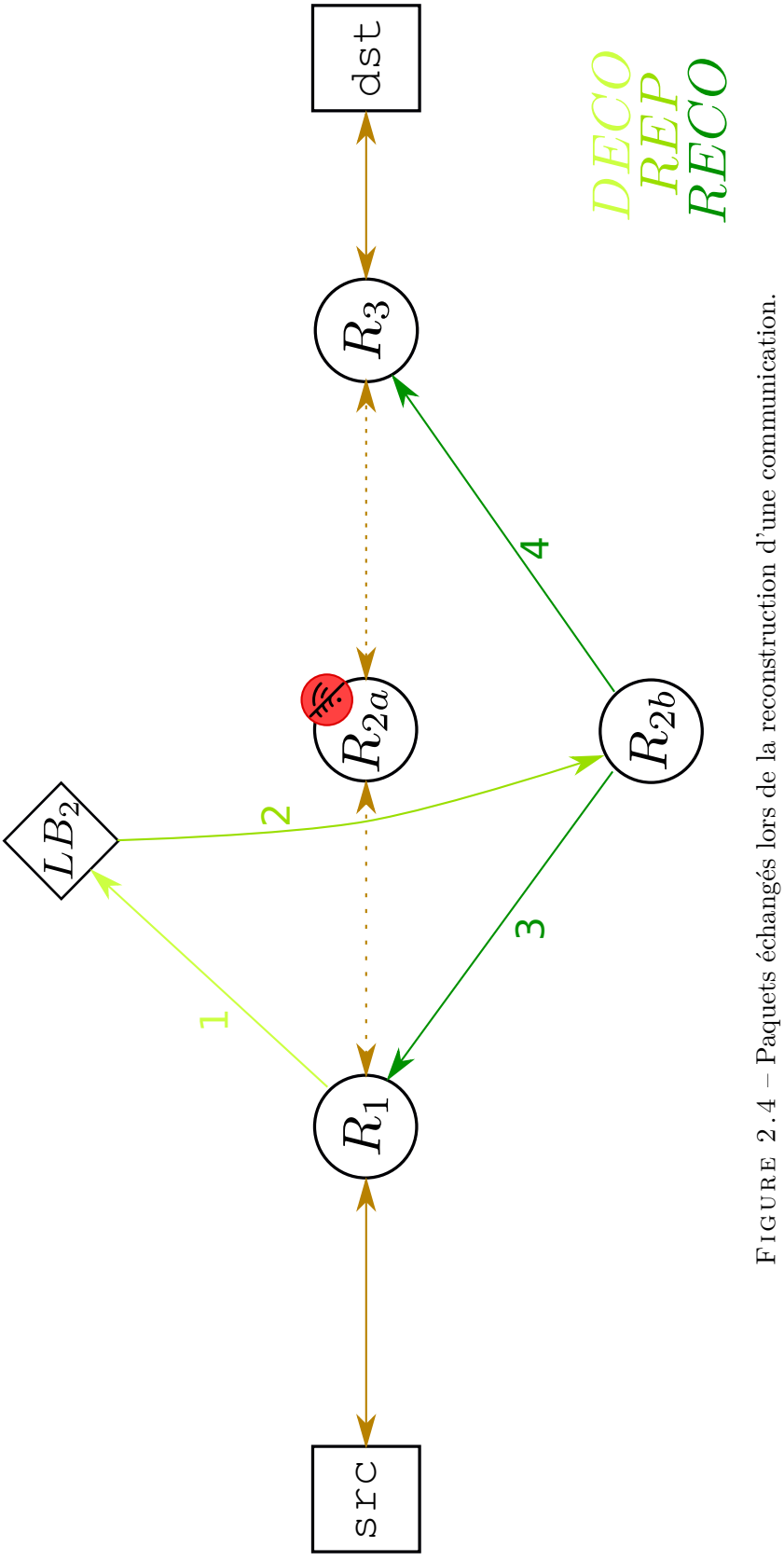


FIGURE 2.4 – Paquets échangés lors de la reconstruction d'une communication.

déjà l'empreinte du flux. Bien entendu, chaque client devant se connecter à au moins un relai, le facteur ne passera jamais sous 1, mais il tendra vers 1 à partir du moment où tous les groupes « Guard » connaîtront l'empreinte du flux et le nombre de clients augmente. On peut appeler ce moment la « saturation » du réseau vers ce flux.

En simulant les connexions connues entre les groupes relais, on peut évaluer le nombre de connexions établies entre des relais avant la saturation vers un flux. Nous avons choisi de prendre en compte plusieurs ratios de groupes « Guard ». On se rend compte que pour n fixé, le nombre de connexions avant saturation est proportionnel au nombre total de groupes (et donc de groupes « Guard »). En outre, plus le ratio de groupes « Guard » est faible, moins il y a de connexions avant saturation. En effet, le nombre de connexions avant saturation est instinctivement limité à n fois le nombre de groupes « Guard ».

Il semble alors avantageux d'avoir un faible nombre de groupes « Guard » pour permettre une forte économie de bande passante. Cependant, selon le mode de fonctionnement du groupe, cela peut être aussi un inconvénient. Un des modes de fonctionnement possible d'un groupe est un partage intégral des données (fonctionnement comme un ordinateur centralisé redondé). Dans ce mode, tous les membres du groupe peuvent faire des liens entre tous les clients dont la communication partage la même empreinte. Il faudrait alors un mode de confiance minimum où les données ne sont partagées qu'au nombre minimum de relais permettant la récupération des communications, à l'aide par exemple de techniques bien connues de partages de secret [Shamir 1979, Liu 2010]. Étant donné ce mode de fonctionnement, il est alors avantageux pour un même nombre de relais « Guard » de regrouper ceux-ci dans un faible nombre de groupes ayant plus de membres.

Cependant, avoir trop peu de noeuds « Guard » peut diminuer le nombre de connexions jusqu'à un point où les relais non-« Guard » sont trop peu exploités. La répartition dans les groupes doit donc se peaufiner expérimentalement selon des vraies données d'utilisation afin de trouver le point de fonctionnement optimal en équilibrant le nombre de groupes et leur balisage « Guard », le nombre de membres des groupes et le coût de fonctionnement d'un load balancing avec partage minimal des données.

Le coût pour `dst` est lui-aussi réduit. Le nombre de connexions sera désormais majoré en fonction du nombre de groupes « Guard » et non par le nombre de clients. Cette nouvelle configuration est donc adaptée à une forte montée en charge, en correspondance avec nos attentes.

Memory. Contrairement à TOR, il y a un certain nombre de variables qui doivent être stockées durant une communication pour pouvoir la rétablir. Nous avons volontairement ignoré la gestion de la mémoire dans le protocole pour en faciliter la lecture, mais on peut ajouter à tout algorithme attendant une réponse (les algorithmes QAS) que si celle-ci n'arrive pas dans un certain délai, la création de communication est considérée comme ayant échoué et est abandonnée et toutes les variables concernées libérées.

Trust. Contrairement au protocole TOR classique, des comportements malveillants peuvent être plus facilement punis. Des scores de confiance peuvent être attribués aux noeuds et évoluer avec le temps. Ces scores peuvent être gérés de

manière interne aux groupes, mais les relais en amont et en aval peuvent signaler des anomalies au `load balancer` du groupe d'un relai suspicieux, pour pouvoir diminuer sa confiance si les comportements suspicieux continuent. En faisant un tel signalement, le relai met en jeu une partie de sa propre confiance, qu'il perdrait en signalant à tort un relai honnête. Tous ses mécanismes ont déjà eu beaucoup de mises en pratique et s'inscrivent dans une longue lignée de réflexions inspirées du problème des généraux byzantins.

Stability. De manière identique à la gestion de confiance, une évaluation de la stabilité des relais peut être mise en place, pour servir d'aide au choix d'un relai responsable d'une communication. De cette manière, on peut déjà faire diminuer la fréquence des ruptures de communication. De plus, l'impact des déconnexions est énormément amoindrie, car il n'y a plus besoin que de quatre échanges avant un rétablissement de communication (au lieu de $2 + 4 + \cdots + 2n$ pour TOR). La perte peut être encore plus amoindrie grâce à l'utilisation de buffers. La mise en place de ceux-ci bien qu'intéressante est à réfléchir car elle implique un coût supplémentaire en mémoire, même si le mécanisme de timeout limite les risques.

En outre, bien que le protocole, tel que présenté, ne couvre que les déconnexions non planifiées, on peut ajouter facilement une méthode de transfert de charge permettant à un relai de demander le transfert de ses communications avant de se déconnecter. Le mécanisme est similaire, en remplaçant l'algorithme DECOc du relai en amont par un algorithme DECOr où le relai envoie lui-même un paquet DECO à LB_i. Le reste du transfert de connexion s'effectue normalement, mais cette fois les données peuvent continuer à transiter via le relai, et donc sans perte de données (éventuellement, on aura la création d'un léger lag si le nouveau relai a un ping un peu plus élevé avec les relais en amont et en aval).

Directory. La gestion de l'annuaire étant répartie, elle est grandement simplifiée. L'annuaire « global » ne concerne que les groupes, pas leurs membres. Les annuaires internes peuvent être gérés en interne au groupe pour le bon fonctionnement du load balancing. Les clés de chiffrement ABE peuvent être très réduites pour peu que les attributs soient calculables plutôt que stockés. La quantité de données que doit télécharger un noeud avant de pouvoir commencer à établir une communication est donc désormais constante.

Une possibilité supplémentaire qui s'offre à ce système est de créer des intersections entre les groupes, ceux-ci ayant alors des noeuds en commun. Ce choix est à double tranchant : si par hasard un noeud est aussi dans le groupe qui devrait être son successeur, il peut simplifier le chemin d'un saut. En revanche cela crée une perte d'anonymat en augmentant l'effet de collusions.

Cryptography. Ce protocole est basé sur des primitives cryptographiques génériques, dont le fonctionnement est similaire, que celles-ci soient classiques ou post-quantiques. Un démonstrateur peut donc être construit avec SHA256, AES256, RSA2048 et l'ABE d'OpenABE [Zeutro 2018] et fonctionner en attendant l'arrivée d'ordinateurs quantiques plus menaçants. Il pourra ensuite laisser place par exemple à SHA512, Chacha20, Crystals-KYBER et un PQ-ABE comme [Liu 2022] pour y résister.

Bien que dans cette version du protocole, ABE soit utilisé en tant que chiffrement

de groupe, de multiples extensions d'ABE permettent d'incorporer des propriétés additionnelles utiles à l'amélioration du protocole.

La première propriété à étudier est le choix entre un KP-ABE ou un CP-ABE [Goyal 2006, Bethencourt 2007]. Ce choix permet en général la même expressivité (il n'y a pas de chose possible avec l'un mais pas avec l'autre) mais les performances ne sont pas réparties de la même façon entre la génération de clés, le chiffrement et le déchiffrement. Ici aussi, des données réelles d'utilisation permettraient de trouver le meilleur compromis. Il semble sémantiquement plus instinctif d'utiliser un KP-ABE, car le client appose sur le message l'attribut d'un groupe, et ne s'occupe pas de la gestion des droits d'accès.

Les propriétés de multi-autorité ou d'autorité décentralisée [Chase 2007, Müller 2008] sont aussi très intéressantes car la distribution des clés ABE devient un point critique du protocole et il s'agit de limiter au maximum les possibilités pour un noeud malveillant de fournir des clés à d'autres noeuds en collusion.

D'autres propriétés d'intérêt viendraient compléter la gestion de confiance. Idéalement, un ABE avec clés révocables [Liang 2010] ou périssables [Liu 2018a], voire même des techniques de traitor-tracing permettraient une répercussion rapide des évolutions de la confiance au niveau cryptographique et pas seulement réseau.

Bien que la plupart des propriétés « avancées » d'ABE soient souvent basées sur les couplages de points de courbes elliptiques et donc appartiennent à la cryptographie classique, celles-ci sont petit à petit répliquées en cryptographie basée sur les réseaux euclidiens et donc résistants à l'ordinateur quantique.

Il est malheureusement important de noter l'écart significatif entre les implémentations disponibles d'ABE classiques et post-quantiques. Si les premières sont multiples, optimisées et facilement prêtes à l'utilisation, les post-quantiques sont dures ou impossibles à trouver, les papiers de recherche ne donnant presque jamais un accès libre aux implémentations de leurs expériences, laissant au lecteur le travail de re-coder de zéro l'algorithme présenté. Un effort dans ce domaine semble dès à présent crucial pour éviter un effet de perte d'expressivité lors d'une éventuelle transition précipitée vers de la cryptographie post-quantique.

Assez récemment, une librairie nommée Palissade, principalement tournée vers le chiffrement homomorphe basé sur les réseaux euclidiens a été rendue disponible sur GitHub [NJIT 2021]. Celle-ci contient un CP-ABE post-quantique, qui à défaut de présenter les propriétés mentionnées précédemment, est amplement suffisant pour commencer des implémentations entièrement post-quantiques et les études de performances de notre protocole.

Anonymity. L'anonymat dépend de beaucoup de paramètres, dont le nombre et la taille des différents groupes. Nous allons simplifier en imaginant qu'ils ont tous autant de membres : g groupes de taille m, la longueur du chemin est n. La probabilité de relier `src` à `dst` est donc de $\frac{1}{gm(gm+1)}$ pour les $(g-2)m$ noeuds qui ne sont pas aux extrémités de la communication, qui eux ont une probabilité de $\frac{1}{(g-2)m+1}$. En effet, tous les noeuds du groupe ont les mêmes capacités de déchiffrement. Ce score peut être amélioré dans la pratique avec une distribution conservatrice des données au sein du groupe. Ainsi, on a naturellement fait baisser la métrique de linkability moyenne car plus de personnes impliquées ont désormais des informations sur une

des extrémités. La métrique « maximum » aussi baisse car les noeuds d'un groupe savent que les autres membres de leur groupe, et ceux du groupe précédent (ou suivant) ne sont pas l'origine (ou la destination) de la communication.

Notre protocole ne change malheureusement aucunement la métrique de détectabilité, les attaques et les contre-mesures connues restent applicables et limitantes. Les attaques actives sur ces métriques sont en revanche désormais limitées par le déploiement d'une gestion de confiance.

En revanche, les conditions minimales permettant d'éviter le suivi des paquets correspondant à un même établissement de communication sont bien respectées. Premièrement, le format des paquets et leur longueur ne donne aucune information sur `src`, `dst`, ou le reste du chemin. Les champs chiffrés des paquets QAS1 sont tous déchiffrés et donc non reliables à cause de la propriété IND-CCA2. Tous les tokens apparaissant en clair (paquets SAQ2,QAS3, et RECO), sont ensuite jetés. On peut vérifier que pour toute information entrante d'un relai, soit un champ de la partie chiffrée a changé, soit la clé, soit un chiffrement a été ajouté ou supprimé. On ne peut donc pas associer des paquets par leur contenu. En revanche, comme pour TOR classique, le timing des réceptions et des émissions de paquets permettent de relier ceux-ci lorsqu'il y a peu de trafic et les contre-mesures impliquent souvent des baisses de performance.

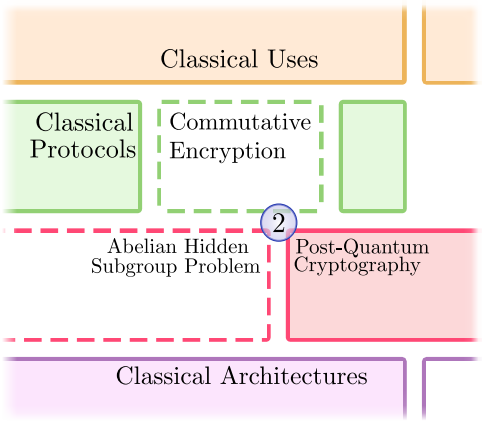
Dénis de Service. Les possibilités sont partiellement mitigées pour les raisons déjà évoquées. De plus, la sélection du relai étant faite par le `load balancer` parmi les membres du groupe et non plus par le client implique qu'il n'est plus possible de cibler un relai spécifique, seulement un groupe, dont la résistance au DoS est donc naturellement plus élevée.

Conclusion. Étant un protocole conçu pour les communications en direct One-to-Many de masse, QasTor se rapproche d'une utilisation habituelle des utilisateurs qui gagne en popularité. Les améliorations en termes de qualité de service devrait permettre de maintenir les pics de nombre d'utilisateurs plus longtemps dans la durée, ce qui améliorera la protection de la vie privée proposée et la capacité du réseau. Nous espérons que ce nouveau protocole permettra à TOR de retrouver un nouvel effet boule-de-neige pour protéger la vie privée du plus grand nombre.

Cependant, les protocoles complexes et multipartites comme QasTor sont inquiétés par la menace de l'arrivée d'ordinateurs quantiques. Ceux-ci vont impacter directement la sécurité de certaines primitives. Nous allons donc dans le prochain chapitre consacrer du temps à étudier quel est l'impact de ces changements sur l'ensemble des protocoles pouvant être mis en place de manière sécurisée.

Chapitre 3

Interception Réglementée basée sur du Chiffrement Commutatif Post-Quantique



Ce chapitre cherche à définir quels protocoles basés sur la commutativité des chiffrements peuvent être préservés lors du passage au post-quantique afin de garantir des utilisations telles que l'interception réglementée.

Sommaire

Depuis le début du XXI^{eeme} siècle, les systèmes connectés améliorent notre qualité de vie en prenant une part toujours plus grande de notre vie quotidienne. Presque tous les éléments de nos vies peuvent être accompagnés par des objets connectés. Parmi ceux-ci, on retrouve des objets mobiles (montres connectées, smartphones...), ou bien fixes (domotique, climatisations connectées...) et de tailles variées, allant de la discrète caméra de sécurité connectée, aux aspirateurs connectés, aux frigos, et aux voitures connectées. Ces objets ont pu changer la façon de percevoir à la fois l'espace domestique, l'espace professionnel et l'espace public.

L'espace urbain, avec l'arrivée de voitures connectées et de plus en plus autonomes, voit de fortes perspectives d'évolution. On peut même être rêveur sur la façon dont ceux-ci vont changer la ville : des voitures pouvant aller se garer seules permettraient de séparer les espaces résidentiels de ceux dédiés au stationnement. L'auto-partage permettrait de rentabiliser les voitures qui sont garées en moyenne 23 heures sur 24. La communication entre véhicules permettrait de synchroniser les freinages et accélérations afin de réduire les embouteillages. La centralisation du calcul d'itinéraire pourrait éviter l'encombrement de voies à la circulation réduite et donc amener à repenser les rues en laissant plus de places aux mobilités dites « douces ».

L'amélioration de la conduite et la diminution du coût du transport et des conséquences écologiques associées ne sont même pas les seuls gains promis par la voiture connectée. La présence de multiples capteurs permet de centraliser des informations (l'état de la route pour des réparation plus rapides, la disponibilité de places de parking), voire de reporter les données des accidents et participer à leur analyse.

> Le traitement préférentiel de ces informations selon leur valeur utilitaire est un sujet complexe. En effet, une voiture qui capte qu'une place de parking est libre et qui transmet cette information a pu aider un autre usager à sortir de la circulation plus efficacement. Mais le temps passant, cette information va perdre sa valeur, comme il est de plus en plus probable qu'une autre voiture ait pris cette place. Optimiser la gestion de cette valeur est une problématique

> forte de la thèse de J. Ibarz, qui a su me partager son intérêt des voitures connectées.

Ces aspects peuvent sembler aujourd'hui encore hors d'atteinte, mais il y a fort à parier que le renouvellement progressif du parc automobile pour des voitures de plus en plus « intelligentes » accélérera ces améliorations. Ce qui est certain, c'est que ces évolutions passent par la multiplication progressive des capteurs à bord des voitures. Notamment des capteurs qui décrivent l'état mécanique de la voiture (pression des pneus, état des freins...), des capteurs de navigation (radar, GPS...) mais aussi des capteurs internes (« infotainment », présence sur les sièges, caméras anti-sommeil...). Ces derniers sont par conception amenés à émettre des données qui concernent directement les passagers, mais certains (dashcams, radars...) peuvent aussi en émettre qui concernent les personnes dans l'environnement de la voiture.

Il faut alors pouvoir maîtriser qui accède à ces données. Les données internes seraient intéressantes pour un constructeur automobile ou un garagiste, les données urbaines pour une mairie, les enregistrements d'accidents pour la police ou la justice, les données de navigation pour le conducteur... Le traitement de ces données est déjà très complexe à cause entre autres de leur variété et de la diversité de leurs destinataires, mais la gestion de celles-ci est rendue encore plus délicate avec les contraintes matérielles.

En effet, les voitures ne sont pas que des ordinateurs à roues. Leur tâche principale est de garantir la sécurité des passagers et de leur environnement. Les calculs tenant à la sécurité (« safety ») sont alors prioritaires et le traitement des données à valeur ajoutée est facultatif. De plus, un véhicule en déplacement peut avoir plus de mal à trouver un destinataire pour ses données. Les connexions ne sont pas toujours stables et des protocoles peuvent prévoir des transmissions d'un véhicule à un autre véhicule, ou bien à une infrastructure de bord de route (bornes VANET). Il s'agit donc de choisir pour les données des méthodes de protection qui intègrent ces contraintes.

Pour venir couronner le tout, le traitement des données automobiles, tout comme celui de toutes les données personnelles, est sujet à des régulations elles-même sujettes à des évolutions dans le temps. On peut penser à l'exemple le plus marquant en France, celui de la RGPD (« Règlement Général sur la Protection des Données »), qui donne des consignes de gestion des données, mais qui nécessite une collaboration étroite entre des experts juristes et des experts en sécurité pour une application efficace. Ces régulations comportent des notions de consentement au partage des données personnelles. Pour les personnes à l'intérieur du véhicule, cela peut être déjà délicat à gérer, mais cela semble presque impossible à faire pour les personnes à l'extérieur, d'autant plus si on prend en compte les possibilités d'inférence en recoupant des données.

> Ce sujet est au coeur du travail de R.Adelin [Adelin 2022], avec qui j'ai eu le plaisir de collaborer et dont la thèse démontre la difficulté de ces traitements. Il propose des méthodes de traduction des contraintes légales en méthodes de chiffrement et de gestion des données et dont il prouve la sécurité.

Mais, comme évoqué dans le chapitre 2, les améliorations des méthodes de protection de vie privée affectent symétriquement les utilisateurs légitimes et les utilisateurs malveillants et potentiellement dangereux. Afin que ces améliorations ne se fassent pas au prix de la sécurité de leurs citoyens, de multiples pays (Canada, États-Unis [Canada 2019, Association 2001]) ont des exigences légales auprès des opérateurs de réseaux afin qu'ils permettent l'audit des données des communications en clair. Pour concilier cette nécessité et celle de respecter la vie privée des utilisateurs, on souhaite mettre en place des protocoles dits d'interception réglementée (« lawful interception » ou LI).

Le mécanisme principal permettant de concilier ces deux objectifs est celui de ne donner à l'autorité chargée de l'application de la loi (ex : police, gendarmerie, armée...) l'accès aux données que de manière temporaire et uniquement dans des circonstances très spécifiques et réglementées comme une décision de justice.

Historiquement, l'interception était pratiquée au niveau physique, en dénudant les câbles téléphoniques et en y installant des pinces. Des restrictions légales (voire constitutionnelles aux États-Unis) sont apparues pour limiter l'utilisation de cette méthode de protection contre les perquisitions et les intrusions abusives [Rosenzweig 1946]. Cependant, la responsabilité du contrôle des activités illégales sur un réseau a été reportée sur le gestionnaire du réseau [Larsson 1957]. Les régulations étaient alors uniquement faites par la loi, mais pas par des moyens techniques ou cryptographiques. Ainsi, les fournisseurs d'accès aux réseaux étaient capables d'accéder au message en clair.

Par la suite, le chiffrement de bout en bout s'est développé pour améliorer la vie privée des utilisateurs [Kent 1979], mais cela a aussi causé des inquiétudes, car les forces de l'ordre étaient incapables désormais d'accéder aux données en clair, malgré l'aide des fournisseurs de réseau.

C'est alors dans les années 1990 qu'un compromis réellement efficace fait son arrivée : les « key escrow mechanisms », ou mécanisme d'entiercement de clés (ou de séquestre de clés, ou de dépôt de clé selon les traductions). L'idée de ce système est de déposer les clés de chiffrement à un tiers de confiance qui est alors chargé de les garder secrètes, sauf si une décision de justice lui demande de les remettre aux forces de l'ordre (Law Enforcement Agency ou LEA en anglais). Pour ne plus se reposer sur un tiers de confiance, on peut simplement partager la clé en deux parties ou plus. Chaque dépôt a alors accès à une partie de la clé qui ne permet pas de déchiffrer le message. Mais lorsque les forces de l'ordre, munies d'une décision de justice récupèrent les clés auprès de tous les dépôts, elles peuvent reconstituer la clé et accéder au message en clair. Au niveau cryptographique, ces mécanismes étaient basés sur des variations des cryptosystèmes ElGamal ou Diffie Hellman, comme [Desmedt 1995]. Ces techniques ont été améliorées et diversifiées avec le temps, notamment pour de meilleurs performances, comme dans les travaux [Han 2011, Azfar 2011].

Les cryptosystèmes ElGamal et Diffie Hellman fonctionnent grâce à des exponentiations dans des groupes $\mathbb{Z}/n\mathbb{Z}$ (les entiers modulo n). Comme cette opération est commutative, on dit que ce chiffrement est commutatif. Comme les systèmes cryptographiques de dépôt de clés impliquent plusieurs participants (les dépôts), dont la grande majorité a le même rôle, et pour lesquels l'ordre de participation

n'est pas important, il est totalement naturel d'utiliser des chiffrements commutatifs (Commutative Encryption ou CE en anglais). Ces chiffrements ont une très forte adaptabilité et peuvent être utilisés pour concevoir des systèmes dans lesquels une entité ne peut pas par elle-même appliquer toutes les restrictions d'accès. Ils ont comme avantage de pouvoir subir de multiples chiffrements et déchiffrements successifs dans plusieurs ordres différents et sont donc parfaits dans notre contexte fortement évolutif où les politiques de contrôle d'accès doivent se conformer à de multiples obligations, nationales, commerciales, contractuelles...

Cependant, les chiffrements commutatifs actuels sont fragiles face à l'arrivée prochaine d'ordinateurs quantiques. Avec les algorithmes quantiques tels que celui de Shor, les problèmes du logarithme discret ou Diffie Hellman Bilinéaire deviennent polynomiaux (faciles à résoudre). Les protocoles de chiffrements commutatifs basés sur ElGamal et Diffie Hellman, dont la sécurité était garantie par la complexité de ces problèmes sont alors exposés à la cryptanalyse quantique. Même les schémas les plus modernes d'interception réglementée avec dépôt de clé [Nuñez 2019, Arfaoui 2021, Bultel 2022] voient leur sécurité menacée.

Or, on ne trouve à l'heure actuelle aucune étude générique permettant d'adapter des schémas de chiffrement commutatifs pour les rendre résistants à l'ordinateur quantique. De plus, il n'y a aucun mécanisme d'interception réglementée résistant à l'ordinateur quantique que nous ayons pu trouver dans la littérature.

Notre objectif est donc de fournir une adaptation du mécanisme d'interception réglementée avec dépôt de clés qui soit post-quantique. Comme la littérature sur les chiffrements commutatifs est principalement composée de schémas conçus pour une situation très précise, nous choisissons de prendre l'approche opposée et de proposer une étude générique, qui permet de définir les propriétés fondamentales des schémas de chiffrement commutatif, comme leurs propriétés de sécurité. Grâce à cela nous chercherons à proposer une méthode pour adapter des systèmes de haut niveau comme ceux de l'interception réglementée à dépôt de clés pour qu'ils soient post-quantiques.

Ce chapitre s'articulera donc de la manière suivante :

— Tout d'abord, nous établirons les définitions relatives au chiffrement commutatif et au sur-chiffrement, sa version simplifiée. Nous présenterons une classification des schémas de chiffrement commutatif existants et nous préciserons ce que l'état de l'art dit actuellement de leur résistance à l'ordinateur quantique.

— Ensuite, nous établirons un formalisme, des définitions et propriétés mathématiques ayant tous pour objectif de décrire l'ensemble des constructions cryptographiques à chiffrement commutatif ou à sur-chiffrement (Super Encryption ou SE en anglais). Nous prendrons en compte les propriétés de sécurité attendues de l'utilisation de ces schémas dans des systèmes multi-utilisateurs afin de délimiter mathématiquement quelles sont les constructions qui sont possibles et sécurisées avec une méthode mais pas avec l'autre.

— Ce formalisme pourra ensuite être étendu progressivement. D'abord en intégrant des encapsulations pour remplacer des chiffrements. Puis en intégrant des primitives post-quantiques parmi ces encapsulations, enfin en ajoutant un

mécanisme de partage de secret qui permettra d'ajouter la possibilité d'une interception réglementée. Cette possibilité n'est aujourd'hui ni trouvée dans la littérature sur l'interception réglementée, ni dans celle sur le chiffrement commutatif. Elle devra permettre à plusieurs États de répondre à leurs besoins en interception réglementée en dépit d'un éventuel ordinateur quantique capable de cryptanalyse, sans avoir besoin de concevoir de nouvelles primitives post-quantiques.

— Notre formalisme et notre cas d'étude nous ont aussi permis de distinguer quelques propriétés supplémentaires qui peuvent être garanties par le chiffrement commutatif, et d'isoler celles qui peuvent êtres produites par un chiffrement post-quantique générique de celles qui ont besoin d'un chiffrement conçu spécifiquement pour ces propriétés.

— Nous illustrerons un cas d'utilisation de notre méthode d'interception réglementée par chiffrement commutatif post-quantique. L'exemple a été choisi pour illustrer que notre méthode peut répondre aux besoins trouvés dans des milieux aussi complexes que les voitures connectées, et pour montrer que les solutions existantes basées sur le sur-chiffrement n'y auraient pas répondu.

3.1 Contexte et Définitions

Nous présentons ici les éléments connus dans la littérature concernant le chiffrement commutatif.

3.1.1 Chiffrement Commutatif

Souvent, des chiffrements classiques sont basés sur une opération commutative sans pour autant se servir de cette propriété. C'est le cas des chiffrements RSA, ElGamal et Diffie-Hellman. Cependant, cette propriété quand elle est présente permet de transformer le système de chiffrement en système de signature. Le déchiffrement pouvant être effectué en premier, il devient alors une signature et le chiffrement effectué ensuite devient la vérification de la signature.

Dans ces cas-ci, on utilise rarement le terme de chiffrement commutatif. En général, cette notion est réservée pour rendre explicite la manière dont le chiffrement se compose avec lui-même, c'est-à-dire ce qui se passe lorsque l'on re-chiffre un chiffré plusieurs fois, avec des clés qui peuvent être différentes.

Le chiffrement commutatif peut surtout être vu en contraste avec le sur-chiffrement. En effet, un chiffré, écrit sous forme de chaîne de bits, peut toujours être interprété comme le message d'un nouvel algorithme de chiffrement. Dans ce cas, si l'on veut retrouver le message, l'ordre des déchiffrements doit être l'ordre inverse des chiffrements, alors que dans le cas d'un chiffrement commutatif, tout ordre de re-chiffrement sera correct.

> De mon point de vue, la meilleure façon de visualiser la différence entre le sur-chiffrement et le chiffrement commutatif est l'analogie avec des verrous apposés à des valises. Pour le sur-chiffrement, un chiffrement correspond à mettre une

valise dans une valise encore plus grande, et verrouiller celle-ci, alors que pour le chiffrement commutatif, cela correspond à ajouter un verrou à une valise déjà verrouillée.

(a) Super-Encryption (SE) / Sur-chiffrement

(b) Commutative Encryption (CE) / Chiffrement Commutatif

FIGURE 3.1 – Accessibilité des étapes de déchiffrement pour SE and CE.

On voit bien que pour le sur-chiffrement, seul le dernier verrou est accessible, alors qu'on peut enlever les verrous dans n'importe quel ordre pour le chiffrement commutatif. Les objets mathématiques qui représentent ces fonctionnements sont respectivement une pile et un ensemble.

L'usage de chiffrement commutatif est alors adapté pour des applications multipartites, dont le nombre de chiffrements et de déchiffrements est grand, mais dont l'ordre de ces étapes est quelconque. Deux définitions formelles du chiffrement commutatif peuvent être trouvées dans l'état de l'art. Cependant, bien que très similaires, elles n'ont pas du tout les mêmes conséquences.

Définition 3.1.1 (dite « restreinte »)**.** Soit f une *Fonction de Chiffrement*. On dira que f est commutative si, pour toutes les clés de chiffrement k_1 et k_2, et pour tout message en clair m, la condition suivante est satisfaite :

$$f_{k_2} \circ f_{k_1}(m) = f_{k_1} \circ f_{k_2}(m) \tag{3.1}$$

Définition 3.1.2 (dite « étendue »)**.** Soit f une *Fonction de Chiffrement* et f^{-1} la fonction de déchiffrement correcte associée. On dira que f est commutative si, pour toutes les clés de chiffrement k_1 et k_2, et leurs clés de déchiffrement correspondantes k'_1 et k'_2, et pour tout message en clair m, la condition suivante est satisfaite :

$$f^{-1}_{k'_1} \circ f^{-1}_{k'_2} \circ f_{k_1} \circ f_{k_2}(m) = m \tag{3.2}$$

On remarquera que dans la définition « restreinte », les déchiffrements peuvent être effectués dans n'importe quel ordre, tant que le chiffrement est correct. C'est la

définition la plus utilisée [Khayat 2008, Huang 2012, Liu 2018b]. Elle est notamment étudiée dans [Shamir 1980], où elle est associée à plusieurs diagrammes commutatifs différents. Ils sont présentés ci-dessous :

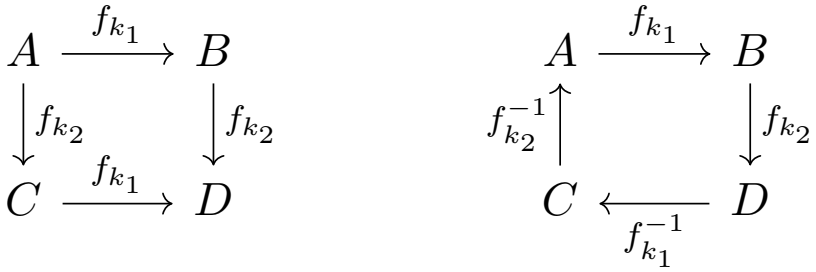


FIGURE 3.2 – Diagrammes commutatifs du chiffrement commutatif "restreint"

La définition « restreinte » est certainement privilégiée car elle est plus simple, mais elle exclut certaines constructions, comme les schémas cryptographiques proposés dans [Moldovyan 2019, Dai 2010], qui répondent à la définition « étendue » mais pas à la définition « restreinte ». La raison pour laquelle ces deux schémas sont ainsi est qu'ils incorporent une part d'aléatoire au chiffrement que le déchiffrement est capable d'ignorer. Ainsi, chaque chiffrement avec une probabilité écrasante sera unique et par conséquent $f_{k_1} \circ f_{k_2}(m)$ sera en pratique toujours différent de $f_{k_2} \circ f_{k_1}(m)$.

> La différence est que la valeur de deux chiffrements consécutifs doit être égale selon la définition « restreinte » (il y a une contrainte sur l'égalité dans D), mais peut différer selon l'ordre des chiffrements d'après la définition « étendue ». Dans les deux cas, après les déchiffrements et peu importe leur ordre, on retrouve le bon clair (contrainte sur l'égalité dans A).

À noter, pour les cryptographes avertis, il ne faut pas confondre le chiffrement commutatif avec le chiffrement homomorphe à un groupe (Group Homomorphic Encryption ou GHE en anglais). En effet, dans celui-ci on peut observer une commutativité dans le groupe des messages chiffrés avec une même clé.

$$f_k(m) + f_k(m') = f_k(m + m') \tag{3.3}$$

Les systèmes de ce genre sont principalement conçus pour permettre des calculs sur des données chiffrées sans avoir besoin de les déchiffrer au préalable. Cela permet par exemple à des utilisateurs de confier leurs données chiffrées à un tiers pour du traitement algorithmique, sans craindre que leurs données personnelles soient connues.

Les systèmes GHE ont été prouvés comme ne pouvant pas être résistants face à un ordinateur quantique [Armknecht 2014] si le groupe des chiffrés est Abélien. Cependant, des méthodes de chiffrement totalement homomorphe (FHE), c'est-à-dire homomorphe pour l'addition et la multiplication, sont elles considérées comme pouvant être post-quantiques tant qu'elles évitent les attaques connues sur les GHE.

3.1.2 Définitions génériques des systèmes à chiffrement commutatif

Souvent, les manuscrits de recherche présentent un nouveau schéma de chiffrement qui est fait pour pouvoir commuter avec lui-même. Mais ils peuvent commuter aussi avec d'autres schémas. C'est pourquoi nous introduisons une notion plus générique de « *Famille de chiffrements commutatifs* ».

Définition 3.1.3. Une famille $\mathcal{F}$ de fonctions de chiffrements commutatifs est appelée *Famille de chiffrement commutatif (Commutative Encryption Family, CEF)* si, pour tous $f, g \in \mathcal{F}$ et leurs fonctions de déchiffrement respectives f^{-1} et g^{-1}, pour toutes les clés de chiffrement k_1, k_2 et leurs clés de déchiffrement correspondantes k_1' et k_2', et pour chaque message en clair m, la condition suivante est satisfaite :

$$g_{k_1'}^{-1} \circ f_{k_2'}^{-1} \circ g_{k_1} \circ f_{k_2}(m) = m \tag{3.4}$$

Comme il arrive, surtout pour les primitives post-quantiques, que les algorithmes de chiffrement ou de déchiffrement ne soient pas déterministes, cette notion doit être élargie aux schémas où cette égalité est vérifiée avec une probabilité écrasante.

Pour étudier les propriétés exclusives à ces familles, nous avons besoin d'un point de comparaison sans commutativité, c'est-à-dire les « *familles de sur-chiffrement* ».

Définition 3.1.4. Une *famille de sur-chiffrement (Super-Encryption Family, SEF)* $\mathcal{F}$ est une famille de fonctions de chiffrement telle que pour chaque paire de fonctions f et $g \in \mathcal{F}$, il y ait une fonction bijective facilement calculable et facilement inversible entre l'espace des chiffrés, ou l'espace des clés de f et l'espace des clairs de g.

Les familles de sur-chiffrement permettent de chiffrer ou d'encapsuler récursivement. Contrairement aux familles de chiffrement commutatif, il faut respecter un ordre précis de déchiffrements ou de décapsulations pour retrouver un clair correct.

> Un exemple bien connu avec lequel le lecteur est désormais bien familier est celui de TOR, dans lequel RSA-OAEP est utilisé récursivement par plusieurs relais (Cf. chapitre 2).

3.1.3 Familles de chiffrement les plus connues

Cette liste n'est pas exhaustive, mais elle couvre les familles de chiffrement commutatif les plus connues.

Xor CEF : Les schémas de type « One-Time Pad »

Le schéma de One-Time Pad pour une longueur de message fixe n est commutatif. Tous les chiffrements qui se basent aussi sur le `xor` du message et d'une clé sont dans cette famille.

$$\forall m, k_1, k_2,\ (m \oplus k_1) \oplus k_2 = (m \oplus k_2) \oplus k_1$$

Les cas d'utilisation de Xor CEF (hors one-time pad) sont typiquement les techniques de dissimulation de données ou de watermarking (application de filigranes) qui commutent avec le chiffrement. Cela permet par exemple de produire une vidéo, la chiffrer, et la confier à un distributeur qui pourra ajouter un filigrane sans pour autant pouvoir lire la vidéo [Zhang 2013, Zhang 2011, Xu 2019].

Prime modulo exponentiation CEF : Schémas de type ElGamal
Étant donné un nombre premier p :

$$\forall m, k_1, k_2, (m^{k_1}\ mod\ p)^{k_2}\ mod\ p = (m^{k_2}\ mod\ p)^{k_1}\ mod\ p$$

.

Les constructions se basant sur une exponentiation avec un modulo premier sont de loin les plus nombreuses dans la littérature du chiffrement commutatif classique [Huang 2012, Khayat 2008, Liu 2018b, Dai 2010]. La sécurité de toutes ces constructions est dépendante du fait qu'il ne soit pas possible en pratique de résoudre le problème du logarithme discret.

Two factors modulo exponentiation CEF : Schémas similaires à RSA
Étant donné $n = p \cdot q$, tels que p et q soient deux nombres premiers :

$$\forall m, k_1, k_2, (m^{k_1}\ mod\ n)^{k_2}\ mod\ n = (m^{k_2}\ mod\ n)^{k_1}\ mod\ n$$

.

Les exemples les plus célèbres dans cette famille sont le RSA et ses variantes (PKCS, OAEP), et le protocole à trois échanges de Shamir, parfois appelé protocole sans clé, ou « Mental Poker ». L'utilisation de RSA dans un protocole de chiffrement commutatif n'est pas sécurisé pour plus que deux participants : pour former un CEF, il faut que tous les schémas de la famille partagent le même n. Or, il est désormais bien connu que d'avoir plusieurs instances de RSA qui partagent le même modulo crée des failles de sécurité majeures [Boneh 1998, Hinek 2010, SIMMONS 1983]. De même, tout schéma multipartite dans cette famille est compromis tant que la sécurité est basée sur la factorisation du modulo.

Prime modulo multiplication CEF : Schémas basés sur les couplages
Étant donné p un nombre premier :

$$\forall m, k_1, k_2, (m \cdot k_1\ mod\ p) \cdot k_2\ mod\ p = (m \cdot k_2\ mod\ p) \cdot k_1\ mod\ p$$

.

La plupart des schémas de chiffrement basés attributs (ABE) appartiennent à la famille des multiplications modulo un premier. Pour illustrer, un très large éventail de schémas ABE sont construits à partir d'une fonction bilinéaire e sur deux groupes d'ordre p $\mathbb{G}_1$ et $\mathbb{G}_2$ qui ont pour générateurs g_1 et g_2 qui sont parfois publics ou secrets, une clé maître $\alpha \in \mathbb{Z}_p$ et $E_k(m) = m \cdot e(g_1, g_2)^{\alpha k}$, $D_k(c) = c \cdot e(g_1, g_2)^{-\alpha k}$.

La valeur $e(g_1, g_2)^{\alpha k}$ peut être retrouvée par l'information contenue dans un schéma de partage de clé uniquement si les attributs valident les conditions de

contrôle d'accès [J. Bethencourt 2007, Goyal 2006]. Les deux catégories principales d'ABE (CP et KP) ont été présentées dans le Chapitre 2. Dans les schémas basés sur les couplages, la sécurité est basée sur la difficulté du problème Diffie-Hellman bilinéaire, c'est-à-dire que $e(g_1, g_2)^{\alpha k}$ n'est pas trouvable par un attaquant.

> Les couplages (pairings en anglais), parfois appelés accouplements, sont des fonctions $e : G_1 \times G_2 \to G_T$ ayant deux propriétés :
>
> 1. La bilinéarité : soient g_1 et g_2 les générateurs respectifs de G_1 et G_2 (tous les éléments non nuls de ces groupes sont des puissances de leurs générateurs), on a $\forall i, j \in \mathbb{Z}^2, e(g_1^i, g_2^j) = e(e_1, e_2)^{ij}$
> 2. La non dégénérescence : $e(g_1^i, g_2^j) = 0 \iff g_1^i = 1$ ou $g_2^j = 1$.
>
> Il y a de nombreuses fonctions de couplages. Les plus connues sont celles de Weil, celles de Tate et toutes leurs dérivées ate, Oate, eta... Les compromis sécurité/performance des couplages sont intensivement étudiés comme ils sont très utiles à la cryptographie basée courbes elliptiques.
>
> Avec R. Adelin, nous avons proposé une méthode pour prendre en compte ces optimisations des couplages dans les algorithmes d'ABE. L'idée générale est de profiter du fait qu'une partie de l'algorithme de calcul du couplage ne nécessite que l'un des deux arguments. Il faut alors concevoir le schéma cryptographique pour que cet argument soit dans la clé et non dans le message, pour pouvoir faire du pré-calcul [Nugier 2019].

Isogénies CEF

Les isogénies sont des objets mathématiques de haut niveau qui représentent des transformations d'une courbe elliptique en d'autres courbes elliptiques. Cette action de transformation est commutative, et des schémas comme [Costello 2016] étaient supposés résistants à l'ordinateur quantique jusqu'à récemment où le candidat SIKE basé sur SIDH à la standardisation de cryptographie post-quantique du NIST a été découvert vulnérable à une attaque qui peut s'exécuter en une heure [Castryck 2022]. [Smith 2018] donne les connaissances mathématiques nécessaires pour comprendre le calcul des isogénies des courbes elliptiques.

FNAA CEF : Finite Non-commutative Associative Algebras

Des formes de chiffrements commutatifs qui correspondent à la définition « étendue » mais pas à la définition « restreinte » émergent suite aux efforts liés à la création d'instances dures du Hidden Discrete Logarithm Problem (qui sont ainsi des instances non-Abéliennes du Hidden Subgroup Problem sans contraintes sur l'ordre des déchiffrements). Par exemple, une proposition a été faite pour se servir de l'algèbre des quaternions [Moldovyan 2010], mais ce schéma a par la suite été réduit polynomialement au problème du logarithme discret classique [Kuzmin 2017].

Cependant, dans [Moldovyan 2019], les auteurs utilisent une algèbre finie à m dimensions, dans laquelle l'opération de multiplication est non commutative, d'où le nom de FNAA. Dans ce schéma, selon l'ordre des chiffrements, le chiffré ne sera pas le même, mais le déchiffrement sera correct dans n'importe quel ordre.

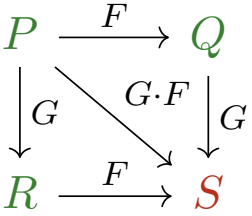
3.1.4 Le chiffrement commutatif contre l'ordinateur quantique

Parmi les éléments présentés jusqu'ici, quels sont ceux connus dans les travaux de recherche comme résistants ou pas à l'ordinateur quantique ? L'arrivée de l'ordinateur quantique est principalement inquiétant pour les fonctions correspondant à la définition « restreinte » de la commutativité ($f_{k_2} \circ f_{k_1}(m) = f_{k_1} \circ f_{k_2}(m)$). J.M. Couveignes a proposé une méthode d'analyse des différentes généralisations du schéma de Diffie-Hellman [Couveignes 2006]. Dans ces « espaces principaux homogènes » (Principal Homogeneous Spaces, PHS), le chiffrement permet de cacher la commutativité de l'action du groupe Abélien des clés sur l'espace des messages.

> Je tiens au passage à remercier B. Smith, qui a continué les travaux de Couveignes. Son manuscrit [Smith 2018] met au clair le raisonnement derrière le framework des PHS et y intègre les isogénies. Je le remercie d'avoir répondu à mes questions et de m'avoir aidé à comprendre les hypothèses de sécurité des PHS.

Dans ce framework, la sécurité des schémas est présentée comme dépendante de la difficulté de deux problèmes, que Couveignes a nommé *vectorisation* (à partir de $F(x)$ et de x, trouver F) et *parallelisation* (à partir de x, $F(x)$ et de $G(x)$ trouver $(G.F)(x)$). Ces problèmes ne sont pas prouvés polynomialement équivalents face à un ordinateur classique (si on sait résoudre la vectorisation on sait résoudre la parallelisation, mais pas l'inverse), mais ils le sont face à un ordinateur quantique [Galbraith 2018]. Si ces deux problèmes sont durs à résoudre dans un PHS, on parle d'espace homogène dur (Hard Homogeneous Space, HHS).

(a) Vectorisation



(b) Parallelisation

FIGURE 3.3 – Deux problèmes durs dans un HHS. Les éléments connus sont en vert, ceux à trouver en rouge.

Par exemple, dans Xor CEF et Prime modulo multiplication CEF, le problème de vectorisation est facile, même pour un ordinateur quantique : $(F \oplus x) \oplus x = F$ et $(F \cdot x) \cdot x^{-1} = F$, où x^{-1} est facilement trouvé grâce à l'algorithme d'Euclide étendu. Les schémas basés sur la difficulté de la factorisation ou du problème du logarithme discret ont leur problème de vectorisation dur pour un ordinateur classique, mais facile pour un ordinateur quantique grâce à l'algorithme de Shor [Shor 1999]. Ainsi tous ces PHS (qui ne sont donc pas des HHS), ne peuvent pas être utilisés dans des schémas de chiffrement commutatif où les résultats des étapes de chiffrement ou de déchiffrement sont rendus publics les uns après les autres.

Par ailleurs, lorsque le problème de vectorisation n'est pas basé sur un problème facile à résoudre pour un ordinateur quantique, il peut tout de même parfois être réduit comme une instance du Abelian Hidden Shift Problem (à partir de F, G Tels que $\forall x, F(x+s) = G(x)$ trouver s). Kuperberg a trouvé une réduction supplémentaire comme une instance du Dihedral Hidden Subgroup Problem. Donc dans ce cas, le problème de vectorisation hérite d'une complexité quantique de $L(1/2, \sqrt{2})$ avec de nombreux compromis entre complexité quantique, complexité classique et complexité mémoire proposées dans [Regev 2004b, Kuperberg 2005b, Kuperberg 2011b, Childs 2014].

Cependant, le seul schéma basé sur le Abelian Hidden Shift Problem qui n'était pas réduit à un problème sur le logarithme discret, SIKE, a été récemment victime d'une attaque avec un ordinateur classique [Castryck 2022]. Si une adaptation permettait de palier à cette vulnérabilité, ce serait le seul candidat à la standardisation à être (potentiellement) post-quantique tout en répondant à la définition « restreinte ». Mais on peut maintenant supposer cette définition de la commutativité et la sécurité post-quantique comme étant inconciliables.

Les schémas qui correspondent à la définition « étendue » mais pas à la définition « restreinte » de la commutativité, ne sont donc pas dans le framework des PHS. À l'heure actuelle, il n'y a pas d'attaque quantique sur ces schémas. Par exemple le schéma [Moldovyan 2019], basé sur des FNAA, est une instance non Abélienne du Hidden Discrete Logarithm Problem, et donc est supposé post-quantique. Ce schéma est cependant distant mathématiquement des autres schémas du NIST, et donc moins étudié et optimisé.

> Voici une intuition qu'il serait intéressant de formaliser : même si les FNAA de [Moldovyan 2019] ne peuvent pas être qualifiés de PHS, ils ne s'en différencient uniquement que par la partie aléatoire du premier chiffrement, les chiffrements suivants étant déterministes. Ils valident l'équation suivante :
>
> $$f_{k_3} \circ f_{k_2} \circ f_{k_1}(m) = f_{k_2} \circ f_{k_3} \circ f_{k_1}(m) \tag{3.5}$$
>
> Ce qui, bien que différent, est fort ressemblant à le définition « restreinte » de la commutativité. Les attaques sur les PHS seraient-elles alors applicables sur les chiffrements après le premier ? A l'inverse, [Dai 2010] intègre de l'aléatoire à chaque chiffrement ce qui éloigne de toute comparaison avec un PHS, mais n'est pas post-quantique car basé sur le logarithme discret. Il n'est pas clair si ces deux idées peuvent être conciliables.

On ne trouve pas dans la littérature sur la cryptographie post-quantique d'étude sur le chiffrement commutatif pour faire du chiffrement ou déchiffrement multipartite. L'étude de la commutativité s'arrête à la réduction de schémas individuels en Abelian Hidden Subgroup Problem. La commutativité devient informellement synonyme de cryptographie classique, les schémas de chiffrement commutatif sont délaissés, l'effort se concentre pour produire de nouvelles primitives post-quantiques, alors que de nombreux aspects fondamentaux du chiffrement commutatif sont ignorés. Du côté positif, il suffit d'un seul schéma post-quantique utilisé récursivement pour avoir du sur-chiffrement post-quantique. En revanche, il n'est nulle part établi quelles

propriétés sont propres au chiffrement commutatif et absentes du sur-chiffrement. On ne sait pas quels systèmes étaient dépendants du chiffrement commutatif et lesquels peuvent être remplacés par du sur-chiffrement...

Typiquement, tous les schémas d'interception réglementée avec dépôt de clé sont basés sur le chiffrement commutatif, et sont tous classiques. Nous n'avons trouvé aucun équivalent post-quantique. De plus, tous les schémas restants dans la standardisation du NIST ne sont pas commutatifs [NIST 2022a].

Le formalisme que nous présentons dans la section suivante a pour objectif de répondre à ces problématiques, en donnant des outils d'étude génériques des schémas commutatifs et nous proposons grâce à ces outils des moyens d'adapter des systèmes basés sur des chiffrements classiques en des versions post-quantiques, dont les systèmes d'interception réglementée. De plus, nous n'aurons besoin que des standards du NIST et bénéficieront donc de toutes les améliorations dont ils sont sujets.

3.2 Étude générique du chiffrement commutatif

Cette section commence par présenter le jeu de sécurité associé à des schémas considérés individuellement. Cette notion pourra ensuite être étendue à des combinaisons de schémas. Un outil est introduit pour représenter le contrôle que ces schémas ont sur l'ordre d'intervention des différents utilisateurs. Cela permet ensuite la comparaison des ensembles des constructions possibles avec le sur-chiffrement et avec le chiffrement commutatif.

3.2.1 Propriétés de sécurité des schémas commutatifs et de sur-chiffrement

3.2.1.1 Propriétés des schémas considérés individuellement

Premièrement, la contrainte évidente à l'utilisabilité d'un schéma cryptographique $\mathcal{S}$ est qu'il soit *correct* : la génération des paires de clés privées/clés secrètes est telle que s'en servir permet bien de retrouver le message en clair. Pour un schéma simple $\mathcal{S}$, les espaces de ses clés de chiffrement et de ses clés de déchiffrement sont notés respectivement $K_{\mathcal{S}}$ et $K'_{\mathcal{S}}$. Ses fonctions de chiffrement et de déchiffrement avec une clé k appliquées au message m ou au chiffré c sont notées respectivement $E_k(m)$ et $D_k(c)$.

L'ensemble des paires de clés (k, k') données par l'algorithme de générations de clés de $\mathcal{S}$ peuvent être représentées par l'application $\phi : K_{\mathcal{S}} \to \mathcal{P}(K'_{\mathcal{S}})$ telle que $k' \in \phi(k)$.

Définition 3.2.1. Le schéma est dit *correct* si pour tout clair m, l'équation suivante est vérifiée avec une probabilité écrasante : $\forall k \in K_{\mathcal{S}}, \forall k' \in \phi(k), D_{k'}(E_k(m)) = m$.

Cette définition de ϕ permet de facilement prendre en compte des primitives où les clés de chiffrement et de déchiffrement ne sont pas associées de manière bijective,

comme c'est par exemple le cas pour les chiffrements IBE, FIBE, ABE, HABE et toutes leurs autres variations.

Deuxièmement, il faut aussi que seule la possession d'une clé de déchiffrement correspondante à la clé de chiffrement utilisée permette de retrouver le clair. La définition standard de sécurité IND-CPA sera utilisée. On préférera utiliser dans le reste de ce chapitre le terme de schéma *individuellement sûr* pour mieux la distinguer de la sécurité de la composition de plusieurs schémas dans un chiffrement commutatif ou un sur-chiffrement.

Dans le jeu de sécurité suivant, l'adversaire représente un attaquant essayant de distinguer de quel message le chiffré provient. On aurait pu aller plus loin et exiger que les chiffrés soient indistinguables d'une chaîne de bits choisis uniformément, mais ce n'est pas nécessaire pour la suite et la plupart des schémas IND-CPA ont déjà cette propriété.

Setup : S'il y en a, le challenger $\mathcal{C}$ donne tous les paramètres publics du système à l'adversaire $\mathcal{A}_{dv}$. $\mathcal{A}_{dv}$ peut demander d'obtenir des clés de chiffrement et de déchiffrement et des paires clair-chiffré d'autres schémas du même CEF/SEF.
Phase 1 : $\mathcal{A}_{dv}$ récupère auprès du challenger une liste de clés de déchiffrement $H_j \in K'_{\mathcal{S}}$. Il peut demander au challenger le déchiffrement de chiffrés qu'il choisit.
Challenge : $\mathcal{A}_{dv}$ fournit deux messages de même longueur m_0 et m_1. $\mathcal{C}$ choisit $k_i \in K_{\mathcal{S}}$ tel que $\phi_i(k_i) \cap \{H_j\} = \emptyset$, lance une pièce équilibrée b, et chiffre m_b avec k_i. Le chiffré $CT = E_{k_i}(m_b)$ est donné à $\mathcal{A}_{dv}$.
Phase 2 : $\mathcal{A}_{dv}$ peut étendre sa liste de clés de déchiffrement tant que $\phi_i(k_i) \cap \{H_j\} = \emptyset$ reste vrai. Il peut demander davantage de déchiffrements de chiffrés qu'il choisit tant qu'ils sont différents de CT.
Guess : $\mathcal{A}_{dv}$ essaye de deviner b en renvoyant une valeur b'. Il gagne si $b' = b$.

L'*avantage* d'un adversaire disposant d'un temps de calcul et d'un stockage polynomial dans ce jeu est défini comme $\mathcal{A}_i = |Pr[b' = b] - \frac{1}{2}|$.

Définition 3.2.2. Le schéma est considéré *individuellement sûr* si $\mathcal{A}_i < \frac{1}{2^\lambda}$ contre un adversaire en temps polynomial si λ est le paramètre de sécurité.

3.2.1.2 Schémas de chiffrement commutatif

L'objectif des schémas de chiffrement commutatif est de composer des schémas individuels pour faire intervenir de multiples participants (noeuds) et pouvoir gérer efficacement une construction complexe. On pourra répertorier trois types de noeuds (trois rôles différents) dans un schéma de chiffrement commutatif : des noeuds *Sources* (S) qui déterminent le clair et le chiffrent, les *Cibles* (Targets, ou T) pour qui ces messages sont émis, et des *Relais* (R) qui feront circuler le message, tout en pouvant effectuer des chiffrements ou déchiffrements supplémentaires, tels les proxys des schémas de proxy-reencryption (PRE) [Ateniese 2006]. Formellement, le schéma de chiffrement commutatif sera défini ainsi :

Définition 3.2.3. Un *Schéma de chiffrement commutatif* (Commutative Encryption Scheme ou CES en anglais) G, pour une CEF $\mathcal{F}$, est un graphe $G = (V, E, O_e, O_d)$ avec :

- $V = S \cup R \cup T$ où S (resp. T) est un ensemble non vide de noeuds sources (resp. cibles), et R est un ensemble de noeuds relais.
- $\forall v \in V,\ (v, v) \notin E\,;\ \forall v \in S,\ (w, v) \notin E\,;\ \forall v \in T,\ (v, w) \notin E.$
- $\forall (v, w) \in E$ il y a un schéma $\mathcal{VW} \in \mathcal{F}$ qui lui est associé.
- $O_e = \{O_{e,v} | v \in V\}$, où $O_{e,v}$ est une relation d'ordre partiel sur l'ensemble des arcs entrants du noeud $\{(v, w) | (v, w) \in E\}$.
- $O_d = \{O_{d,v} | v \in V\}$, où $O_{d,v}$ est une relation d'ordre partiel sur l'ensemble des arcs sortants du noeud $\{(w, v) | (w, v) \in E\}$.

Une des définitions classique d'un graphe est $G = (V, E)$ ou V est l'ensemble des noeuds (*Vertices*) et E l'ensemble des arêtes (*Edges*) reliant les noeuds. Ici, la définition est sur-chargée pour prendre en compte un ordre entre les arcs et un schéma qui leur est associé (on incorpore ainsi l'ordre des schémas à la définition).

Définition 3.2.4. Un *Schéma de sur-chiffrement* (Super Encrytpion Scheme ou SES en anglais) est défini selon les mêmes contraintes, mais pour une SEF $\mathcal{F}$ et avec $O_{e,v}$ et $O_{d,v}$ des ordres totaux pour tous $v \in V$.

Propriété 1. *L'ensemble de tous les SES est inclus dans l'ensemble de tous les CES. Il y a une application triviale d'un SES vers CES (dont l'ordre partiel sera aussi total). La réciproque est fausse.*

Pour définir formellement le déroulement d'un CES/SES en pratique, nous définissons trois algorithmes. Le premier *Keys* permet de distribuer des clés aux noeuds, de manière à ce qu'il y ait une clé de chiffrement pour chaque arc sortant et une clé de déchiffrement pour chaque noeud entrant. Le suivant *Activate* présente le mode d'utilisation de ces clés par les noeuds. Le dernier *Run* représente le déroulement d'un CES/SES à partir des deux précédents algorithmes.

Algorithm 1 $Keys(v)$

$Encryption\_keys \leftarrow ()$
$List\_enc \leftarrow$ Sorted $(\{(w,v) \in E\}, O_{e,v})$
for $(w,v) \in List\_enc$ **do**
  $Encryption\_keys \leftarrow$ Append$(Encryption_keys, k_{\mathcal{WV}} \in K_{\mathcal{WV}})$
end for
$Decryption\_keys \leftarrow ()$
$List\_dec \leftarrow$ Sorted $(\{(v,w) \in E\}, O_{d,v})$
for $(v,w) \in List\_dec$ **do**
  $Encryption\_keys \leftarrow$ Append$(Encryption_keys, k'_{\mathcal{VW}} \in K'_{\mathcal{VW}})$
end for
Give $Encryption\_keys$ and $Decryption\_keys$ to v

Algorithm 2 $Activate(v, M)$

$C \leftarrow ()$
for $m \in M$ **do**
  $c \leftarrow m$
  for $k' \in Decryption\_keys$ **do**
    $ct \leftarrow D_{k'}(ct)$
  end for
  for $k \in Encryption\_keys$ **do**
    $c \leftarrow E_k(ct)$
  end for
  $C \leftarrow$ Append(C, c)
end for
Output C

Algorithm 3 $Run(node\_list, m)$

for $v \in E$ **do**
  $Keys(v)$
end for
$M = ()$
$out = ()$
for $v \in node\_list$ **do**
  if $v \in S$ **then**
    $M \leftarrow$ Append$(M, Activate(v, m))$
  else if $v \in R$ **then**
    $M \leftarrow$ Append$(M, Activate(v, M))$
  else if $v \in T$ **then**
    $out \leftarrow$ Append$(out, Activate(v, M))$
  end if
end for
Output out

> En langage courant, comment un CES/SES ainsi défini se déroule en pratique ? Premièrement un message est choisi, puis des noeuds sources $s_i \in S$ vont s'activer, ce qui signifie qu'ils chiffrent le message avec toutes leurs clés dans un ordre conforme à O_{e,S_i}. La relation d'ordre $(s_i, r_1)O_{e,s_i}(s_i, r_2)$ peut se lire « (r_i, r_1) avant (s_i, r_2) ». Elle impose que le chiffrement dans le schéma $\mathcal{S}_i\mathcal{R}_1$ soit effectué avant le chiffrement dans le schéma $\mathcal{S}_i\mathcal{R}_2$. Les chiffrés résultants sont diffusés aux autres noeuds. Lorsque des relais $r_i \in R$ vont ensuite s'activer, ils vont déchiffrer les chiffrés reçus avec toutes leurs clés de déchiffrement dans un ordre conforme à O_{d,r_i}, chiffrer le résultat avec toutes leurs clés de chiffrement dans un ordre conforme à O_{e,r_i}, puis diffuser le résultat final. Quand un noeud cible $t_i \in T$ est activé, il déchiffre avec toutes ses clés de déchiffrement dans un ordre conforme à O_{d,t_i}, mais garde le résultat secret.

On peut étendre la définition du caractère correct d'un schéma individuel pour la rendre applicable aux CES/SES.

> **Définition 3.2.5.** Pour un ensemble de listes de noeuds dites *valides*, un CES/SES G est dit *correct* si pour tout clair m,
> $node\_list$ est valide $\Rightarrow m \in Run(node\_list, m)$, avec une probabilité écrasante.

Cette définition implique notamment que si des schémas individuels correspondant à des arcs d'un CES/SES ne sont pas corrects, ceux-ci peuvent être oubliés, car les noeuds s'en servant ne permettront pas aux cibles d'accéder au message. On peut identiquement supposer que les clés données sont telles que pour un arc (v, w), la clé de déchiffrement $k'_{\mathcal{VW}}$ donnée à w appartient à $\phi(k_{\mathcal{VW}})$, la clé de chiffrement donnée à v, toute autre paire de clés ne pouvant pas servir si le schéma est individuellement sûr.

Exemple détaillé : La figure 3.4 présente un exemple de CES/SES dont tous les arcs sont associés à des schémas corrects. Les ordres de chiffrement et de déchiffrement sont notés sur les bouts des arcs lorsque ceux-ci ne sont pas triviaux. Nous allons vérifier ensemble que ce CES/SES est correct s'il n'y a qu'une liste valide $node\_list = (s_1, r_1, r_2, r_3, r_4, t_1)$. On peut remarquer qu'il suffit de le faire pour un SES, l'ordre des chiffrements ne changeant rien au caractère correct des CES.

— Après $Activate(s_1, m)$, $M = (E_{k_{\mathcal{S}_1\mathcal{R}_1}} \circ E_{k_{\mathcal{S}_1\mathcal{R}_2}} \circ E_{k_{\mathcal{S}_1\mathcal{T}_1}}(m))$ (les chiffrements ont été effectués dans l'ordre indiqué au dessus des arcs).

— Lors de $Activate(r_1, M)$, le déchiffrement du seul élément de M renvoie $E_{k_{\mathcal{S}_1\mathcal{R}_2}} \circ E_{k_{\mathcal{S}_1\mathcal{T}_1}}(m)$, qui une fois rechiffré vaut $E_{k_{\mathcal{R}_1\mathcal{R}_2}} \circ E_{k_{\mathcal{S}_1\mathcal{R}_2}} \circ E_{k_{\mathcal{S}_1\mathcal{T}_1}}(m)$. Cette valeur est ajoutée à la fin de M.

— Lors de $Activate(r_2, M)$, le dernier élément de M après deux déchiffrements vaut $E_{k_{\mathcal{S}_1\mathcal{T}_1}}(m)$, car ceux-ci se sont faits dans l'ordre inverse des chiffrements, ces derniers ne commutant pas. Cette valeur chiffrée deux fois dans l'ordre écrit sur les arcs vaut $E_{k_{\mathcal{R}_2\mathcal{R}_3}} \circ E_{k_{\mathcal{R}_2\mathcal{T}_1}} \circ E_{k_{\mathcal{S}_1\mathcal{T}_1}}(m)$. Cette valeur est ajoutée à la fin de M.

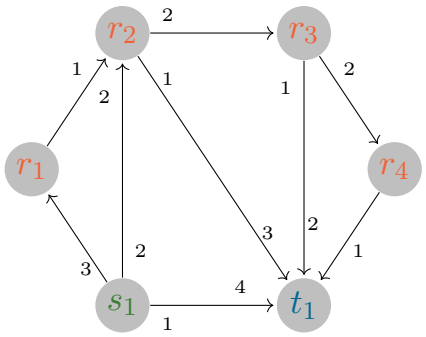


FIGURE 3.4 – Exemple de CES/SES avec quatre relais.

— Lors de $Activate(r_3, M)$, le déchiffrement du dernier élément de M renvoie $E_{k_{\mathcal{R}_2\mathcal{T}_1}} \circ E_{k_{\mathcal{S}_1\mathcal{T}_1}}(m)$, qui une fois rechiffré deux fois dans l'ordre écrit sur les arcs vaut $E_{k_{\mathcal{R}_3\mathcal{R}_4}} \circ E_{k_{\mathcal{R}_3\mathcal{T}_1}} \circ E_{k_{\mathcal{R}_2\mathcal{T}_1}} \circ E_{k_{\mathcal{S}_1\mathcal{T}_1}}(m)$. Cette valeur est ajoutée à la fin de M.

— Lors de $Activate(r_4, M)$, le déchiffrement du dernier élément de M renvoie $E_{k_{\mathcal{R}_3\mathcal{T}_1}} \circ E_{k_{\mathcal{R}_2\mathcal{T}_1}} \circ E_{k_{\mathcal{S}_1\mathcal{T}_1}}(m)$, qui une fois rechiffré vaut $E_{k_{\mathcal{R}_4\mathcal{T}_1}} \circ E_{k_{\mathcal{R}_3\mathcal{T}_1}} \circ E_{k_{\mathcal{R}_2\mathcal{T}_1}} \circ E_{k_{\mathcal{S}_1\mathcal{T}_1}}(m)$. Cette valeur est ajoutée à la fin de M.

— Une fois arrivés à $Activate(t_1, M)$, suite à tous les déchiffrements du dernier élément de M, on obtient m car les déchiffrements se font dans l'ordre inverse des chiffrements, ces derniers ne commutant pas.

Remarques :

1. Au cours de ce processus, tous les autres éléments de M n'ont pas été utilisés, car pour ceux-ci, soit le noeud n'a pas de clé dans le dernier schéma de chiffrement utilisé (le déchiffrement rendra un résultat inutilisable), soit il y a eu précédemment un déchiffrement dans un mauvais schéma qui a rendu le résultat inutilisable.
2. Si l'on avait changé les ordres des arcs il n'y aurait eu aucune liste de noeuds valide donnant un résultat correct pour un SES (et peu importe l'ordre, un CES aurait été correct).
3. Les activations avant un noeud source n'ont aucun effet, M étant vide. On pourra dons supposer que *node_list* commence toujours par un noeud source.
4. Les résultats étant concaténés dans M sans effacer les résultats précédents, toute *node_list* qui contient $(s_1, r_1, r_2, r_3, r_4, t_1)$ en tant que sous-liste, et pas forcément de manière contiguë, peut aussi être valide tout en préservant le caractère correct du schéma.
5. Lorsque une boucle apparaît dans le graphe, tous les noeuds en amont de la boucle ne peuvent pas contribuer à faire apparaître m dans M. Nous considérerons donc les graphes sans boucle.

Le caractère correct du CES/SES doit être accompagné de deux autres propriétés pour son utilisation en pratique. La première est de s'assurer que seuls des noeuds cibles puissent récupérer le message en clair, qui sera simplement appelée

sécurité du clair. La seconde est la capacité à limiter les listes de noeuds permettant aux noeuds cibles d'accéder au clair. L'ensemble de ces listes sera ainsi appelé *Valid Set*. Les CES/SES limitant l'accès à m à ces listes seront dits *respectant* le Valid Set.

3.2.1.3 Sécurité du clair d'un schéma de chiffrement commutatif ou de sur-chiffrement

Le jeu de sécurité suivant représente une collusion entre tous les relais aussi appelée comportement « honnête-mais-curieux » (les relais participent comme prévu au schéma, mais vont aussi essayer de récupérer autant de données que possible, à l'inverse d'un comportement malveillant où ils ne joueraient même pas leur rôle). A cause de cette collusion, les activations des relais n'apparaîtront pas dans le jeu de sécurité. Le CES/SES sera dit garantissant la *sécurité du clair* si les relais ne peuvent pas deviner le clair (et donc seuls les noeuds cibles le peuvent).

Setup : Exécuter la phase Setup du jeu de sécurité de chaque schéma individuel $\mathcal{VW}$ utilisé dans G.
Phase 1 : L'adversaire $\mathcal{A}_{dv}$ obtient une liste $H_{\mathcal{VW}}$ de clés de déchiffrement de $K'_{\mathcal{VW}}$ pour tout schéma $\mathcal{VW}$. Il peut demander au challenger le déchiffrement de chiffrés qu'il choisit dans les schémas qu'il veut.
Challenge : Le challenger exécute $Keys(v)$ pour tous les noeuds de V tel que pour tout schéma $\mathcal{VW}, \phi(k_{\mathcal{VW}}) \cap \{H_i\} = \emptyset$. Si $v \in R$, les clés sont aussi données à $\mathcal{A}_{dv}$. $\mathcal{A}_{dv}$ fournit deux messages de même longueur m_0 et m_1. Puis le challenger lance une pièce équilibrée b. Il fixe $M = ()$ puis exécute $M =$Append$(M, Activate(s, m_b)))$ pour tout $s \in S$. Ensuite, il donne M à $\mathcal{A}_{dv}$.
Phase 2 : Pour chaque schéma $\mathcal{VW}$, $\mathcal{A}_{dv}$ peut étendre sa liste de clés tant que $\phi(k_{\mathcal{VW}}) \cap \{H_{\mathcal{VW}}\} = \emptyset$ reste vrai. Il peut demander davantage de déchiffrements de chiffrés qu'il choisit tant qu'il ne s'agit pas d'éléments de M.
Guess : $\mathcal{A}_{dv}$ essaye de deviner b en renvoyant une valeur b'. Il gagne si $b' = b$.

L'avantage de l'adversaire est défini comme $\mathcal{A} = |Pr[b' = b] - \frac{1}{2}|$.

Définition 3.2.6. Un CES/SES est considéré comme garantissant la *sécurité du clair* si $\mathcal{A} < \frac{1}{2^\lambda}$ contre un adversaire en temps polynomial si λ est le paramètre de sécurité.

Proposition 2. *Soit $G = (V, E, O_e, O_d)$ avec $V = (S, R, T)$ un CES/SES, $\forall\ S_i \in S,\ \exists\ T_j \in T$ tel que $(S_i, T_j) \in E$ et $\mathcal{S}_i\mathcal{T}_j$ soit individuellement sûr, alors G garantit la sécurité du clair.*

Démonstration. ($\Rightarrow$) Supposons que $\forall s_i \in S,\ \exists t_j \in T$ tel que $(s_i, t_j) \in E$ et $\mathcal{S}_i\mathcal{T}_j$ soit individuellement sûr. Selon le jeu de sécurité de $\mathcal{S}_i\mathcal{T}_j$, l'avantage de l'adversaire est $\mathcal{A} < \frac{1}{2^\lambda}$ et cet avantage n'est pas augmenté grâce à la connaissance d'autres

schémas. Pour chaque noeud de $s_i \in S$, $Activate(s_i, m)$ ajoute à M un élément chiffré au moins une fois via un schéma partagé avec un $t_i \in T$. Les clés de ces schémas n'étant pas données à l'adversaire, l'avantage le plus fort que l'adversaire ait dans le jeu de sécurité des schémas individuels est une borne supérieure de l'avantage dans le jeu de sécurité de G. Comme tous les jeux individuels sont surs, on peut conclure que G garantit la sécurité du clair.

($\Leftarrow$) Supposons que G garantisse la sécurité du clair. Par contraposée, supposons que pour un certain $s_i \in S$, $\forall t_j \in T$, soit $(s_i, t_j) \notin E$ soit $\mathcal{S}_i\mathcal{T}_j$ n'est pas individuellement sûr. Dans le premier cas où il existe un $s_i \in S$, tel que $\nexists t_j \in T$, tel que $(s_i, t_j) \in E$, s_i chiffre seulement avec des clés dont des clés de déchiffrement correspondantes ont été données à des relais et donc à l'adversaire dans la phase Challenge, donc $\mathcal{A}_{dv}$ peut déchiffrer et accéder au message m et avoir un avantage dans le jeu de sécurité. Dans le cas où des arcs $(s_i, t_j) \in E$ existent mais aucun n'est individuellement sûr, $\mathcal{A}_{dv}$ a un avantage $\mathcal{A}_{\mathcal{S}_i\mathcal{T}_j} > \frac{1}{2^\lambda}$ dans les jeux de sécurité des $\mathcal{S}_i\mathcal{T}_j$ et donc a un avantage au moins égal au plus petit avantage individuel, et donc $\mathcal{A} > \frac{1}{2^\lambda}$ dans le jeu de sécurité de G. □

3.2.2 Propriétés d'ordonnancement d'un schéma commutatif ou de sur-chiffrement

3.2.2.1 Séquences d'utilisateurs valides

Un des objectifs (voire des prérequis) des schémas cryptographiques multipartites est d'avoir un certain degré de contrôle sur l'ordre de participation des utilisateurs. Ceux-ci sont des personnes ou des entités morales (des entreprises, services publics...). Nous devons relier cette notion au concept de noeuds tel que précédemment défini. Comme un même utilisateur peut gérer plusieurs noeuds, nous pouvons définir une application ρ qui à un noeud renvoie l'utilisateur auquel il appartient. Les CES/SES ne devant pas avoir de boucles, si un utilisateur doit intervenir plusieurs fois dans un schéma, cela se fera via un noeud différent pour chaque intervention. Il est alors possible d'étendre les concepts définis précédemment :

Définition 3.2.7. U est un *utilisateur source* si $\exists v \in S | \rho(v) = U$ de même, U est un *utilisateur cible* si $\exists v \in T | \rho(v) = U$

On peut supposer qu'un utilisateur n'est pas à la fois source et cible, car il pourrait s'envoyer le message à lui-même sans contrôle d'accès.

> Cette dernière proposition peut sembler logique, mais en pratique un utilisateur peut chercher à s'envoyer une donnée à lui-même. C'est par exemple le cas des utilisateurs de services d'hébergement de données chiffrées (parfois appelés coffres-forts numériques). La notion de temps et d'oubli du clair permet de distinguer l'utilisateur ayant émis les données de celui qui les récupère, et la transmission de la clé peut être vue comme un envoi de données à travers le temps. On est ici confronté à un exemple du « *paradoxe du bateau de Thésée* » que l'on résoudra en définissant quand cela se présente deux utilisateurs

différents.

On peut aussi étendre la notion d'activation d'un noeud à celle d'activation d'un utilisateur en donnant *user_list* en paramètre à *Run* à la place de *node_list* et en ajoutant ces lignes au début de l'algorithme :

$node\_list = ()$
for $U \in user\_list$ **do**
  $s\_list \leftarrow (v \in S | \rho(v) = U)$
  $r\_list \leftarrow (v \in R | \rho(v) = U)$
  $t\_list \leftarrow (v \in T | \rho(v) = U)$
  $node\_list \leftarrow \text{Append}(node\_list, s\_list, r\_list, t\_list)$
end for

On peut aussi étendre l'idée d'ensemble de séquences de noeuds valides dans la définition d'un CES/SES correct à des listes d'utilisateurs valides. On utilisera la notation VS pour Valid Set en anglais, avec $VS = \{X_1, X_2, \cdots, X_m\}$ et $X_i = (U_{i,1}, \cdots, U_{i,n})$. Un schéma correct doit alors avoir $X_i \in VS \Rightarrow m \in Run(X_i, m)$.

3.2.2.2 CES/SES respectant un Valid Set.

L'idée de cette partie est d'étudier quand l'implication est vraie dans l'autre sens : $m \in Run(X, m) \Rightarrow X \in VS$. Comme remarqué plus tôt, l'activation de noeuds supplémentaires est possible sans changer l'accès à m. Nous utiliserons donc la notion d'ensemble de sous-listes.

$$\overline{X} = \overline{(x_i)_{1 \leq i \leq n}} = \{X\} \cup \left(\bigcup_{i=0}^{n} \overline{(x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n)} \right) \text{ and } \overline{()} = \emptyset$$

Nous nous intéressons donc aux cas où $m \in Run(X, m) \Rightarrow \overline{X} \cap VS \neq \emptyset$, c'est-à-dire que seulement si une liste d'utilisateurs contient une sous-liste qui est dans VS (l'intersection de l'ensemble des sous-listes et de VS est non-nulle) alors un noeud cible peut retrouver le clair.

Le jeu de sécurité suivant prend en paramètre un Valid Set et représente une situation où l'adversaire agit comme un noeud cible qui souhaite deviner le message alors que les utilisateurs n'ont pas été activés dans une des séquences valides. Les relais ne sont pas en collusion avec l'adversaire. L'idée est que si l'adversaire n'a pas d'avantage, seules les activations respectant une des séquences valides permettent à un noeud cible de retrouver le message.

Setup : Exécuter la phase Setup du jeu de sécurité de chaque schéma individuel $\mathcal{VW}$ utilisé dans G.
Phase 1 : L'adversaire $\mathcal{A}_{dv}$ obtient une liste de clés de déchiffrement $H_{\mathcal{VW}} \notin K'_{\mathcal{VW}}$ pour tout schéma $\mathcal{VW}$. Il peut demander au challenger le déchiffrement de chiffrés qu'il choisit dans les schémas qu'il veut.
Challenge : Le challenger exécute $Keys(v)$ pour tous les noeuds de V tel que pour tout schéma $\mathcal{VW}, \phi(k_{\mathcal{VW}}) \cap \{H_i\} = \emptyset$. Si $v \in T$, les clés sont aussi données à $\mathcal{A}_{dv}$. $\mathcal{A}_{dv}$ fournit deux messages de même longueur m_0 et m_1, et une *user_list* $X = (U_1, U_2, \cdots, U_n)$. Puis le challenger lance une pièce équilibrée b. Il exécute $out = Run(X, m_b)$. Il donne M et out à $\mathcal{A}_{dv}$.
Phase 2 : Pour chaque schéma $\mathcal{VW}$, $\mathcal{A}_{dv}$ peut étendre sa liste de clés tant que $\phi(k_{\mathcal{VW}}) \cap \{H_{\mathcal{VW}}\} = \emptyset$ reste vrai. Il peut demander davantage de déchiffrements de chiffrés qu'il choisit tant qu'il ne s'agit pas d'éléments de M ou out.
Guess : $\mathcal{A}_{dv}$ essaye de deviner b en renvoyant une valeur b'. Il gagne si $b' = b$.

Pour que le schéma soit correct, il faut que si $X \in VS$, $m_b \in out$. L'avantage de l'adversaire dans ce jeu est $\mathcal{A} = |Pr[b' = b \text{ si } \nexists t \in T \text{ tel que } (U_1, U_2, \cdots, U_n, \rho(t)) \cap VS = \emptyset] - \frac{1}{2}|$.

Le fait d'avoir donné les clés des cibles à $\mathcal{A}_{dv}$ permet de représenter à la fois le cas où c'est une personne extérieure au CES/SES, mais aussi celui où il s'agit d'une cible souhaitant ignorer tout ou une partie du contrôle d'accès et trouver le message avant le moment prévu pour le dernier déchiffrement. De là aussi vient l'ajout d'un dernier noeud cible t dans la définition de l'avantage.

Définition 3.2.8. On dira d'un CES/SES qu'il *respecte* VS si $\mathcal{A} < \frac{1}{2^\lambda}$ contre un adversaire en temps polynomial avec λ le paramètre de sécurité.

Si un schéma est à la fois correct, sûr, et qu'il respecte VS, il permet de s'assurer que nul ne peut deviner le message sans une séquence valide, et que seul un noeud cible retrouve le message d'origine (avec probabilité écrasante) dès lors que la séquence est valide.

Lemme 3. *Pour chaque Valid Set contenant une seule user_list,* $VS = \{X\}$, *il y a toujours au moins un CES/SES correct qui garantit la sécurité du clair et respecte* VS.

Démonstration. Il n'y a qu'à prouver ce résultat pour un SES et la propriété 1 l'étendra à un CES en choisissant les schémas dans une même CEF, et en gardant les mêmes ordres que le SES.

Soit $VS = \{X\}$, avec $U = (U_1, U_2, \cdots, U_n)$. Le cas simple où $X = (U_1, U_2)$ peut être laissé au lecteur (qui pourrait par exemple décider d'appeler U_1 « Alice » et U_2 « Bob »), nous nous concentrons sur les cas où $n > 2$.

Construisons le SES $G = (V, E, O_e, O_d)$ comme suit.

— L'ensemble des noeuds de G est $V = (S, R, T)$,

— $S = \{s_1\}$ avec $\rho(s_1) = U_1$,

— $R = \{r_i \text{ for } 1 < i < n\}$ avec $\forall 1 < i < n, \rho(r_i) = U_i$,

— $T = \{t_n\}$ avec $\rho(t_n) = U_n$,

— $E = \{(r_i, r_{i+1}) | 1 < i < n\}\} \cup (s_1, t_n) \cup (s_1, r_2) \cup (r_{n-1}, t_n)$ et tous les schémas associés sont individuellement surs et corrects,

— $(s_1, t_n)O_{e,s_1}(s_1, r_2)$ et $(r_{n-1}, t_n)O_{d,t_n}(s_1, t_n)$. Comme les autres noeuds n'ont qu'un arc entrant et un arc sortant, il n'y a pas besoin de spécifier un ordre.

La sécurité individuelle de $\mathcal{S}_1\mathcal{T}_n$ garantit la sécurité du clair par la proposition 2. Il reste à prouver que le SES respecte VS. Il faut pour cela prouver qu'il y a toujours au moins un schéma individuellement sûr dans lequel $\mathcal{A}_{dv}$ n'a pas de clé de déchiffrement valide au fur et à mesure que les utilisateurs de la séquence X demandée par $\mathcal{A}_{dv}$ sont activés, ce qui permettra de s'assurer qu'à chaque instant $\mathcal{A} < \frac{1}{2^\lambda}$.

Comme remarqué précédemment, il est possible de supposer que X commence avec U_1, car des activations avant n'auraient aucun effet (M resterait vide). Après $Activate(s_1, m)$, la valeur ajoutée à M qui sera donc donnée à $\mathcal{A}_{dv}$ est chiffrée dans les schémas $\mathcal{S}_1\mathcal{T}_n$ et $\mathcal{S}_1\mathcal{R}_2$. On procède par induction, avec pour supposition que les éléments de M sont soit invalidés par un déchiffrement dans le désordre, soit chiffrés avec deux schémas individuellement surs $\mathcal{S}_1\mathcal{T}_n$ et $\mathcal{R}_{j-1}\mathcal{R}_j$. $Activate(r_{i\neq j}, M)$ avec tout noeud autre que r_j ne donne aucun avantage supplémentaire à l'adversaire contre le schéma $\mathcal{R}_{j-1}\mathcal{R}_j$, et ne produit que des éléments invalidés par un déchiffrement dans le désordre. $Activate(r_j, M)$ renvoie un seul message non-invalidé par déchiffrement (car $\mathcal{R}_{j-1}\mathcal{R}_j$ est correct), et celui-ci est toujours chiffré dans deux schémas : $\mathcal{R}_1\mathcal{R}_n$ et $\mathcal{R}_j\mathcal{R}_{j+1}$.

Le cas où $j = n - 1$ est le seul dans lequel $\mathcal{A}_{dv}$ possède des clés dans les deux schémas (il a les clés de t_n). Cependant, cela implique aussi qu'il existe un t tel que $\overline{(x_1, \cdots, x_n, \rho(t))} \cap VS \neq \emptyset$, à savoir $t = t_n$ ce qui est un cas exclu dans la définition du respect de VS.

Donc seule une *user_list* dont une sous-liste est valide pourra permettre à un noeud cible de retrouver le clair. □

Proposition 4. *Pour tout Valid Set (avec un nombre quelconque de séquences), il existe au moins un CES/SES correct qui garantit la sécurité du clair et respecte VS.*

Démonstration. Similairement au Lemme 3, la preuve n'est nécessaire que pour un SES. Par la construction du Lemme 3, le respect de chaque séquence valide est garanti par un graphe. Construisons l'union « naïve » de ces graphes (équation ci-dessous pour l'union de deux graphes, mais généralisable pour toute quantité dénombrable) :

$$\begin{aligned} G_1 \cup G_2 &\iff ((S_1, R_1, T_1), E_1, O_{e_1}, O_{d_1}) \cup ((S_2, R_2, T_2), E_2, O_{e_2}, O_{d_2}) \\ &\iff ((S_1 \cup S_2, R_1 \cup R_2, T_1 \cup T_2), E_1 \cup E_2, O_{e_1} \cup O_{e_2}, O_{d_1} \cup O_{d_2}) \end{aligned} \tag{3.6}$$

Dans cette construction, le raisonnement par induction du Lemme 3 n'est pas correct car la fusion de deux noeuds identiques provenant de graphes construits pour des *user_list* différentes crée des situations où le nombre de chiffrements pourrait être plus grand que deux. Il est possible de remédier à ce problème en s'assurant de l'unicité des noeuds. Cela peut se faire en définissant pour toute *user_list* X_i une application d'indiçage $u_i(V) = \{v_i |\ v \in V\}$ qui conserve $\phi : \phi(v_i) = \phi(v)$. Cela permet d'adapter l'union des graphes :

$$G_1 \cup G_2 \iff ((u(S_1) \cup u(S_2), u(R_1) \cup u(R_2), u(T_1) \cup u(T_2)), E_1 \cup E_2, O_{e_1} \cup O_{e_2}, O_{d_1} \cup O_{d_2}) \tag{3.7}$$

Le graphe résultant est donc une union de sous-graphes disjoints. Chacun d'eux garantit la sécurité du clair grâce à l'existence d'un arc correspondant à un schéma individuellement sûr entre son noeud source et son noeud cible. De même, comme chaque sous-graphe est un CES/SES correct, après activation des relais dans un ordre respectant une des séquences, au moins un noeud cible peut retrouver le message correct.

Il reste donc à prouver que l'union de G_1 et G_2, respectant respectivement VS_1 et VS_2 garantit le respect de $VS_1 \cup VS_2$.

Soit un graphe construit selon le Lemme 3 pour faire respecter VS_1. Si on ajoute au graphe un sous-graphe disjoint, la preuve du respect de VS_1 ne peut rester correcte que si aucune séquence composée uniquement des noeuds ainsi ajoutés peut permettre à $\mathcal{A}_{dv}$ d'avoir un avantage malgré une séquence non valide.

On peut appliquer symétriquement le même raisonnement au sous-graphe que l'on vient d'ajouter si celui-ci fait respecter VS_2. Ainsi, soit l'ordre des activations choisi par l'adversaire valide VS_1, soit il valide VS_2, soit aucun avantage n'est gagné lors de l'union des graphes. En répétant l'union de graphes deux à deux, on peut conclure que l'union d'une quantité dénombrable de CES/SES garantit le respect de l'union des Valid Sets dont ils garantissent individuellement le respect. Cette construction par union donne donc pour tout VS un CES/SES qui respecte VS. □

3.2.3 Écriture factorisée des Valid Sets.

En tant que liste de listes, les Valid Sets ne sont pas un outil pratique pour décrire et résoudre des situations complexes où énumérer tous les ordres valides serait exponentiellement long. Cependant, les Valid Sets peuvent être écrits dans une forme équivalente qui sera bien plus courte et bien plus pratique dans des applications concrètes. Cette forme est celle d'expressions reliant des utilisateurs avec les opérateurs binaires suivants : le « ou » (exclusif) noté $\vee$, le « et » $\wedge$, et le « puis » $\rightarrow$.

Voici quelques exemples pour se forger une intuition de l'utilisation de ces opérateurs :

— Dans le schéma d'envoi chiffré classique d'Alice vers Bob :
$VS = Alice \rightarrow Bob$.

— Dans un schéma de Proxy Re-Encryption de S vers T [Ateniese 2006] : $VS = S \to proxy \to T$

— Si deux proxys *peuvent* être utilisés, cela devient : $VS = S \to (proxy_1 \vee proxy_2) \to T$

— Si deux proxys *doivent* être utilisés mais peu importe l'ordre, cela devient : $VS = S \to (proxy_1 \wedge proxy_2) \to T$.

Ce qui rend ces expressions pratiques, c'est que ces trois opérateurs ont une sémantique identique au langage naturel. Cela les rend bien plus instinctifs que les listes de listes. De plus, ces expressions peuvent être manipulées grâce aux propriétés de « factorisation » suivantes :

Propriété 5. $A \wedge B \iff (A \to B) \vee (B \to A)$

Cette propriété peut être lue comme « Un message qui doit passer par A et B, doit soit passer par A puis par B, soit par B puis par A ».

Propriété 6. $\to$ *se distribue sur le* $\vee$*, c'est-à-dire :*

$$A \to (B \vee C) \iff (A \to B) \vee (A \to C)$$
$$(B \vee C) \to A \iff (B \to A) \vee (C \to A)$$

Ces deux propriétés permettent de factoriser les Valid Sets pour pouvoir décrire de manière efficace des cas complexes. Il est possible de faire l'inverse en écrivant toutes ces expressions dans une forme « canonique » $(A_1 \to \cdots \to A_n) \vee (B_1 \to \cdots \to B_m) \vee \cdots$ dans laquelle on voit bien l'équivalence avec la liste de listes $\{(A_1, \cdots, A_n), (B_1, \cdots, B_n), \cdots\}$. Cela prouve au passage l'équivalence de ces notations.

La figure 3.5 propose un exemple de CES/SES qui respecte $VS = S \to (A \vee B) \to (C \wedge D) \to T$. Pour trouver ce graphe, nous avons écrit VS en forme canonique puis utilisé la construction de la preuve de la Proposition 4. Remarquez bien que dans cette figure, chaque noeud a un nom unique grâce à un indiçage, alors que les utilisateurs sont identifiés grâce à la couleur des noeuds.

Alors que la Proposition 2 garantit que l'on pourra toujours construire un CES/SES fonctionnel pour chaque Valid Set, on a bien pour intuition en regardant la figure 3.5 que cette construction n'est pas optimale. Le fait qu'une notation factorisée des Valid Sets existe nous suggère que d'autres graphes d'ordres ou de tailles plus basses respectant les mêmes Valid Sets pourraient aussi exister.

3.2.4 Avantages du chiffrement commutatif sur le sur-chiffrement

Grâce à toutes les définitions précédentes, il est enfin possible de poser proprement la question suivante : « Quels sont les CES qui pour un VS fixé, soient corrects, garantissent la sécurité du clair et respectent VS mais qui ne pourraient pas être remplacés par un SES sans qu'une de ces propriétés soient invalidées ? »

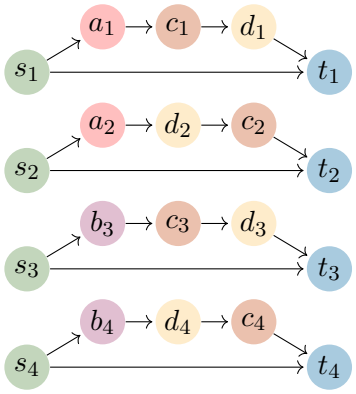
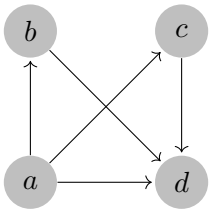


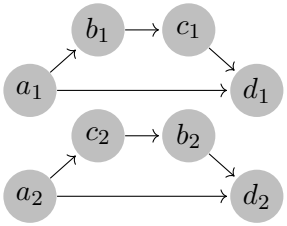
FIGURE 3.5 – Exemple d'utilisation de la construction de la preuve de la proposition 4 pour $S \to (A \vee B) \to (C \wedge D) \to T$.

Les Propositions 2 et 3 réduisent le problème car la sécurité du clair est garantie par l'existence du schéma individuellement sûr $\mathcal{ST}$ et le fait que tous les Valid Sets peuvent être respectés par à la fois un CES et un SES.

La différence viendra (comme d'habitude) du fait qu'un CES sera correct peu importe l'ordre affecté aux arcs (pour les étapes de chiffrement et déchiffrement), alors que pour les SES, certains ordres rendront le résultat non correct. Il existe même toute une catégorie de graphes (V, E), pour lesquels aucun ordres O_e, O_d de chiffrement et déchiffrement ne permet $G = (V, E, O_e, O_d)$ d'être correct.



(a) Le plus petit graphe "problématique".



(b) Le plus petit graphe de SES correct équivalent.

FIGURE 3.6 – Deux SES respectant $A \to (B \wedge C) \to D$

Lemme 7. *Le graphe présenté en Figure 3.6 ne peut pas être le graphe d'un SES correct si ses schémas de chiffrement appartiennent à un SEF dont aucune sous-famille n'est commutative.*

Démonstration. Supposons que $\mathcal{F}$ est un SEF dont aucune sous-famille n'est commutative. Le graphe respecte $VS = A \to (B \wedge C) \to D$. On peut donc dire que $(a, d)O_{e,a}(a, b)$ et $(a, d)O_{e,a}(a, c)$. Sans perte de généralité, supposons $(a, b)O_{e,a}(a, c)$. Durant $Activate(a, M)$, le noeud chiffre donc dans $\mathcal{AD}$ puis dans $\mathcal{AB}$, puis dans $\mathcal{AC}$. Plusieurs cas sont ensuite possibles.

Cas 1 : $Activate(d, M)$ est effectué avant $Activate(b, M)$ et $Activate(c, M)$. Certains chiffrements n'ont pas eu lieu (soit dans $\mathcal{BD}$ soit dans $\mathcal{CD}$), le déchiffrement par D dans tous les schémas dans lequel il possède des clés donnera un résultat incorrect, les schémas ne commutant pas entre eux.

Cas 2 : $Activate(b, M)$ est effectué avant $Activate(c, M)$. Le dernier chiffrement effectué était dans le schéma $\mathcal{AC}$, et les schémas ne commutant pas entre eux, le déchiffrement dans $\mathcal{AB}$ renvoie un résultat incorrect.

Cas 3 : $Activate(c, M)$ est effectué avant $Activate(d, M)$. c déchiffre dans $\mathcal{AC}$ puis chiffre dans $\mathcal{CD}$. Plus tard lors de $Activate(b, M)$, le déchiffrement dans $\mathcal{AB}$ ne commutant pas donnera aussi un résultat incorrect.

Ainsi, il n'y a aucune séquence d'utilisateurs qui permette à d d'avoir un résultat correct. □

Proposition 8. *Le plus petit graphe (en ordre et en taille) qui soit « problématique » pour les SES (ne peut pas être correct peu importe l'ordre choisi) est celui en figure 3.6a. Cela peut être vérifié par énumération de tous les graphes plus petits.*

Conséquence 1 : Soit (V, E) le graphe d'un SES. Si on enlève tous les noeuds appartenant à des mineurs isomorphes au graphe problématique et tous leurs prédécesseurs directs et indirects, on donne aux noeuds cibles exactement les mêmes capacités d'accès au clair que le graphe d'origine.

Conséquence 2 : Un algorithme itératif permet de générer tous les SES ayant un seul noeud source s et un seul noeud cible t et qui ne contienne pas le graphe « problématique ». On construit progressivement un graphe en commençant par le graphe (0) de la Figure 3.7 : on remplace un par un les arcs en pointillé par les graphes (1), (2), ou (3) jusqu'à ce qu'il n'y ait plus de graphes en pointillé. On peut vérifier que le graphe ainsi construit est un SES qui peut être correct car le point de départ est un SES qui peut être correct et chaque modification n'a créé aucun mineur problématique (deux chemins ou plus, de longueur deux ou plus reliant deux noeuds). Cette construction couvre toutes les possibilités pour des SES corrects : on peut le prouver visuellement en arrangeant les noeuds de manière chronologique (début d'un arc toujours à gauche et la fin à droite) et en mettant toujours le noeud c de la modification (2) au dessus de l'arc (a, b) pour pouvoir suivre l'évolution de la pile (« stack ») des chiffrements.

Conséquence 3 : Pour un même VS, le plus petit graphe d'un SES qui respecte VS et qui admet un ordre qui lui permet d'être correct ne peut pas être plus petit que le plus petit graphe d'un CES qui est correct et respecte VS.En effet, si un SES (V, E, O_e, O_d) est le plus petit à respecter ces critères, alors le CES $(V, E, \emptyset, \emptyset)$ les respecte aussi.

De plus, le plus petit SES correct, qui respecte le même VS que le graphe problématique, est obtenu en dupliquant le graphe mais en ne gardant qu'un des deux chemins de longueur 2 ou plus entre a et d. Si le graphe problématique est un mineur d'un graphe d'un SES, on peut appliquer cette transformation sur le mineur et la reproduire jusqu'à ce qu'il n'y ait plus de mineur problématique.

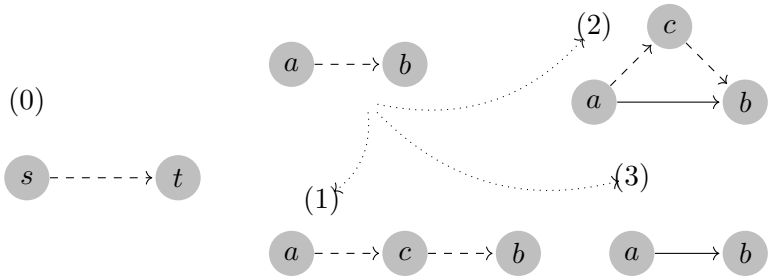


FIGURE 3.7 – Correct SE schemes Lock key graph construction

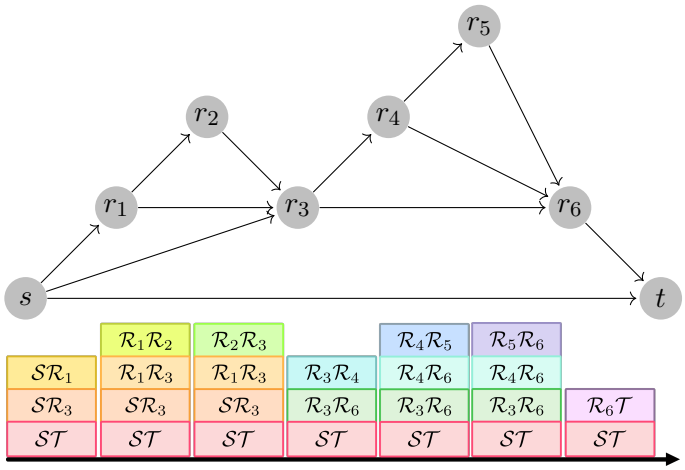


FIGURE 3.8 – Exemple d'un SES correct et pile des chiffrements. Ordre des chiffrements de bas en haut et déchiffrements de haut en bas

Ainsi, les graphes des CES peuvent n'être que plus petits ou de même taille pour un même VS. Aussi, plus le graphe d'un CES contient des mineurs problématiques, plus le graphe d'un SES qui respecte VS sera gros. Cette différence n'a pas d'impact sur la sécurité. Outre que de diminuer l'efficacité, ne pas pouvoir gérer les mineurs problématiques limite les cas d'utilisations des SES.

Conséquence 4 : Le Lemme 2 implique que « A applique un contrôle d'accès sur B, C, et D » et « B et C appliquent un contrôle d'accès sur D » ne peuvent pas être vrais en même temps pour un schéma de sur-chiffrement correct. Ainsi, étant donné le respect d'un même Valid Set, seul un schéma de chiffrement commutatif permettrait des situations de confiance entre les relais, c'est-à-dire permettre à plusieurs d'entre eux de s'assurer de l'identité des autres et des destinataires.

Il se distingue ainsi une réponse claire à notre question : « Quels sont les schémas de chiffrement commutatif qui, pour un Valid Set fixé, sont corrects, garantissent la sécurité du clair et respectent le Valid Set, qui ne puissent pas être remplacés par un schéma de sur-chiffrement sans qu'une de ces propriétés soient invalidée ? ». Il s'agit de ceux qui contiennent un mineur problématique, ce qui correspond aux situations où deux relais ou plus doivent s'assurer de l'identité des autres participants du schéma, ce qui est indispensable pour assurer un mécanisme d'interception réglementée.

3.3 LIQUoRICE - Lawful Interception with post-QUantum ResIstant Commutative Encryption

D'après le résultat de la section précédente, il n'est pas possible de se servir de sur-chiffrement pour gérer un dépôt de clé, les forces de l'ordre n'ayant aucun moyen de s'assurer que les autres parties de clés qu'ils ne possèdent pas ont bien été données aux dépôts prévus. Le chiffrement commutatif étant nécessaire, il faut alors l'adapter afin de le faire résister aux ordinateurs quantiques. Cette adaptation va d'abord passer par l'extension des schémas de chiffrement commutatif à des encapsulations, puis il sera possible d'introduire des primitives post-quantiques, et enfin une dernière modification permettra de gérer un dépôt de parties de clés.

3.3.1 Schémas d'encapsulations commutatives

L'encapsulation est le moyen de profiter de l'avantage des chiffrements symétriques (le chiffré est de même taille ou légèrement plus grand que le message) et de la capacité à partager un secret des chiffrements asymétriques, grâce à l'utilisation de fonctions de hachage. La sécurité de ce procédé a notamment été étudiée dans [Shoup 2000]. Tous les chiffrements asymétriques, s'ils sont dotés d'une fonction de hachage résistante à la recherche de collisions peuvent être transformés en mécanismes d'encapsulation de clé (Key Encapsulation Mechanism ou KEM en anglais).

On peut effectuer la même transformation sur les schémas de chiffrement commutatif. On ne chiffrera alors plus le message plusieurs fois, mais une seule fois, avec un algorithme symétrique et une clé w, par exemple $AES_w(m)$. Ce sera alors la clé, plus petite, et de taille indépendante de la longueur message, qui sera sujette au CES. Toute application de contrôle d'accès effectué sur cette clé est indirectement appliquée au message.

On peut aussi remplacer la famille de schémas de chiffrement d'un CES par des KEM. Pour cela, il faut au préalable choisir un groupe $(\mathbb{G}, +)$ qui est « commutatif » au sens de la définition étendue $\forall a, b, c \in \mathbb{G}, a + b + c - b - c = a$ ($a + b + c$ et $a+c+b$ peuvent être différents). Il faut aussi que les opérations $+$ et $-$ soient faciles à calculer.

Il est alors possible de remplacer un schéma de chiffrement $\mathcal{S}_i$ par une encapsulation, c'est-à-dire que pour tout m_i de l'espace des clairs de $\mathcal{S}_i$, pour $w \in \mathbb{G}$, étant donné une fonction de hachage résistante à la recherche de collisions H_i qui aille de l'espace des clairs de $\mathcal{S}_i$ dans $\mathbb{G}$, alors,

$$\begin{array}{lll} C \leftarrow E_{k_i}(w) & \text{équivaut à} & C' \leftarrow E_{k_i}(m_i), C \leftarrow w + H_i(m_i) \\ w \leftarrow D_{k'_i}(C) & \text{équivaut à} & m_i \leftarrow D_{k'_i}(C'), w \leftarrow C - H_i(m_i) \end{array} \tag{3.8}$$

Un changement fondamental est ainsi opéré : la commutativité des schémas de chiffrement transformés en encapsulations n'est plus nécessaire, étant donné qu'elle est désormais portée par le groupe $\mathbb{G}$. Il est donc possible de reprendre la définition d'un CES en enlevant le besoin pour $\mathcal{F}$ d'être une famille de chiffrements

commutatifs, mais à la place une famille de chiffrements quelconques, et en ajoutant une famille de fonctions de hachage correspondantes pour en faire des encapsulations.

Si le groupe $\mathbb{G}$ valide la définition « restreinte » de la commutativité, du point de vue de w, ces encapsulations sont équivalentes à des chiffrements dans un schéma de one-time pad itéré, tant que les r_i sont choisis aléatoirement de manière indistinguable de uniforme et que les $\mathcal{S}_i$ sont individuellement surs. Cependant, comme évoqué précédemment, les problèmes de vectorisation et parallélisation dans ces groupes étant souvent faciles, il faut prendre des précautions supplémentaires pour éviter de créer des problèmes de sécurité en utilisant des encapsulations.

Par exemple, l'adversaire obtenant le message en entrée et en sortie d'un relai dont l'activation ne contient qu'une encapsulation obtiendrait $C = w + \Sigma_i H_i(m_i)$ et $C' = w + \Sigma_i H_i(m_i) + H_{i+1}(m_{i+1})$. Si le problème de vectorisation dans $\mathbb{G}$ est facile (c'est-à-dire, $\forall a, b \in \mathbb{G}$, étant donné a et $a + b$, il est facile de trouver b) alors l'adversaire déduit facilement $C' - C = H_{i+1}(m_{i+1})$, ce qui rend l'encapsulation de m_i inutile vis-à-vis de la sécurité du clair ou du respect d'un VS, ce qui est bien le cas dans $\mathbb{G} = (\mathbb{F}_{2^n}, \oplus)$.

Le raisonnement est identique pour les décapsulations, et ainsi tous les relais n'effectuant qu'une seule étape d'encapsulation/décapsulation n'appliquent en réalité aucun contrôle d'accès si la vectorisation est facile. On peut les enlever du graphe d'un CES sans changer les capacités d'accès au message. Les relais restants ont alors tous un degré d'au moins 2.

L'idée est alors que $\forall m, a, b \in \mathbb{G}$, étant donné m et $m + a + b$ un oracle de résolution de la vectorisation ou de la parallélisation ne donne aucun avantage pour trouver a ou b s'ils sont choisis aléatoirement de manière indistinguable de l'uniforme car les valeurs intermédiaires $m + a$ ou $m + b$ ne sont jamais fournies.

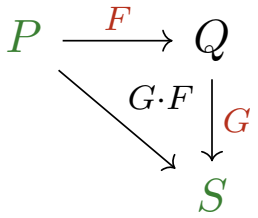


FIGURE 3.9 – Problème dur malgré des oracles de parallélisation et vectorisation. Les éléments connus sont en vert, ceux à trouver en rouge.

On peut s'assurer que cette contrainte soit satisfaite par construction pour garantir qu'un CES ne soit pas vulnérable à une attaque par vectorisation en choisissant le graphe de façon à ce que tous les relais soient de degré au moins deux. Cela n'a pas d'impact sur l'ensemble des VS que les schémas peuvent respecter car la construction du Lemme 3 vérifie cette propriété.

Les noeuds sources ne sont pas obligés d'avoir un degré d'au moins deux : le message, s'il s'agit d'une clé d'encapsulation, est lui-même uniformément aléatoire. On peut reprendre la figure 3.9 avec $P = 1_{\mathbb{G}}$, $Q = w$ et $S = w + H_1(r_1)$ pour constater que la sécurité du clair n'est pas non plus remise en question, tous les noeuds source devant avoir au moins un chiffrement vers un noeud cible. Ainsi,

malgré toutes les informations que puissent mettre en commun les relais, ceux-ci pourront au mieux obtenir $w + m_{\mathcal{ST}}$ et ne pourront pas en déduire w si $\mathcal{ST}$ est individuellement sûr.

3.3.2 Schémas de Chiffrement Commutatif Résistants aux Ordinateurs Quantiques

Comme la commutativité des schémas individuels n'est plus nécessaire, la possibilité est ouverte de se servir de schémas Post-Quantiques qui ne commutent pas. Tant qu'une fonction de hachage résistante aux collisions est disponible, l'encapsulation permet un contrôle d'accès identique à des chiffrements commutatifs.

Notamment, pour un CES, les propriétés de sécurité du clair, du caractère correct et du respect d'un VS, ne dépendent que du fait que les schémas qui le composent soient corrects et individuellement surs. Si cette dernière propriété est définie pour un attaquant disposant d'un ordinateur quantique, la conclusion est la même.

On peut alors construire un schéma de chiffrement commutatif post-quantique par exemple avec $\mathbb{G} = (\mathbb{F}_{2^n}, \oplus)$ et CRYSTALS-KYBER, la primitive fraîchement standardisée par le NIST. Nous surnommerons les schémas construits ainsi des CES « Xor-Kyber ».

Le problème de la vectorisation n'a donc même pas besoin d'être dur dans $\mathbb{G}$ sous deux conditions : que tous les relais aient des degrés d'au moins deux, et qu'il y ait un arc entre chaque noeud source et un noeud cible. Par conséquent, dans toutes les situations respectant ces contraintes, il sera certainement préférable d'utiliser un CES « Xor-Kyber » que de passer par une construction spécialisée comme des FNAA pour résister à un ordinateur quantique.

Ce résultat devrait réduire la motivation qui vise à rechercher et à développer de nouveaux schémas commutatifs post-quantiques (respectant la définition « élargie » de la commutativité mais pas celle « restreinte ») comme les FNAA. Nous montrerons plus tard que la grande majorité des fonctionnalités motivant l'utilisation de schémas commutatifs sont conservées par les CES « Xor-Kyber » et que très peu nécessitent ces schémas sur-mesure.

> On peut avoir l'intuition que cette méthode de groupement des opérations deux par deux pourrait aussi être applicable pour permettre l'utilisation d'un GHE post-quantique, contrairement à ce que [Armknecht 2014] prouve.
>
> Cette intuition vient du fait que la définition de la commutativité « restreinte » sur les clés que nous avons étudiée est très symétrique à la commutativité sur les messages des GHE, en voyant le chiffrement comme fonction à deux variables : $E(E(m, k_1), k_2) = E(m, k_1 + k_2) = E(E(m, k_2), k_1)$ ressemble à $E(m_2, k) + E(m_1, k) = E(m_1 + m_2, k) = E(m_1, k) + E(m_2, k)$.
>
> Si on suit cette intuition et qu'on force des additions d'au moins deux éléments simultanément dans le GHE, la phase de challenge du jeu de sécurité IND-CPA (implicite dans [Armknecht 2014] p.7 l.4) change de la première formulation à la seconde :
>
> « $\mathcal{A}_{dv}$ donne m_0 et m_1 au challenger, qui lance une pièce équilibrée b et donne $E_k(m_b)$ à $\mathcal{A}_{dv}$. »

> « $\mathcal{A}_{dv}$ donne m_0 et m_1 au challenger, qui lance une pièce équilibrée b, tire m_2 uniformément dans l'ensemble des clairs, et donne $E_k(m_2) + E_k(m_b)$ à $\mathcal{A}_{dv}$. »
>
> Le fait de pouvoir résoudre le *subgroup membership problem* devrait permettre à l'adversaire de retrouver $m_2 + m_b$, mais pas de deviner b. On applique en somme un masquage aux opérations. En revanche, contrairement aux schémas commutatifs, cela peut avoir plus d'impacts sur les utilisations des GHE.

3.3.3 Interception Réglementée basée sur du Chiffrement Commutatif Post-Quantique

Nous avons maintenant tous les éléments à notre disposition pour ajouter un mécanisme d'interception de clé dans les schémas de chiffrement commutatif se servant d'encapsulations, ce qui nous permettra de mettre en place un dépôt de clés.

Premièrement, les données sont chiffrées avec un algorithme de chiffrement symétrique grâce à la clé w. Contrairement à ce qui était fait dans la section précédente, il ne faut plus tirer aléatoirement w dans le groupe $\mathbb{G}$. A la place, il faudra déterminer w en fonction des différents $H_i(m_i)$. Nous présenterons d'abord le cas d'une interception directe, avant d'ajouter une méthode de partage de clé pour la distribuer entre des dépôts.

Soit a un relai de confiance. Un noeud source s tel que $(s, a) \in E$ peut choisir de donner à a des capacités d'interception. Cela peut être nécessaire si ce relai doit appliquer une politique de contrôle d'accès qui dépend du contenu du message, ou s'il appartient aux forces de l'ordre dans le cas des interceptions réglementées. Soit $\mathcal{SA}$ le schéma individuellement sûr et correct dans lequel s et a ont des clés de chiffrement et de déchiffrement correspondantes. s tire aléatoirement de manière uniforme un élément $m_{\mathcal{SA}}$ qui sera l'élément d'encapsulation du schéma $\mathcal{SA}$. Cela permet de calculer $w_{\mathcal{SA}} = H_{\mathcal{SA}}(m_{\mathcal{SA}})$. Il peut ensuite se servir d'une fonction de hachage H dont les entrées et les condensats sont dans $\mathbb{G}$, pour définir w comme $w = H(w_{sa})$.

Le schéma de chiffrement commutatif avec encapsulations peut se poursuivre sur w normalement, avec pour valeur spécifique $r_{\mathcal{SA}}$ utilisée dans le schéma $\mathcal{SA}$. Cela permet ainsi à a de retrouver $w_{\mathcal{SA}}$ pendant $Activate(a, M)$ (en calculant $H_{\mathcal{SA}}(m_{\mathcal{SA}})$). Le hachage par H permettra de retrouver w, la clé du chiffrement symétrique des données.

> Au cours du processus de publication des contributions présentées dans ce chapitre, il nous a été signalé la ressemblance de cette méthode à une porte dérobée (Backdoor en anglais). Or ce n'est pas le cas ici. Le noeud source doit faire le choix délibéré de ne pas choisir w aléatoirement. Il est donc pleinement conscient et informé de qui accédera au message. De plus, une couche de chiffrement de bout en bout peut toujours être ajoutée avant l'encapsulation des données, ce qui serait équivalent en termes de droits d'accès.

Cette construction ne rentre pas dans le jeu de sécurité définissant la sécurité du clair tel que précédemment défini à cause du choix de clés non-aléatoire et l'ajout

d'un relai de confiance. Il faut adapter le jeu de sécurité pour que les clés de a ne soient pas données à $\mathcal{A}_d v$. De plus, si H résiste à la recherche de collisions, il n'y a pas d'avantage à trouver w_{sa} à partir de $w_{sa} + H(w_{sa})$, donc ce choix n'est pas problématique.

Ainsi a a un accès permanent au message. Il faut maintenant le limiter aux situations d'interception réglementée, à savoir après la décision d'une autorité juridique.

Pour ceci, il va falloir transformer des relais en dépôts de clés en partageant celles-ci grâce à un schéma de partage de secret. Nous allons présenter un exemple où la clé est séparée en deux parties, la seconde étant donnée à un relai n. Cependant, toutes les méthodes plus avancées, basées sur des interpolations (pour faire du « k parmi n ») [Shamir 1979], ou toutes les méthodes linéaires (Linear Secret Sharing Scheme ou LSSS en anglais, permettant de créer des conditions booléennes de « et » ou « ou » entre des utilisateurs) fonctionneront aussi [Liu 2010]. Cela permettra de créer des façons plus flexibles de répondre aux problématiques d'interception réglementée.

Le prérequis à cette transformation est bien sûr que les relais qui ont des segments de la clé ne vont pas céder ceux-ci à a, sauf sous obligation légale. La confiance en ces relais n'est pas nécessaire, mais la non-collusion avec a l'est, ce qui est plus facile à garantir et aussi une hypothèse de base des systèmes de dépôt de clés.

s commence par choisir aléatoirement de manière uniforme $m_{\mathcal{SA}}$ et $m_{\mathcal{SN}}$ dans les espaces de clairs de schémas individuellement surs et corrects $\mathcal{SA}$ et $\mathcal{SN}$. Il calcule $w_{\mathcal{SA}} = H_{\mathcal{SA}}(m_{\mathcal{SA}})$ et $w_{\mathcal{SN}} = H_{\mathcal{SN}}(m_{\mathcal{SN}})$. Puis il fixe $w = H(w_{\mathcal{SA}}) + H(w_{\mathcal{SN}})$. Le schéma de chiffrement commutatif continue alors normalement mais avec les valeurs choisies pour ces deux schémas.

Il faudra ainsi que a demande légalement à n l'accès à $H(w_{sn})$ afin de pouvoir reconstruire la clé w et ainsi accéder aux données. n cependant ne sera jamais capable d'accéder aux données étant donné que a est de confiance.

De cette manière, nous avons à notre disposition la première méthode d'interception réglementée basée sur du chiffrement commutatif, grâce à l'utilisation d'une primitive post-quantique standardisée. Contrairement au sur-chiffrement, elle permet aux forces de l'ordre de contrôler quels sont les dépôts des autres parties de la clé. Notre étude formelle du chiffrement commutatif se conclut ainsi, mais il nous reste à étudier les impacts de notre construction sur la couche protocole, ce que nous allons faire en illustrant l'utilisation de notre formalisme dans le domaine des voitures connectées.

3.4 Interception Réglementée Post-Quantique appliquée aux voitures connectées

Nous présentons ici l'intégration du mécanisme d'interception réglementée basé sur du chiffrement commutatif dans un système de remontée de données de véhicules connectés. Dans cet exemple, nous allons nous concentrer uniquement sur des transmissions « V2I » (Vehicle-to-Infrastructure communications). Cet exemple est construit comme une évolution du système proposé dans [Adelin 2022].

Les véhicules connectés sont soumis à une quantité incroyable de régulations,

comme ils sont sujets aux contraintes de résilience des systèmes critiques temps-réel, aux contraintes de sécurité et de respect de la vie privée liée à la manipulation de données utilisateurs, et aux contraintes commerciales de valorisation de ces données. Les voitures ayant une espérance de vie pouvant dépasser une dizaine voire une quinzaine d'années, il est crucial que les systèmes de protection des données adoptés par celles-ci puissent s'adapter à des changements législatifs et de nouvelles contraintes sur chacun de ces sujets.

Les entités qui vont interagir dans ce scénario sont les suivantes :

— Le constructeur du véhicule, qui doit s'assurer que les données émises soient protégées ou accessibles selon la loi (par exemple : si les données moteur devaient être accessibles par le constructeur du véhicule).

— Le véhicule, V qui va essayer quand sa connexion le permet de transmettre les données qu'il crée. Le conducteur peut émettre des réserves sur les données qui sont rendues accessibles et par qui elles le sont, dans la limite de la loi.

— Un relai fixe de bord de route P, part d'un réseau VANET (Vehicular Ad-Hoc Network), récolte ces données et les renvoie sur le réseau classique. Il sert de proxy pour s'assurer que les prochains participants soient bien impliqués dans le schéma cryptographique.

— Le fournisseur d'accès au réseau N est désigné comme dépôt de clés.

— Les forces de l'ordre A ont des droits d'interception (par exemple en cas de litige, accident grave, vol...)

— Un serveur Cloud C va stocker les données pour les rendre disponibles. Il pourra donc appliquer du contrôle d'accès supplémentaire au moment de la demande d'une donnée.

— Des consommateurs de données D vont chercher au près du cloud des données qui les intéressent. Ces consommateurs peuvent être des mairies, des services météo, des services de recommandation d'itinéraire, des assureurs... Les conducteurs et les constructeurs peuvent eux aussi agir comme des consommateurs auprès du cloud.

L'ordre dans lesquels on souhaite que ces participants interviennent est : véhicule, puis proxy, puis fournisseur de réseau et forces de l'ordre (peu importe l'ordre), puis cloud et finalement consommateurs de données. Cela donne $VS = V \to P \to (A \wedge N) \to C \to D$.

Quelles sont les différentes relations de contrôle d'accès entre les participants ? La réponse nous permet d'établir le graphe G. Tout d'abord, le véhicule doit transmettre des parties des clés au dépôt de clés et aux forces de l'ordre. Le conducteur veut appliquer un contrôle sur les consommateurs de ses données, et le véhicule a un chiffrement vers le proxy pour s'assurer du passage des données par celui-ci. Le proxy est garant qu'un chiffrement a bien été effectué pour les forces de l'ordre et pour le fournisseur de réseau. Il peut en plus désigner un cloud pour l'hébergement des données. Les forces de l'ordre et le fournisseur du réseau étant impliqués dans le système de dépôt de clé doivent vérifier mutuellement leur identité. De plus, les forces de l'ordre pourraient limiter le lieu où les données peuvent être hébergées, pour des questions de souveraineté nationale des données par exemple. Et finalement, le cloud

est responsable de la dernière étape de contrôle du consommateur de données, les droits d'accès pouvant dépendre du temps depuis l'émission de celles-ci. Le graphe ainsi établi est présenté en figure 3.10. Des signatures peuvent être ajoutées à chaque étape de transmission en parallèle du CES.

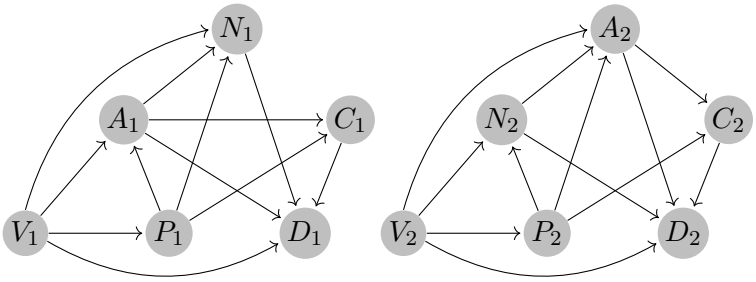


FIGURE 3.10 – CES pour une remontée d'informations de voitures connectées

Nous allons utiliser le groupe $(\mathbb{F}_{2^{512}}, \oplus)$ avec $H = SHA3$ pour être résistant à la recherche de collisions par un ordinateur quantique. Nous nous servirons de AES pour l'encapsulation des données, et sur tous les schémas où des clés sont préalablement partagées entre les parties ($\mathcal{A}_1\mathcal{N}_1, \mathcal{N}_2, \mathcal{A}_2$, éventuellement $\mathcal{P}_1, \mathcal{N}_1$ et $\mathcal{P}_2, \mathcal{N}_2$). Pour la plupart des autres schémas, devant utiliser de la cryptographie asymétrique, nous utiliserons CRYSTALS-KYBER. Cependant, certains schémas peuvent être choisis avec plus de précautions : les schémas $\mathcal{VD}$ peuvent se servir d'un ABE post-quantique [Wang 2020] afin de profiter de la dérivation à partir de la loi proposée dans [Adelin 2022], et $\mathcal{CD}$ peut aussi être un ABE géré par une autre autorité. Si la bande passante devait être réduite, comme les clés publiques de N et de A ne devraient pas changer régulièrement, $\mathcal{VN}$ et $\mathcal{VA}$ pourraient plutôt être Classic McEliece, qui a les chiffrés les plus petits des schémas post-quantiques en standardisation.

Au début du CES, pour permettre l'interception, le véhicule détermine aléatoirement de manière uniforme $r_{\mathcal{VA}}, r_{\mathcal{VN}}$ dans l'espace des clairs des schémas $\mathcal{VA}$ et $\mathcal{VN}$ et calcule $w_{\mathcal{VA}} = H_{\mathcal{VA}}(r_{\mathcal{VA}})$ et $w_{\mathcal{VN}} = H_{\mathcal{VN}}(r_{\mathcal{VN}})$. Il peut alors chiffrer les données m avec $CT = AES_w(m), w = H(w_{\mathcal{VA}} \oplus w_{\mathcal{VN}})$. Suite aux encapsulations dans les deux autres schémas, le véhicule va alors envoyer $w \oplus w_{\mathcal{VP}} \oplus w_{\mathcal{VA}} \oplus w_{\mathcal{VN}} \oplus w_{\mathcal{VD}}$ et les éléments d'encapsulation $(E_{k_{\mathcal{VP}}}(r_{\mathcal{VP}}), \cdots)$ au proxy, suite à quoi le CES se déroule normalement.

Comme tous les schémas choisis sont corrects, individuellement surs, qu'il y en a un reliant les noeuds sources à un noeud cible, et que tous les noeuds sont de degré au moins deux, la sécurité du clair est garantie, même dans un groupe dans lequel la vectorisation est facile. Le graphe impose bien le respect de VS. De plus, cette construction n'aurait pas été possible avec un schéma de sur-chiffrement, le graphe comportant plusieurs mineurs problématiques. Aussi, l'encapsulation permet de créer de la commutativité entre des schémas (AES, CRYSTALS-KYBER, ABE) qui n'avaient pas de commutativité au préalable.

Cet exemple devrait permettre d'étendre des protocoles tels que [Adelin 2022] à des cas d'utilisations plus complexes, là où les protocoles avec une seule étape de chiffrement trouvent des limites.

3.5 Fonctionnalités des schémas de chiffrement commutatif

Notre étude générique nous a permis de mettre en valeur une variété de bénéfices apportés par le chiffrement commutatif. Dans cette section, les propriétés en italique ne sont jamais apparues explicitement dans la littérature sur le chiffrement commutatif. La plupart des fonctionnalités listées sont utilisées dans notre application aux voitures connectées.

3.5.1 Fonctionnalités dont un CES « Xor-Kyber » permet le passage au Post-Quantique.

Respecter des chemins préétablis pour la donnée : Bien que cette fonctionnalité soit utilisée dans plusieurs schémas de chiffrement, une étude générique de la manière dont les ordres d'intervention des utilisateurs sont limités n'a jamais été faite à notre connaissance. Cette notion que nous avons appelé « respect du VS » a pour seul prérequis le comportement honnête mais curieux des relais. Ce comportement peut être obtenu grâce à des techniques dissuasives ou des protocoles de vérification (traitor tracing, watermarking, etc).

Délégation du contrôle d'accès à des noeuds honnêtes : Cette propriété est souvent trouvée pour des problématiques où l'émetteur est un appareil connecté mobile ou de faible capacité de calcul. Les mises à jour et changements de politiques peuvent être peu pratiques. Un relai honnête peut alors appliquer ces changements en tant que proxy pour plus d'adaptabilité.

Retarder l'application du contrôle d'accès : Les politiques de contrôle d'accès à des données peuvent être dépendantes du temps. Par exemple, une voiture émettant une information de place de parking libre génère une valeur importante dans les 30 minutes qui suivent mais ne fait plus qu'informer sur les trajets de son conducteur sans rien fournir en contrepartie par la suite. On peut alors souhaiter que les droits d'accès de certains services à certaines données dépendent du moment d'accès.

Bien que certains chiffrements basés attributs soient conçus pour permettre l'intégration de contraintes temporelles dans le contrôle d'accès, il n'en demeure pas moins que sans sur-chiffrement ou sans chiffrement commutatif, ces droits sont déterminés à l'émission des données. La seule autre méthode d'application de contrôle d'accès retardée est un mécanisme de requête (des clés par exemple), ce qui demande une interaction avec l'émetteur souvent impossible, car des appareils légers ne peuvent pas se permettre d'être en permanence accessibles et de gérer des données anciennes [Ateniese 2006].

Avec du chiffrement commutatif, les politiques de contrôle d'accès peuvent être appliquées au moment de l'accès. Un relai de stockage dans un cloud peut stocker des données qu'il ne rechiffrera que lorsqu'elles seront réclamées. Le re-chiffrement peut avoir une politique de contrôle basée sur le temps écoulé depuis la réception de la donnée, s'effectuer sans interaction avec la source et avoir une meilleure disponibilité. De plus, certaines données ayant à être stockées longtemps (parfois des années), cela permet d'avoir un contrôle d'accès cohérent avec la législation en vigueur au

moment du déchiffrement et pas du chiffrement. Il n'y a pas besoin de se servir de schémas conçus autour de cette fonctionnalité [Sahai 2012], tout standard peut servir.

> C'est la plus grosse différence entre les capacités des schémas ABE et les CES. ABE permet certes des politiques d'accès qui dépendent du temps, mais elles sont appliquées au chiffrement. Grâce au chiffrement commutatif, celles-ci peuvent être déterminées à l'accès et donc avoir évolué depuis le chiffrement. Cette différence majeure a en partie motivé la combinaison de ABE et de sur-chiffrement dans la publication avec R. Adelin.

Externalisation du calcul : Dans le contexte de l'Internet des objets ou des objets en temps réel, le chiffrement est énergivore et prend du temps, deux ressources précieuses pour des appareils légers. Le chiffrement commutatif permet de limiter ce coût en effectuant un contrôle minimal et en laissant un relai, ayant plus de temps et de capacités de calcul, appliquer un contrôle d'accès plus complet.

Décentralisation : L'encapsulation brise toute interdépendance entre des schémas individuels d'un CES. Chaque schéma peut alors être géré par une autorité de génération et de distribution des clés (par exemple une PKI) différente si nécessaire. De plus, certains schémas sont eux-mêmes conçus pour répartir les pouvoirs de génération de clés (autorité dite « décentralisée »). Ces responsabilités peuvent donc être partagées de manière aussi précise que nécessaire.

Composition de schémas : Le paradigme actuel de création de schémas cryptographiques est « bottom-up » : on recherche dans la littérature un cryptosystème le plus proche des contraintes fonctionnelles ou on en construit un nouveau. Cela peut ne pas satisfaire toutes les contraintes. Le chiffrement commutatif permet l'approche opposée, à savoir commencer par déterminer des ordres d'accès (Valid Sets), puis les contrôles appliqués entre les utilisateurs (arcs du graphe), et au final de choisir n'importe quel schéma individuellement sûr pour chacun de ces arcs, sans être contraint par les autres arcs.

Cela permet de considérer à la fois les contraintes de sécurité, de contrôle d'accès, de limites de performance, de temps de calcul et les limites d'infrastructure. Par exemple, certains schémas se spécialisent pour proposer une bande passante aussi petite que possible, ou un temps de calcul minimal au chiffrement ou au déchiffrement, ce dernier cas pouvant par exemple être désirable pour un noeud qui reçoit des données en permanence...

Capacité de mise à jour : A tout instant, l'un des schémas du CES peut être remplacé par un autre. Cela limite alors les besoins de distribution de clés à celles de ce schéma et aux noeuds concernés par celui-ci, sans avoir à effectuer une nouvelle fois toute la mise en place du CES.

Groupes relais : Plusieurs schémas de chiffrement one-to-many (par exemple les schémas ABE ou PQ-ABE) permettent de voir les relais et les cibles comme des groupes de noeuds et non pas des noeuds isolés. Lorsque c'est le cas, plusieurs étapes de chiffrement pour des groupes de noeuds agissent comme des intersections entre ces groupes, assurant que les noeuds ayant toutes les capacités de déchiffrement nécessaires valident les contrôles d'accès de tous les relais en amont.

Équilibrage des charges dans le réseau : Lorsque les relais sont des groupes de noeuds, il est possible que les noeuds du groupe s'accordent pour déterminer lequel sera responsable du calcul afin d'optimiser le ping, la bande passante et le temps de calcul des noeuds. De plus les VS offrant de multiples chemins possibles permettent aussi d'optimiser les charges et les routes entre les groupes relais.

3.5.2 Fonctionnalités nécessitant un CES « FNAA » pour un passage au Post-Quantique

Il s'agit donc de propriétés nécessitant que des noeuds du graphe du CES ne soient pas de degré au moins deux.

Le Protocole Sans Clé : Cette fonctionnalité est certainement celle qui en premier aura motivé l'étude des chiffrements commutatifs (plus précisément ceux qui commutent avec eux-mêmes). Ce protocole permet à Alice d'envoyer un message à Bob sans aucune information mutuelle (aucun échange préalable, même pas au travers d'une tierce personne). Pour cela, Alice choisit un message m, des clés $k_a, k'_a \in \phi(k_a)$, et envoie $E_{k_a}(m)$ à Bob. Il choisit des clés $k_b, k'_b \in \phi(k'_b)$ et répond à Alice en lui envoyant $E_{k_b} \circ E_{k_a}(m)$. Alice peut déchiffrer grâce à la commutativité et renvoyer $E_{k_b}(m)$ à *Bob*, qui peut alors déchiffrer et retrouver le message [Huang 2012].

Transfert Aveugle (*Oblivious Transfer* n'a pas de traduction parfaite en français) : C'est une extension de la fonctionnalité précédente qui permet à Alice de partager une information parmi n à Bob sans pour autant savoir laquelle Bob a pu apprendre [Dai 2010].

Watermarking : L'utilisation de la commutativité, principalement du `xor`, permet de faire commuter des étapes de chiffrement et des étapes de traitement de données telles que le filigrane (Watermarking). Une vidéo ou une image peuvent être compressées, chiffrées et recevoir un filigrane par un noeud relai. Cependant, ces techniques sont très fortement dépendantes de la méthode de chiffrement et d'application du filigrane [Xu 2019, Zhang 2013, Zhang 2011].

Il s'avère que ces trois propriétés ne peuvent pas utiliser une construction « Xor-Kyber » car leurs graphes sont tous celui de la figure 3.11 dans lequel des relais n'ont pas un degré au moins deux et seraient donc sujets à une attaque par vectorisation.

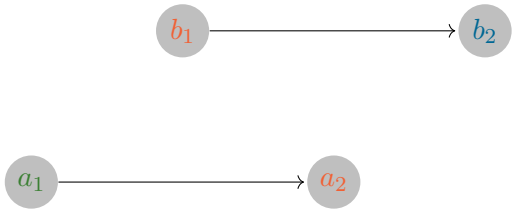


FIGURE 3.11 – Le graphe du protocole "Sans Clé"

Si on cherchait à reconstruire le protocole sans clé dans un CEF « Xor-Kyber », après $Activate(a_1, w)$, le message est encore sûr car deux éléments de $\mathbb{G}$ ont été déterminés en même temps ($w \oplus w_{a_1 a_2}$ ne donne aucune information sur w ou sur $w_{\mathcal{A}_1\mathcal{A}_2}$). Cependant, $Activate(b_1, M)$ ajoute $w \oplus w_{\mathcal{A}_1\mathcal{A}_2} \oplus w_{\mathcal{B}_1\mathcal{B}_2}$ et donc si le

problème de vectorisation est simple dans $\mathbb{G}$, l'attaquant est capable de retrouver $w_{\mathcal{B}_1\mathcal{B}_2}$ et donc de retrouver w après que $Activate(a_2, w)$ ait ajouté $w \oplus w_{\mathcal{B}_1\mathcal{B}_2}$ à M. Ainsi, pour pouvoir être utilisés de manière post-quantique, ces fonctionnalités doivent se reposer sur l'utilisation de schémas répondant à la définition « étendue » mais pas « restreinte » de la commutativité tels les FNAA.

Cependant, toutes ces utilisations dérivées de « Mental Poker » [Shamir 1981], jeter une pièce à distance de Blum, le problème des millionaires de Yao et le Transfert Aveugle de Rabin, ont tous comme point commun d'être vulnérables à des attaques par homme du milieu car il n'y a pas d'authentification possible sans échange préalable de données. A l'inverse, s'il est possible d'échanger un secret au préalable, il est possible d'encapsuler un jeu de « Mental Poker » classique (non post-quantique) basé sur le DLP si les joueurs n'ont pas d'ordinateur quantique. S'ils ont la chance d'en avoir, ils peuvent aussi jouer au mental poker avec une primitive quantique [Jia 2011]. Il y a donc peu de raisons connues actuellement de se servir de ces primitives spécialisées (FNAA).

3.6 Conclusion

Dans ce chapitre, nous avons proposé le premier schéma d'interception réglementée post-quantique, et des outils applicables à des scénarios complexes où des utilisateurs et des noeuds de natures variés sont impliqués pour permettre un contrôle d'accès fin et dynamique.

Nous avons commencé par une étude des propriétés déjà connues du chiffrement commutatif et de sa résistance aux ordinateurs quantiques. En étudiant génériquement et de manière formelle les constructions possibles avec du chiffrement commutatif et du sur-chiffrement, nous avons pu montrer que les constructions nécessitant des vérifications entre les relais ne peuvent pas se baser sur le sur-chiffrement.

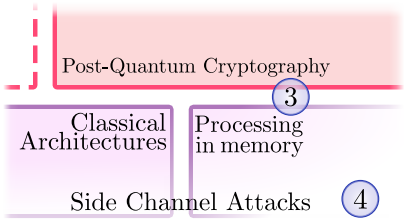
La transformation de chiffrements en encapsulations permet l'utilisation de n'importe quelle primitive post-quantique pour créer un schéma post-quantique de chiffrement commutatif, sous seules contraintes que les relais effectuent au moins deux opérations et que chaque source ait un chiffrement direct vers une cible. Cela permet aussi de choisir les encapsulations de manière à ce que la clé protégeant les données puisse être reconstruite à partir de fragments gardés par des dépôts jusqu'à ce que les forces de l'ordre les récupèrent sur ordre de justice.

Nous avons illustré l'utilisation de nos outils dans un scénario de véhicules connectés ou les contraintes en temps de calcul, bande passante, et contrôle d'accès sont complexes, et changeants.

Nous avons pu restreindre le besoin de primitives spécialisées telles les FNAA à des cas similaires au protocole sans clé, et permis la combinaison de primitives standardisées plus loin que par sur-chiffrement. Naturellement, des schémas de chiffrement commutatif impliquant plusieurs étapes de chiffrement et déchiffrement sont ceux qui peuvent bénéficier le plus de chaque amélioration de performance qui peut être apportée à ces standards.

Chapitre 4

Rôle des Architectures Émergentes dans l'Évaluation de Primitives Post-Quantiques



Ce chapitre étudie l'optimisation de primitives post-quantiques avec du calcul en mémoire (3) et les conséquences sur les fuites par canaux auxiliaires (4).

Sommaire

4.1 Introduction

La caractérisation, la création et l'optimisation de primitives cryptographiques post-quantiques sont des défis dont l'importance augmente à chaque fois qu'une nouvelle avancée de l'ordinateur quantique est annoncée. En 2019, Google annonçait avoir atteint la « suprématie quantique » [Arute 2019, Lesage 2019] et en 2021 IBM égalait cette prétention [IBM 2021]. Si ce terme est très bien choisi sur un plan publicitaire afin de motiver des levées de fonds, la réalité - même si elle correspond à des avancées scientifiques impressionnantes - est bien moins grandiose que ce qu'il laisse paraître.

En effet, « suprématie quantique » indique la capacité d'un ordinateur quantique à effectuer un calcul qu'un ordinateur classique est incapable d'effectuer. Cependant, le calcul choisi pour l'expérience menée par Google peut dans les grandes lignes être résumé à la simulation d'un ordinateur quantique. Simplifié ainsi à « nous avons construit un ordinateur quantique qui simule un ordinateur quantique plus facilement qu'un ordinateur classique », l'accomplissement semble moindre. L'ordinateur quantique peut donc actuellement être parfois meilleur qu'un ordinateur classique, mais pas encore plus utile.

En ce qui concerne l'utilisation des deux algorithmes quantiques redoutés par les cryptographes (ceux de Shor et de Grover [Shor 1999, Grover 1997]), l'ordinateur quantique tarde encore à faire ses preuves. Par exemple en 2018, l'ordinateur quantique savait au mieux factoriser 4 088 459 [Dash 2018]. Ce chiffre pouvant s'écrire sur 22 bits, on est encore bien loin des 1024 ou 2048 bits des modulos des chiffrements RSA. Néanmoins, certains se posent la question alarmiste « L'apocalypse quantique est-elle pour demain ? » [SemiColonWeb 2019].

Les avis sur la menace quantique pesant sur les systèmes cryptographiques sont alors tout sauf unanimes. Certains voient RSA cassé dans un assez court terme, d'autres peut-être dans quelques dizaines d'années et encore d'autres signalent que ce ne sera peut-être jamais le cas, à cause des contraintes de températures ou de pureté de matériaux utilisés [Buffoni 2022, de Leon 2021]. Malgré tout, il est certain que les efforts de recherche dans l'ordinateur quantique se multiplient aujourd'hui, ne serait-ce que pour leur capacité de distribution quantique de clés, qui peut être faite avec un niveau de sécurité arbitrairement haut. Il convient donc au moins par précaution d'être prêts à l'arrivée d'un ordinateur quantique pouvant effectuer efficacement des attaques sur nos moyens de chiffrement actuels.

Les efforts les plus concrets sur le sujet sont menés par le NIST qui a entraîné la communauté cryptographique globale dans la recherche des meilleures primitives post-quantiques pour une standardisation. Les standards nommés en résultat de ce processus seront très certainement dans un premier temps intégrés aux toolkits cryptographiques en tant que recommandation, puis en tant qu'option par défaut puis en tant qu'obligation au fur et à mesure que la menace quantique se fera sentir de plus en plus proche.

L'avantage de ce genre de compétition internationale est le respect le plus diligent du principe de Kerckhoffs, l'idée étant que si la communauté des experts cryptographiques dans sa plus grande partie n'arrive pas à trouver de vulnérabilités sur un système, il y a d'autant moins de chances qu'une vulnérabilité soit trouvée

par un groupe indépendant malveillant. La première phase de la compétition élimine ainsi les candidats exposés aux vulnérabilités les plus évidentes et permet de faire avancer les preuves de réduction des hypothèses de sécurité dans des problèmes supposés durs.

Une fois que seuls des candidats supposés résistants restent en lice, on fusionne éventuellement ceux conceptuellement très proches afin de réduire leur nombre, puis on se concentre sur leurs performances. En effet, un standard cryptographique ayant vocation à être utilisé de nombreuses fois par jour sur presque tous les appareils connectés de la planète se doit de consommer le moins de ressources possibles, en termes de temps de calcul, de bande passante et d'espace mémoire.

Pour rendre l'évaluation plus uniforme, le NIST a fixé quelques architectures cibles sur lesquelles les primitives seront évaluées. Par exemple pour l'optimisation logicielle, on étudiera les performances sur un processeur générique et micro-contrôleur ARM. L'optimisation matérielle est aussi importante, les candidats sont encouragés à présenter des résultats d'implémentation sur un accélérateur FPGA [NIST 2019].

Comme déjà évoqué, il y a à l'heure actuelle un premier KEM qui a été standardisé (CRYSTALS-KYBER [Schwabe 2021]) et quelques candidats restants ont été intégrés à un quatrième round de compétition [NIST 2022a]. Il y a plusieurs raisons au fait que contrairement aux précédents efforts de standardisation, celui-ci mènerait à plusieurs standards. Premièrement, pour la résilience face à l'ordinateur quantique, CRYSTALS-KYBER et les candidats restants ne se reposant pas sur la difficulté à résoudre des mêmes problèmes, il est possible que l'un soit résolu mais pas les autres. La redondance permettra alors de s'adapter à des « surprises » dans les années à venir. Secondement, ces primitives ont des caractéristiques très différentes qui leur confèrent des avantages et des inconvénients les rendant appropriées à des situations elles aussi très différentes. Afin de satisfaire au mieux le plus de contraintes d'utilisations possibles, il est intéressant d'avoir du choix dans les primitives selon le scénario.

Les candidats restants dans la standardisation des KEM sont basés sur les codes correcteurs d'erreurs (Classic McEliece, BIKE [teambikesuite.org 2022], HQC [Aguilar Melchor 2017]) et sur les isogénies de courbes elliptiques (SIKE [Azarderakhsh 2017]). La table 4.1 présente une comparaison simplifiée des candidats et du nouveau standard. Des quatre candidats restants, Classic McEliece peut être considéré le plus sûr, notamment en vertu de son ancienneté. Du nom de son créateur, la primitive a été présentée en 1978 en tant que compétiteur de RSA, mais la taille des variables de l'algorithme était inadaptée aux ordinateurs de l'époque. En comparaison aux autres primitives finalistes, Classic McEliece propose la plus petite taille de chiffrés, ce qui est une propriété très intéressante dans les scénarios ou l'on cherche à limiter l'utilisation de bande passante, ou si celle-ci est coûteuse (typiquement pour les applications spatiales). Cependant, comme cela peut être vu dans la table 4.1, Classic McEliece souffre d'une clé publique qui est bien plus grande que celle de ses compétiteurs et d'un temps de génération de clé publique bien plus long. Les points de comparaison plus proches, les primitives HQC et BIKE s'en distinguent car elles utilisent des codes correcteurs structurés, ce qui permet d'améliorer les performances et de diminuer le coût de stockage de la clé. Cependant, ils introduisent un doute supplémentaire sur l'impact de la structure de

Primitive	Classic McEliece	BIKE	HQC	SIKE	CRYSTALS KYBER
Public Key	261 120	3 115	3 024	330	800
Secret Key	6 452	5 223	3 064	374	1 632
Ciphertext	128	3 083	6 017	346	768
KeyGen	25.73	19.20	17.60	22.49	14.51
Encrypt	15.48	16.83	18.32	23.20	15.14
Decrypt	17.73	20.67	19.24	23.29	14.77

TABLE 4.1 – Caractéristiques et performances des primitives du round 4 sur un Intel Core i3-10110U de 2019 ; 2 x 2100MHz. Tailles en octets, temps de calcul exprimés en log_2 du nombre de cycles. Jeux de paramètres pour une sécurité Catégorie I (AES128). 32 Octets de secret partagé. Données issues de [Bernstein 2021]. CRYSTAL-KYBER est ajoutée pour comparaison.

la clé sur sa sécurité.

En plus de ses deux avantages principaux (la sécurité et la taille du chiffré), Classic McEliece exhibe aussi un niveau très haut de parallélisme arithmétique. Cela crée un intérêt très fort quant à son implémentation sur des accélérateurs matériels dédiés ou encore sur des architectures émergentes ayant de fortes capacités de calcul parallèle. Parmi celles-ci, des architectures proposent du calcul en mémoire (Processing in Memory ou PIM en anglais), en permettant à des espaces de stockage tels que les caches ou la RAM d'effectuer des calculs avec des opérandes bien plus larges que ceux que l'ALU du processeur pourrait prendre en charge. De plus, les modifications nécessaires pour intégrer du calcul en mémoire à un processeur peuvent être minimales en surface de puce. Ce type de technologies a maintes fois été prouvé efficace pour réduire non seulement le temps de calcul, mais aussi les mouvements de données et la consommation d'énergie d'une application. Un exemple très en vogue vers lequel le calcul en mémoire est souvent dirigé est celui des réseaux de neurones et du machine learning en général dans lequel un haut niveau de parallélisme est très apprécié [Aga 2017, Eckert 2018, Fujiki 2019].

> A l'opposé de calculs séquentiels, des calculs parallélisables ont la propriété que le calcul de l'un ne dépend pas du résultat du précédent. Plusieurs opérations parallèles peuvent alors être effectuées soit dans le désordre, soit être distribuées sur plusieurs ordinateurs en même temps, soit dans notre cas être effectuées toutes en même temps si l'architecture de l'ordinateur nous le permet.

Il est donc dommage que la standardisation ne prenne actuellement pas en compte de telles architectures émergentes. Cela entraîne un manque de visibilité sur l'efficacité réelle des primitives sur le matériel sur lequel elles seront certainement exécutées au cours de leur durée de vie (celle-ci étant voulue la plus longue possible). Afin de ne pas ignorer des synergies entre les primitives et les architectures d'exécution (à l'image de la synergie entre le parallélisme de Classic McEliece et celui fourni par le calcul en mémoire), il est important d'inclure des discussions multi-niveaux

dans l'intérêt d'une comparaison des performances équitable sur le long terme, et d'un co-design plus poussé entre les primitives et les architectures.

Dans ce chapitre, nous allons donc proposer la première évaluation de primitives post-quantiques sur une architecture comportant du calcul en mémoire et notamment en cache. Nos résultats montreront qu'en comparaison à une implémentation sur un processeur classique, le temps de calcul de la génération de la clé peut être réduit de 59% (voire jusqu'à 92% pour la clé publique seule) et le temps de chiffrement peut être réduit de 76% avec l'algorithme actuellement utilisé dans la primitive, mais jusqu'à 98% si on adapte l'algorithme pour profiter plus de l'architecture en stockant la clé publique de manière transposée.

Pour une comparaison juste et plus éclairée, nous allons aussi proposer des résultats dans une architecture qui inclut des calculs vectoriels. Toutes les expériences ont été réalisées grâce à un simulateur de puce RISC-V personnalisée, qui inclut à la fois un module de vectorisation, un cache permettant du calcul en mémoire, et des compteurs de performance matériels adaptés pour analyser précisément le comportement du coeur de processeur et du cache. A notre connaissance, c'est le seul simulateur à l'heure actuelle qui permet à la fois l'utilisation d'opérations vectorielles et en mémoire. De plus, le simulateur est publiquement disponible, et présenté de manière à ce que nos expériences soient facilement reproductibles [Migliore 2022].

> Au cours de l'exploration de ce sujet, nous avons commencé par envisager l'utilisation de simulateurs existants pour réaliser nos expériences. Grâce à l'aide précieuse de Y. Mao et L. Laffargue, nous avons pu utiliser Chipyard pour réaliser nos premières expériences. Les résultats étaient très prometteurs, ce qui a confirmé notre intuition sur cette problématique. Cependant, ce simulateur était à la fois très limité dans la variété des expériences que nous aurions pu mener et trop lent pour permettre une méthode scientifique rigoureuse dans le temps limité d'une thèse (L'exécution du code de référence, sans optimisations avait pris deux mois !).
> Les recherches de ce chapitre auraient alors été impossibles sans le travail formidable de V. Migliore, et de son expertise sur RISC-V. Le simulateur nous a ouvert beaucoup de dimensions supplémentaires dans notre étude, à la fois la vectorisation, différents niveaux d'optimisation, et une reconfigurabilité beaucoup plus efficace de l'architecture.

La suite de ce chapitre se compose donc de la façon suivante. La section 4.2 présente le calcul en mémoire et son utilisation en cryptographie. La section 4.3 présente les algorithmes de Classic McEliece et comment il est possible de l'accélérer grâce à du calcul en cache. La section 4.4 est dédiée à la présentation du simulateur, des expériences et des résultats qui en sont issus. Ensuite, la section 4.5 est une discussion sur l'applicabilité des optimisations que nous avons apportées à Classic McEliece sur les primitives des rounds 3 et 4. La section 4.6 présente nos pistes d'évaluation de ces changements sur les fuites par canaux auxiliaires et comment l'évolution des architectures pourrait amener à réduire celles-ci. Finalement, la section 4.7 apporte des conclusions sur notre travail.

4.2 Calcul en Mémoire : Background et Atouts en cryptographie

4.2.1 Background

En 1965, Gordon Moore postule dans « Cramming more components onto integrated circuits » que « *The complexity for minimum component costs has increased at a rate of roughly a factor of two per year* » et que « *there is no reason to believe it will not remain nearly constant for at least 10 years* » [Moore 1998]. Cette hypothèse a été vérifiée bien plus longtemps que ce qu'il avait prévu et jusqu'en 2020 les progrès en miniaturisation des transistors et de réduction des coûts de production semblaient suivre une logique log-linéaire.

Cependant, dans une architecture Von-Neumann classique, cette progression semble atteindre une certaine limite, se heurtant à trois obstacles (« walls ») [Siegl 2016] :

— The Memory Wall : Si la fréquence des processeurs augmente, celle des mémoires est bien plus lente. Si ce delta est trop grand, l'accès aux données crée un goulot d'étranglement des temps de calcul.

— The Bandwidth Wall : Pour des raisons identiques, il en va de même pour la capacité de transfert des données entre les différents niveaux de cache. Des compromis sont recherchés entre la latence et la bande passante des bus de données afin de maximiser l'utilisation du coeur.

— The Power Wall : La limite peut-être la plus connue est celle due aux besoins de dissipation de chaleur liée à l'augmentation des fréquences et à la diminution de la taille des transistors. [Kumar 2015] établit en 2015 que les limites physiques permettront au maximum à la loi de Moore de durer jusqu'en 2050.

En 2022, Jensen Huang, CEO de NVIDIA, prétend que « *Moore's Law's dead... It's completely over* » [Witkowski 2022]. Bien que très fortement controversée pour le moment, son homologue concurrent d'Intel prétendant l'inverse quelques jours après [Machkovech 2022], cette affirmation force la réflexion sur les futurs moyens d'amélioration des performances de calcul. Une solution qui vient à l'esprit est de concevoir l'architecture afin de minimiser l'impact des blocages (Walls) identifiés. Le « Memory Wall » et le « Bandwidth Wall » étant dus aux disparités entre les éléments de stockage et les éléments de calcul, on peut donc envisager soit d'intégrer plus de mémoire dans le coeur de processeur, soit d'intégrer du calcul dans la mémoire.

Afin de maximiser les effets tout en minimisant les coûts d'une telle transformation, des technologies de calcul en mémoire (PIM) font leur apparition. Ils ont pour objectif d'adapter des zones de stockage pour leur permettre des opérations de calcul basiques pour le moins de surface de puce supplémentaire possible. Parmi les conséquences immédiates de ces transformations : moins de déplacements de données jusqu'au coeur, un parallélisme plus fort et ainsi des économies à la fois en temps de calcul mais aussi en consommation énergétique (et donc en chaleur à dissiper).

La valeur de ces procédés a été démontrée plusieurs fois, avec des architectures variées, qu'elles soient simples ou multi-coeur, et que la nouvelle capacité de calcul

ait été donnée au(x) cache(s), ajoutée à proximité du cache, ou à la mémoire principale. Les résultats expérimentaux montrent à la fois une stabilité suivant les exigences industrielles de robustesse et des gains impressionnants en performance et en consommation, avec des résultats d'autant plus flagrants que l'application choisie est parallélisable.

Dans ce chapitre nous nous concentrerons sur la technologie qui nous semble la plus « légère » : le *bit-line computing*. Pour pouvoir intégrer du bit-line computing, une mémoire SRAM doit être modifiée de manière à avoir des capteurs verticaux (*sensors*) en plus d'activateurs horizontaux. Ainsi, lorsqu'une ou plusieurs lignes de la mémoire seront activées, les voltages qui traversent les capteurs verticaux peuvent permettre de calculer le résultat d'un `nor` ou d'un `and` et de stocker celui-ci dans une ligne avec le statut « write-enabled ».

Nous avons choisi d'étudier cette technologie car elle peut s'appliquer à un cache, et comme elle n'implique que des changements minimaux, elle aura plus de chances d'intégrer nos ordinateurs, a minima plus rapidement que des technologies demandant plus de changements.

Le concept du bit-line computing intégré à un cache est notamment porté par le papier Compute Cache [Aga 2017], qui propose l'intégration de calcul dans un cache pour une surface supplémentaire de puce de 8% uniquement. Des portes logiques au bout de chaque colonne de la mémoire (au bout de chaque capteur vertical) permettent toutes les opérations binaires basiques dont la copie ou la comparaison. Cela permet une gestion des données plus intelligente, et qui demande peu ou pas de travail de l'ALU, et qui permet un meilleur débit pour ces calculs simples. Les données ont moins besoin de se déplacer, cela limite ainsi le problème du « Memory Wall ». De plus un parallélisme est possible à un niveau encore plus élevé que celui du calcul vectoriel, avec des opérandes qui selon la taille du cache pourraient aller jusqu'à 8KB.

Les applications identifiées et évaluées par les auteurs du papier sont faites pour profiter de ces avantages : compter les mots dans un fichier texte, appliquer des opérations de masquage bit à bit (utile dans le traitement d'image par exemple), multiplier des matrices de bits... Toutes ces tâches partagent comme limitations le débit du processeur et l'afflux des données vers celui-ci. Les résultats sont excellents : Compute Cache donne en moyenne sur ces opérations un débit multiplié par 54 et une réduction en consommation énergétique de 70 à 90% [Aga 2017].

Ce design a depuis été adapté et amélioré pour prendre en compte les besoins aux applications de réseaux de neurones. Pour effectuer les additions et multiplications nécessaires, ainsi que des réductions modulaires, des opérations de calcul sur des opérandes verticaux *bit-array computing* sont ajoutées. Cela nécessite un stockage supplémentaire en fin de colonne, mais l'augmentation de surface totale du processeur reste inférieure à 2%, pour une augmentation de parallélisme flagrante. Cela crée un besoin d'avoir des opérandes parfois écrits horizontalement et parfois verticalement. Pour gérer plus efficacement ces changements, ce modèle incorpore des *Transpose Gateway Units*, petits éléments de mémoire accessibles dans deux directions.

La limitation principale est la gestion de l'emplacement des données en mémoire. Le bit-line-computing nécessite l'alignement des opérandes en mémoire. Cela requiert un développement très méticuleux, et pour certaines applications, peut entraîner des

contraintes de conception, des variables étant adaptées aux dimensions du matériel permettant de meilleures performances et une gestion simplifiée. Progressivement, une partie des auteurs de Compute Cache travaille à masquer la gestion fine de la mémoire pour faciliter le travail de développement [Fujiki 2019] et faire profiter plus d'applications d'un cache doté de capacités de calcul.

4.2.2 Definitions et Notations

Dans cette sous-section, nous listons plus formellement les notations associées aux opérations de cache. En particulier, nous utilisons celles de [Aga 2017]. De même la « configuration de référence » qui sera utilisée tout au long du chapitre est une configuration de très petit cache capable de calcul (configuration aussi issue de [Aga 2017]). Elle servira à illustrer les ordres de grandeurs minimaux pour les composants doués de calculs en caches. Cette configuration est illustrée avec toutes les définitions dans la figure 4.1.

Cache : Un *cache* est une unité de mémoire permettant le stockage éphémère de données provenant de la mémoire centrale. Le cache est divisé en lignes de cache qui stockent des lignes de données (généralement 64 octets) copiées depuis la mémoire centrale. Beaucoup plus petit que celle-ci, il a pour enjeu de prévoir les données nécessaires au processeur pour en faciliter l'accès.

Set : Dans les caches modernes, les lignes de caches sont regroupées dans des *sets*. Le cache peut décider librement quelles données devraient rester dans un set ou être flushées. En général on décide de garder les données les plus récentes pour favoriser la réutilisation de données déjà présentes dans le cache. Le set dans lequel une donnée est stockée dépend de son adresse en mémoire.

Tag : Le *tag* est un identifiant unique utilisé par le cache pour distinguer des données stockées dans plusieurs lignes d'un même set. Le tag est aussi dépendant de l'adresse de la donnée en mémoire.

Structure typique d'adressage dans le cache : Pour un cache typique, la structure des adresses est comme suit :
`| TAG | SET | OFFSET |` ou OFFSET permet d'identifier un octet parmi les autres de la même ligne. La taille (en bits) des différents champs dépend des dimensions et de la structure du cache (taille d'une ligne pour `OFFSET` et le nombre de sets pour `SET`).

Bit-line : Une *bit-line* est un ensemble de bits partageant le même capteur vertical permettant les opérations en cache. Celles-ci peuvent donc exclusivement avoir lieu entre des bits de la même bit-line.

Block Partition : Une *Block Partition* (BP) est un groupe de plusieurs sets (2 dans la configuration de référence) dont les bit-lines sont en commun. C'est donc avec l'offset l'élément d'adressage qui permet de savoir si deux données peuvent interagir dans du calcul en cache.

Bank : Une *Bank* est un groupe de Block Partitions regroupées par des contraintes géométriques d'alignement. La distinction entre Banks est physique mais n'a aucun impact logiciel ni sur les calculs en cache.

Structure d'adressage pour le calcul en cache : La structure des adresses est légèrement différente : le champ `SET` est subdivisé pour laisser apparaître les

informations de localité nécessaires pour les calculs sur des bit-lines. Les adresses ont la structure suivante, avec la taille des champs dépendant de la géométrie du cache :

`| TAG | SET IN BP | BANK | BP | OFFSET |`

Dans la configuration de référence, pour des adresses de 32 bits, les champs sont de longueurs respectives 22,1,2,1,6.

Cache Aligned Data : Deux bits sont dits alignés en cache si dans leurs adresses, les champs `BANK`, `BP`, et `OFFSET` sont identiques et qu'ils partagent la même position au sein de leur octet.

Bit-Line Operation : A notre connaissance, les opérations de calcul en cache sont uniquement des opérations binaires. Les Bit-Line Operations sont effectuées simultanément sur toutes les bit-lines de la même Bank Partition la ou les lignes activées sont définies par les adresses (`addr_a` et `addr_b`) passées en paramètre. Le résultat est aussi stocké dans une ligne de cache de la même Bank Partition où l'écriture est autorisée, ou dans un registre (`addr_dst` ou `ret`).

La même opération peut être effectuée dans un, plusieurs ou tous les BPs simultanément. Le nombre de BPs dans lesquels le calcul sera effectué en simultané est indiqué dans `n_lines` pour le décodage de l'instruction. Il s'agit d'une puissance de deux. Nous considérons les opérations suivantes :

```
CCPY(addr_dst, addr_a, n_lines)
CCMP(ret, addr_a, addr_b, nb_lines)
CAND(addr_dst, addr_a, addr_b, n_lines)
COR(addr_dst, addr_a, addr_b, n_lines)
CXOR(addr_dst, addr_a, addr_b, n_lines)
CNOT(addr_dst, addr_a, n_lines)
CSET(addr_dst, val, n_lines)
```

L'opération `CCMP` fixe le registre `ret` à 0 si les opérandes sont identiques, 1 sinon. L'opération `CSET` remplira une ligne de zeros si `val` $= 0$, et sinon avec des 1. Il ne diffère du `cc_buz` de Compute Caches que par une porte `not` conditionnelle.

Bit-Array operation : Il s'agit d'une composition d'opérations bit-line qui permet à une fonction d'être effectuée sur des entrées de taille arbitraire grâce à leur représentation via un circuit binaire. Cela signifie que tous les bits d'un même élément sont alignés en cache. On dit aussi que l'élément est stocké verticalement. Le nombre d'opérations bit-line dépend du nombre de portes du circuit binaire, mais toutes les bit-lines peuvent être impliquées en même temps. On supposera deux opérations bit-array principales pour la somme et la multiplication (on peut supposer des variantes pour un calcul avec retenue ou modulaire).

```
C_ADD(addr_dst, addr_a, addr_b, size, n_lines)
C_MUL(addr_dst, addr_a, addr_b, size, n_lines)
```

Le niveau de parallélisme atteignable entraîne un meilleur débit pour des opérations de complexité binaire faible sur des nombres d'opérandes très grands et multiples de la taille d'une ligne de cache. Les opérations bit-array peuvent nécessiter un ou plusieurs bits de mémoire. Ceux-ci peuvent être pris dans la mémoire principale du cache, mais pour accélérer et faciliter les calculs, ils peuvent aussi être placés juste à côté des portes logiques des opérations bit-line, au bout des

capteurs verticaux, dans une zone appelée *Column Peripheral*, qui ne contient que les variables temporaires de calculs en bit-array, et qui est donc plus facile à gérer que la mémoire du cache.

Transpose Gateway Units : Une mémoire SRAM dotée de moyens bi-dimmensionels d'accès aux données. Cela permet de charger des données écrites horizontalement et de les lire verticalement ou vice-versa. Ainsi des calculs bit-array en cache et des calculs classiques du processeur peuvent être alternés sur les mêmes données avec une perte de temps minimale à l'exécution.

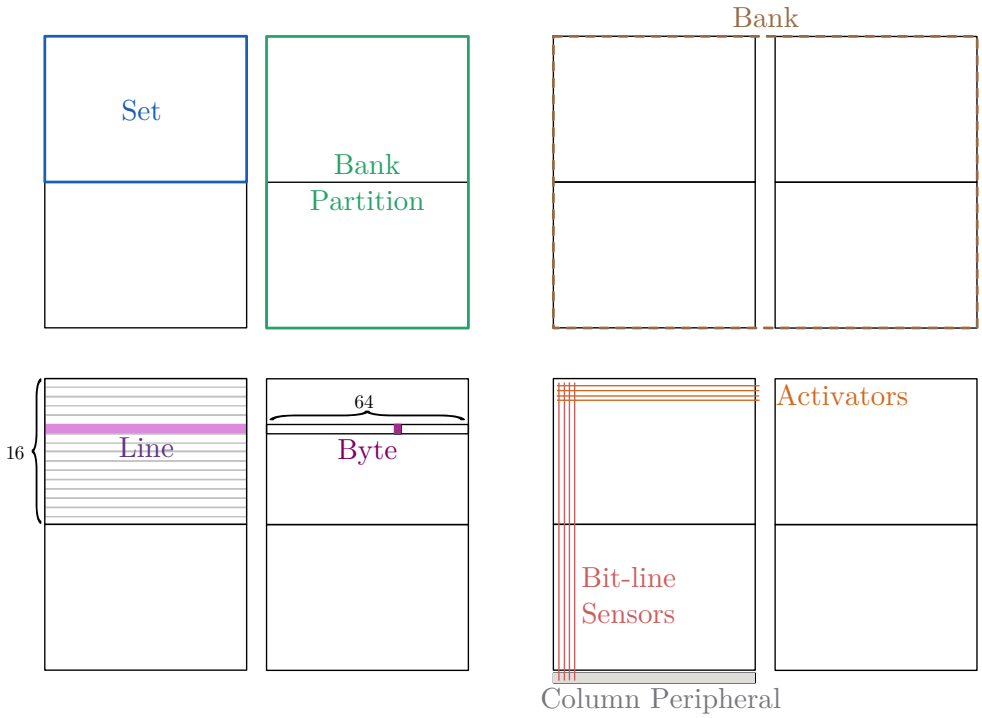


FIGURE 4.1 – Exemple de géométrie d'un cache L1 adapté aux calculs en mémoire.

Pour aider à visualiser les notions de bit-packing, d'alignement en cache et de représentation verticale des données, celles-ci sont illustrées dans la figure 4.2. Cette dernière contient différentes représentations d'une matrice 10 par 10 d'éléments de 12 bits. Ceux-ci étant normalement stockés sur 16 bits, on observe un gain de 25% d'espace si on bitpacke les données (certains éléments de la matrice ne sont plus alignés aux bits aux octets mémoire). Si on aligne les données en cache, on peut faire des opérations en cache entre les lignes de la matrice, par exemple, un `xor`. Si on stocke les données verticalement on peut faire n'importe quel calcul entre deux lignes de la matrice (succession d'opérations binaires, comme les opérations bit-array). Cependant, si on peut remarquer que pour une matrice aussi petite cette représentation peut ne pas être utile, pour une matrice 4096 par 10, le nombre d'opérations en cache serait le même, et aucun espace mémoire serait perdu par l'alignement.

> Écrits sur 12 bits, 33 et 11 sont souvent représentés sur quatre octets (`0x00`, `0x21`, `0x00`, `0x0B`) alors qu'ils peuvent l'être sur trois (`0x02`, `0x10`, `0x0B`).

4.2.3 Calcul en Mémoire pour la Cryptographie

En plus d'une optimisation purement algorithmique, trois approches sont généralement suivies et documentées dans l'objectif d'améliorer les performances des primitives cryptographiques en prenant en compte l'architecture :

1. Optimiser l'algorithme en se servant des caractéristiques de l'architecture sans changer celle-ci (adapter les variables à la taille des registres, les opérations à l'Instruction Set...).
2. Concevoir un accélérateur matériel spécialisé pour l'exécution d'une primitive préalablement optimisée algorithmiquement.
3. Utiliser une approche nommée « co-design » cherchant à faire coïncider au maximum la primitive et les caractéristiques de l'architecture (nouvelles instructions, accélérateurs pour des parties stratégiques du calcul, adaptation de l'algorithme à ces capacités...)

Cependant, ces approches ne remettent jamais en question l'architecture Von Neumann Classique et hors de celle-ci, les alternatives sont très rarement explorées dans la littérature, en particulier les architectures de calcul en mémoire.

Ces approches ayant peu d'impact sur la surface de puce mais permettant une forte économie de temps et d'énergie dans plusieurs scénarios, la cryptographie devrait s'en emparer et les intégrer dans les processus de conception, comme l'ont été les opérations vectorielles. De plus, les calculs en mémoire présentent des avantages vis-à-vis des co-processeurs : ils sont adaptables à des changements de jeux de paramètres ou de primitives avec le temps et ne sont pas réservés à des utilisations cryptographiques.

Dans le contexte de la cryptographie, la méthode de calcul en mémoire qui semble être la plus adaptée est le calcul en cache. En effet, dans de nombreuses primitives les opérations logiques et arithmétiques peuvent être nombreuses et variées et une proximité à l'ALU est désirable, les interactions avec le coeur étant relativement fréquentes. C'est la raison pour laquelle nos expériences ont été effectuées dans un cache, mais elles pourraient être répliquées dans une RAM ayant les mêmes capacités de calcul.

4.3 Considérations de design pour l'Accélération de Classic McEliece

4.3.1 Le Cryptosystème Classic McEliece

Le cryptosystème Classic McEliece est la plus vieille primitive post-quantique, proposée en [McEliece 1978] et basée sur les codes correcteurs d'erreur. L'utilisation première de ceux-ci est d'encoder des données qui transiteront via un canal bruité

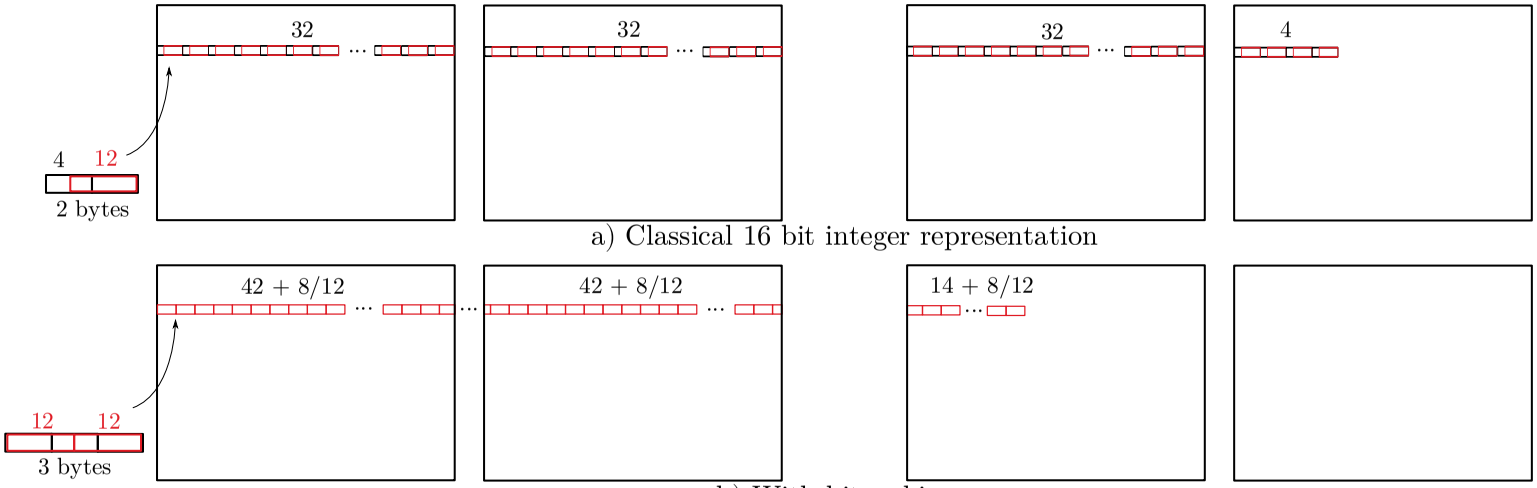


FIGURE 4.2 – Plusieurs façons de représenter en mémoire une matrice 10×10 d'éléments de 12 bits. Les grand rectangles noirs sont les Sets de la mémoire, en rouge les données.

et d'améliorer les chances que le décodeur puisse identifier les erreurs et retrouver le message correct.

> Le code correcteur d'erreurs le plus simple est celui de la triplication : 0 est encodé `000` et 1 est encodé `111`. Le décodeur est alors le choix de la majorité : `111`, `110`, `101` et `011` ayant plus de `1` que de `0` seront décodés comme 1 et `000`, `001`, `010` et `100` ayant plus de `0` seront décodés 0. Si le canal a 25% de chances de changer un bit, un récepteur a normalement 75% de chances de recevoir le bon bit, mais cela monte à 84% avec la triplication. Selon le taux d'erreurs, des codes optimisent le ratio longueur du message / longueur du mot de code, ou d'autres caractéristiques et se construisent selon des algorithmes très variés. Je remercie au passage J.C. Deneuville qui aura pris du temps pour m'expliquer l'utilisation des codes correcteurs d'erreurs en cryptographie post-quantique au début de ma thèse.

L'idée du cryptosystème Classic McEliece est que si le décodeur est gardé secret et qu'il ne peut pas être déduit de l'encodeur, cela transforme les codes correcteurs d'erreurs en fonctions à trappe : pour chiffrer, l'émetteur encode son message et ajoute le maximum de bits de bruit que le code puisse débruiter. Le destinataire possédant le décodeur est le seul à pouvoir enlever ces erreurs avec une probabilité forte.

Classic McEliece a été conçu à base de codes de Goppa, pour la vitesse de l'algorithme de décodage. Jusqu'à présent, ce cryptosystème ne présente pas de vulnérabilité. Quelques années après, Niederreiter proposa son propre cryptosystème basé sur les codes de Goppa binaires [Niederreiter 1986], mais les deux ont ensuite été prouvés équivalents [Li 1994].

> Les codes de Goppa sont des généralisations des codes de Reed-Solomon, dont le principe général est de sur-échantillonner un polynôme. Par exemple pour définir un polynôme de degré 5, il faut 6 points. Si on envoie donc 8 points à la place, on pourra toujours retrouver le polynôme de départ si 2 points contiennent des erreurs.

L'algorithme KeyGen de génération des clés se compose ainsi :

1. `sk_gen()` : La clé secrète contient trois parties : la première est générée en prenant au hasard un polynôme $g \in \mathbb{F}_q[x]$ où $q = 2^m$ avec $m = 12$ ou $m = 13$ selon le jeu de paramètres, la deuxième est une liste de n éléments $\alpha_1 \cdot \alpha_n$ de $\mathbb{F}_q$ et la dernière est une chaîne de caractères $s \in \mathbb{F}_2^n$
2. `pk_gen()` : La clé publique est générée ainsi :
   (a) **Initialization** : Tri des éléments de la clé secrète pour obtenir le support L.
   (b) **Roots** : Application du polynôme g à chaque élément de L.
   (c) **Inversion** : Calcul de l'inverse $g(L)$ et stockage en tant que inv.
   (d) **Zeroing** : Passer tout l'espace mémoire de mat à 0.
   (e) **Filling** : Faire t fois ce qui suit : (illustration figure 4.11a)

— Transposer successivement chaque groupe de 8 éléments de inv (m bits) en m octets de mat, de manière à ce que les éléments de $\mathbb{F}_q$ soient stockés verticalement.

— Définir inv comme la multiplication coefficient par coefficient de inv et L

(f) **Gaussian** : mat est réduite à sa forme systématique (Fig 4.3), grâce à l'algorithme du pivot de gauss. Si cela échoue car elle n'admet pas de forme systématique, redémarrer KeyGen du début.

(g) **Copy** : Copier la partie droite de mat (tout sauf le carré identité, qui est inutile) afin de la bitpacker (la rendre contiguë en mémoire, sans padding). Ce résultat est retourné comme étant la clé publique.

3. `controlbits()` : Cela change la représentation de la liste des α de la clé secrète pour améliorer les temps de déchiffrement. L'algorithme se sert d'un réseau de Benes pour trouver des paires d'éléments de $\mathbb{F}_q$ à échanger de manière à ce que les α_i soient les premiers éléments de la liste.

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & a_{1,m} & \cdots & a_{1,n} \\ 0 & 1 & \cdots & 0 & a_{2,m} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & a_{m,m} & \cdots & a_{m,n} \end{pmatrix}$$

FIGURE 4.3 – Matrice en forme systématique (identité dans le carré principal).

L'algorithme Encrypt de chiffrement se compose ainsi :

1. `gen_e()` : Générer un vecteur d'erreurs rempli de zéros sauf à un nombre fixe d'endroits distribués aléatoirement. Ce sera le clair, ou une fois hashé la clé échangée.
2. `syndrome()` : Multiplier la matrice de la clé publique par le vecteur d'erreurs et renvoyer le résultat en tant que chiffré.

L'algorithme Decrypt de déchiffrement code se compose ainsi :

1. `support_gen()` : se sert des bits de contrôle de la clé secrète pour retourner le support L.
2. `synd()` : grâce au polynôme g de la clé secrète, au support L et au chiffré, renvoyer le syndrome s.
3. `bm()` : l'algorithme Berlekamp-Massey renvoie le polynôme minimal pour s.
4. `root()` : renvoie les racines du polynôme sur le support L. Le vecteur d'erreurs contient des 1 aux positions de ces racines. Si ce vecteur est de poids (nombre de 1) correct, il s'agit du clair, on peut le hasher pour trouver la clé partagée.

Le goulot d'étranglement de KeyGen est la génération de la clé publique, et en particulier l'étape Gaussian : la réduction de la matrice en forme systématique. En comparaison, le temps de génération de la clé secrète est négligeable (voir section 4.4).

De plus l'algorithme de systématisation peut aboutir à un échec. Il convient alors de détecter ce problème au plus tôt pour éviter des calculs inutiles. On va alors calculer la trigonalisation du carré principal de la matrice en gardant une matrice de transformation en mémoire, puis lorsqu'on s'est assuré de l'absence de zéros sur la diagonale on procède à la transformation du reste de la matrice.

L'algorithme d'élimination Gaussienne est le meilleur algorithme connu pour cette tâche et les performances sont donc seulement liées au débit du calcul des opérations arithmétiques. Or, les opérations du pivot de Gauss sont extrêmement parallélisables, étant donné que la même multiplication (le même `and` dans le cas d'une matrice de bits) doit être effectuée sur tous les éléments d'une ligne de la matrice. Cela est parfaitement compatible avec les modules de vectorisation de certains processeurs (Intel AVX, ARM NEON...).

4.3.2 Accélérer Classic McEliece avec le Calcul en Cache

Dans cette sous-section, nous allons évoquer les différents éléments de Classic McEliece qui peuvent être modifiés pour accommoder au mieux les opérations en cache. On évoquera en premier les jeux de paramètres du cryptosystème.

4.3.2.1 Jeux de paramètres et tailles des opérandes

Dans le code de référence de Classic McEliece soumis au NIST, cinq jeux de paramètres sont proposés, avec des niveaux de sécurité et des performances associées différentes. Un récapitulatif de ceux-ci est fourni en table 4.2. On retrouve trois paramètres qui varient selon la sécurité désirée, desquels sont déduits la taille des clés :

- `GFBITS` : La taille des éléments de $\mathbb{F}\_q$, ou q est donc égal a $2^{\texttt{GFBITS}}$.
- `SYS_N` : La longueur d'une ligne de la matrice.
- `SYS_T` : Le nombre d'éléments de $\mathbb{F}\_q$ par colonne.
- `PK_NROWS` : `SYS_T`×`GFBITS` le nombre de bits d'une colonne.
- `PK_NCOLS` : `SYS_N`−`PK_NROWS` le nombre de colonnes de la matrice une fois qu'on a enlevé le carré principal pour la stocker de manière plus compacte.

Comme la taille de la matrice varie beaucoup d'un jeu de paramètres à l'autre, une discussion sur le stockage de la matrice en mémoire est essentielle. Dans les architectures actuelles, la meilleure façon de manipuler la matrice de manière efficace est de l'avoir bitpackée. Cela présente l'avantage de minimiser l'espace mémoire nécessaire au calcul. Cependant, les bits d'une même colonne de la matrice ne partagent ainsi pas la même bit-line et aucun calcul en cache n'est possible entre eux. Pour le rendre possible il faudra alors ajouter du padding à la fin de chaque ligne pour faire coïncider tous les éléments de la même colonne de la matrice sur la même bit-line.

En général, des petits caches comme des L1 seront trop petits pour y mettre la matrice entière. En revanche les calculs étant effectués ligne par ligne, il suffit de s'assurer que plusieurs d'entre elles puissent rentrer en même temps en cache. Ce sera le cas même pour des caches de tailles minimales. Par exemple pour un cache

LD1 de 16 Kb (256 lines de 64 bytes), une ligne de la matrice prend 16 lignes (6.25 %) du cache et il en faut au moins quatre pour les opérations de systématisation (deux lignes de cache, un masque et une variable auxiliaire). Dans notre configuration de référence minimale (issue de [Aga 2017]), les performances ne sont pas limitées par la taille du cache mais par la vitesse de chargement des données dans le cache.

Des paliers d'efficacité sont atteints lorsque la taille d'une ligne est un multiple de la taille maximale d'un opérande en cache (nombre de BPs du cache multiplié par la taille d'une ligne). Par exemple avec 8 BPs, le jeu de paramètres « 8192128 » (dans lequel une ligne de matrice est écrite sur exactement 8 lignes de cache) n'a même pas besoin de padding et les opérations en cache ont un parallélisme maximal. Pour « 348864 », il faudra un peu de padding, et on n'exploite pas tout le parallélisme. Le jeu de paramètres le moins adapté est « 460896 », qui aura besoin de 7 lignes de cache de padding et aura besoin de 2 opérations, la seconde sur relativement peu de données.

Sachant par exemple qu'il faudra le même nombre de calculs en cache par ligne pour le jeu de paramètres « 348864 », que les lignes fassent 3488 ou 4096 bits, on pourrait chercher si des lignes plus longues mais moins nombreuses ne permettraient pas de garantir la même sécurité (tout en gardant `GFBITS` bas et en dessous de 16 pour un traitement efficace du polynôme dans le coeur de processeur et minimiser `controlbits()`.

Parameter Set	Matrix dimensions		Cache lines used		PK size (bits)	Secu. Cat.
	Rows	Columns	Row	Matrix		
348864	768	3488	7	5 232	2 088 960	1
460896	1248	4608	9	11 232	4 193 280	3
6688128	1664	6688	14	21 736	8 359 936	5
6960119	1508	6960	14	20 500	8 378 552	5
8192128	1664	8192	16	26 624	10 862 736	5

TABLE 4.2 – Jeux de paramètres pour la soumission de Classic McEliece à la standardisation du NIST et les conséquences sur le nombre de lignes de cache utilisées en supposant des lignes de 64 octets.

Il faut aussi noter que des processeurs haut de gamme peuvent souvent être équipés avec un cache L3 très large (10 MB et parfois même plus). Dans ce genre de cas, il est possible d'y stocker la matrice entière. Cela annule la limitation de la vitesse du chargement des données dans le cache (accès à la mémoire) et ne laisse plus que le temps des opérations arithmétiques. Dans le reste du chapitre, nous supposerons l'utilisation d'un cache petit comme défini plus tôt en sachant qu'il s'agit ainsi d'un pire cas, et que plus de capacités de calcul en mémoire (plus de BPs, plus de lignes par set) ne feront qu'améliorer les performances.

4.3.2.2 KeyGen

sk_gen() :

La première partie de KeyGen, `sk_gen` ne peut pas particulièrement être accélérée avec du calcul en mémoire car elle est principalement composée de l'expansion d'une graine aléatoire et d'opérations entre des éléments de $\mathbb{F}_q$ qui ne bénéficient ainsi pas d'opérandes plus grands avec des jeux d'instructions classiques. Ce n'est pas en réalité un problème puisque `sk_gen` représente un temps de calcul négligeable dans KeyGen, Nous ne nous tentons donc pas de l'optimiser.

pk_gen() :

Dans `pk_gen`, comme évoqué précédemment, nous devons changer la représentation de *mat* en mémoire pour pouvoir se servir de calcul en cache. En langage C, l'attribut `__attribute__((aligned (CACHE_ROW)))` permet de ne pas gérer manuellement le padding.

Initialization, Roots, et **Inversion :**

Ces trois étapes créent la première ligne de la matrice (inv) alors que les éléments de $\mathbb{F}_q$ sont encore stockés horizontalement. Identiquement, comme ces étapes ne sont pas parallélisables et de durée négligeable (comme confirmé dans nos résultats expérimentaux), nous ne chercherons pas à les optimiser.

Zeroing :

La remise à zéro de la mémoire est très directe avec le calcul en cache, en réduisant le nombre d'opérations d'un facteur de $r/(l_{size} \times n_{BP})$, où l_s est la taille d'une ligne de cache, n_{BP} est le nombre de lignes calculées en parallèle et r la taille d'un registre.

```
for (i = 0; i < PK_NROWS; i++)
    for (j = 0; j < SYS_N/8; j++)
      mat[i][j] = 0;
/* Can be optimised as */
for (i = 0; i < PK_NROWS; i++)
    CSET(mat[i],0,8);
```

Ainsi, les données des 8 lignes de cache dont les adresses coïncident avec l'adresse de `mat[i]` sauf dans leurs 9 derniers bits (6 bits d'offset, 1 de BP, et 2 de bank dans notre exemple) vont être chargées dans le cache (et progressivement verrouillées en attendant la présence de toutes les autres), puis mises à zéro simultanément. Tout cela a évité le chargement dans des registres et a laissé le processeur libre de faire d'autres calculs pendant ce temps.

Filling :

Il n'y a pas de parallélisme immédiat dans cette étape, le calcul des $\alpha_j^{(i-1)}/g(\alpha_j)$ ne profitent pas d'opérandes plus grands car ils requièrent des multiplications consécutives dans $\mathbb{F}_q$. Comme précédemment, ce n'est pas un souci puisque les temps de calcul sont faibles.

Gaussian :

C'est l'étape la plus gourmande en temps de calcul, mais c'est heureusement aussi celle avec le plus de parallélisme. Le pivot de Gauss (adapté pour être effectué en temps constant selon les exigences de sécurité side-channel du NIST) consiste à créer un masque de 1 ou de 0 selon la comparaison de l'élément diagonal avec celui

à éliminer. Ce masque est appliqué par un `and` sur la ligne de l'élément diagonal et ce résultat temporaire permet d'appliquer un `xor` à la ligne de l'élément à éliminer. Comme évoqué plus haut, on ne calcule pas vraiment sur toute la ligne, dans un premier temps on élimine le triangle inférieur, puis on ne procède sur le reste de la matrice que si elle admet bien une forme systématique.

Le goulot d'étranglement vient du fait que la complexité de l'algorithme et celle de son implémentation diffèrent d'un facteur $\texttt{SYS\_N}/r$, où r correspond aux nombres de bits traités en parallèle (par exemple, la taille des registres). Ainsi, en utilisant des opérandes suffisamment larges pour contenir une ligne entière, on retire ce facteur de ralentissement. On le voit dans le code ci-dessous par la disparition d'une boucle sur la longueur de la ligne.

```
for (k = row + 1; k < PK_NROWS; k++){
    mask = mat[row][i] ^ mat[k][i];
    mask >>= j;
    mask &= 1;
    mask = -mask;

    for (c = 0; c < SYS_N/8; c++)
        mat[row][c] ^= mat[k][c] & mask;
}
/* Can be optimised as */
for (k = row + 1; k < PK_NROWS; k++){
    bit = mat[row][i] ^ mat[k][i];
    bit >>= j;
    bit &= 1;
    CSET(mask, bit); //The mask is stored in 8 cache lines instead of a
     byte
    CAND(aux, mask, mat[k]); //The auxiliary value is also stored in 8
    cache lines
    CXOR(mat[row], mat[row], aux);
}
```

La précaution que prend le code optimisé en retardant le calcul après la vérification que la matrice est systématisable n'est plus nécessaire, le temps de calcul sur le carré principal étant le même que sur une ligne entière.

Sur ces caches à correspondance directe, des problèmes peuvent survenir lorsque plusieurs opérandes et/ou la destination d'un calcul en cache partagent un même emplacement dans le cache. Pour éviter ces collisions et augmenter le débit de calcul, nous proposons de fixer l'espace du cache pris par les deux variables « permanentes » (aux et $mask$ dans le snippet de code précédent). Ces variables sont utilisées pour toutes les lignes de la matrice. En contrepartie, nous adaptons les adresses des lignes de la matrice pour qu'elles ne correspondent jamais à ces deux lignes. Cela améliore les performances en évitant d'avoir parfois à faire les opérations de manière classique et tous les cache miss que cela aurait causé. Cependant, cela augmente le padding nécessaire et l'espace total de la matrice d'un ratio de $2/number\_cache\_ways$.

```
uint64_t buf[ 1 << GFBITS ];
unsigned char mat[ PK_NROWS ][ SYS_N/8 ];

/* Is replaced by */

//cache ways is the number of lines in a BP
#define CACHE_SIZE (CACHE_LINE_SIZE*CACHE_WAYS*CACHE_BANK_PARTITIONS)
#define ROW_LEN (CACHE_LINE_SIZE * CACHE_BANK_PARTITIONS)
//Extra cache rows needed to store the matrix without collisions
#define EXTRA_ROWS (2*(1-(PK_NROWS + (CACHE_WAYS-3)) /(CACHE_WAYS -2)))
//real row number, once padding rows are accounted for
#define ROW(i) ((i) + 2*((i) / (CACHE_WAYS-2)))

unsigned char vars[ (2 + PK_NROWS + EXTRA_ROWS)*ROW_LEN+ sizeof(gf)*(
    SYS_T+1 + SYS_N*2)] __attribute__((aligned (CACHE_SIZE)));
unsigned char * aux = vars ;
unsigned char * mask = vars + ROW_LEN ;
unsigned char * mat = vars + 2*ROW_LEN;
```

Copying :

Augmenter la taille de la matrice avec du padding ne ralentit pas le calcul puisque le temps d'accès reste le même. Le résultat en sortie doit être une clé bitpackée, la plus petite possible. Le calcul en cache ne permet pas ce genre de déplacement, transversal aux bit-lines. Le code de référence copie registre par registre la matrice, sauf les bits du carré principal devenu l'identité (ils n'amènent aucune information au déchiffrement). Lorsque la matrice est paddée pour le calcul en cache, le nombre d'opérations de copie restent inchangées.

Sur toutes ces opérations, le calcul en cache donne donc une réduction du nombre total d'opérations arithmétiques et en mouvement de données, en échange de plus de mémoire requise pour le calcul.

4.3.2.3 Encrypt

Premièrement, la création du vecteur d'erreurs ne bénéficie pas d'opérandes plus larges, il s'agit de remplir un vecteur de zéros, sauf pour un nombre précis d'éléments choisis aléatoirement. Cette manipulation d'aléatoire n'implique aucun calcul dont le niveau de parallélisme permette de lui faire bénéficier de l'utilisation de calcul en mémoire.

Dans le code de référence, la clé publique est étendue afin de rajouter la matrice identité à gauche de la matrice clé publique, comme elle avait été enlevée en fin de `pk_gen()` pour rendre la clé plus compacte. Le vecteur d'erreurs devient le clair, qui multiplié par la matrice donnera le syndrome qui sera le chiffré. La multiplication est faite de manière classique, avec une multiplication et une accumulation des éléments ligne par ligne et octet par octet (8 éléments multipliés en même temps). Ensuite un calcul de parité sur l'octet accumulateur donne le bit correspondant du syndrome. L'algorithme est illustré dans la figure 4.4.

Les variations que nous proposons et dont nous allons étudier les performances sont les suivantes.

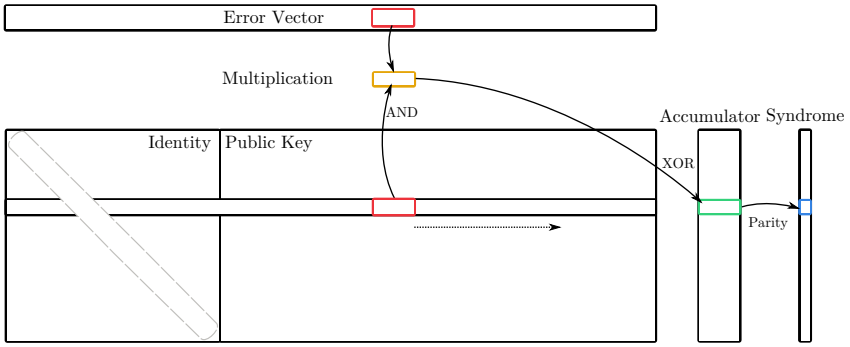


FIGURE 4.4 – `Algorithme syndrome()` de référence.

NoId :

La matrice n'a pas à être étendue pour être de nouveau dans sa forme systématique. A la place, nous nous contenterons de copier les premiers octets (correspondants à l'identité) du vecteur d'erreurs directement dans le résultat et l'accumulation commencera à partir de ces valeurs. Du temps devrait être gagné car les lignes sont plus courtes à parcourir. Nous avons appliqué cette modification à toutes les autres variations. L'algorithme est illustré dans la figure 4.5.

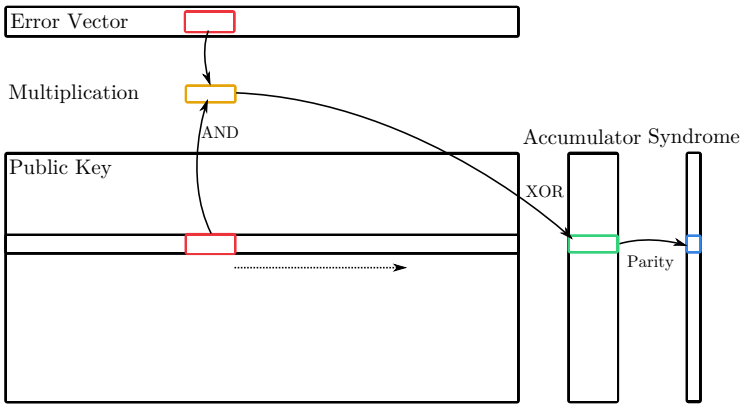


FIGURE 4.5 – `syndrome()` sans partie identité.

With Cache Operations :

L'objectif est de ne plus parcourir les lignes, mais d'effectuer les multiplications sur toute une ligne simultanément. Il n'y aura alors plus que l'accumulation et le calcul de parité à faire octet par octet. Pour faire cela, il faut cependant que le vecteur d'erreurs et les lignes de la matrice soient alignés en mémoire. Cela signifie que la matrice ne peut plus être bitpackée et donc qu'elle prendra plus de place en mémoire. L'algorithme est illustré dans la figure 4.6.

```
1 for (j = 0; j < SYS_N/8; j++)
2   b ^= row[j] & e[j];
3 /* is replaced by */
4 CAND(row, row, e);
5 for (j = 0; j < SYS_N/8; j++)
6   b ^= row[j];
```

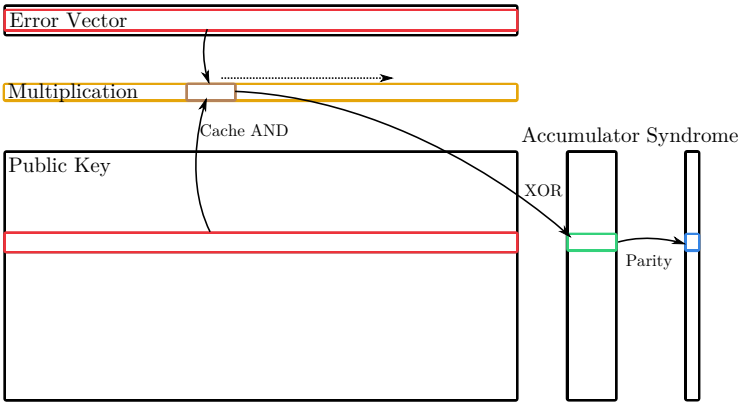


FIGURE 4.6 – `syndrome()` avec opérations en cache.

Transposed : Calculer les multiplications colonne par colonne plutôt que ligne par ligne permet de faire une accumulation dans un bit plutôt que dans un octet, ce qui enlève le besoin de calcul de parité. Pour permettre cette méthode de calcul, nous supposerons que la matrice est donnée dans sa forme transposée. Pour la ligne i de la matrice transposée (colonne i de la matrice d'origine), il n'y a plus qu'à fixer un octet selon le i^{eme} bit du vecteur d'erreurs (255 si c'est un 1, 0 sinon), multiplier et accumuler directement le résultat. L'algorithme est illustré dans la figure 4.7.

Transposed with cache operations :

Pour augmenter le parallélisme, il est possible d'effectuer les multiplications sur plus d'une ligne de la matrice transposée en même temps. Par exemple, avec le jeu de paramètres le plus petit, les lignes de la matrice transposée font 768 bits. C'est-à-dire qu'ils peuvent être contenus dans deux lignes de cache, et donc deux BPs. Dans le petit cache que nous avons pris en exemple, qui a 8 BPs, on peut alors avoir en même temps quatre lignes de la matrice chacune à l'adresse du début d'une paire de BPs. On peut fixer séparément les quatre éléments de multiplication selon les bits du vecteur d'erreur, et ensuite procéder à la multiplication et l'accumulation simultanément pour les quatre lignes. Il faudra cependant penser à la fin à agréger les accumulateurs de chaque paire de BPs dans un seul résultat. L'algorithme est illustré dans la figure 4.8.

Une transposition classique durant Encrypt pourrait cependant complètement défaire toutes les économies de temps faites sur la multiplication matrice vecteur. Ainsi, si le gain en performances le justifie, on pourrait envisager de « préparer » la clé publique de manière transposée dans le KeyGen, de la même manière que `controlbits()` prépare la clé secrète, ou alors on pourrait transposer la clé lors de sa réception, avant le chiffrement dans des scénarios dans lesquels ce temps n'a pas à être minimisé.

Une architecture de calcul en mémoire plus évoluée comme celle de Neural Cache [Eckert 2018] et qui inclurait des Transpose Gateway Units pourrait rendre la transposition plus efficace et enlever l'implication du coeur dans ce processus. Dans ces architectures, le calcul de manière transposée devrait donc être plus rapide et plus intéressant.

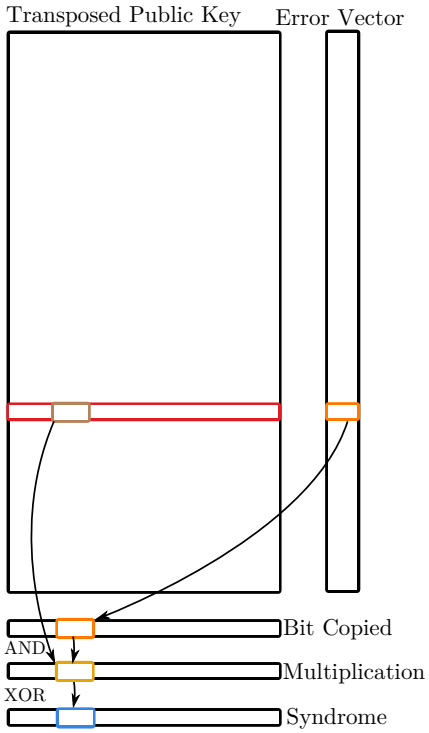


FIGURE 4.7 – `syndrome()` avec matrice transposée.

4.3.2.4 Decrypt

L'objectif du déchiffrement est de retrouver le vecteur d'erreurs à partir du chiffré. Pour cela il cherche le mot de code le plus proche du chiffré dans le code de Goppa défini dans la clé secrète, il calcule la différence avec le chiffré et vérifie si celle-ci a bien le poids (nombre de uns) que doit avoir le clair.

Des accélérateurs matériels comme des Transpose Gateway Units pourraient se révéler très utiles pour le déchiffrement dans la mesure où la routine `support_gen` utilise de multiples transpositions pour déduire le support des bits de contrôle.

La seule autre partie où une accélération grâce à des Bit-Line Operations est clairement visible est le zeroing des grandes variables à l'instar de r, qui fait la taille d'une ligne de la matrice et de laquelle le mot de code le plus proche est recherché. Toutes les autres opérations sont sur de petits opérandes tels que des éléments de $\mathbb{F}_q$.

Nous n'évaluerons pas ces gains minimes de gestion de données, car ceux-ci sont attendus de manière transversale sur toutes les applications, et ne sont pas spécifiques à la cryptographie. Les articles de calcul en mémoire présentent eux-mêmes ces résultats.

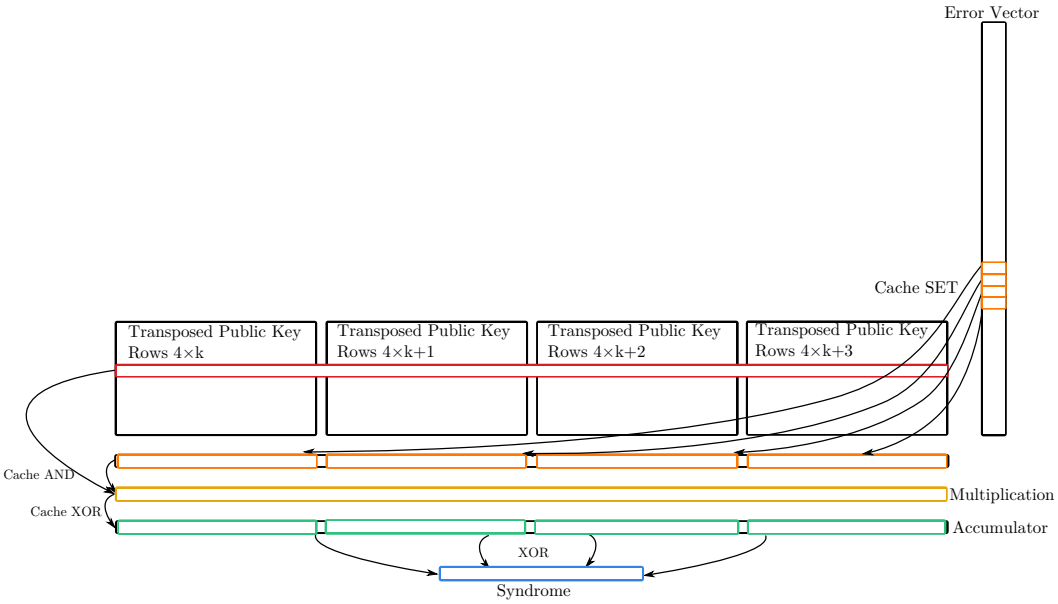


FIGURE 4.8 – `syndrome()` avec matrice transposée et opérations en cache.

4.4 Expérimentation

4.4.1 Banc d'essais RISC-V

Avant de pouvoir évaluer l'efficacité des optimisations que nous proposons, nous avons étés confrontés au défi de trouver un banc d'essai adapté à nos expériences. Il faut à la fois qu'il implémente le coeur RISC-V, les extensions de calcul vectoriel, une hiérarchie de cache, et soit adaptable à des opérations en cache. La communauté RISC-V est très active [Brown 2022] et il est possible de trouver pour chacun de ces points un simulateur adapté, même pour les calculs vectoriels, supportés par le simulateur Spike par exemple. Cependant, à notre connaissance, il n'existe aucune implémentation qui prenne en compte le calcul en mémoire ou en cache.

Une solution que nous avions envisagée dans un premier temps était l'intégration des opérations en cache à un outil déjà existant. Cela est rendu difficile car les opérations doivent être intégrées à un niveau micro-architectural pour permettre une gestion spécifique de la géométrie du cache (en bank partitions), la gestion des Bit-Line Operations et des conflits qu'elles peuvent amener, le verrouillage des données dans le cache et les transferts des données entre la RAM et le cache.

Après un premier essai infructueux qui a consisté à intégrer un contrôleur cache dans le simulateur Chipyard, nous nous sommes penchés sur la conception de notre propre simulateur en langage C, qui implémente un RISC-V mono-coeur, avec support des jeux d'instructions RV32I et RV32M et une partie de l'extension de calcul vectoriel (les opérations nécessaires pour l'évaluation de McEliece). Il intègre une hiérarchie de caches à correspondance directe, à géométrie entièrement configurable, et des instructions pour toutes les Bit-line Operations basiques (celles listées dans [Aga 2017]). Pour avoir une estimation précise du nombre de cycles de calcul, nous avons intégré la gestion de quelques éléments micro-architecturaux, tels que la

gestion des aléas du flot de contrôle, le contrôleur mémoire entre le cache et la RAM, ainsi que les micro-opérations internes exécutées pendant les instructions de calcul en cache.

Le simulateur est disponible sur GitLab [Migliore 2022]. Nous y avons mis des implémentations de la version de référence de Classic McEliece, une version accélérée par vectorisation et une version se servant du calcul en cache. Il est mis en place et documenté de manière à ce que les expériences détaillées ci-dessous puissent être reproduites le plus facilement possible.

Afin de mieux comprendre les interactions entre le RISC-V, le cache et la RAM, nous avons intégré des compteurs de performance matérielle :

- Compteurs d'instructions
  - Nombre d'instructions
  - Nature des instructions
- Compteurs de cycles
  - Cycles d'horloge globale
  - Cycles RISC-V
  - Cycles d'opérations cache
  - Cycles dûs aux aléas du flot de contrôle
- Compteurs Mémoire
  - Nombre de transferts entre le cache et la RAM
  - Utilisation du tas et de la pile
  - Allocations Mémoires

La liste complète des fonctionnalités est accessible dans la documentation sur le dépôt.

De plus, l'approche qui a été utilisée pour l'interconnexion des composants matériels est très flexible pour permettre des reconfigurations très rapides. Par exemple, si un cache n'est pas utile dans une expérience, le RISC-V peut être directement connecté à la RAM avec très peu d'édition de code. Cela est aussi vrai s'il y a besoin d'évaluer les performances d'un accélérateur matériel dans une architecture complète. De fait, le contrôleur qui nous permet les calculs en cache est vu comme un accélérateur matériel par le simulateur.

4.4.2 Résultats d'Implémentation

Nous avons fait les expériences en simulant trois architectures différentes : une première version générique, dotée d'aucun accélérateur, une version disposant de calcul vectoriel, et une version disposant de capacités de calcul en cache. Pour la version du RISC-V avec calcul vectoriel, nous avons adapté l'implémentation AVX2 fournie dans la soumission Classic McEliece du NIST pour nous servir de nos instructions. Les changements apportés pour la version avec calcul en cache étaient présentés en section 4.3.2.

Pour refléter un cas d'utilisation dans lequel les capacités de calcul en mémoire sont minimales et le parallélisme de la primitive est aussi minimal, nous avons décidé

de baser nos expériences sur le plus petit jeu de paramètres de Classic McEliece « 348864 » (voir table 4.2) et de configurer le cache comme celui proposé dans Compute Cache [Aga 2017]. Rappel : lignes de 64 octets, 16 lignes par set, 2 sets par BP, 2 BP par bank, 4 banks au total.

Cependant, pour aussi pouvoir isoler les effets du déplacement des données de la variation du temps de calcul, nous allons aussi faire varier le nombre de lignes par set, de 16 à 1024, taille à laquelle la matrice peut entrer entièrement dans le cache, et qui nous place dans un scénario très semblable à du calcul effectué directement dans la RAM.

4.4.2.1 KeyGen

Comme évoqué dans la section 4.3, KeyGen est l'algorithme qui bénéficie le plus de calcul en mémoire. Le tableau 4.10 offre une vision globale des besoins en temps et en mémoire de chacune des routines et algorithmes de KeyGen (avec un cache de 1024 par set, optimisation GCC avec l'option -O3). On peut voir en effet que Gaussian et Zeroing sont les deux étapes qui présentent une forte accélération à l'aide du calcul en cache.

Le tableau 4.3 présente de manière plus précise le nombre de cycles de l'étape Gaussian pour des nombres de lignes par set qui varient. Nous avons aussi représenté ces données sous forme de graphes comme une fonction du nombre de lignes par set de cache, avec pour référence 16 lignes par set, pour les trois façons de conduire les opérations arithmétiques, dans la figure 4.9.

Ways	Reference	Vector	Cache
16	1 037 890 479	562 045 422	266 384 330
32	1 008 175 343	544 386 638	256 151 754
64	993 256 239	534 758 990	250 866 890
128	985 981 839	526 796 846	245 638 090
256	966 934 895	510 378 350	232 160 458
512	855 361 103	401 017 326	140 247 338
1024	793 189 167	329 896 718	17 098 761

TABLE 4.3 – Performances de l'élimination Gaussienne pour différentes tailles de cache (en cycles). Balise d'optimisation du compilateur -O3.

On peut voir que la forme des courbes est très similaire. La vectorisation divise approximativement le temps de calcul en deux par rapport à l'implémentation de référence (450 millions de cycles en moins dans toutes les configurations) et passer au calcul en cache divise encore le temps de calcul par deux (300 millions de cycles supplémentaires sont gagnés). Cette expérience est faite pour isoler la contribution du mouvement des données dans le temps de calcul. Ainsi, la différence de performance entre deux colonnes du tableau est principalement l'effet de gain de vitesse en opérations arithmétiques et la différence entre deux lignes est principalement la différence de temps passé à charger les données en mémoire.

On en conclut ainsi par exemple que dans un cache avec 64 lignes par set, 20%

du temps de calcul de Gaussian est dû au mouvement de données dans l'architecture de référence, mais passe à 38% quand les opérations arithmétiques accélèrent grâce au calcul vectoriel, et à 93% avec l'accélération due au calcul vectoriel.

Cela explique pourquoi lorsque le cache est suffisamment large pour contenir la matrice entière, l'étape Gaussian est 19 fois plus rapide avec le calcul en cache qu'avec le calcul vectoriel, et 46 fois plus rapide que le calcul de référence. On en déduit globalement que bien que le déplacement des données ait un impact similaire peu importe la méthode de calcul choisie, plus les opérations arithmétiques sont rapides plus le déplacement est un facteur limitant.

Ainsi, pour de petits caches, le goulot d'étranglement sera encore le temps de mouvement des données vers le cache (Bandwidth Wall). Cependant, comme la géométrie de cache a été choisie pour représenter un pire cas et que les résultats sont déjà remarquables (4×), on sait que grâce à du calcul en mémoire dans des éléments plus grands tels que des parts de la RAM, les effets seront exacerbés.

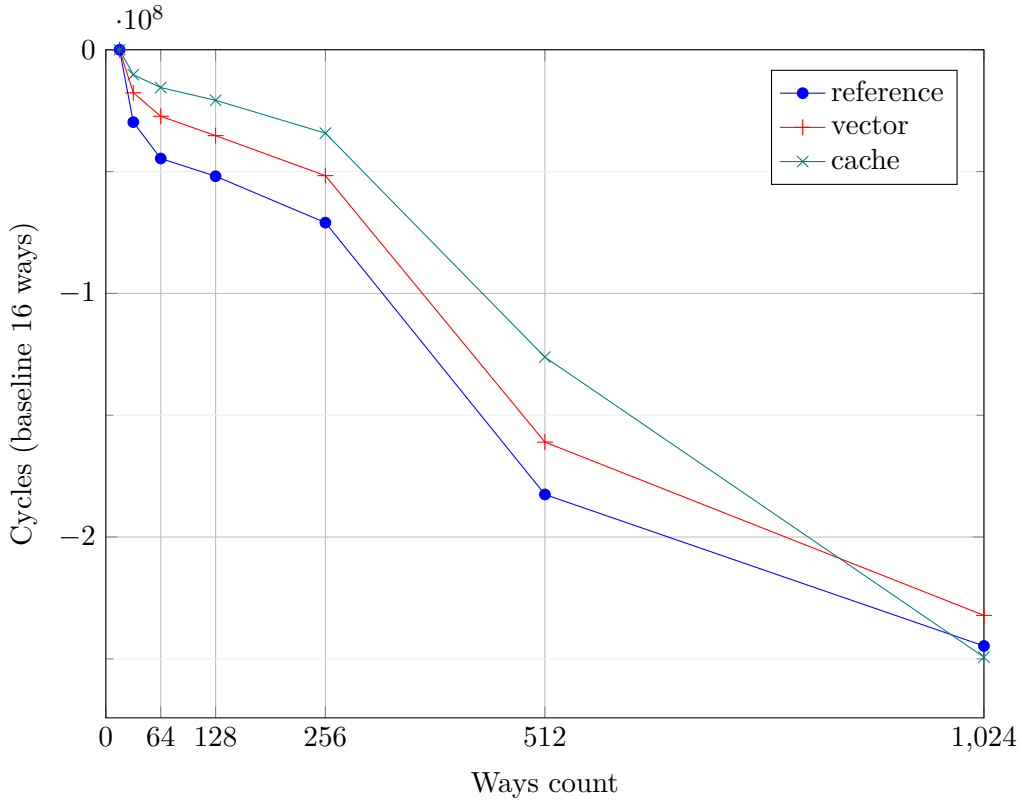


FIGURE 4.9 – Évolution du nombre de cycles de calcul en fonction du nombre de lignes de cache par set pour l'élimination Gaussienne. La base pour chaque architecture est la configuration à 16 lignes.

Le tableau 4.10 montre que la phase Zeroing aura pu être accélérée d'un ratio de 20, et le passage de la matrice sous forme systématique d'un ratio de 46.4. Au total le calcul de `pk_gen` a été 12.6 fois plus rapide. En proportion des temps de calcul de `pk_gen`, l'étape Gaussian qui représentait 94.4% en représente désormais 25.7%.

Measure	Cycles			RAM Used (Bytes)			RAM Allocated (Bytes)		
Architecture	Reference	Vector	Cache	Reference	Vector	Cache	Reference	Vector	Cache
SKgen	26 683 052	26 683 052	26 683 052	42 549	42 549	42 549	51 042	51 042	51 042
Initialization	5 053 223	5 167 998	5 053 223	64 582	64 582	64 586	424 106	424 090	1 007 782
Roots	14 089 820	14 089 820	14 089 820	14 114	14 114	14 114	14 146	14 146	14 146
Inversion	2 005 620	2 005 620	2 005 620	7 100	7 100	7 100	14 238	14 238	14 238
Zeroing	335 772	338 076	16 614	334 848	334 852	0	334 850	390 922	0
Filling	25 421 134	25 448 910	27 876 551	348 804	348 804	348 800	381 726	390 926	440 068
Gaussian	793 189 167	329 896 718	17 098 761	334 860	344 069	73 728	381 726	390 934	392 802
Copy	456 603	458 908	508 421	522 244	522 248	552 244	691 554	700 746	1 142 442
PKgen	840 551 389	377 406 084	66 463 458	667 722	676 931	667 718	691 678	700 894	1 242 574
Controlbits	449 397 559	449 397 559	449 424 663	56 948	56 948	56 948	90 854	90 854	90 854
KeyGen	1 316 632 015	853 486 710	542 756 873	691 375	700 584	733 323	691 662	700 584	1 242 558

FIGURE 4.10 – Génération des clés : comparaison des performances sur trois architectures et leurs jeux d'instructions : RISC-V standard, avec opérations vectorielles et des opérations en cache. Balise d'optimisation du compilateur -O3.

En contrepartie, l'étape de remplissage Filling est passée de 3% à 41.8%. Ainsi, contrairement à ce que nos connaissances a priori laissaient penser, cette étape n'est plus négligeable dans la génération de la clé publique.

Pour l'optimiser, il faudrait cependant repenser entièrement l'étape. Pour bénéficier d'une architecture de calcul en bit-line, il faut augmenter le parallélisme des opérations. Les deux types d'opérations effectuées sont des multiplications d'éléments de $\mathbb{F}_q$ de 12 ou 13 bits, écrits sur deux octets, et la transposition de ceux-ci pour les placer dans la matrice en vertical. Notre proposition est de commencer par transposer `inv` et L pour que les éléments soient écrits verticalement (prérequis de toute opération bit-line) et d'effectuer les multiplications grâce à des multiplications bit-array.

Il faudrait cependant vérifier expérimentalement cette proposition. En effet, il est possible que la profondeur du circuit binaire de la multiplication d'éléments de 12 bits et donc le nombre d'opérations bit-line nécessaires contrebalance le parallélisme des 3488 multiplications effectuées en parallèle. Pour cela il faudrait une architecture de type Neural Cache [Eckert 2018].

Les optimisations faites sur l'opération de Zeroing font que cette dernière reste très négligeable dans le calcul : de 0.04% à 0.02% plutôt que à 0.5%. Le gain de temps sur cette opération confirme la conclusion de [Aga 2017] en montrant que cette modification de l'architecture amènera des gains de performances pour toutes les applications, même si ce n'est qu'en rendant la gestion de la mémoire plus efficace. L'utilisation très basse de la RAM dans les étapes de Zeroing et Gaussian est due au fait que les principales variables sont en cache.

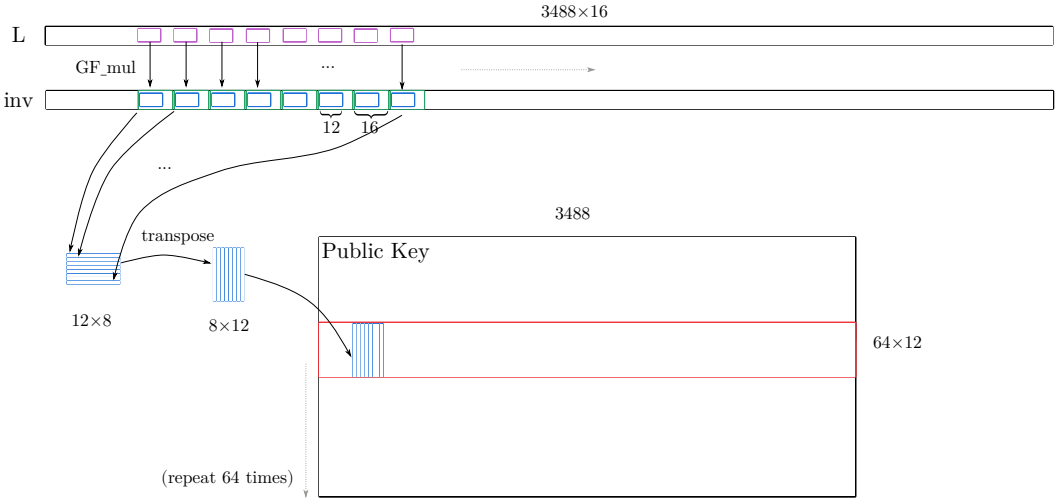
Problématiquement, `controlbits()` représente désormais 82.6% de KeyGen, et est donc un nouveau goulot d'étranglement et donc une cible pour de prochaines optimisations. La structure de l'algorithme de `controlbits()` alterne entre deux types d'opérations. Certaines pourraient être parallélisées si les opérandes venaient à être écrits verticalement comme par exemple :

```
1 for (x = 0;x < n;++x) B[x]=(A[x]<<16)|(B[x]&0xffff);
```

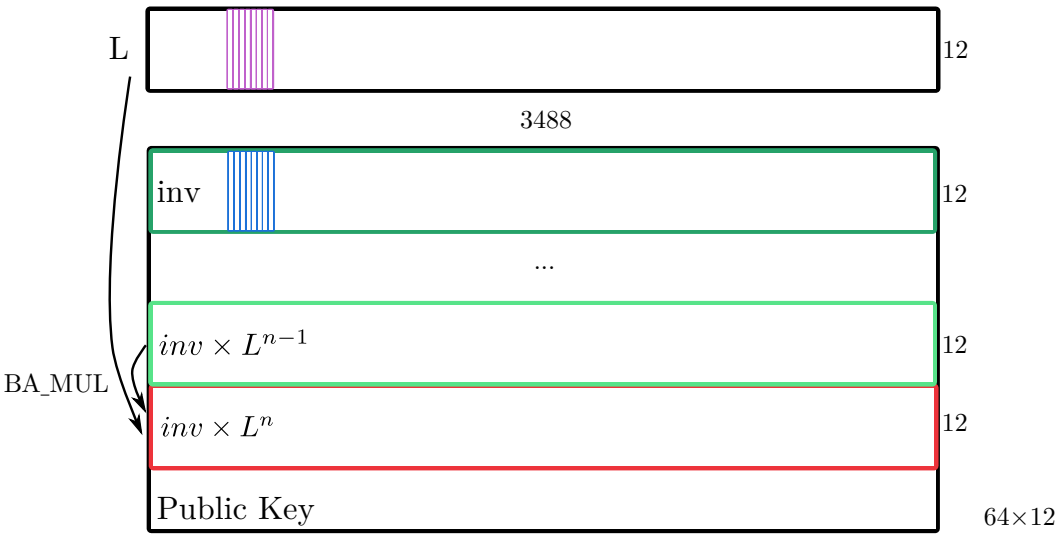
De telles opérations devraient être très rapides avec des opérations en cache. Cependant, d'autres opérations nécessitent d'échanger de multiples éléments de place, par exemple pour les trier, ce qui ne peut pas être fait efficacement à cause des contraintes du calcul bit-line.

La solution proposée dans la littérature, est comme attendu un co-processeur, car il a été prouvé que ceux-ci permettent une implémentation très performante de ces opérations d'échange et de tri [Kohútka 2017]. Pour se tenir à notre approche généraliste, on peut imaginer évaluer la capacité des mémoires à accès multi-dimensionnel pour résoudre ce problème. Cependant, la taille des données qu'il faudrait transposer de multiples fois au cours du calcul pourrait se révéler bien trop grand pour atteindre des performances intéressantes. De plus, des architectures avec des mémoires à accès multi-dimensionnel larges ne semblent pas être accessibles dans un futur proche.

Par ailleurs, on peut évaluer l'impact sur les besoins d'allocation de mémoire : la matrice bitpackée du code de référence utilisait 768×3488 bits, mais pour l'alignement en cache, elle doit utiliser $768 \times 8 \times 512$ bits. L'augmentation théorique est donc de 17.4%. Cependant, nous avons ajouté des contraintes sur les adresses



(a) Algorithme Filling de référence pour le remplissage de la matrice



(b) Filling modifié pour des architectures qui permettent les calculs bit-array

FIGURE 4.11 – Filling : référence et notre proposition

pour éviter les collisions empêchant les calculs en cache. De plus, le compilateur semble vouloir séparer les opérandes de calcul en cache des autres variables et l'augmentation d'allocation mémoire mesurée est de 79.7%. Un contrôle plus précis de l'emplacement devrait pouvoir réduire ce problème.

> J'ai pu observer au cours du développement ce qui sera d'après moi la plus grande barrière à la diffusion du calcul en mémoire : les efforts supplémentaires qui doivent être mis dans le contrôle précis des variables et leur adressage en mémoire. Bien que ce problème soit identifié et que des efforts semblent être faits sur ce sujet, je ne pense pas qu'une utilisation généralisée soit envisageable sans une compatibilité bien plus forte avec les applications existantes, ou un coût d'adaptation bien plus faible.

Notre simulateur nous a aussi permis de compiler le firmware à différents niveaux d'optimisations. La comparaison entre les performances des options O0, O2 et O3 a montré que les niveaux d'optimisations plus bas donnent des meilleurs ratios de gain de performance, mais des performances moindres en valeur absolue. Cela est totalement attendu car les optimisations ont été faites au niveau algorithmique mais pas au niveau assembleur.

Malheureusement, nous n'avons pas pu nous servir de l'option Os, qui optimise l'allocation de la mémoire. En effet GCC ne sait pas comment se comporter en présence d'opérandes de calcul en cache. Si cette technologie se développait, une re-conception du compilateur pourrait être envisagée.

4.4.2.2 Encrypt

Les résultats de nos expériences sur le chiffrement sont présentés dans le tableau 4.4. On peut y voir qu'en effet, le fait de retirer le calcul de la multiplication sur la partie identité de la matrice retire 22% du temps de calcul, ce qui correspond parfaitement à la réduction de 22% de la taille des lignes de la matrice. Les variations suivantes sont basées sur celle sans l'identité. Le calcul en cache permet de retirer 67% supplémentaires du temps de calcul. Si à la place, on décide de transposer les calculs, on gagne 64% de temps de calcul. Cependant, si on transpose la matrice et qu'on répartit les lignes pour augmenter l'efficacité du calcul en cache, on arrive à un calcul 47.6 fois plus rapide (64 fois plus rapide que le code de référence).

Measure	Cycles	RAM Used (Bytes)	RAM Allocated (Bytes)
Reference	3 012 219	262 088	269 286
No Id	2 239 978	261 992	269 190
Cache	734 918	261 748	270 818
Transposed	796 970	349 208	350 426
Transposed + Cache	47 311	916	256 102

TABLE 4.4 – Performances du chiffrement pour cinq algorithmes de calcul du syndrome (en cycles). Balise d'optimisation du compilateur -O3.

Ways	Reference	Cache	T. + Cache
16	3 195 547	872 646	212 008
32	3 169 211	845 350	211 048
64	3 156 475	832 230	210 856
128	3 150 395	825 990	210 664
256	3 144 251	819 686	210 568
512	3 120 571	818 854	182 600
1024	3 012 219	734 918	47 311

TABLE 4.5 – Performances du chiffrement (en cycles) selon le nombre de lignes par set du cache. Balise d'optimisation du compilateur -O3.

Tout comme avec KeyGen, nous observons dans le tableau 4.5 que le nombre de cycles économisés par le calcul en cache est stable autour de 2.25 millions de cycles, et 700 000 cycles supplémentaires sont économisés grâce à la version transposée avec calcul en cache. La conclusion est aussi la même : plus l'algorithme est optimisé et plus les opérations arithmétiques sont rapides, plus le goulot d'étranglement devient le mouvement des données entre la mémoire et le cache et les performances sont ainsi bien meilleures dans des mémoires larges.

Cela indique clairement que si on ne trouve pas de réduction majeure du temps de calcul de l'algorithme `controlbits()` grâce au changement d'architecture, la clé publique aurait toutes les raisons d'être transposée et paddée durant KeyGen, qui est de toute façon lent (à cause de `controlbits()`). Notre simulation a montré qu'une transposition très naïve (bit à bit) coûte 29 542 317 cycles, et que le nombre de cycles gagnés dans le chiffrement est de 687 607 cycles. Ce compromis donne pour un grand cache un chiffrement 15.5 fois plus rapide, contre un ralentissement de la génération de clé de 5.4% uniquement, qui pourrait être bien moindre avec une transposition optimisée voire accélérée matériellement. De plus KeyGen ne doit être exécuté qu'une fois, contrairement à Encrypt. En suivant la même logique, la transposition devrait aussi être faite pour les architectures classiques, sans impact sur la taille de clé, le bitpacking étant toujours possible.

4.5 Applicabilité aux autres candidats du NIST

Le cryptosystème Classic McEliece est le seul candidat dans les rounds 3 et 4 du NIST qui bénéficie clairement et directement des opérations bit-line. Les autres candidats ne peuvent se servir dans leur état actuel de ces opérations que pour de la gestion générique des données (copie, nettoyage ou déplacement de données), mais pas pour des opérations arithmétiques booléennes parallélisables.

Dans un souci d'exhaustivité, nous allons chercher en détail les éléments de chacune des primitives des rounds 3 et 4 qui pourraient être adaptés pour bénéficier de ces calculs. Nous allons aussi identifier quelles modifications et améliorations du calcul en bit-line pourraient être utiles à ces primitives. Certaines sont déjà en développement (comme le bit-array computing et les Transpose Gateway Units déjà évoqués) et d'autres qui ne sont que théoriques (opérateurs de décalage ou rotation).

4.5.1 Candidats du Round 4 et CRYSTALS-KYBER

4.5.1.1 HQC : Hamming Quasi Cyclic

Tout comme Classic McEliece, HQC est basé sur les codes correcteurs d'erreurs [Aguilar Melchor 2017]. L'aspect le plus intéressant de cette primitive est que, contrairement à Classic McEliece, le décodeur du code n'est pas la clé secrète. C'est la gestion du niveau de bruit du chiffré qui permettra de faire que le décodeur ne sera utile qu'au destinataire du message.

La primitive est basée sur des codes systématiques quasi-cycliques. *Cycliques* fait référence au fait que la matrice soit circulante, ce qui veut dire que chaque ligne de celle-ci est la même que celle au dessus, mais avec une rotation des éléments vers la droite d'une colonne. *Quasi* décrit le fait que la matrice ne soit pas circulante dans son entièreté, mais seulement bloc par bloc. Tout comme la clé publique de Classic McEliece, *systématiques* veut dire que le carré principal de la matrice est l'identité (l'identité est bien une matrice circulante).

L'avantage principal de HQC, relativement à Classic McEliece, est que le format strict de la clé publique permet à celle-ci d'être considérablement plus petite étant donné que seule la première ligne de chaque bloc non-identité est nécessaire à la reconstitution de la matrice entière.

L'opération la plus importante de ce cryptosystème est la multiplication, de la matrice par un vecteur, cette fois écrit sous forme de vecteur creux. Cela est effectué une fois pendant KeyGen, deux fois pendant Encrypt et une fois de plus pendant Decrypt. Cette opération consiste en des `xor` de la première ligne de la matrice sur laquelle on a effectué un nombre de rotations équivalent à chaque position du vecteur d'erreurs creux (la majorité des éléments étant nuls, seules les positions de ceux non-nuls sont écrites). Le vecteur d'erreurs contient alors 67/101/133 éléments non nuls selon le jeu de paramètres quand chaque ligne est respectivement de longueur 24 677 / 46 747 / 70 853.

Une architecture capable de calcul en cache ne pourrait qu'aider ce processus en accélérant les opérations de xor, mais ne serait d'aucune utilité pour accélérer l'opération de rotation, qui doit être répétée pour satisfaire les exigences de programmation en temps constant. La seule modification qui pourrait accélérer cette part du calcul serait éventuellement une modification des column peripherals qui permettrait aux bits d'une colonne d'affecter les colonnes voisines. Bien que des opérateurs de décalage d'un cran semblent envisageables au sein d'une même Bank Partition voire d'une même Bank, la distance physique entre les Banks rend le décalage de grands opérandes très peu envisageable actuellement. De même, les opérateurs de rotation demandant de replacer le dernier élément à la place du premier semblent durs à créer. Mais une astuce logicielle à base de copie du vecteur permet de remplacer la rotation par un simple décalage.

4.5.1.2 BIKE

BIKE [teambikesuite.org 2022] est une troisième primitive basée sur les codes correcteurs d'erreurs. Plus précisément, elle est basée sur les codes Quasi-Cycliques à matrices de contrôle de parité modérément denses. Les deux opérations cruciales

pour cette primitive sont une inversion modulaire et une multiplication polynomiale. Nous n'avons pas trouvé de stratégie directe permettant une accélération de l'inversion modulaire. Cependant, dans sa forme actuelle, la multiplication est implémentée selon l'algorithme de Karatsuba, qui a un mode portable et un mode vectoriel. Il pourrait être envisagé de remplacer la multiplication par la même que HQC, si celle-ci était appuyée par un opérateur matériel de décalage entre des opérandes très grands (la longueur des éléments multipliés h_1 et h_0^{-1} vont de 12 323 bits à 40 973 selon le jeu de paramètres).

4.5.1.3 SIKE

La primitive SIKE [Azarderakhsh 2017] est basée sur des isogénies de courbes elliptiques de Montgomery avec le corps $\mathbb{F}_{p^2}$, où p est de la forme $2^{e_2}3^{e_3} - 1$ et est codé sur un nombre de bits allant de 434 à 751 bits selon le jeu de paramètres. Comme les additions et multiplications de points de courbes elliptiques sont des opérations de grands nombres, ce sont précisément l'inverse de ce pourquoi le calcul en mémoire est adapté (beaucoup d'opérations sur de petits éléments). Il n'y a aucun gain possible grâce à l'amélioration du parallélisme dû au calcul en cache.

4.5.1.4 CRYSTALS-KYBER

Déjà standardisée, CRYSTALS-KYBER [Schwabe 2021] est basée sur la difficulté du problème LWE (Learning With errors) dans les modules de réseaux euclidiens (problème MLWE). Elle diffère des autres candidats car son approche pour la multiplication est basée sur la NTT (Number Theoretic Transform). Elle exploite la propriété que pour deux éléments f et g d'un anneau R_q, le produit $f \cdot g = NTT^{-1}(NTT(f) \circ NTT(g))$, où $\circ$ est la multiplication coefficient par coefficient est plus rapide à calculer que la convolution classique. L'algorithme KeyGen gagne du temps en générant la clé directement dans le domaine transformé NTT.

Il semble qu'il n'y ait aucune fonction du cryptosytème CRYSTALS-KYBER qui puisse bénéficier de calcul bit-line ou bit-array, si ce n'est la multiplication coefficient par coefficient dans la routine `polyvec_pointwise_acc`, mais la transposition nécessaire pour l'accélération pourrait ralentir l'accumulation. Même avec des Transpose Gateway Units, le nombre de coefficients des jeux de paramètres actuels (256) n'est pas suffisant pour justifier l'utilisation de calcul en mémoire plutôt que l'utilisation de calcul vectoriel.

4.5.2 Candidats du Round 3

4.5.2.1 NTRU

La primitive NTRU du round 3 du processus de standardisation du NIST arrive avec deux variantes : *ntruhps* et *ntruhrss*, qui évite le re-chiffrement de validation [Chen 2021]. Les deux sont basées sur le même opérateur principal, une multiplication polynomiale et partagent les outils de manipulation des polynômes et de leurs coefficients.

Ces polynômes ont des degrés allant de $N = 509$ à 821 selon le jeu de paramètres. Les coefficients appartiennent à $\mathbb{Z}_q$ avec $log_2(q) = 11, 12$ ou 13 et sont donc écrits sur 11 à 13 bits et stockés sur 2 octets. Dans leur méthode de stockage actuel, très peu d'opérations pourraient bénéficier de calculs en bit-line, comme par exemple une parallélisation de l'application du masque, qui permet de passer à 0 les bits supplémentaires dans l'écriture sur deux octets des éléments de $\mathbb{Z}_q$, pour faire ainsi les opérations de modulo en fin d'opération.

Pour aller plus loin, il serait possible d'écrire les coefficients verticalement en mémoire. Cela débloquerait des améliorations possibles sur plusieurs sub-routines avec une architecture capable d'opérations bit-array, notamment sur les routines `poly_Z3_to_Zq`, `poly_trinary_Zq_to_Z3`, ainsi que sur les autres applications parallèles sur les coefficients.

On s'attend à ce que ces opérations aient une latence augmentée due à la profondeur de leur circuit binaire pour les opérations bit-array, mais celle-ci serait compensée sur l'algorithme complet par l'augmentation du débit (à l'image des résultats présentés dans [Eckert 2018]). Cette latence ne devrait pas être problématique dans des environnements mono-thread « in order » où le calcul est effectué sur tous les coefficients des polynômes avant de démarrer la prochaine routine. Dans le cas de NTRU, cette transformation doit être évaluée car le niveau de parallélisme (821 bits dans le meilleurs cas) ne permet pas forcément de compenser les opérations supplémentaires.

Cependant, si on combine ceci avec les opérateurs de décalage entre lignes que nous avons évoqué plus tôt, il est possible de changer l'algorithme de calcul de convolution de polynômes. Pour simplifier le code, nous allons supposer que N est un multiple de la taille d'une ligne de cache, disons 1024 (pour garder l'ordre de grandeur), et que les éléments sont stockés verticalement.

```
for(k=0; k<NTRU_N; k++){
    r->coeffs[k] = 0;
    for(i=1; i<NTRU_N-k; i++)
        r->coeffs[k] += a->coeffs[k+i] * b->coeffs[NTRU_N-i];
    for(i=0; i<k+1; i++)
        r->coeffs[k] += a->coeffs[k-i] * b->coeffs[i];
}
```

On peut transformer le code ci-dessus en celui ci-dessous avec des opérations bit-array dont les paramètres sont les adresses des données écrites verticalement, la taille verticale des données (ici 13 lignes) et le nombre de BPs nécessaires (ici 4).

```
for(k=0; k<NTRU_LOGQ; k++){
    CSET(aux->bits[k], 0, 2);
    CSET(r->bits[k], 0, 2);
}
for(i=0; i<NTRU_N;i++){
    C_MUL(aux, a, b, 13, 2);
    C_ADD(r, r, aux, 13, 2);
    for(k=0; k<NTRU_LOGQ; k++){
        CLSHIFT(a->bits[k], a->bits[k],2);
    }
}
```

Ainsi, si le débit est suffisamment amélioré, cela créerait une motivation pour réévaluer les paramètres de sécurité au profit d'éléments plus petits mais en plus grand nombre, avec un degré alors proche mais inférieur à un multiple de la taille d'une ligne de cache.

4.5.2.2 NTRU Prime

En ce qui concerne l'applicabilité des calculs en cache, la différence majeure entre NTRU et NTRU Prime [Bernstein 2016] est que dans ce dernier le modulo q comme le nom l'indique est premier plutôt qu'une puissance de 2. L'opération principale, qui est encore la multiplication polynomiale et toutes les considérations précédentes sur NTRU s'appliquent aussi ici. Cependant, les opérations modulo q ne sont plus triviales (oubli des bits de poids trop fort) et effectuer des passages au modulo dans des calculs bit-array augmenterait énormément le nombre d'opérations binaires nécessaires, ce qui irait à l'encontre du gain en parallélisme qui n'est pas si grand pour ces jeux de paramètres.

4.5.2.3 Saber

Saber [KULeuven-COSIC 2021] est une primitive basée sur le problème Learning With Rounding. Tout comme pour NTRU, les éléments principaux du cryptosystème sont des polynômes et l'opération cruciale est la multiplication de ces polynômes, nécessaire à la multiplication rapide de matrices et vecteurs de polynômes.

La multiplication est faite avec une combinaison des algorithmes Toom-Cook, Karatsuba et la multiplication classique. La transposition des données avec leur taille et nombre actuels ne permettrait aucun gain de calcul en mémoire, comme aucun opérande ne serait plus grand que le degré du polynôme c'est-à-dire 256 bits (taille plus adaptée au calcul vectoriel) et les coefficients étant écrits sur 13 bits pour tous les jeux de paramètres (c'est la taille de la matrice et du vecteur qui varie).

Il faudrait une transformation très forte de la primitive pour lui permettre de profiter de calcul bit-array, soit des paramètres beaucoup plus grands ou réévalués au profit de degrés plus grands et d'éléments plus petits.

4.5.2.4 FrodoKEM

FrodoKEM [Bos 2016] est un cryptosystème classique basé sur le problème LWE avec pour objectif la simplicité de son implémentation. La clé publique est générée via une expansion de graine aléatoire : soit SHAKE soit AES. Elle possède $n = 640/976/1344$ lignes et colonnes d'éléments de $\mathbb{Z}_q$, avec $log_2(q) = 15/16/16$, chacun étant alors stocké sur deux octets. Tout comme pour les autres primitives basées sur des variantes de LWE, l'opération la plus importante est la multiplication matrice-vecteur.

En bref rappel, KeyGen génère une grande matrice A de taille n-par-n, une matrice verticale S de taille n-par-8, une matrice d'erreurs E verticale de taille n-par-8 et calcule $B \leftarrow AS + E$. Les graines permettant de générer A et B deviennent les clés publiques et S la clé secrète. Encrypt choisit une matrice horizontale S' de taille 8-par-n, une matrice d'erreurs E'' de taille 8-par-n, et pour un message

M renvoie le chiffré $C_1 \leftarrow S'A + E'$ et $C_2 \leftarrow S'B + E'' + M$. Decrypt calcule $M' \leftarrow C_2 - C_1 S$.

Le calcul de la multiplication des éléments n'est pas possible en bit-array si l'écriture des éléments n'est pas changée de verticale à horizontale, ce qui permettrait d'effectuer la multiplication de la même manière que dans l'algorithme Encrypt transposé que nous avons proposé plus haut. Ce changement accélérerait le calcul des produits vecteur-matrice, et serait le plus efficace avec des matrices peu hautes et une grande largeur proche d'un multiple de la taille d'une ligne de cache. S'il était fait ainsi, le produit d'un élément de $\mathbb{Z}_q$ du vecteur pourrait être appliqué en parallèle à toute une ligne de la matrice avec du calcul bit-array.

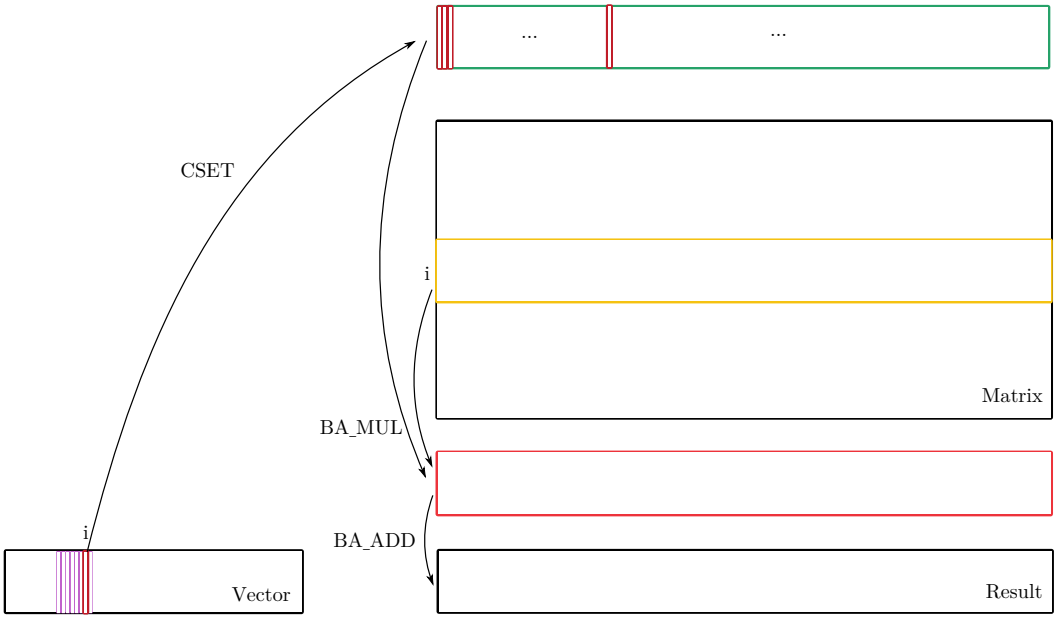


FIGURE 4.12 – Accélération de produits vecteur-matrice grâce au calcul bit-array.

Supposons que nous adoptions ce changement de représentation des éléments de A. Le produit $S'A$ du chiffrement serait accéléré, mais la génération de clé serait fortement dégradée. Une solution de contournement de ce problème serait d'avoir un algorithme d'expansion de clé qui puisse produire A directement en forme transposée, auquel cas KeyGen pourrait calculer $B^\intercal \leftarrow S^\intercal A^\intercal + E^\intercal$.

Calculé de la sorte, ce produit aussi bénéficierait du calcul bit-array, mais il nécessiterait la transposition de $B^\intercal$ a posteriori. Comme S n'est pas le sujet de calculs bit-array, il peut être généré normalement comme une matrice verticale d'éléments écrits à l'horizontale mais lu comme une matrice transposée pour garder l'accélération des produits $B^\intercal \leftarrow S^\intercal A^\intercal + E^\intercal$ de KeyGen et $C_1 S$ de Decrypt.

Dans ce scénario, pour améliorer encore l'efficacité, le compromis entre la taille de la matrice et la taille de ses éléments peut être revu pour favoriser des éléments plus nombreux et plus petits, pour augmenter le parallélisme et diminuer le nombre d'opérations binaires de la multiplication. De plus si les éléments peuvent être réduits sous le seuil critique de 8 bits, des Transpose Gateway Units peuvent accélérer encore plus efficacement la transposition.

4.5.3 Récapitulatif

De la même manière que toutes les primitives ont été par le passé revisitées afin de profiter du calcul vectoriel, elles devraient être à nouveau repensées afin de prendre en compte un accès à du calcul parallèle en mémoire. En résumé : on préfère avoir plus d'éléments si les calculs sur ceux-ci sont plus simples. Pour l'application des différentes améliorations évoquées du calcul en mémoire, elles sont regroupées sur le tableau 4.6.

Category	Primitive	Operands Too Small	Bit-Array + TGU	Shift or Rotation
Learning With Errors	KYBER	X		
	SABER	X		
	NTRU		X	X
	NTRU Prime		X	X
	FrodoKem		X	
Error Correcting Codes	HQC			X
	BIKE			X
Isogenies	SIKE	X		

TABLE 4.6 – Améliorations du calcul en mémoire qui permettent des optimisations des primitives du round 3. Rouge : pas d'optimisation sans changement de la primitive, vert : améliorations d'architectures déjà présentes dans l'état de l'art, orange : propositions d'améliorations.

4.6 Évaluation des Impacts sur la Sécurité des Canaux Auxiliaires

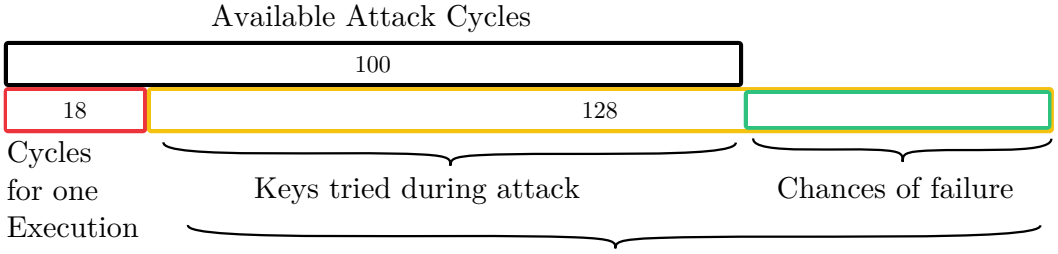
4.6.1 Attaques par canaux auxiliaires et modèle du poids de Hamming

On peut observer les caractéristiques physiques des objets qui font des calculs cryptographiques afin d'en déduire les valeurs probables des variables durant l'exécution de ces algorithmes. L'objectif est que certaines informations sur ces variables puissent aider à retrouver la clé. L'attaque par canaux auxiliaires certainement la plus connue concerne RSA, où le temps de calcul de l'algorithme square-and-multiply dépend de la valeur de l'exposant, qui est la clé. A l'image de cette attaque, nous nous concentrerons dans cette section sur les attaques passives et non-intrusives.

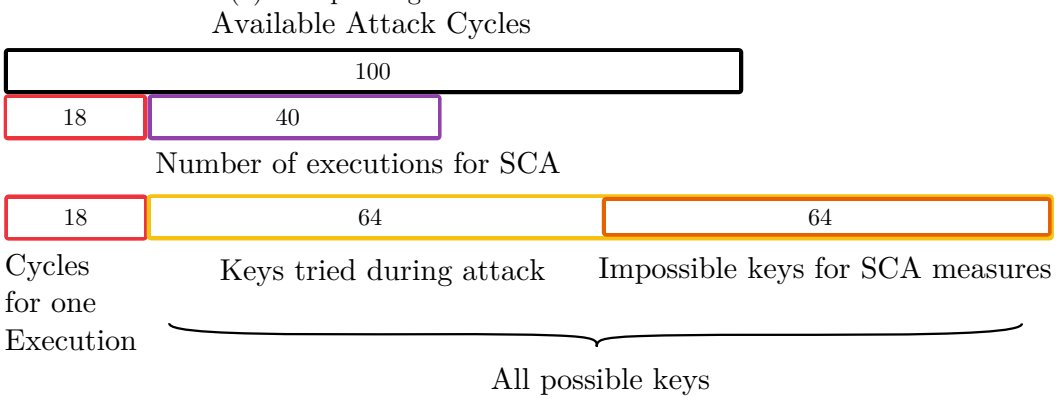
> Je recommande au passage le court article [Standaert 2010] pour une introduction rapide et claire des attaques par canaux auxiliaires.

Avec ces mesures physiques, un adversaire cherche à améliorer son modèle de l'algorithme cryptographique pour en obtenir un plus utile que la boîte noire. En effet, si l'attaquant dispose de 2^{100} cycles de calcul pour tenter de trouver la clé d'un chiffrement sécurisé (IND-CCA2, IND-CPA) présenté en tant que boîte noire, le meilleur algorithme est le brute-force. On peut donc connaître ses chances de réussite selon le temps d'exécution de l'algorithme et le nombre de clés possibles. Pour illustrer, s'il y a 2^{128} clés possibles, qu'un chiffrement prend 2^{18} cycles et que l'attaquant a 2^{100} cycles de calcul à sa disposition, la probabilité de réussite de l'attaquant est de $2^{(100-18-128)} = 2^{-46}$.

L'adversaire peut décider de passer du temps (mesurable en nombre d'exécutions du code) à faire des mesures physiques. Dans notre exemple précédent, on peut imaginer que l'attaquant arrive à déterminer la première moitié de la clé (par exemple car les adresses mémoires utilisées pendant le calcul dépendent de sa valeur). Pour déterminer précisément cette première moitié de la clé, il doit cependant effectuer 2^{40} fois l'algorithme. Parmi les 2^{100} cycles dont dispose l'adversaire, $2^{(18+40)}$ sont passés à découvrir 64 bits de clé. Il reste à l'adversaire $2^{100} - 2^{(18+40)} \approx 2^{100}$ cycles pour trouver les 64 bits restants, ce qu'il fera facilement car $2^{100} \gg 2^{(18+64)}$



(a) Lorsque l'algorithme est exécuté en boite noire.



(b) Lorsque l'architecture fait fuiter une moitié de la clé.

FIGURE 4.13 – Capacité de l'attaquant de mener une attaque sur une primitives cryptographique. Les valeurs sont en log_2.

> Le nombre de cycles, en tant que mesure des capacités de l'attaquant, est une bonne métrique, mais je maintiens qu'une métrique sur le coût monétaire de l'attaque serait plus approprié, surtout quand le matériel pour des attaques par canaux auxiliaires peut en avoir un très élevé. En effet l'efficacité de l'attaque dépend de la précision de la mesure. Il y a un compromis entre le prix de l'appareil de mesure et le nombre de mesures. Il en faudra plus pour établir des statistiques permettant de compenser un matériel moins précis (quand c'est possible). Ou, à l'inverse il faudra payer un prix d'entrée très élevé pour un appareil de mesure à la pointe.

Il convient donc de ne pas évaluer uniquement la complexité d'attaque sur le problème mathématique sur lequel se repose l'algorithme, mais d'évaluer la complexité totale, en prenant en compte les gains d'information que l'adversaire pourrait avoir en effectuant des mesures par canaux auxiliaires.

Pour éviter des fuites d'information trop évidentes et exploitées par le passé, le fait que le temps de calcul et les adresses des variables soient constants est fortement désiré [NIST 2022b]. Cela étant dit, ces mesures n'empêchent pas le temps de calcul de pouvoir varier au niveau matériel, comme exposé récemment dans l'attaque Hertzbleed [Wang 2022]. Il revient au concepteur de ces machines de réduire ces fuites dans de prochaines conceptions. Les deux autres sources de fuites qui ne peuvent pas être corrigées au niveau logiciel/algorithmique sont la consommation énergétique du calcul et les émissions électromagnétiques causées par celui-ci.

Typiquement, les flip-flops (bascules de transistors) permettant au processeur de stocker ses variables, et celles de la mémoire utilisent une tension différente selon si leur valeur est stable ou en train de changer. Si celle-ci change (un 0 devient 1 ou un 1 devient un 0), la consommation est légèrement plus forte que si elle ne change pas. En mesurant la consommation totale du processeur ou de différents composants, on peut ainsi avoir une mesure (bruitée) du nombre de bit-flips effectués durant un cycle, chaque bascule contribuant à hauteur similaire à la consommation totale (dans ce modèle de fuite).

Il est alors possible d'essayer d'établir une relation entre la mesure de voltage et les instructions effectuées avec le processeur. En isolant une instruction et en connaissant finement l'architecture, on peut déterminer quels sont les calculs effectués et sur quelles variables. Ainsi on peut tenter d'en déduire de l'information sur celles-ci. Cela porte communément le nom de Hamming Weight Model [Standaert 2010].

Par exemple, sur un ordinateur 32 bits, si un registre est à 0 et qu'une donnée x est chargée dans ce registre, chaque passage d'un bit à 1 dans x contribuera plus à la consommation, qui contiendra alors une composante proportionnelle au poids de Hamming de x. Si l'attaquant répète de multiple fois l'expérience, il est capable d'isoler statistiquement la contribution de x de celle du « bruit » (calculs ne concernant pas la variable attaquée).

La mesure de la consommation électrique peut se faire sur l'alimentation d'un composant et fait donc apparaître toutes les contributions de manière équivalente. A l'inverse, les émissions électromagnétiques, diminuant avec la distance permettent d'étudier plus précisément certaines zones des composants suffisamment distantes

physiquement, mais demandent un outillage précis, d'autant que des facteurs extérieurs peuvent plus facilement créer des variations dans les mesures.

Défensivement, il y a des pistes de recherche en cours d'exploration pour rendre les architectures de plus en plus hermétiques aux attaques par canaux auxiliaires [Standaert 2010]. Il est possible que les mesures proposées ne soient pas adaptées à toutes les situations (light-weight, temps réel, low-cost...). On peut donc par précaution ne pas se reposer sur la sécurité fournie par la technologie mémoire pour une primitive se voulant généraliste.

L'autre défense envisagée suppose que des fuites auront inévitablement lieu. Il convient alors de faire en sorte que celles-ci ne soient pas exploitables par l'attaquant (masquage des variables secrètes). Ces méthodes étant coûteuses en performance, nous allons d'abord étudier les moyens avec lesquels l'attaquant pourrait exploiter ces informations.

4.6.2 Modèle d'attaquant, entropie des calculs

Le modèle dans lequel nous allons évaluer la sécurité par canaux auxiliaires est le suivant : nous allons supposer que l'attaquant dispose d'un matériel d'attaque infiniment précis qui donne un accès non-bruité aux poids de Hamming des transitions des registres du processeur et des stockages mémoire. Il a aussi la capacité de savoir quand est effectué une certaine instruction. Ainsi la méthode d'attaque est la suivante : l'attaquant tient un tableau de toutes les clés possibles, et effectue des mesures sur une exécution de l'algorithme. A chaque instruction, les informations obtenues lui permettent parfois de déduire que certaines des clés de la liste ne sont pas possibles : elles n'auraient pas correspondu aux valeurs physiques mesurées. Elles sont enlevées de la liste et l'analyse continue jusqu'à ce que la taille restante de la liste soit dans les capacités de brute-force de l'attaquant (dans notre exemple $2^{(100-18)}$).

Ce modèle d'attaquant « parfait » ne prend pas en compte la complexité de la déduction des clés qui peuvent être enlevées. Un modèle d'attaquant « réel » prend en compte une limite dans sa capacité de déduction. Par exemple, si une opération entre les 8 premiers bits de la clé et les 8 suivants (par exemple une multiplication modulaire) cause une transition de poids de Hamming 4, l'attaquant doit (sauf s'il dispose d'un algorithme plus performant) énumérer toutes les combinaisons possibles des 16 premiers bits pour vérifier si le poids de Hamming résultant est bien de 4. On voit bien alors que si une opération implique plus de bits secrets que la capacité de calcul dont dispose l'attaquant, celui-ci doit ignorer l'opération. Il peut essayer de continuer l'attaque mais en ayant éventuellement perdu de l'information sur les variables impliquées.

En réalité, l'attaquant n'aura pas toujours accès à des mesures suffisamment précises pour pouvoir totalement rayer certaines possibilités de valeurs que peuvent prendre certaines parties de la clé. On peut donc associer à chaque valeur possible une probabilité plus ou moins forte (on listera donc des tuples valeur/probabilité dans le tableau). C'est alors l'entropie H qui remplacera la colonne Size et qui permettra de relier les informations apprises à la complexité restante à l'attaque brute-force. Ainsi on pourra dire que l'attaquant dispose d'une attaque, si après une

Bits	Size	Possible Values
1	1	0, 1
2	1	0, 1
3	1	0, 1
⋮	⋮	⋮
128	1	0, 1

(a) Début de l'algorithme, aucune information

Bits	Size	Possible Values
1-8	6.12	0000 1111, 0001 0111, ⋯
9-16	5.8	0000 0111, 0000 1011, ⋯
17-24	0	1111 1111
⋮	⋮	⋮
121-128	6.12	0000 1111, 0001 0111, ⋯

(b) Une écriture en mémoire 8 bits par 8 bits fait fuiter les poids de Hamming

Bits	Size	Possible Values
1-16	10.10	0000 1111 0001 1111, 0000 1111 0011 0111, ⋯
17-24	0	1111 1111
⋮	⋮	⋮
121-128	6.12	0000 1111, 0001 0111, ⋯

(c) Fuite du poids de Hamming de la multiplication des 2×8 permiers bits.

Bits	Size	Possible Values
1	1	0, 1
2	1	0, 1
⋮	⋮	⋮
16	1	0, 1
17-24	0	1111 1111
⋮	⋮	⋮
121-128	6.12	0000 1111, 0001 0111, ⋯

(d) Une opération trop grande pour faire des déductions réécrit sur les 16 premier bits

TABLE 4.7 – Exemple de suivi des clés possibles au cours d'un calcul. La colonne Size représente le log_2 du nombre de valeurs possibles

opération, la somme de la colonne entropie H est plus petite que le nombre d'essais que peut faire l'attaquant avec sa capacité d'attaque.

> Pour rappel, l'entropie d'une distribution a pour propriété notable que pour une distribution sur n valeurs, elle est maximale et vaut $log_2(n)$ si cette distribution est uniforme. Elle vaut à l'inverse 0 sur une distribution concentrée sur une seule valeur. Un dé à 8 faces a une entropie de 3 s'il est équilibré. Si le dé est truqué, son entropie sera plus basse. C'est une mesure d'« uniformité » de la distribution. Si l'entropie est très faible, on peut « parier » avec plus de confiance sur le résultat. Similairement le brute-force peut se faire en « pariant » en premier sur les clés les plus probables.

Pour s'assurer de la résistance de son schéma cryptographique, il n'est pas nécessaire de calculer soi-même pour plusieurs exécutions les clés restantes (se mettre dans la peau de l'attaquant) pour avoir une représentation statistique de la réalité. On peut attribuer une fois pour toute, à chaque opération, une perte d'entropie pour chaque opérande et une quantité d'information mutuelle créée entre ceux-ci.

> Une information mutuelle entre deux variables est une information sur ces deux variables qui n'a pas de conséquences sur l'une d'elles prise séparément. Par exemple pour deux dés à 6 faces, savoir que la somme des lancers est paire ne donne aucune information sur l'un des deux dés, mais il n'y a plus que 18 cas possibles au lieu de 36. Un bit d'information mutuelle est ainsi créé.

Il n'y a donc pas à suivre quelles sont les valeurs possibles des variables secrètes, mais uniquement leur nombre. On peut reprendre les tableaux précédents sans liste des valeurs, en travaillant de manière uniquement statistique. Tant que l'entropie totale de la clé (somme de la colonne) est supérieure à la capacité de l'attaquant, même un attaquant « parfait » n'a pas d'attaque. Si ce n'est pas le cas, il faut que la somme des entropies des opérandes de certaines opérations soit supérieure à la limite de calcul pour que l'attaquant « réel » n'ait pas d'attaque.

Ces modèles d'attaquants sont similaires à celui trouvé dans [Köpf 2007], il cherche à minimiser l'entropie de partitions de la clé. Nous supposons juste en plus que l'information est le poids de Hamming des bit-flips de toutes les variables au cours du calcul, ce qui enlève la complexité théorique de gérer une attaque adaptative.

En vue de la création d'un outil d'analyse de résistance aux canaux auxiliaires, le travail à entreprendre serait d'établir l'entropie perdue pour chaque opération d'une architecture connue, par exemple les opérations binaires classiques :

— $\texttt{Xor}(r_1, r_2) \to r_3$
— $\texttt{And}(r_1, r_2) \to r_3$
— $\texttt{Or}(r_1, r_2) \to r_3$
— $\texttt{Not}(r_1) \to r_2$
— $\texttt{Copy}(r_1) \to r_2$

— `AddWithCarry`$(r_1, r_2) \rightarrow r_3$

— `AddWithoutCarry`$(r_1, r_2) \rightarrow r_3$

— `ModMult`$(r_1, r_2) \rightarrow r_3$

— `Mult`$(r_1, r_2) \rightarrow (r_3, r_4)$

Ainsi que des opérations plus complexes ou composées comme :

— `AddMult`$(r_1, r_2, r_3) \rightarrow (r_4, r_5)$

— `ModAddMult`$(r_1, r_2, r_3) \rightarrow r_4$

Ces entropies doivent être évaluées pour chacune de ces opérations, pour chaque taille de registre, pour plusieurs modes possibles, tout en prenant en compte pour chacun des registres si la variable qu'il contient est secrète ($H > 0$) ou publique ($H = 0$), ainsi que si certains des registres d'entrée sont aussi des registres de sortie.

Pour certaines de ces opérations comme `Xor`, des formules analytiques peuvent être trouvées facilement pour donner des formules faciles à calculer selon la taille des registres. La figure 4.14 ci-dessous montre que l'entropie de r_1 perdue lors de `Xor`$(r_1, r_2) \rightarrow r_2$ est linéaire dans le logarithme de la taille du registre. Mais pour d'autres fonctions moins évidentes, on peut recourir à l'énumération et l'application directe de la formule de l'entropie de Shannon. Lorsque ce n'est pas possible, par exemple pour `AddMult` qui implique un total de 5 registres (si ce sont des registres de 32 bits c'est déjà 2^{160} possibilités), on peut au moins chercher à donner une approximation de la quantité d'information apprise en extrapolant les statistiques obtenues sur un nombre réduit de calculs avec des registres tirés aléatoirement.

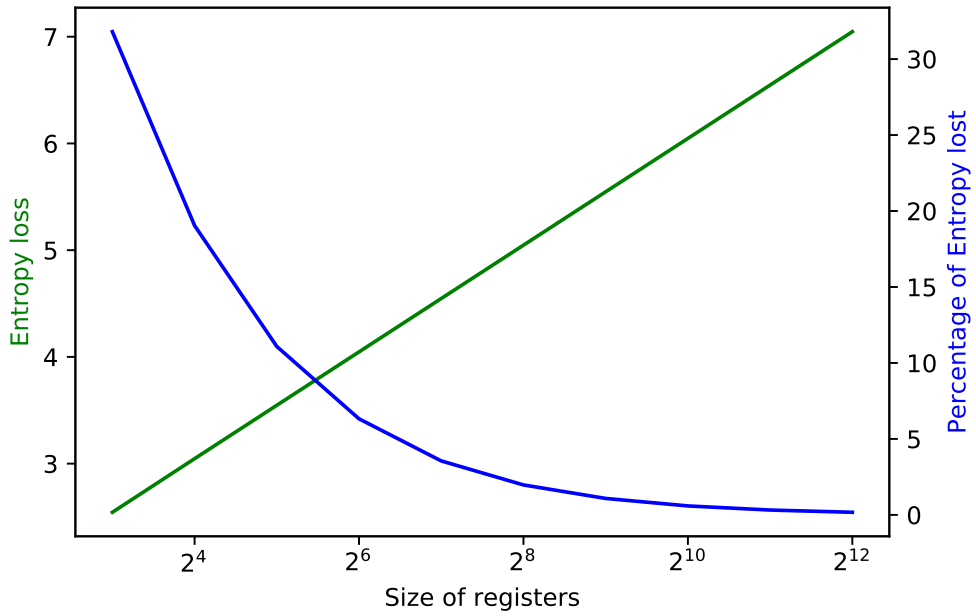


FIGURE 4.14 – Entropy loss during `Xor`$(r_1, r_2) \rightarrow r_2$ in the Hamming Weight model

Nous avons commencé à développer une preuve de concept d'un simulateur permettant d'évaluer au cours de l'exécution d'un algorithme la quantité d'entropie restante dans chacune des variables secrètes. Pour commencer par un cas d'utilisation simple, nous avons appliqué ce modèle à Salsa20 et ChaCha20, car ces primitives ne sont basées que sur 3 opérations de 32 bits : le `Xor`, le `Shift` et le `Add`.

Les résultats de cette simulation n'ont pas encore été soumises au processus de publication scientifique de vérification par les pairs, mais ils semblent donner quelques indications sur les manières d'améliorer la sécurité :

— Certaines opérations comme le $\texttt{Xor}(r_1, r_2) \to r_2$ ne créent pas d'information mutuelle et rendent la perte d'information « asymétrique ». Pour ChaCha20, cela indique qu'il est intéressant de vérifier s'il n'existe pas une disposition des blocs publics `expand 32 byte k` qui soit plus sécurisée que la disposition actuelle sur les 4 premier blocs.

— Le nombre d'opérations et donc de fuites dans ce modèle est tel que le seul moyen qui apparaît comme pouvant empêcher l'entropie de descendre en dessous du paramètre de sécurité, est de créer beaucoup d'information mutuelle entre certaines variables. Ainsi, l'espace des variables est si grand qu'on ne peut pas tracer l'ensemble des valeurs possibles avec ces fuites.

— Les fuites de certaines informations sont indépendantes statistiquement, comme les premiers `Add`, `Xor` et `Shift` de chaque quarter round de ChaCha20. En simulant toutes les opérations comme si elles étaient indépendantes statistiquement, nous nous assurons de calculer une borne supérieure des informations que l'attaquant peut obtenir, car lorsqu'elles sont liées, c'est qu'une partie des informations que l'attaquant obtient est redondante.

On peut maintenant chercher à évaluer les impacts sur les primitives qu'aurait le passage à une architecture de calcul en mémoire en suivant ce modèle.

4.6.3 Impact de la taille des opérandes et des variables

Opérandes plus grands : Comme montré sur la figure 4.14, la quantité d'information donnée par la fuite du poids de Hamming d'une variable est linéaire dans le log de la taille de la variable qui fuite. Par conséquent, plus l'opérande est grand, plus la quantité d'information qui fuite est grande, mais en proportion de la longueur de la clé, la fuite est de plus en plus négligeable.

Ainsi on voit que la taille des variables est la première défense naturelle qu'il peut y avoir contre les attaques par canaux auxiliaires. Avoir à calculer le `Xor` de deux variables de 64 bits par paquets de 16 à cause d'une limite de l'architecture est voué à faire fuiter beaucoup plus d'information que le même calcul sur une architecture 64 bits.

De même on voit l'intérêt de mécanismes SIMD comme la vectorisation pour ajouter une protection : entre huit multiplications successives dans l'ALU (32bits) et ces mêmes multiplications en simultané avec AVX, la seconde version fuitera une proportion d'information beaucoup plus basse. De plus, la création d'information mutuelle entre les variables sera impossible à suivre pour l'attaquant à cause du nombre de combinaisons possibles.

Similairement, on peut étendre ce raisonnement sur des opérations en cache ou en mémoire, dont les fuites seraient alors extrêmement dures à exploiter (apprendre le poids de Hamming de 4096 bits est inexploitable sauf si on sait par exemple que les 4090 premiers bits sont des 0). Des multiplications en Bit-Array sur 3488 variables de 12 bits (Filling de KeyGen de Classic McEliece) donneraient une quantité bien moindre d'informations à l'attaquant que les mêmes multiplications faites successivement à l'horizontal. De plus, le mélange de calculs verticaux et horizontaux permis par les mémoires multi-dimensionnelles peut être utilisé pour créer efficacement de l'information mutuelle entre les variables.

Dans la version de *Encrypt* de Classic McEliece transposée et en cache que nous proposions, une opération pourrait faire fuiter le message : `CSET`. Si l'espace mémoire fixé par les `CSET` est mis à zéro avant l'algorithme, le poids de Hamming des transitions de chaque `CSET` n'aura que deux valeurs possibles : 0 ou 1024. Cela dévoilera le premier bit du vecteur d'erreurs et peut se répéter sur les suivants. Afin que cela n'arrive pas, et qu'aucun lien soit fait entre les bits du message, il faut au préalable créer un masque de zéros sur les bits correspondant aux BPs 0,2,4,6 et 1 sur bits des BPs 1,3,5,7. Si une copie de ce masque est faite sur la ligne des `CSET` avant ceux-ci, le seul poids de Hamming de transition possible est de 512.

Variables plus petites que les opérandes : Si un algorithme utilise une variable plus petite que la taille maximale d'opérandes disponibles (par exemple 16 bits sur une architecture 64 bits). Il est dommage d'ignorer tous les bits non utilisés et de les laisser à 0. Selon l'opération que doit subir cette variable, il serait judicieux de se servir de bits qui ne sont pas impliqués dans le calcul pour stocker du bruit, dont la contribution au poids de Hamming du résultat réduirait l'information exploitable par l'attaquant.

Pour illustrer, la figure suivante (4.15) montre comment le fait de ne réserver que x bits d'un registre de taille y fait varier l'entropie perdue. On remarque que déjà avec autant de bruit que de données, la perte est fortement réduite.

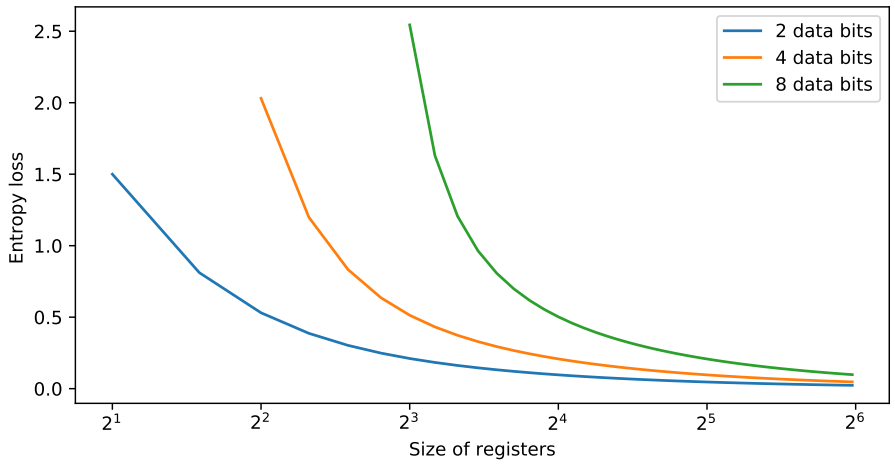


FIGURE 4.15 – Perte d'entropie selon la quantité de données et la taille du registre

On voit bien ainsi que lors du traitement d'une donnée, plus il y a de bruit traité en même temps que la donnée, moins des informations sur celle-ci vont fuiter. Cependant, on sera contraint pour certaines opérations de laisser aussi des champs inoccupés pour garder la justesse du calcul, comme pour la multiplication par exemple.

Variables plus grandes que les opérandes : Ce mode de traitement bruité peut être transposé aux variables plus grandes que la taille de l'opérande. L'objectif est de répartir le calcul en divisant la variable sur plusieurs opérandes, de la manière dont opérations d'addition et de multiplication sont gérées dans les bibliothèques de traitement de grands nombres.

Classiquement, dans ces librairies les nombres sont écrits en remplissant entièrement des registres à partir des bits les plus faibles. Par exemple, sur une architecture 64 bits, une variable de 80 bits aura ses 16 bits de poids fort écrits sur 16 bits parmi 64 et les 64 bits de poids faible remplissent entièrement un autre mot mémoire.

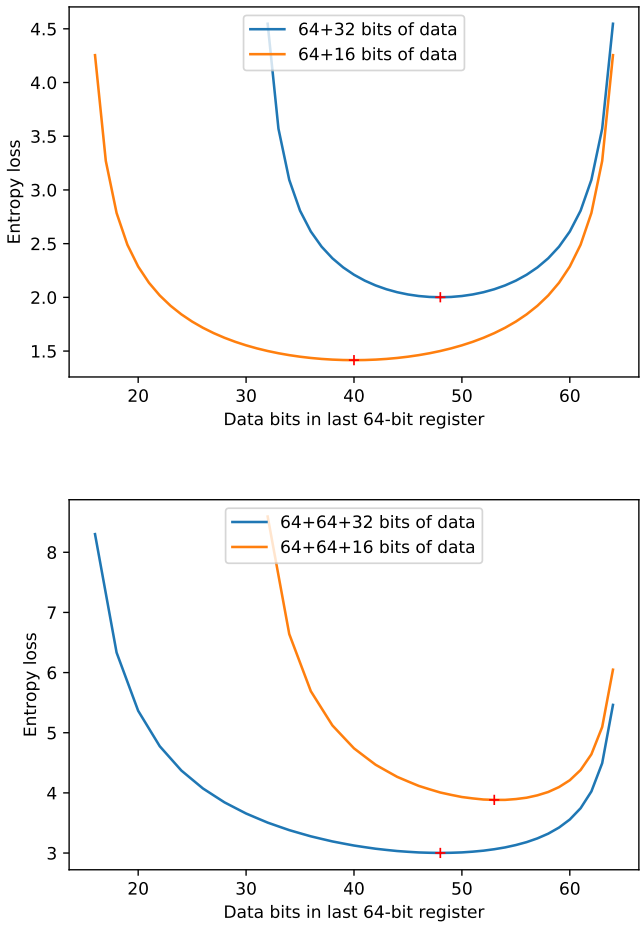


FIGURE 4.16 – Perte d'entropie selon la répartition des bits de bruit

Or, la figure 4.16, obtenue par l'application de la formule de l'entropie condi-

tionnelle permet de vérifier cela par exemple pour le `xor` et indique que la fuite totale d'information est minimale quand les données sont réparties équitablement entre les opérandes (pour un nombre d'opérandes fixe). En effet on comprend qu'à l'inverse, une partie de données fortement bruitée fuite très peu d'informations, mais en contrepartie la partie peu bruitée en perd beaucoup.

Il reste donc à évaluer quel est le nombre de registres optimal pour représenter une certaine donnée. On remarque sur la figure 4.17 qu'il y a un compromis : soit la proportion des registres réservés aux données est forte et il peut y en avoir peu mais beaucoup d'informations fuitent, soit elle est faible et peu d'informations fuitent, mais le nombre de registres nécessaires explose. Comme ce nombre contraint aussi les performances de calcul, il vaut certainement mieux pencher pour plutôt peu de bruit. Par exemple avec 75% de données, on n'a plus qu'un tiers de la fuite d'information pour les architectures 64 bits.

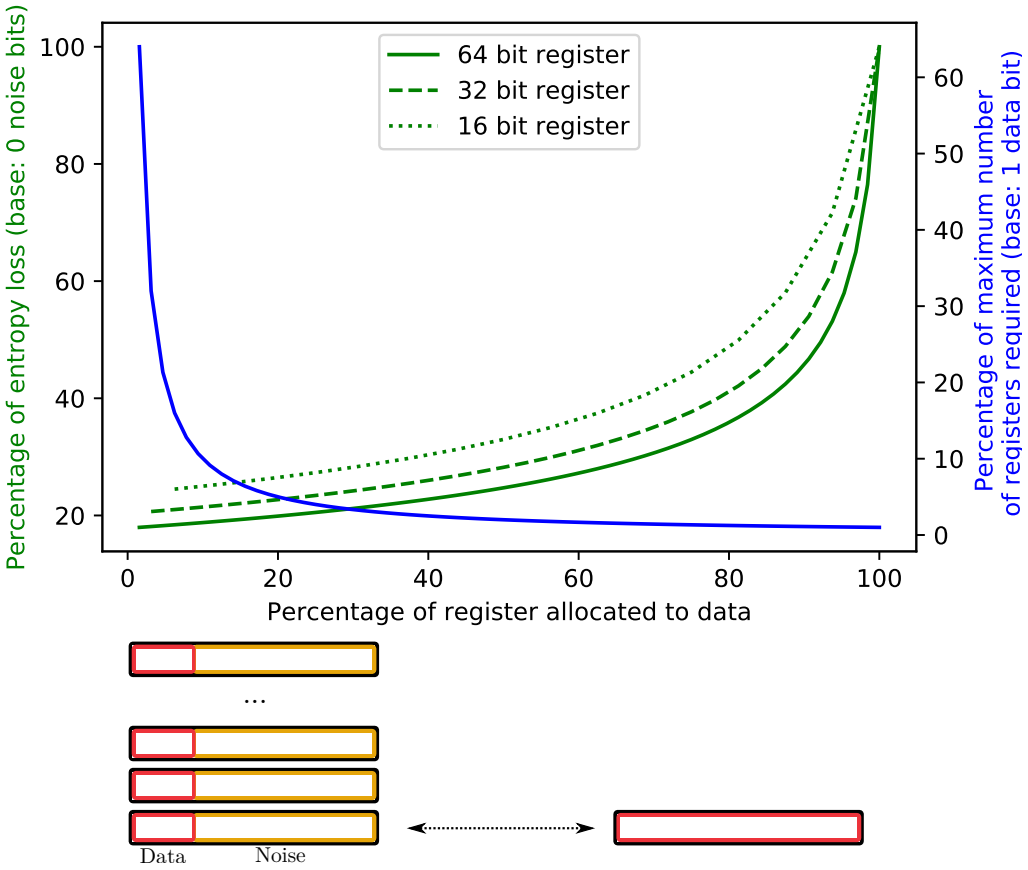


FIGURE 4.17 – Effets de différents taux de données et de bruit sur les fuites d'informations et l'espace de stockage nécessaire pour une donnée

La gestion de ce bruit doit actuellement se faire de manière logicielle, ce qui implique soit des réductions de performances soit un temps de développement plus long pour inclure une gestion de look-up-tables dépendantes de l'architecture. Il est

possible d'envisager une prise en charge de ce bruit au niveau matériel, en revisitant le jeu d'instructions pour incorporer des opérations bruitées aux outils disponibles. Par exemple, une architecture 64 bits pourrait avoir un `xor64`, un `xor32` avec 32 bits de bruit, et un `xor16` avec 48 bits de bruit. Si de telles opérations sont envisagées, il conviendrait de les prendre en compte dans les évaluations de fuite d'entropie selon la quantité de bruit choisie. On pourrait par exemple évaluer l'algorithme ChaCha20 sur un ordinateur 64 bits quand les opérations ont 32 bits de bruit. On peut alors envisager de reprendre l'évaluation des coûts de masquages de primitives [Migliore 2019].

Cette approche peut être proposée pour remplacer un mécanisme fortement étudié actuellement : celui de l'ajout logiciel ou matériel de masques sur les données. Par exemple masquer deux variables avec un `xor` avec une variable aléatoire permet que le `xor` des variables masquées ne cause aucune fuite. Identiquement, il est possible de créer des masques additifs ou des masques multiplicatifs. Ce qui est problématique avec ces masques est qu'ils ne sont pas compatibles entre eux. Il faut alors changer de masque entre deux opérations différentes, ce qui est très coûteux en performance.

Notre approche correspond à un masquage « imparfait » : certes, il reste de l'information qui fuite, mais si la quantité d'information de cette fuite est maîtrisée et qu'elle ne permet pas à l'attaquant d'obtenir une complexité de brute-force suffisamment basse, il n'y a pas besoin de recourir à des méthodes plus coûteuses. Un simulateur plus poussé permettrait par exemple d'identifier les opérations qui causent les plus grandes pertes d'entropie pour ne masquer que celles-ci, ce qui réduirait les coûts au global.

L'ajout de bruit est totalement adapté aussi sur les opérations en mémoire :

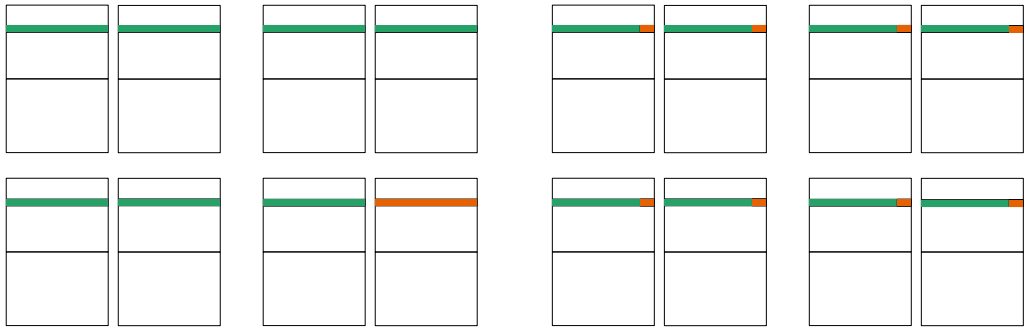
— Dans le chiffrement McEliece transposé avec calcul en cache, le reste des lignes de caches non utilisées peuvent par exemple contenir du bruit ce qui ajoute une protection supplémentaire sans surcoût en complexité.

— Identiquement, si Filling est effectué en BitArray, 3488 calculs en même temps demandent l'implication de 7 BPs, mais pour des raisons de lecture d'instruction, 8 sont mobilisés. On peut alors remplir ce dernier BP (le padding des lignes de la matrice) non pas de zéros mais de bruit pour réduire les fuites.

Le fait d'ajouter du bruit en bout d'un opérande aussi grand peut cependant avoir ses limites : si la consommation sera la même peu importe quels bits sont des données et quels bits sont du bruit, ce n'est pas le cas des émissions électromagnétiques. La distance physique entre le premier BP (données) et le dernier BP (bruit) peut être suffisamment grande pour que plusieurs sondes permettent d'enlever la contribution du BP bruit sur la mesure du BP données. Cette attaque dépend de la précision de la sonde de l'attaquant (et donc de ses moyens financiers) et de la conception physique de la puce (distance entre les banks).

Il conviendrait d'évaluer à quelle distance physique les bits de bruit n'influent plus assez sur les mesures pour limiter la fuite d'information sur les données. La contre-mesure qui permettrait d'empêcher d'isoler le bruit des données serait de « panacher » le bruit et les données, en répartissant les bits de bruit entre les BPs, à l'exacte image de comment les bits de bruits devaient être ajoutés aux variables

écrits sur plusieurs mots mémoire décrits précédemment.



(a) Bruit positionné après les données (b) Bruit réparti avec les données

FIGURE 4.18 – Répartition des bits de bruit (rouge) à travers les Bank Partitions

Dans notre modèle où l'attaquant n'est limité que par la déduction de quelles valeurs sont possibles pour les variables secrètes, le calcul en mémoire semble donc constituer un avantage majeur. En effet, il permet une fuite d'information moindre en proportion et de créer plus d'information mutuelle entre les variables. Cela crée des opérations à partir desquelles l'attaquant ne peut pas déduire d'information à cause de leur taille. Cela permet aussi d'inclure des bits de bruit au calcul, qui réduit les fuites de certaines opérations sur des variables plus petites, ou réparties sur plusieurs opérandes. Ces principes nécessitent cependant des études et outils de simulation beaucoup plus poussés et leur incorporation progressive au niveau matériel afin, comme les calculs en mémoire, de devenir petit à petit transparents pour un développeur.

4.7 Conclusion

Le point le plus important à retenir de ce chapitre est que l'évolution des architectures et l'intégration d'accélérateurs matériels génériques tels que le calcul en mémoire auront des effets secondaires positifs sur les performances de certaines primitives post-quantiques et leur sécurité par canaux auxiliaires.

En particulier, nous nous sommes concentrés sur Classic McEliece, un chiffrement asymétrique post-quantique qui est connu pour sa sécurité avec des petits chiffrés, mais pour son temps de génération de clés plus grand que ses compétiteurs. Nos résultats ont montré que KeyGen et Encrypt peuvent être améliorés grâce au calcul en mémoire.

Pour `pk_gen`, être capable de calcul en cache permet une accélération pouvant aller jusqu'à 12.6 fois. En conséquence, nous avons appris que des étapes de la génération de la clé telles que Filling, qui pouvaient précédemment être considérées comme négligeables pourraient bénéficier d'une adaptation de leur conception à ces nouvelles architectures.

Pour Encrypt, nous avons montré que lorsque la clé publique est fournie dans la forme transposée, ce nouveau type d'architecture peut entraîner une accélération

allant jusqu'à 28 fois. Si `controlbits` reste un goulot d'étranglement, il conviendrait alors que la transposition se fasse en fin de KeyGen. De plus, ce concept illustre l'opportunité que le calcul en mémoire offre pour re-penser les algorithmes de manière non-conventionnelle (« outside the box ») pour mieux optimiser les ressources.

L'application de nos connaissances nouvelles aux autres primitives des rounds 3 et 4 du processus de standardisation nous donne les résultats suivants :

— Quelques primitives (Crystals-KYBER et SIKE [Schwabe 2021, Azarderakhsh 2017]) ne devraient pas bénéficier du calcul en mémoire comme elles sont faites pour être optimales sur les architectures actuelles, ou à l'aide d'opérations vectorielles sur 256 bits.

— A l'exception de celles-ci, les primitives profiteraient d'une réévaluation de leurs paramètres afin d'améliorer leurs performances sur des architectures capables de calcul en cache. Cela serait fait principalement en augmentant le parallélisme des opérations, si possible en diminuant en même temps leur complexité en portes logiques.

— Les architectures actuelles favorisent l'utilisation de données écrites horizontalement, alors que le calcul bit-array serait souvent plus efficace si elles étaient écrites verticalement. Cependant, dans nos architectures actuelles, la transposition est coûteuse en performances, ce qui limite le gain de temps possible. Une combinaison de calcul en bit-array et de Transpose Gateway Units pourrait se révéler efficace pour accélérer NTRU, NTRU prime, et FrodoKEM [Chen 2021, Bernstein 2016, Bos 2016].

— Si nous devions prendre une approche Top-down, et concevoir des opérateurs de calcul en mémoire visant à améliorer les performances de primitives cryptographiques, un opérateur de rotation ou de décalage pourrait être un bon objectif. Cela permettrait d'accélérer HQC [Aguilar Melchor 2017], BIKE [teambikesuite.org 2022], NTRU [Chen 2021], et NTRU Prime [Bernstein 2016]. Cependant, cela pourrait se révéler être un réel défi scientifique car il faudrait pouvoir transférer des données du column peripheral extérieur d'un BP à un autre de manière efficace.

— Les primitives basées sur les codes correcteurs d'erreurs comme Classic McEliece seront généralement celles qui bénéficieront le plus des améliorations du calcul en mémoire, car ce sont celles qui ont des tailles d'opérandes les plus adaptées.

Notre étude s'est concentrée sur les KEMs, mais elle pourrait parfaitement être étendue aux algorithmes de signature, où des résultats similaires sont à attendre. Il faudrait aussi idéalement l'étendre à d'autres champs de la cryptographie connus pour travailler sur de larges quantités de données ou avec de larges opérandes comme c'est le cas pour le Fully Homomorphic Encryption.

De plus, il serait intéressant de répéter les expériences en supposant un cache k-associatif plutôt qu'un cache à correspondance directe, ou avec une hiérarchie de cache à plusieurs niveaux pour déterminer à quel niveau il est plus intéressant de faire les calculs. Aussi, l'intégration dans nos expériences d'une simulation précise d'une architecture « Neural Cache » ou « Duality Cache » permettrait d'étudier les

accélérations du calcul bit-array et des Transpose Gateway Units. En simulation, l'évaluation des accélérations possibles avec un opérateur de rotation est désirable.

Dans un modèle de fuite en distance de Hamming, le fait de disposer d'opérandes plus grands permet une fuite d'information moindre en proportion. Cela permet aussi l'intégration de bruit au calcul sans pertes de performances, et ainsi de faire des opérations sans avoir besoin de masquage.

Conclusion et Perspectives

Conclusion

Au travers de la présente thèse, nous avons cherché à étudier l'évolution des différentes couches de la sécurité cryptographique à l'approche de l'ordinateur quantique. Nous avons identifié les changements qui seront propres à chaque couche, chacune constituant un cœur de sujets de recherche en tant que tel.

1. Au niveau de la couche utilisation, nous avons identifié une demande des utilisateurs d'avoir à la fois des services d'accès à une grande quantité de données en simultané ainsi qu'une vie privée toujours plus protégée, notamment par l'intraçabilité de la navigation.
2. Au niveau des protocoles, ceux d'entre eux qui étaient basés sur la commutativité des chiffrements peuvent subir des attaques d'ordinateurs quantiques, capables de résoudre le Hidden Subgroup Problem dans les groupes Abéliens. En conséquence, les protocoles d'interception réglementée ne sont plus sécurisés, à l'image d'une grande partie des systèmes multipartites complexes.
3. Au niveau des primitives, c'est la cryptographie asymétrique qui est à changer. La standardisation de nouvelles primitives est en cours. Il est certain que des optimisations très poussées des nouveaux standards seront proposées.
4. Au niveau des architectures, on note l'arrivée de nouvelles techniques permettant de doter les mémoires de capacités de calcul. D'un autre côté, les mesures par canaux auxiliaires sont de plus en plus précises et permettent l'accès à de plus en plus de variables secrètes des algorithmes.

Les modifications qui ont lieu dans chaque couche ont nécessairement un impact sur les couches supérieures et inférieures. Nous avons donc cherché à illustrer le besoin de cohérence transversale entre les différentes couches. Pour cela, nous avons choisi un problème lié spécifiquement à chaque interface entre deux couches. Les contributions que nous avons apporté au cours de cette thèse ont été représentées sur la figure 4.19.

Protocoles adaptés aux futures utilisations et aux futures primitives. L'évolution de la consommation numérique implique la conception de nouveaux protocoles, plus adaptés aux demandes. Les besoins d'anonymat et de livestreaming mutualisé pouvaient sembler contradictoires. Nous avons choisi de développer un protocole nouveau sur la base de Tor, conçu pour concilier ces exigences et aussi prendre en compte la nécessité d'adapter les primitives utilisées au milieu post-quantique.

Les primitives de chiffrements par attributs permettent la gestion du routage par des groupes de noeuds. Cette transformation permet de se détacher de nombreuses limites de Tor : la gestion complexe du Directory, la qualité de service limitée et la possibilité de déconnexion, ainsi que le manque de capacités à lutter contre un noeud malveillant. De plus, le changement de mode d'établissement de la connexion permet

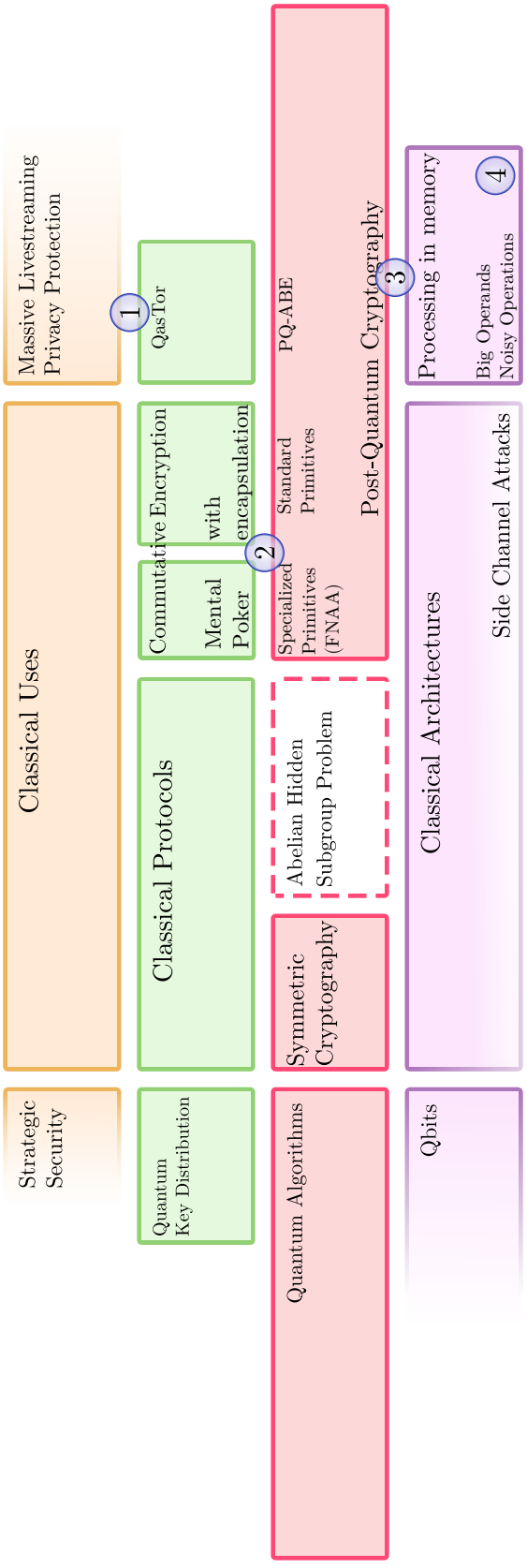


FIGURE 4.19 – Contributions aux différentes couches de la protection des données.
1-4 : Contributions présentées dans cette thèse.
Pointillés : Sécurité compromise
Dégradé Horizontal : Domaine en cours d'expansion
Dégradé Vertical : Domaine menacé par des avancées scientifiques

plusieurs fonctionnalités cruciales dont la capacité à partager un même flux, ce qui limite fortement la bande passante totale sur le réseau et diminue significativement le temps d'établissement d'une connexion.

Le protocole doit bien sûr être adapté aux besoins. L'algorithme de répartition de noeuds dans les groupes et les algorithmes de load balancing sont notamment critiques pour l'optimisation du protocole. Les contraintes fortes de la couche utilisation vont même se ressentir sur la couche primitive. En effet le choix de primitives, notamment le chiffrement par attributs choisi, aura un impact fort sur les performances et donc sur la vitesse d'adoption d'une telle technologie.

Préserver les protocoles de chiffrement commutatif. Les protocoles qui faisaient intervenir divers utilisateurs au moyen d'étapes successives de chiffrement et de déchiffrement le faisaient souvent au moyen de chiffrements commutatifs. Ces protocoles ne sont pas aujourd'hui post-quantiques. Nous avons cherché à permettre une transition vers le post-quantique, tout en intégrant les besoins spécifiques aux cas d'usage qui sont faits des chiffrements commutatifs (interception réglementée) et l'adaptabilité aux architectures cibles variées de ces cas d'usage (voitures connectées par exemple).

Nous avons premièrement cherché à déterminer dans quelles situations le surchiffrement était suffisant. Nous avons prouvé que celui-ci ne permet pas de faire en sorte, au besoin, que les relais puissent tous s'assurer de l'identité des autres participants au schéma. Dans ces cas, seul le chiffrement commutatif est adapté.

L'étude générique de ces protocoles nous a menés à deux solutions principales pour le transfert vers le post-quantique. D'une part, si le graphe qui représente les chiffrements effectués entre les utilisateurs dans le protocole n'a que des noeuds de degré 2 ou plus, il est possible de se servir d'encapsulations de n'importe quelle primitive post-quantique IND-CCA2 (typiquement, un standard du NIST) vers un groupe Abélien, et de tout de même s'assurer qu'aucun avantage n'est donné à l'attaquant même si le problème de vectorisation est facile dans ce groupe. D'autre part, si ce graphe doit contenir des noeuds de degré 1, il reste possible de se servir de primitives spécialisées, par exemple basées sur les FNAA. Cependant, les seules utilisations connues de ces protocoles sont le mental poker, le protocole sans clé, le problème des millionnaires et le transfert aveugle.

Excepté pour ces cas très spécifiques, le chiffrement commutatif basé encapsulations est adapté à des cas d'utilisation complexes, car les schémas peuvent être individuellement remplacés et adaptés à des évolutions de contexte, fréquentes dans l'exploitation commerciale de données personnelles et touchant à la sécurité. Avec la capacité de déléguer et retarder l'application du contrôle d'accès, cela permet une adaptation fine des primitives aux architectures sur lesquelles elles s'exécuteront (light-weight, ABE...).

Adapter les primitives post-quantiques à leur futur contexte d'exécution. La standardisation menée par le NIST a pu se concentrer uniquement sur l'optimisation des algorithmes des primitives grâce à la définition d'architectures cibles. Cela a retiré de la conversation toute vision transversale primitive-architecture. Cependant, nous avons illustré à quel point les primitives et les architectures peuvent co-évoluer.

Les évolutions des architectures, vers la diversification des espaces de calcul

et notamment l'intégration de calcul en mémoire, ont un impact fort sur les performances des primitives. Parmi les plus étudiées actuellement, Classic McEliece présente un parallélisme idéal à l'application de calcul en cache. Et nous avons pu mettre en valeur les gains de performances très forts possibles avec celui-ci. Ces nouvelles architectures vont jusqu'à permettre une nouvelle conception des primitives autour de celles-ci et peuvent faire changer les jeux de paramètres optimaux. Pour illustrer ce genre de répercussions, nous avons montré qu'un changement très fort du chiffrement de Classic McEliece ainsi que du stockage de la clé est nécessaire pour optimiser celui-ci pour le calcul en cache.

Inversement, les primitives et leurs problématiques de sécurité indiquent les évolutions que peuvent prendre les architectures : nous avons montré qu'un grand nombre de primitives post-quantiques profiteraient fortement de calcul vertical en mémoire, ainsi que de moyens efficaces de transposer de grandes quantités de données. De plus, pour réduire les fuites d'informations sur les variables secrètes de ces architectures, celles-ci et les algorithmes devraient être adaptés à la gestion d'opérandes plus grands, et contenant un mélange de données et de bruit. De tels changement architecturaux auront par rebond des conséquences au niveau des algorithmes, qui devront adapter la gestion de leurs variables pour profiter au plus des défenses proposées tout en optimisant les performances.

Perspectives.

Au cours des recherches, nous avons trouvé plusieurs pistes d'exploration propres à chacun des sujets.

— QasTor est encore à l'état de preuve de concept. Il y a encore des algorithmes du protocole qui doivent être améliorés, mais cela nécessite de disposer de données de test, tels que des moyennes d'utilisation de bande passante, des valeurs précises de temps d'exécution des algorithmes, etc.

— Notre méthode de transfert de protocoles de chiffrement post-quantique manque d'un exemple concret. Certes, nous avons grâce à nos outils pu développer un nouveau protocole pour les voitures connectées, mais il serait certainement plus intéressant de reprendre un protocole déjà existant et de le transformer pour le rendre post-quantique.

— Notre évaluation des primitives post-quantiques sur les architectures intégrant du calcul en mémoire est limitée par le manque de calcul vertical et de mémoires multi-dimensionnelles. Incorporer ceux-ci ainsi que la gestion de l'entropie des variables secrètes dans notre simulateur permettrait de pousser la discussion vers de nouvelles pistes d'amélioration.

Cette manière d'isoler les sujets n'est pas idéale, comme nous l'avons déjà illustré, il faudrait intégrer ces améliorations dans une vision encore plus globale : l'adoption éventuelle d'un protocole comme QasTor, comme déjà évoqué, est très dépendant de la qualité de service proposée, et donc notamment des performances des primitives choisies, et celles-ci sont aussi dépendantes des architectures. Pourrait-on essayer de construire la primitive PQ-ABE la plus adaptée à QasTor ? Si des architectures

de calcul en mémoire deviennent disponibles, il semble que les primitives basées sur les codes correcteurs sont plus adaptées que les primitives LWE. Est-il possible d'avoir un ABE basé sur les codes correcteurs, et qui ainsi puisse s'exécuter plus vite, pour améliorer encore un peu l'expérience utilisateur ? De la même manière pour les voitures connectées, quelles sont les meilleures primitives pour chacune des encapsulations du schéma ? Quels sont les noeuds qui ont le plus besoin de résistance aux canaux auxiliaires ? Pour eux, est-il acceptable de réduire les performances pour améliorer leur sécurité ? Quelles primitives sont les plus adaptées à une application simple des lois et contrats du domaine ?

Cette vision transversale de la cryptographie est une piste qui mène d'un côté à l'amélioration de l'expérience utilisateur, pour qui la protection des données doit être intégrée aux applications sans pertes de fonctionnalités ou ressenti de moindre performances et de l'autre côté à une augmentation de l'expressivité de la cryptographie, ce qui permet que l'exploitation de données et la sécurité ne soient pas des antonymes de la vie privée.

Publications

Les travaux présentés dans ce manuscrit ont été effectués à Toulouse au Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) du Centre National de la Recherche Scientifique (CNRS) au sein de l'équipe Tolérance aux Fautes et Sûreté de Fonctionnement. À l'heure à laquelle sont écrites ces lignes, ces travaux ont donné lieu aux publications suivantes :

R. Adelin, C. Nugier, E. Alata, et al., «*Facing emerging challenges in connected vehicles : a formally proven, legislation compliant, and post-quantum ready security protocol*». Dans Journal of Computer Virology and Hacking Techniques (JICV), 2022, Springer.

C. Nugier, D. Leblanc-Albarel, A. Blaise, et al., «*An Upcycling Tokenization Method for Credit Card Numbers*». In SECRYPT 2021-18th International Conference on Security and Cryptography. 2021. Hal.

C. Nugier, E. Alata, «*Brevet prioritaire France FR2205615*». Institut National de la Propriété Intelectuelle (INPI), 2022

Bibliographie

[Abu-Ghazaleh 2019] Abu-Ghazaleh, N., Ponomarev, D. et Evtyushkin, D. *How the spectre and meltdown hacks really worked*. IEEE Spectrum, vol. 56, no. 3, pages 42–49, 2019. (Cité en page 14.)

[Adelin 2022] Adelin, R., Nugier, C., Alata, É., Nicomette, V., Migliore, V. et Kaâniche, M. *Facing emerging challenges in connected vehicles : a formally proven, legislation compliant, and post-quantum ready security protocol*. Journal of Computer Virology and Hacking Techniques, pages 1–28, 2022. (Cité en pages 57, 88 et 90.)

[Aga 2017] Aga, S., Jeloka, S., Subramaniyan, A., Narayanasamy, S., Blaauw, D. et Das, R. *Compute caches*. Dans 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), pages 481–492. IEEE, 2017. (Cité en pages 98, 101, 102, 110, 117, 119 et 122.)

[Agrawal 2021] Agrawal, K. *Signal Statistics : Usage, Revenue, & Key Facts*. août 2021. [Online] (Cité en page 28.)

`https://www.feedough.com/signal-statistics-usage-revenue-key-facts/`.

[Aguilar Melchor 2017] Aguilar Melchor, C., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.-C., Gaborit, P., Persichetti, E. et Zémor, G. *Hamming Quasi-Cyclic (HQC)*. novembre 2017. [Online] (Cité en pages 97, 126 et 144.)

`https://hal.archives-ouvertes.fr/hal-01946880`. Submission to the NIST post quantum standardization process. 2017.

[Akhoondi 2012] Akhoondi, M., Yu, C. et Madhyastha, H. V. *LASTor : A low-latency AS-aware Tor client*. Dans 2012 IEEE Symposium on Security and Privacy, pages 476–490. IEEE, 2012. (Cité en page 34.)

[ANSSI 2014] ANSSI. *Référentiel Général de Sécurité Version 2.0, Annexe B1*. 2014. [Online] (Cité en page 7.)

`https://www.ssi.gouv.fr/uploads/2014/11/RGS_v-2-0_B1.pdf`.

[Arfaoui 2021] Arfaoui, G., Blazy, O., Bultel, X., Fouque, P.-A., Jacques, T., Nedelcu, A. et Onete, C. *How to (Legally) keep secrets from mobile operators*. Dans European Symposium on Research in Computer Security, pages 23–43. Springer, 2021. (Cité en page 59.)

[Armknecht 2014] Armknecht, F., Gagliardoni, T., Katzenbeisser, S. et Peter, A. *General Impossibility of Group Homomorphic Encryption in the Quantum World*. 2014. [Online] (Cité en pages 62 et 86.)

`https://arxiv.org/abs/1401.2417`.

[Arute 2019] Arute, F., Arya, K., Babbush, R. et et al. *Quantum supremacy using a programmable superconducting processor*. 2019. [Online] (Cité en page 96.)

`https://doi.org/10.1038/s41586-019-1666-5`.

[Association 2001] Association, A. B. *ABA Standards for Criminal Justice : Electronic Surveillance, Section A : Electronic Surveillance of Private Communications, 3d ed.*, 2001. Accessed : 2023-01-14. (Cité en page 58.)

[Ateniese 2006] Ateniese, G., Fu, K., Green, M. et Hohenberger, S. *Improved proxy re-encryption schemes with applications to secure distributed storage.* ACM Transactions on Information and System Security (TISSEC), vol. 9, no. 1, pages 1–30, 2006. (Cité en pages 12, 69, 80 et 91.)

[Azarderakhsh 2017] Azarderakhsh, R., Campagna, M., Costello, C., Feo, L., Hess, B., Jalali, A., Jao, D., Koziel, B., LaMacchia, B., Longa, P.*et al. SIKE–Supersingular Isogeny Key Encapsulation.* URL : https ://sike. org, 2017. (Cité en pages 97, 127 et 144.)

[Azfar 2011] Azfar, A. *Implementation and performance of threshold cryptography for multiple escrow agents in VoIP.* Dans International Joint Conference on Advances in Signal Processing and Information Technology, pages 143–150. Springer, 2011. (Cité en page 58.)

[Barbera 2013] Barbera, M. V., Kemerlis, V. P., Pappas, V. et Keromytis, A. D. *CellFlood : Attacking Tor onion routers on the cheap.* Dans European Symposium on Research in Computer Security, pages 664–681. Springer, 2013. (Cité en page 33.)

[Barker 2011] Barker, J., Hannay, P. et Szewczyk, P. *Using traffic analysis to identify the second generation onion router.* Dans 2011 IFIP 9th International Conference on Embedded and Ubiquitous Computing, pages 72–78. IEEE, 2011. (Cité en page 33.)

[Bernstein 2016] Bernstein, D. J., Chuengsatiansup, C., Lange, T. et Van Vredendaal, C. *NTRU Prime.* IACR Cryptol. ePrint Arch., vol. 2016, page 461, 2016. (Cité en pages 129 et 144.)

[Bernstein 2021] Bernstein, D. J. et (editors), T. L. *eBACS : ECRYPT Benchmarking of Cryptographic Systems*, Nov 2021. (Cité en page 98.)

[Bethencourt 2007] Bethencourt, J., Sahai, A. et Waters, B. *Ciphertext-Policy Attribute-Based Encryption.* Dans 2007 IEEE Symposium on Security and Privacy (SP '07), Berkeley, France, mai 2007. IEEE. (Cité en page 53.)

[Boneh 1998] Boneh, D.*et al. Twenty years of attacks on the RSA cryptosystem.* Notices of the AMS, vol. 46, no. 2, pages 203–213, 1998. (Cité en page 64.)

[Boneh 2001] Boneh, D. et Franklin, M. *Identity-based encryption from the Weil pairing.* Dans Annual international cryptology conference, pages 213–229. Springer, 2001. (Cité en page 18.)

[Boneh 2020a] Boneh, D. *Online Cryptography Course by Dan Boneh.* 2020. [Online] (Cité en page 5.)
`https://crypto.stanford.edu/~dabo/courses/OnlineCrypto/`.

[Boneh 2020b] Boneh, D. et Shoup, V. *A Graduate Course in Applied Cryptography.* 2020. [Online] (Cité en page 5.)
`http://toc.cryptobook.us/`.

[Bos 2016] Bos, J., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V., Raghunathan, A. et Stebila, D. *Frodo : Take off the ring ! practical, quantum-secure key exchange from LWE.* Dans Proceedings of the 2016 ACM SIGSAC conference on computer and communications security, pages 1006–1018, 2016. (Cité en pages 129 et 144.)

[Brown 2022] Brown, S. *RISC-V Is Thriving : Here's What You Need to Know.* novembre 2022. [Online] (Cité en page 117.)
`https://community.cadence.com/cadence_blogs_8/b/corporate/posts/riscv`.

[Bub 2016] Bub, J. Bananaworld : Quantum mechanics for primates. Oxford University Press, 2016. (Cité en page 14.)

[Buffoni 2022] Buffoni, L., Gherardini, S., Zambrini Cruzeiro, E. et Omar, Y. *Third Law of Thermodynamics and the Scaling of Quantum Computers.* Phys. Rev. Lett., vol. 129, page 150602, Oct 2022. [Online] (Cité en page 96.)
`https://link.aps.org/doi/10.1103/PhysRevLett.129.150602`.

[Bultel 2022] Bultel, X. et Onete, C. *Pairing-free secure-channel establishment in mobile networks with fine-grained lawful interception.* Dans Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing, pages 968–970, 2022. (Cité en page 59.)

[Bump 2022] Bump, P. *Analysis / Timeline : How two extremist groups planned for Jan. 6.* Washington Post, 2022. [Online] (Cité en page 28.)
`https://www.washingtonpost.com/politics/2022/03/15/timeline-how-two-extremist-groups-planned-jan-6/`.

[Canada 2019] Canada. *Criminal Code R.S., 1985, c.C-46, s.184 1993, c.40, s.3 2004, c.12, s.4 2019, c.25, s.64.* 2019. [Online] (Cité en page 58.)
`https://laws-lois.justice.gc.ca/eng/acts/C-46/page-26.html#s-184.2`. From : Government of Canada, Accessed : 2023-01-14.

[Castryck 2022] Castryck, W. et Decru, T. *An efficient key recovery attack on SIDH (preliminary version).* 2022. [Online] (Cité en pages 65 et 67.)
`https://eprint.iacr.org/2022/975`.
`https://eprint.iacr.org/2022/975`.

[Cattafesta 2021] Cattafesta, F. *Une nouvelle loi européenne veut la peau du chiffrement des messageries.* 2021. [Online] (Cité en page 28.)
`https://www.macg.co/services/2021/07/une-nouvelle-loi-europeenne-veut-la-peau-du-chiffrement-des-messageries-122913`.

[Cayre 2021] Cayre, R., Galtier, F., Auriol, G., Nicomette, V., Kaâniche, M. et Marconato, G. *InjectaBLE : Injecting malicious traffic into established Bluetooth Low Energy connections.* Dans 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pages 388–399. IEEE, 2021. (Cité en page 13.)

[Chase 2007] Chase, M. *Multi-authority Attribute Based Encryption.* Theory of Cryptography, pages 515–534, 2007. (Cité en page 53.)

[Chen 2021] Chen, C., Danba, O., Hoffstein, J., Hülsing, A., Rijneveld, J., Saito, T., Schanck, J., Schwabe, P., Whyte, W., Xagawa, K.*et al. NTRU : A submission to the NIST post-quantum standardization effort*, 2021. (Cité en pages 127 et 144.)

[Childs 2014] Childs, A., Jao, D. et Soukharev, V. *Constructing elliptic curve isogenies in quantum subexponential time.* Journal of Mathematical Cryptology, vol. 8, no. 1, pages 1–29, 2014. (Cité en page 67.)

[Clauser 1969] Clauser, J. F., Horne, M. A., Shimony, A. et Holt, R. A. *Proposed Experiment to Test Local Hidden-Variable Theories.* Phys. Rev. Lett., vol. 23, pages 880–884, Oct 1969. [Online] (Cité en page 15.)
`https://link.aps.org/doi/10.1103/PhysRevLett.23.880`.

[Costello 2010] Costello, C. et Stebila, D. *Fixed Argument Pairings.* Dans Abdalla, M. et Barreto, P. S. L. M., editeurs, Progress in Cryptology – LATINCRYPT 2010, pages 92–108, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. (Cité en page 49.)

[Costello 2016] Costello, C., Longa, P. et Naehrig, M. *Efficient Algorithms for Supersingular Isogeny Diffie-Hellman.* Dans Robshaw, M. et Katz, J., editeurs, Advances in Cryptology – CRYPTO 2016, pages 572–601, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. (Cité en page 65.)

[Couveignes 2006] Couveignes, J. M. *Hard Homogeneous Spaces.* ., 2006. (Cité en page 66.)

[Cryptage 2006] Cryptage. *La lettre de George Sand.* 2006. [Online] (Cité en page 5.)
`http://www.cryptage.org/lettre-george-sand.html`.

[Dai 2010] Dai, W. *Commutative-like encryption : a new characterization of ElGamal.* arXiv preprint arXiv :1011.3718, 2010. (Cité en pages 62, 64, 67 et 93.)

[Dash 2018] Dash, A., Sarmah, D., Behera, B. K. et Panigrahi, P. K. *Exact search algorithm to factorize large biprimes and a triprime on IBM quantum computer.* 2018. [Online] (Cité en page 96.)
`https://arxiv.org/abs/1805.10478`.

[de Leon 2021] de Leon, N. P., Itoh, K. M., Kim, D., Mehta, K. K., Northup, T. E., Paik, H., Palmer, B., Samarth, N., Sangtawesin, S. et Steuerman, D. W. *Materials challenges and opportunities for quantum computing hardware.* Science, vol. 372, no. 6539, page eabb2823, 2021. (Cité en page 96.)

[Desmedt 1995] Desmedt, Y. *Securing Traceability of Ciphertexts — Towards a Secure Software Key Escrow System.* Dans Guillou, L. C. et Quisquater, J.-J., editeurs, Advances in Cryptology — EUROCRYPT '95, pages 147–157, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. (Cité en page 58.)

[Dingledine 2004] Dingledine, R., Mathewson, N. et Syverson, P. *Tor : The Second-Generation Onion Router :.* Rapport technique, Defense Technical Information Center, Fort Belvoir, VA, janvier 2004. (Cité en pages 12 et 31.)

[Eckert 2018] Eckert, C., Wang, X., Wang, J., Subramaniyan, A., Iyer, R., Sylvester, D., Blaaauw, D. et Das, R. *Neural cache : Bit-serial in-cache acceleration of deep neural networks.* Dans 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA), pages 383–396. IEEE, 2018. (Cité en pages 98, 115, 122 et 128.)

[ElGamal 1985] ElGamal, T. *A public key cryptosystem and a signature scheme based on discrete logarithms.* IEEE transactions on information theory, vol. 31, no. 4, pages 469–472, 1985. (Cité en page 18.)

[Ettinger 1999] Ettinger, M. et Hoyer, P. *A Quantum Observable for the Graph Isomorphism Problem.* janvier 1999. [Online] (Cité en page 21.) `http://arxiv.org/abs/quant-ph/9901029`. arXiv :quant-ph/9901029.

[Eur-Lex 2016] Eur-Lex. *General Data Protection Regulation.* 2016. [Online] (Cité en page 11.) `ELI:http://data.europa.eu/eli/reg/2016/679/2016-05-04`.

[Evans 2009] Evans, N. S., Dingledine, R. et Grothoff, C. *A Practical Congestion Attack on Tor Using Long Paths.* Dans USENIX Security Symposium, pages 33–50, 2009. (Cité en page 33.)

[Fassinou 2021] Fassinou, B. *Le Parlement européen approuve la surveillance massive des communications privées, malgré la mise en garde sur les risques pour la sécurité et la confidentialité.* 2021. [Online] (Cité en page 28.) `https://www.developpez.com/actu/316563/Le-Parlement-europeen-approuve-la-surveillance-massive-des-communications-privees-malgre-la-mise-en-garde-sur-les-risques-pour-la-securite-et-la-confidentialite/`.

[Fröhlich 2017] Fröhlich, B., Lucamarini, M., Dynes, J. F., Comandar, L. C., Tam, W. W.-S., Plews, A., Sharpe, A. W., Yuan, Z. et Shields, A. J. *Long-distance quantum key distribution secure against coherent attacks.* Optica, vol. 4, no. 1, page 163, jan 2017. [Online] (Cité en page 16.) `https://doi.org/10.1364%2Foptica.4.000163`.

[Fujiki 2019] Fujiki, D., Mahlke, S. et Das, R. *Duality cache for data parallel acceleration.* Dans Proceedings of the 46th International Symposium on Computer Architecture, pages 397–410, 2019. (Cité en pages 98 et 102.)

[Galbraith 2018] Galbraith, S., Panny, L., Smith, B. et Vercauteren, F. *Quantum Equivalence of the DLP and CDHP for Group Actions*, 2018. `https://eprint.iacr.org/2018/1199`. (Cité en page 66.)

[Gallagher 2014] Gallagher, K. *You Can Strengthen Anonymity, Privacy and Free Speech on the Internet by Running a Tor Relay.* 2014. [Online] (Cité en page 30.) `https://freedom.press/news/you-can-strengthen-anonymity-privacy-and-free-speech-on-the-internet-by-running-a-tor-relay/`.

[Galov 2022] Galov, N. *19 Internet Privacy Statistics to Be Paranoid About in 2022*. 2022. [Online] (Cité en page 23.)
https://webtribunal.net/blog/internet-privacy-statistics/.

[Gaudiaut 2021] Gaudiaut, T. *Infographie : Telegram et Signal à l'assaut de WhatsApp*. 2021. [Online] (Cité en page 28.)
https://fr.statista.com/infographie/23954/nombre-telechargements-par-jour-applications-messagerie-telegram-signal-whatsapp/.

[Goyal 2006] Goyal, V., Pandey, O., Sahai, A. et Waters, B. *Attribute-based encryption for fine-grained access control of encrypted data*. Proceedings of the ACM Conference on Computer and Communications Security, pages 89–98, 01 2006. [Online]. Available: https://dx.doi.org/10.1145/1180405.1180418. (Cité en pages 53 et 65.)

[Grover 1997] Grover, L. K. *Quantum Mechanics Helps in Searching for a Needle in a Haystack*. Phys. Rev. Lett., vol. 79, pages 325–328, Jul 1997. [Online] (Cité en page 96.)
https://link.aps.org/doi/10.1103/PhysRevLett.79.325.

[Han 2011] Han, K., Yeun, C. Y., Shon, T., Park, J. et Kim, K. *A scalable and efficient key escrow model for lawful interception of IDBC-based secure communication*. International Journal of Communication Systems, vol. 24, no. 4, pages 461–472, 2011. (Cité en page 58.)

[Hinek 2010] Hinek, M. J. et Lam, C. C. *Common modulus attacks on small private exponent RSA and some fast variants (in practice)*. Journal of Mathematical Cryptology, vol. 4, no. 1, pages 58–93, 2010. (Cité en page 64.)

[Houmansadr 2013] Houmansadr, A., Brubaker, C. et Shmatikov, V. *The parrot is dead : Observing unobservable network communications*. Dans 2013 IEEE Symposium on Security and Privacy, pages 65–79. IEEE, 2013. (Cité en page 33.)

[Huang 2012] Huang, K. et Tso, R. *A commutative encryption scheme based on ElGamal encryption*. Dans 2012 International Conference on Information Security and Intelligent Control, pages 156–159. IEEE, 2012. (Cité en pages 62, 64 et 93.)

[IBM 2021] IBM. *IBM annonce avoir atteint la suprématie quantique*. novembre 2021. [Online] (Cité en page 96.)
https://www.20minutes.fr/high-tech/3178515-20211122-ibm-affirme-avoir-atteint-une-veritable-suprematie-quantique.

[J. Bethencourt 2007] J. Bethencourt, B. W., A. Sahai. *Ciphertext-Policy Attribute-Based Encryption*. IEEE Symposium on Security and Privacy (SP '07)„ 2007. hal-01788815. (Cité en page 65.)

[Jansen 2019] Jansen, R., Vaidya, T. et Sherr, M. *Point break : A study of bandwidth {Denial-of-Service} attacks against tor*. Dans 28th USENIX security symposium (USENIX Security 19), pages 1823–1840, 2019. (Cité en page 33.)

[Jia 2011] Jia, H.-Y., Wen, Q.-Y., Song, T.-T. et Gao, F. *Quantum protocol for millionaire problem.* Optics communications, vol. 284, no. 1, pages 545–549, 2011. (Cité en page 94.)

[Kent 1979] Kent, S. T. *Protocol design considerations for network security.* Dans Interlinking of Computer Networks, pages 239–259. Springer, 1979. (Cité en page 58.)

[Khayat 2008] Khayat, S. H. *Using commutative encryption to share a secret.* Electrical Engineering Department, Ferdowsi University of Mashhad, Iran, 2008. (Cité en pages 62 et 64.)

[Kohútka 2017] Kohútka, L. et Stopjaková, V. *A new efficient sorting architecture for real-time systems.* Dans 2017 6th Mediterranean Conference on Embedded Computing (MECO), pages 1–4, juin 2017. (Cité en page 122.)

[KULeuven-COSIC 2021] KULeuven-COSIC. *SABER - SABER is a Module-LWR based KEM submitted to NIST*, Dec 2021. (Cité en page 129.)

[Kumar 2015] Kumar, S. *Fundamental limits to Moore's law.* arXiv preprint arXiv :1511.05956, 2015. (Cité en page 100.)

[Kuperberg 2005a] Kuperberg, G. *A subexponential-time quantum algorithm for the dihedral hidden subgroup problem.* SIAM Journal on Computing, vol. 35, no. 1, pages 170–188, 2005. Publisher : SIAM. (Cité en page 21.)

[Kuperberg 2005b] Kuperberg, G. *A subexponential-time quantum algorithm for the dihedral hidden subgroup problem.* SIAM Journal on Computing, vol. 35, no. 1, pages 170–188, 2005. (Cité en page 67.)

[Kuperberg 2011a] Kuperberg, G. *Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem.* arXiv preprint arXiv :1112.3333, 2011. (Cité en page 21.)

[Kuperberg 2011b] Kuperberg, G. *Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem.* arXiv preprint arXiv :1112.3333, 2011. (Cité en page 67.)

[Kuzmin 2017] Kuzmin, A., Markov, V., Mikhalev, A., Mikhalev, A. et Nechaev, A. *Cryptographic algorithms on groups and algebras.* Journal of Mathematical Sciences, vol. 223, no. 5, pages 629–641, 2017. (Cité en page 65.)

[Köpf 2007] Köpf, B. et Basin, D. *An information-theoretic model for adaptive side-channel attacks.* Dans Proceedings of the 14th ACM conference on Computer and communications security - CCS '07, page 286, Alexandria, Virginia, USA, 2007. ACM Press. (Cité en page 136.)

[Larsson 1957] Larsson, C. *Telecom Companies as Crime Investigators.* Stockholm Institute for Scandianvian Law 1957-2010, 1957. (Cité en page 58.)

[Lesage 2019] Lesage, N. *Qu'est-ce que la « suprématie quantique » que Google aurait atteinte ?* septembre 2019. [Online] (Cité en page 96.)

`https://www.numerama.com/tech/550760-quest-ce-que-la-suprematie-quantique-que-google-aurait-atteinte.html`.

[Li 1994] Li, Y. X., Deng, R. H. et Wang, X. M. *On the equivalence of McEliece's and Niederreiter's public-key cryptosystems.* IEEE Transactions on Information Theory, vol. 40, no. 1, pages 271–273, 1994. (Cité en page 107.)

[Liang 2010] Liang, X., Lu, R., Lin, X. et Shen, X. S. *Ciphertext policy attribute based encryption with efficient revocation.* TechnicalReport, University of Waterloo, 2010. (Cité en page 53.)

[Liu 2010] Liu, Z., Cao, Z. et Wong, D. S. *Efficient Generation of Linear Secret Sharing Scheme Matrices from Threshold Access Trees*, 2010. (Cité en pages 51 et 88.)

[Liu 2018a] Liu, J. K., Yuen, T. H., Zhang, P. et Liang, K. *Time-Based Direct Revocable Ciphertext-Policy Attribute-Based Encryption with Short Revocation List.* Applied Cryptography and Network Security, pages 516–534, 2018. (Cité en page 53.)

[Liu 2018b] Liu, M., Nanda, P., Zhang, X., Yang, C., Yu, S. et Li, J. *Asymmetric commutative encryption scheme based efficient solution to the millionaires' problem.* Dans 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pages 990–995. IEEE, 2018. (Cité en pages 62 et 64.)

[Liu 2022] Liu, H. et Jiang, R. *Efficient Revocable Attribute-Based Encryption from R-LWE in Cloud Storage.* Dans Li, X., editeur, Advances in Intelligent Automation and Soft Computing, Lecture Notes on Data Engineering and Communications Technologies, pages 1051–1058, Cham, 2022. Springer International Publishing. (Cité en pages 40 et 52.)

[Loss 1998] Loss, D. et DiVincenzo, D. P. *Quantum Computation with Quantum Dots.* Physical Review A, vol. 57, no. 1, pages 120–126, janvier 1998. [Online] (Cité en page 14.)
`http://arxiv.org/abs/cond-mat/9701055`. arXiv :cond-mat/9701055.

[Lovejoy 2020] Lovejoy, B. *Senators once more try to ban end-to-end encryption ; still don't understand it.* juin 2020. [Online] (Cité en page 28.)
`https://9to5mac.com/2020/06/24/ban-end-to-end-encryption/`.

[Machkovech 2022] Machkovech, S. *Intel : "Moore's law is not dead" as Arc A770 GPU is priced at $329.* septembre 2022. [Online] (Cité en page 100.)
`https://arstechnica.com/gadgets/2022/09/the-intel-arc-a770-gpu-launches-october-12-for-329/`.

[Macnish 2023] Macnish, K. *Surveillance Ethics.* 2023. [Online] (Cité en page 29.)
`https://iep.utm.edu/surv-eth/`.

[Makrushin 2017] Makrushin, D. *The cost of launching a DDoS attack.* 2017. [Online] (Cité en page 33.)
`https://securelist.com/the-cost-of-launching-a-ddos-attack/77784/`.

[McEliece 1978] McEliece, R. J. *A public-key cryptosystem based on algebraic.* Coding Thv, vol. 4244, pages 114–116, 1978. (Cité en page 105.)

[Migliore 2019] Migliore, V., Gérard, B., Tibouchi, M. et Fouque, P.-A. *Masking Dilithium*. Dans Deng, R. H., Gauthier-Umaña, V., Ochoa, M. et Yung, M., editeurs, Applied Cryptography and Network Security, pages 344–362, Cham, 2019. Springer International Publishing. (Cité en pages 25 et 142.)

[Migliore 2022] Migliore, V. et Nugier, C. *RISC-V simulator with PIM operations*. 2022. [Online] (Cité en pages 99 et 118.)
`https://gitlab.laas.fr/vmiglior/risc-v-simulator`.

[Moldovyan 2010] Moldovyan, D. *Non-commutative finite groups as primitive of public key cryptosystems*. Quasigroups and Related Systems, vol. 18, no. 2, pages 165–176, 2010. (Cité en page 65.)

[Moldovyan 2019] Moldovyan, D. N., Moldovyan, A. A., Phieu, H. N. et Nguyen, M. H. *Post-quantum commutative encryption algorithm*. Dans Context-Aware Systems and Applications, and Nature of Computation and Communication, pages 205–214. Springer, 2019. (Cité en pages 62, 65 et 67.)

[Moore 1998] Moore, G. E. *Cramming more components onto integrated circuits*. Proceedings of the IEEE, vol. 86, no. 1, pages 82–85, 1998. (Cité en page 100.)

[Müller 2008] Müller, S., Katzenbeisser, S. et Eckert, C. *Distributed Attribute-Based Encryption*. Dans ICISC, 2008. (Cité en page 53.)

[Nayak 2008] Nayak, C., Simon, S. H., Stern, A., Freedman, M. et Sarma, S. D. *Non-Abelian Anyons and Topological Quantum Computation*. Reviews of Modern Physics, vol. 80, no. 3, pages 1083–1159, septembre 2008. [Online] (Cité en page 14.)
`http://arxiv.org/abs/0707.1889`. arXiv :0707.1889 [cond-mat].

[Newlin 2016] Newlin, M. *MouseJack, KeySniffer and Beyond : Keystroke Sniffing and Injection Vulnerabilities in 2.4 GHz Wireless Mice and Keyboards*. DEFCON, 2016. (Cité en page 13.)

[Niederreiter 1986] Niederreiter, H. *Knapsack-type cryptosystems and algebraic coding theory*. Prob. Contr. Inform. Theory, vol. 15, no. 2, pages 157–166, 1986. (Cité en page 107.)

[NIST 2019] NIST. *Round 2 of the NIST PQC "Competition" - What was NIST Thinking ?*, 2019. Accessed : 2021-08-01. (Cité en page 97.)

[NIST 2022a] NIST. *PQC Standardization Process : Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates*, 2022. Accessed : 2022-09-6. (Cité en pages 68 et 97.)

[NIST 2022b] NIST. *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. 2022. [Online] (Cité en page 133.)
`https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf`.

[NJIT 2021] NJIT. *PALISADE Lattice Cryptography Library (release 1.11.2)*, mai 2021. (Cité en page 53.)

[Nugier 2019] Nugier, C., Remi, A., Vincent, M. et Eric, A. *Multi-Locking and Perfect Argument Order : Two Major Improvements of Attribute-Based Encryption (Long Paper)*. 2019. [Online] (Cité en pages 49 et 65.)
`https://eprint.iacr.org/2019/1219`.
`https://eprint.iacr.org/2019/1219`.

[Nugier 2022] Nugier, C. et Alata, E. *Brevet prioritaire France FR2205615*, 10/06/2022. (Cité en page 36.)

[Nuñez 2019] Nuñez, D., Agudo, I. et Lopez, J. *Escrowed decryption protocols for lawful interception of encrypted data*. IET Information Security, vol. 13, no. 5, pages 498–507, 2019. [Online] (Cité en page 59.)
`https:`
`//onlinelibrary.wiley.com/doi/abs/10.1049/iet-ifs.2018.5082`.
_eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1049/iet-ifs.2018.5082.

[O'donnell 2018] O'donnell, R. *15-859BB : Quantum Computation and Quantum Information 2018*. 2018. [Online] (Cité en page 14.)
`https://www.cs.cmu.edu/~odonnell/quantum18/`.

[Panchenko 2012] Panchenko, A., Lanze, F. et Engel, T. *Improving performance and anonymity in the Tor network*. Dans 2012 IEEE 31st International Performance Computing and Communications Conference (IPCCC), pages 1–10. IEEE, 2012. (Cité en page 34.)

[Peng 2019] Peng, W., Wang, B., Hu, F., Wang, Y., Fang, X., Chen, X. et Wang, C. *Factoring larger integers with fewer qubits via quantum annealing with optimized parameters*. SCIENCE CHINA Physics, Mechanics & Astronomy, vol. 62, no. 6, pages 1–8, 2019. (Cité en page 18.)

[Pries 2008] Pries, R., Yu, W., Fu, X. et Zhao, W. *A new replay attack against anonymous communication networks*. Dans 2008 IEEE International Conference on Communications, pages 1578–1582. IEEE, 2008. (Cité en page 33.)

[Project 2022] Project, T. T. *Welcome to Tor Metrics*. 2022. [Online] (Cité en page 30.)
`https://metrics.torproject.org/`.

[Regev 2003] Regev, O. *Quantum Computation and Lattice Problems*. avril 2003. [Online] (Cité en page 21.)
`http://arxiv.org/abs/cs/0304005`. arXiv :cs/0304005.

[Regev 2004a] Regev, O. *A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space*. arXiv preprint quant-ph/0406151, 2004. (Cité en page 21.)

[Regev 2004b] Regev, O. *A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space*. arXiv preprint quant-ph/0406151, 2004. (Cité en page 67.)

[Rieck 2022] Rieck, K. *Influential Security Papers*. 2022. [Online] (Cité en page 18.)
`https://www.sec.tu-bs.de/~konrieck/topnotch/crypto_papers.html`.

[Rosenzweig 1946] Rosenzweig, M. L. *Law of Wire Tapping.* Cornell LQ, vol. 32, page 514, 1946. (Cité en page 58.)

[Ruiz 2020] Ruiz, P. *Sécurité : le Conseil de l'UE s'apprête à adopter une résolution visant à forcer l'introduction de portes dérobées au sein des applications de messagerie chiffrées pour lutter contre la pédophilie.* 2020. [Online] (Cité en page 28.)
`https://securite.developpez.com/actu/310269/Securite-le-Conseil-de-l-UE-s-apprete-a-adopter-une-resolution-visant-a-forcer-l-introduction-de-portes-derobees-au-sein-des-applications-de-messagerie-chiffrees-pour-lutter-contre-la-pedophilie/`.

[Sahai 2012] Sahai, A., Seyalioglu, H. et Waters, B. *Dynamic credentials and ciphertext delegation for attribute-based encryption.* Dans Annual Cryptology Conference, pages 199–217. Springer, 2012. (Cité en page 92.)

[Saleh 2018] Saleh, S., Qadir, J. et Ilyas, M. U. Shedding light on the dark corners of the internet : A survey of tor research, volume 114. Elsevier, 2018. (Cité en page 31.)

[Schwabe 2021] Schwabe, P. *Crystals-cryptographic suite for algebraic lattices*, Dec 2021. (Cité en pages 97, 127 et 144.)

[Scott 2011] Scott, M. *On the Efficient Implementation of Pairing-Based Protocols.* Dans Chen, L., editeur, Cryptography and Coding, pages 296–308, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. (Cité en page 49.)

[SemiColonWeb 2019] SemiColonWeb et Rodon, J. *L'apocalypse quantique est-elle pour demain ?* juin 2019. [Online] (Cité en page 96.)
`https://blog.revolve.team/2019/06/20/ordinateur-quantique-rsa/`.

[Shamir 1979] Shamir, A. *How to share a secret.* Communications of the ACM, vol. 22, no. 11, pages 612–613, 1979. (Cité en pages 51 et 88.)

[Shamir 1980] Shamir, A. *On the power of commutativity in cryptography.* Dans International Colloquium on Automata, Languages, and Programming, pages 582–595. Springer, 1980. (Cité en page 62.)

[Shamir 1981] Shamir, A., Rivest, R. L. et Adleman, L. M. *Mental poker.* Dans The mathematical gardner, pages 37–43. Springer, 1981. (Cité en pages 12 et 94.)

[Shor 1999] Shor, P. W. *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer.* SIAM review, vol. 41, no. 2, pages 303–332, 1999. (Cité en pages 18, 66 et 96.)

[Shoup 2000] Shoup, V. *Using hash functions as a hedge against chosen ciphertext attack.* Dans International Conference on the Theory and Applications of Cryptographic Techniques, pages 275–288. Springer, 2000. (Cité en page 84.)

[Siegl 2016] Siegl, P., Buchty, R. et Berekovic, M. *Data-centric computing frontiers : A survey on processing-in-memory.* Dans Proceedings of the Second International Symposium on Memory Systems, pages 295–308, 2016. (Cité en pages 22 et 100.)

[SIMMONS 1983] SIMMONS, G. J. *A "WEAK" PRIVACY PROTOCOL USING THE RSA CRYPTO ALGORITHM.* Cryptologia, vol. 7, no. 2, pages 180–182, 1983. [Online] (Cité en page 64.)
https://doi.org/10.1080/0161-118391857900.

[Singh 1999] Singh, S. et Coqueret, C. Histoire des codes secrets : de l'Egypte des Pharaons à l'ordinateur quantique. Le livre de poche. J.-C. Lattès, 1999. (Cité en pages 5 et 11.)

[Smith 2018] Smith, B. *Pre- and post-quantum Diffie–Hellman from groups, actions, and isogenies*, 2018. https://eprint.iacr.org/2018/882. (Cité en pages 65 et 66.)

[Standaert 2010] Standaert, F.-X. Introduction to side-channel attacks, page 27–42. Springer US, Boston, MA, 2010. (Cité en pages 131, 133 et 134.)

[streamfizz 2020] streamfizz. *Les statistiques 2020 sur la diffusion de direct sur internet.* janvier 2020. [Online] (Cité en page 35.)
https://www.streamfizz.com/les-statistiques-a-connaitre-sur-la-diffusion-en-direct-live-streaming/.

[Sulaiman 2013] Sulaiman, M. A. et Zhioua, S. *Attacking tor through unpopular ports.* Dans 2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops, pages 33–38. IEEE, 2013. (Cité en page 33.)

[teambikesuite.org 2022] teambikesuite.org. *BIKE - Bit Flipping Key Encapsulation*, Jan 2022. (Cité en pages 97, 126 et 144.)

[Truelist 2022] Truelist. *Tor Stats - TrueList 2022.* 2022. [Online] (Cité en page 30.)
https://truelist.co/blog/tor-stats/.

[Tue 2022] Tue, J. t. . a.-T. C. *UK Government Apparently Hoping It Can Regulate End-To-End Encryption Out Of Existence.* janvier 2022. [Online] (Cité en page 28.)
https://www.techdirt.com/2022/01/11/uk-government-apparently-hoping-it-can-regulate-end-to-end-encryption-out-existence/.

[Untersinger 2018] Untersinger, M. *L'affaire Cambridge Analytica plonge Facebook dans une crise historique.* Le Monde.fr, mars 2018. [Online] (Cité en page 28.)
https://www.lemonde.fr/pixels/article/2018/03/20/l-affaire-cambridge-analytica-plonge-facebook-dans-une-crise-historique_5273376_4408996.html.

[Viméo 2020] Viméo. *Streaming Stats | 47 Must-Know Live Video Streaming Statistics.* janvier 2020. [Online] (Cité en page 23.)
https://livestream.com/blog/62-must-know-stats-live-video-streaming.

[Wang 2020] Wang, G., Wan, M., Liu, Z. et Gu, D. *Dual System in Lattice : Fully Secure ABE from LWE Assumption*, 2020.
https://eprint.iacr.org/2020/064. (Cité en page 90.)

[Wang 2022] Wang, Y., Shacham, H., Paccagnella, R., Fletcher, C. W., He, E. T. et Kohlbrenner, D. *Hertzbleed : Turning Power Side-Channel Attacks Into Remote Timing Attacks on x86*, 2022. (Cité en pages 14 et 133.)

[Winter 2012] Winter, P. et Lindskog, S. How the great firewall of china is blocking tor. USENIX-The Advanced Computing Systems Association, 2012. (Cité en page 32.)

[Witkowski 2022] Witkowski, W. *'Moore's Law's dead,' Nvidia CEO Jensen Huang says in justifying gaming-card price hike*. 2022. [Online] (Cité en page 100.) `https://www.marketwatch.com/story/moores-laws-dead-nvidia-ceo-jensen-says-in-justifying-gaming-card-price-hike-11663798618`. Section : Industries.

[Xu 2019] Xu, D. *Commutative encryption and data hiding in HEVC video compression*. IEEE Access, vol. 7, pages 66028–66041, 2019. (Cité en pages 64 et 93.)

[Zeutro 2018] Zeutro. *openabe*, 2018. (Cité en pages 40 et 52.)

[Zhang 2011] Zhang, X. *Reversible data hiding in encrypted image*. IEEE signal processing letters, vol. 18, no. 4, pages 255–258, 2011. (Cité en pages 64 et 93.)

[Zhang 2013] Zhang, X. *Commutative reversible data hiding and encryption*. Security and Communication Networks, vol. 6, no. 11, pages 1396–1403, 2013. (Cité en pages 64 et 93.)

Adaptation d'Outils Cryptographiques pour un Contexte Post-Quantique

Résumé :

La cryptographie est utilisée pour protéger la vie privée de tous quotidiennement lors de la navigation sur Internet. On observe de multiples changements en cours ces dernières années, dûs en partie à la possible arrivée prochainement d'ordinateurs quantiques capables de mettre en danger certains de nos chiffrements actuels.

On remarque quatre principaux enjeux associés à ces changements :

1. L'adaptation des primitives, qui doivent être remplacées par de nouvelles, post-quantiques.
2. L'adaptation des protocoles, qui parfois reposent aussi sur des propriétés vulnérables aux ordinateurs quantiques.
3. L'évolution des habitudes des utilisateurs, pour lesquels sont conçus ces protocoles, notamment impliquant une consommation numérique plus importante (flux vidéos en direct) et une demande de protection accrue de la vie privée.
4. Les capacités des ordinateurs, qui évoluent en permanence (par exemple grâce à des accélérateurs matériels), souvent à pour d'autres applications (IA, jeux), peuvent offrir des surfaces d'attaques, notamment par canaux auxiliaires.

La présente thèse soutient que ces enjeux sont interdépendants et qu'une vision groupée de ces problèmes aide à garantir l'adaptation de la cryptographie à l'ère post-quantique.

Ce propos est appuyé par trois pistes de recherche :

1. Proposer un protocole anonymisant de live-streaming massif (contraintes utilisateurs) qui soit aussi post-quantique (contraintes sur les primitives).
2. Étudier les protocoles de chiffrements commutatifs, en particulier d'interception réglementée, ainsi que la manière de les rendre post-quantiques sans créer de faiblesses ou être limités à des sur-chiffrements.
3. Comprendre la manière dont les architectures dotées de calcul en mémoire vont affecter les performances des primitives post-quantiques, et comment un co-design est possible de manière à aussi réduire les fuites par canaux auxiliaires.

Mots clés : Anonymisation, Vie Privée, Optimisation, Calcul en mémoire, Canaux Auxiliaires, Post-quantique.

Adapting Cryptographic Tools to a Post-Quantum Context

Abstract :

Cryptography is used to protect everyone's privacy on a daily basis when browsing the Internet. In recent years, several changes have been taking place, partly due to the possible arrival of quantum computers capable of endangering some of our current cryptographic systems.

We identify four issues associated with these changes :

1. The adaptation of primitives, which must be replaced by new, post-quantum ones.
2. The adaptation of protocols, which sometimes also rely on properties vulnerable to quantum computers.
3. The evolution of the end-users habits, in their use of digital tools, as they tend to consum more live video streams but at the same demand stronger privacy protection.
4. Computers capabilities are constantly evolving (e.g. through hardware accelerators), often due to other applications (AI, games), however these evolutions can offer attack surfaces, in particular by auxiliary channels.

This PhD thesis argues that these issues are interdependent and that a global consideration of these problems helps to ensure that cryptography can be adapted to the post-quantum era. For that purpose, three main contributions are proposed, that consist in :

1. Proposing an anonymizing protocol for massive live-streaming (user constraints) which is also post-quantum (constraints on primitives).
2. Studying commutative encryption protocols, in particular lawful interception ones, in order to make them post-quantum without creating weaknesses or being limited to super-encryption.
3. Understanding how architectures featuring in-memory computing will affect the performance of post-quantum primitives, and how co-design is possible so as to also reduce auxiliary channel leakage.

Keywords : Anonymity, Privacy, Optimization, Processing in memory, Side Channel, Post-quantum.