UNIVERSITÉ DU QUÉBEC À MONTRÉAL

OPÉRATIONS NON RUDIMENTAIRES DE TABLES DE HACHAGE DISTRIBUÉES ET CLAVARDAGE EN GROUPE SÛR BOUT-EN-BOUT

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

SIMON DÉSAULNIERS

AVRIL 2019

REMERCIEMENTS

Je tiens à remercier mes anciens collègues chercheurs et concepteurs de logiciels pour leur accueil chaleureux et leur enthousiasme distingué au travail. C'est dans l'ambiance réalisée par la présence de gens comme Andreas Traczyk, Alexandre Viau, Guillaume Roguez et Adrien Béraud que j'ai pu m'épanouir et connaître l'expérience de travail dans une équipe RD sur des sujets passionnants comme les systèmes distribués, et ce en ayant le souci du respect de l'utilisateur en développant des solutions libres.

Je remercie mes superviseurs de m'avoir épaulé tout au long de mon cheminement. Je suis reconnaissant pour l'apport de Sébastien Gambs dans mes démarches à me perfectionner sur le plan de la recherche de documentation. Son expérience en sécurité des systèmes informatisés m'a judicieusement guidé dans la rédaction de ce mémoire. De plus, Alexandre Blondin Massé a été d'une aide indispensable et essentielle dans la relecture de mes écrits et l'application de rigueur. En particulier, je lui dois ma reconnaissance pour ses conseils en rédaction formelle de sujets mathématiques et algorithmiques. Finalement, ses connaissances en pratiques de conception logicielle ont permis une communication et une productivité inégale entre nous.

Les bases solides pour l'étude de la sécurité des schémas cryptographiques m'ont été transmises de manière rigoureuse par l'enseignement de Louis Salvail. Je lui manifeste donc ma sincère gratitude.

Je ne peux me permettre d'oublier Jaël Gareau qui a été un collègue hors pair avec lequel j'ai pu m'entretenir sur des sujets pointus en mathématiques et infor-

matique. Il a aussi démontré sa dévotion à l'autre en m'attribuant un accès SSH à sa machine afin de me permettre le peaufinage de mon texte au moyen de son logiciel Antidote 10. Pour cela, je lui suis très reconnaissant.

Je remercie évidemment mes amis qui m'ont soutenu dans le périple d'une vie à Montréal. Je mentionne affectueusement William Lamontagne et Marie-Laurence Audet pour leur chaleureuse compagnie. Je note aussi la présence constante de Pharah Lallmahomed depuis la Belgique. Je n'oublie bien sûr pas mes amis d'enfance Étienne Roy-Bourque et Jean-Sébsatien Mathon qui m'ont toujours montré leur appui le plus sincère. Évidemment, je souhaite également remercier tous mes amis dans l'ensemble.

Finalement, j'ai la plus haute gratitude envers mon père qui m'a transmis l'esprit de rigueur et l'intérêt pour les mathématiques.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

TABLE DES FIGURES

RÉSUMÉ

L'ère des télécommunications que connaît le monde d'aujourd'hui a révolutionné la façon dont on communique entre tous. Les immenses câbles sous-marins qui relient les grands centres nationaux font transiter des quantités gigantesques de données de toutes sortes. Autant des transactions bancaires, des messages personnels, du contenu illicite, des signaux d'alerte ou d'appel à l'aide circulent sur le réseau Internet sans souci des blocs fondamentaux érigés pour soutenir ce vaste orchestre communicationnel. Sachant que ces données sont expédiées dans les quatre coins du monde, il est légitime de se demander où vont donc celles-ci. Ou encore, qui peut les lire ? Ou bien peut-on censurer (ou altérer) ces correspondances ? Comment est-on protégé ? Ces interrogations exigent de prêter attention à plusieurs dimensions du problème telles que le mode d'acheminement de ces données ainsi que l'état dans lequel celles-ci voyagent. Le modèle utilisé pour répondre aux questions susmentionnées est bien connu et sert de convention pour les développements actuels des systèmes de communication. Cette convention est celle des réseaux centralisés. La Toile ainsi que le calcul infonuagique en sont des exemples. Tant en termes de transport de données que sur le plan de la sécurité de celles-ci, le même principe prévaut : la centralisation. Depuis plusieurs années, on voit se construire un argumentaire stipulant que ces pratiques échouent considérablement à préserver la vie privée des gens. Par moment, les données des utilisateurs sont même exploitées à leur insu et dans une perspective marchande. En particulier, l'état des communications en direct entre usagers, vulnérables à l'espionnage, appelle à l'urgence au changement. Ce faisant, le clavardage respectueux de la vie privée fait l'objet d'études constantes depuis quelques années. De plus, la topologie réseau — jouant un rôle clef dans la question de vulnérabilité à la censure — suscite elle aussi l'attention des chercheurs. Nous contribuons à ces deux sujets que nous trouvons tous deux essentiels à l'étude de la vie privée du clavardage. En premier lieu, nous présentons des travaux réalisés sur OpenDHT, une plateforme de stockage de données distribuée et libre. Plus particulièrement, nous nous attardons à la question d'indexation distribuée ainsi qu'à différentes optimisations des algorithmes de manutention. Dans un second temps, nous faisons l'état de l'art des différents protocoles de clavardage sûr bout en bout et amenons un enrichissement à l'un d'entre eux en ce qui a trait au problème de la capacité de liaison entre eux de différents messages d'un même émetteur.

RESUMO

En Espéranto — La epoko de telekomunikadoj en la hodiaŭa mondo revoluciis la manieron kiel ni komunikas. La grandegaj submaraj kabloj konektantaj la ĉefajn centrojn naciaj transdonas gigantajn kvantojn da datumoj de ĉiuj specoj. Bankaj transakcioj, personaj mesaĝoj, kontraŭleĝa enhavo, alarmaj signaloj aŭ alvoko por helpo ĉiuj cirkulas en la interreto sen zorgi pri la fundamentaj blokoj starigita por subteni ĉi tiun vastan komunikan orkestron. Sciante, ke ĉi tiuj datumoj estas sendataj en la kvar anguloj de la mondo, estas laŭleĝa demandi kien ili iras. Aŭ denove, kiu povas legi ilin? Aŭ ĉu ni povas cenzuri (aŭ ŝanĝi) ĉi tiujn korespondadojn? Kiel ni garditas? Ĉi tiuj demandoj postulas atenton al pluraj dimensioj de la problemo, kiel la rimedo per kiu ĉi tiuj datumoj veturas kaj la stato de ili vojaĝe. La modelo uzata por respondi la antaŭajn demandojn estas bone konata kaj servas kiel konvencio por aktualaj evoluoj en komunikadaj sistemoj. Ĉi tiu konvencio estas tiu de retoj centralizata. La Tut-Tera Teksaĵo kaj la nuba komputado estas ekzemploj. Laŭ ambaŭ la transporto kaj la sekureco de ĉi tiuj datumoj, la sama principo ĉefinfluas: la centralizado. Ekde pluraj jaroj antaŭe, ni vidas la konstruadon de argumento deklarante, ke ĉi tiuj praktikoj grave malsukcesas antaŭgardi la privatecon de homoj. Kelkfoje datumoj de utiligantoj estas uzataj sen ilia scio kaj laŭ merkata perspektivo. Aparte, la stato de senperaj komunikadoj inter uzantoj, vundeblaj je spionado, petu urĝecan ŝanĝon. Tiel, privateca retbabilado estas la temo de konstantaj studoj ekde kelkaj jaroj. Krome, reto topologio — ludanta ĉefa rolon en la temo de vundebleco je cenzuro — ankaŭ vekas la atenton de esploristoj. Ni kontribuas al ĉi tiuj du temoj, kiujn ni trovas esencaj por la studado de la privateco de la retbabilado. Unue, ni prezentas laborojn faritajn sur OpenDHT, libera platformo por la distribuado kaj stokado de datumoj. Precipe, ni pritraktas la problemon de indeksado distribuante kaj malsamaj optimizoj de stokantaj algoritmoj. Finfine, ni prezentas la plej aktualajn sciojn pri la diversaj fin-al-fine sekuraj retbabiladaj protokoloj kaj alportu riĉigon al unu el ili rilate al la problemo pri la kaplabeco ligante diversajn mesaĝojn de la sama sendanto. — *Fin du passage en Esperanto.*

Mots-clefs : Table de hachage distribuée, Kademlia, indexation, recherche, PHT, préfixe, hachage, arbre, trie, vie privée, protocole, ART, asynchronous ratcheting tree, paternité, non-transitivité, authentification, sécurité, bout en bout, E2E.

INTRODUCTION

L'humanité a connu plusieurs médias de communication. Dans un premier temps, il a été oral et écrit, par des moyens rudimentaires tels que le papier, et acheminé par courrier. Ensuite, le monde a connu une révolution des télécommunication en passant à l'ère du téléphone. Ce changement a eu un rôle déterminant dans les différentes activités humaines de toutes sortes. La seconde révolution récente a été amorcée par le numérique qui rend maintenant virtuellement possible la transmission de messages de façon instantanée presque partout sur terre et même jusqu'en orbite [28]. Comme on le sait, les possibilités ne se limitent pas aux communications par messages, mais s'étendent au divertissement (YouTube), le partage et la contribution dans la sphère informatique (GitHub), artistique (DevianArt), musical (BandCamp), d'archive d'information (MusicBrainz) et la documentation sur tous les sujets (Wikipedia). Au final, la quasi totalité des activités humaines sont connectées à Internet d'une façon ou d'une autre. Autant des transactions bancaires, des messages personnels, du contenu illicite, des signaux d'alerte ou d'appel à l'aide, du partage de connaissances et bien plus, se transmettent via les câbles reliant les divers coins du monde.

Tous ces changements se sont faits si rapidement, que les diverses préoccupations touchant les aspects de liberté des individus n'ont pas suivi adéquatement. Ce que nous connaissons des techniques de sécurisation des communications nous dit que les pratiques d'aujourd'hui protègent avant tout les détenteurs des services, non pas les utilisateurs de ceux-ci. C'est donc entre les mains d'autres que soi qu'est remise la responsabilité de garantir sa vie privée. Or, on a de multiples preuves que ces techniques échouent considérablement [36, 98]. Par moment, les

données des usagers sont même exploitées à leur insu [76] dans une perspective marchande [107]. L'approche conventionnellement employée pour concevoir nos systèmes d'information est celle des communications centralisées [4, 66, 70, 101]. Ce faisant, il est légitime d'exiger qu'on se penche sur cette question afin d'en cibler les aspects à améliorer.

Motivation

Pendant que certains souhaitent rendre illégaux (ou sans efficacité déterminante) les outils de chiffrement protégeant la vie privée des utilisateurs [40, 41, 50, 99], nous pensons qu'il s'agit d'un besoin réel et légitime pour tous les habitants de la planète, car jamais la surveillance à large échelle n'a été aussi facile. En effet, l'histoire précédant l'avènement des communications électroniques montre qu'auparavant, l'énergie requise pour mener des opérations de surveillance, sur une population, pouvait croître rapidement avec la taille de celle-ci. Cela s'explique entre autres par la nécessité d'assigner des agents pour conduire des opérations de surveillance et différents facteurs de limitation physique. Il est indéniable que ce type d'opérations engendrait alors des coûts considérables. Suivant cela, on pouvait raisonnablement inférer que la surveillance de masse ne se faisait pas, ou que légèrement. Il est donc clair qu'un individu bénéficiait d'une vie privée intègre *de facto* dans la majeure partie des cas. Bien sûr, la vie privée d'une personne particulière pouvait facilement être brisée, mais cela ne pouvait pas être vrai pour un très grand nombre de gens.

L'évolution des technologies connue depuis le dernier siècle rend cette réalité complètement révolue puisque les limitations physiques empêchant la surveillance à grande échelle ne s'appliquent pas sur les systèmes utilisés quotidiennement aujourd'hui. Ceux-ci fonctionnent grâce au réseau Internet qui permet l'échange rapide de très importants volumes de données. Cela fournit donc l'ensemble des

conditions suffisantes à la surveillance à large échelle. Les communications peuvent être interceptées, interrompues et stockées automatiquement, rendant *ipso facto* les individus impuissants et violés dans leur droit à la vie privée reconnu formellement depuis 1947 par la commission des droits de la personne [15].

Tel qu'indiqué dans [107], « il est impossible d'imaginer le capitalisme de surveillance sans numérique, mais il est facile d'imaginer le numérique sans capitalisme de surveillance. » Le manque de vie privée sur Internet est dû en partie aux choix de conception des services bâtis suivant la création de différents protocoles dont particulièrement *Hypertext Transfer Protocol* (HTTP) et *Domain Name System* (DNS) dans les années 90. En effet, ces protocoles ont façonné la Toile et engendré une structure fortement centralisée du réseau où certains exercent un pouvoir important sur les autres [68]. Cette formule a fourni le calque pour l'élaboration de la sécurité des communications qui aboutit à SSL en 1993 [72] par Netscape en réponse au besoin de sécuriser les transactions bancaires et commerciales. Ce faisant, le modèle de Netscape ne comprenait pas de préoccupation face à la préservation de la vie privée des usagers. Ce n'est que lors de la montée de l'utilisation massive de solutions de communication bâties sur les mêmes bases qu'on constate les pertes de vie privée. Bien que certaines formules comme *Pretty Good Privacy* (PGP) [106] aient tenté de répondre à ce problème, ceux-ci ne se sont pas imposés comme standard de sécurité des communications [104].

L'atteinte à la vie privée des individus est d'un ordre tout autre comparativement au passé. En effet, il est aujourd'hui possible d'avoir accès en temps réel aux faits et gestes de gens grâce à leur téléphone cellulaire. De plus, toutes conversations entre les utilisateurs des espaces de communication les plus populaires comme Facebook (dans son mode normal [29]), Slack, Discord, Mattermost et la téléphonie VoIP peuvent être facilement mises sur écoute. En temps réel ou *a posteriori*, ces conversations peuvent être analysées par n'importe quelle personne ayant un

FIGURE 0.1: Matrix attaqué (date : 11 avril 2019) Les utilisateurs sont laissés à leur sort, incapables de transmettre pendant plusieurs heures.

accès privilégié aux systèmes de ces services. Sur la plateforme Slack, un « administrateur ayant souscrit à l'offre "Plus", leur abonnement le plus cher, pourra télécharger l'ensemble des données d'un groupe de travail sans que les employés [en] soient avertis. Cela inclut les conversations publiques, les discussions sur les groupes, mais également les messages privés qui, par conséquent, ne le seront plus vraiment. » [32]. De plus, l'accès à ces systèmes peut prendre de multiples formes comme le partage de données par du personnel corrompu, l'intrusion par des pirates informatiques et les erreurs humaines variées. De son côté, Discord décrit explicitement la façon dont les données des usagers sont utilisées [26]. Ils peuvent « également partager des informations agrégées [1] ou non personnelles avec [leurs] partenaires ou autres à des fins commerciales. ». En résumé, selon les conventions actuelles, on demande aux usagers de ces systèmes de *faire confiance* aux concepteurs. Toutefois, il existe des alternatives qui fournissent des propriétés qui établissent un niveau de vie privée se qui se rapproche de celui dont jouissaient les individus des périodes précontemporaines.

À cela s'ajoutent les vulnérabilités qu'incombe l'emploi de modèles centralisés comme base des outils de communications. Comme illustré dans la capture d'écran

1. Des informations agrégées ne garantissent pas un haut degré de vie privée et peuvent être sujettes à réidentification par voie d'analyse de données.

de la figure 0.1, un service comme Matrix agrégeant une quantité considérable d'usagers rend le système global fragile. Bien que Matrix soit décentralisé, il diffère du paradigme distribué et n'est donc pas insensible aux intrusions. Le cas de Matrix n'est pas isolé. Ces attaques surviennent constamment au fil des années. Entre autres, une frappe produite le 21 octobre 2016 [44] a coupé d'Internet une grande partie de la planète partie. Les utilisateurs plus éveillés au sujet des détails d'ordre pratique ont certes pu remarquer que seul le service de résolution de nom de domaine (DNS) était affecté. Cependant, ce service est vital pour le fonctionnement de la Toile et rend donc très difficiles (voir impossibles), dans le modèle actuel, les activités sur celle-ci.

Ces défaillances en sécurité de systèmes de communication sont un sujet fortement étudié et discuté depuis environ une décennie. Différents travaux récents, notamment la table de hachage distribuée OpenDHT [84] et l'analyse du protocole de communication Signal et ses dérivés [16, 17], ont suscité notre intérêt.

Contributions

Dans un premier temps, nous nous sommes penchés sur des besoins particuliers du projet OpenDHT dans le cadre de travaux conjoints entre le milieu entrepreneurial et la sphère universitaire.

- l'optimisation de l'accessibilité des données récupérées dans un système sujet à des variations de connectivité potentiellement accrues, un phénomène nommé « remous » (*churn* en anglais) ;
- un outil de recherche par mot-clef dans un contexte complètement décentralisé afin de permettre le stockage et l'indexation de données telles que :
  - des profils utilisateur pour ainsi bâtir un annuaire pour un logiciel de communication.

— des informations sur des *super nœuds* pour rendre possible la délégation d'opérations en mode déconnecté (par exemple pour appareils mobiles avec un bas niveau de bande passante à leur disposition).

Suivant ces besoins, nous avons travaillé à l'élaboration de solutions concrètes afin de les joindre au projet [2] sous licence libre [3]. Ensuite, les détails de nos travaux ont été publiés [24] dans les actes de la conférence *Ubiquitous Networking : Third International Symposium* (2017).

Dans un dernier temps, nous avons abordé le problème de clavardage sûr bout en bout, c'est-à-dire l'échange de message sans acteur intermédiaire avec lequel des secrets cryptographiques seraient partagés. Ce choix de conception élimine les brèches de vie privée comme celles énoncées plus tôt dans ce document.

L'étude des différents avancements les plus récents, dont nous avons fait un résumé, nous a permis de produire un modèle qui vise à supprimer certaines lacunes observées. Les protocoles de communication sécurisée de référence tels que Signal [16, 59], OTR [12] et ses dérivées [38, 56, 86], demeurent encore à perfectionner. En particulier, la non-transitivité de paternité de message reste à optimiser dans un contexte asynchrone. Notre formule répond à cette question avec une utilisation nouvelle de briques cryptographiques bien connues. Finalement, nos efforts nous ont conduits à énoncer les conjectures 3.3.1, 3.3.2 et 3.3.3.

Structure du document

Dans ce document, nous employons des approches éprouvées constituant des alternatives aux conventions susmentionnées en favorisant plutôt des idées contenues

2. Les travaux peuvent être consultés à l'adresse suivante https://github.com/savoirfairelinux/opendht

3. Licence GPLv3

dans les domaines des réseaux répartis ainsi que la sécurité bout en bout. En particulier, nous présentons des travaux réalisés sur le modèle très connu de table de hachage distribuée [3, 65, 82, 90] et abordons la question d'indexation de données [34, 35, 43, 77, 89, 92] dans un contexte totalement distribué. De plus, le cas des communications sûres bout en bout fait l'objet d'un second chapitre où nous procurons un aperçu des plus récents ouvrages en la matière [12, 16, 17, 38, 56, 59, 60, 61, 86, 94, 95]. Nous terminons avec une proposition en réponse au problème de déni plausible en ce qui a trait aux informations qui nous lient aux messages que nous écrivons dans un système de clavardage.

Dans le chapitre 1, nous fournissons différentes notions préalables. En particulier, nous exposons les concepts de cryptographie telle que les briques essentielles d'analyse de sécurité calculatoire. Ensuite, nous détaillons la structure de table de hachage distribuée, plus particulièrement le modèle Kademlia [65].

Par la suite, le chapitre 2 fait le compte rendu de diverses contributions à la bibliothèque OpenDHT. On y définit les questions de persistance et d'indexation de données, les différentes solutions étudiées ainsi que la stratégie choisie.

Finalement, le chapitre 3 explore le problème de clavardage sûr bout en bout. Pour ce faire, nous établissons la liste des propriétés de sécurité bien connues et désirables pour un système de cyberbavardage sécurisé au fait des plus récentes innovations. Nous continuons avec une revue de l'état de l'art des protocoles de communication sûre bout en bout et terminons en proposant certaines améliorations.

CHAPITRE I

PRÉLIMINAIRES

De façon à bien préparer le lecteur aux chapitres 2 et 3, nous détaillons maintenant les notions nécessaires à la bonne compréhension du reste du document. Plus exactement, nous prenons le temps de mentionner, dans la section 1.1, différents résultats centraux au domaine de la cryptographie en plus de constructions variées qui en découlent. Les différentes conventions d'écriture sont aussi précisées pour une formulation plus concise dans les chapitres à venir. Ensuite, dans la section 1.2 nous abordons quelques éléments généraux sur les réseaux et les divers paradigmes existants. Nous terminons enfin le chapitre avec une section sur les tables de hachage distribuées, un aperçu de quelques exemples et avec des explications plus longues sur le modèle Kademlia.

1.1 Cryptographie

La cryptographie est une discipline de la cryptologie, un sous-domaine des mathématiques, et plus particulièrement de l'informatique, qui recueille l'ensemble des connaissances permettant d'écrire de manière secrète. Renaud Dumont, un professeur de l'Université de Liège, définit la cryptographie comme « l'étude des méthodes donnant la possibilité d'envoyer des données de manière confidentielle sur un support donné » [27]. Ces descriptions, bien qu'anciennement totalement

correctes, ne permettent pas de fournir une définition englobant tous les concepts et techniques aujourd'hui utilisés dans ce champ de connaissances. La définition suivante extraite de *Introduction to Modern Cryptography, Second Edition* comble ce manque.

> L'étude des techniques mathématiques permettant la sécurisation d'information numérique, de systèmes et de calculs distribués contre des attaques perpétrées par des adversaires.
>
> — Katz et Lindell [46]

Ces connaissances sont utilisées ensemble afin de former ce qu'on appelle un « chiffre » (anciennement nommé un « code » dans l'époque précontemporaine). À titre de synonyme, nous employons aussi le terme « méthode de chiffrement » pour désigner la même chose. On dit alors qu'un message rendu illisible par l'intermédiaire d'un chiffre est le procédé de *chiffrer* ce message. Le résultat du chiffrement est appelé « cryptogramme » (ou encore « texte chiffré »). De façon similaire, l'action permettant de retrouver un message à partir du cryptogramme associé est appelée *déchiffrement*. En complément, on compte le domaine de la cryptanalyse qui consiste en l'ensemble des connaissances visant à déterminer le texte clair correspondant à un cryptogramme donné. Nous explorons la terminologie venant de ce domaine dans la section 1.1.1. Finalement, lorsqu'un chiffre est à l'épreuve de tentatives visant à inverser sa sécurité, on dit alors que celui-ci est *sûr*. La sécurité d'un système concerne normalement principalement les propriétés de confidentialité, intégrité et authenticité (par l'authentification). La norme ISO 27000:2018(E) [45] définit ces vocables comme suit :

— la *confidentialité* : « propriété selon laquelle l'information n'est pas rendue disponible ni divulguée à des personnes, des entités ou des processus non autorisés » ;

— *l'intégrité* : « propriété d'exactitude et de complétude » ;

— et *l'authentification* : « moyen pour une entité d'assurer la légitimité d'une caractéristique revendiquée ».

Les méthodes pour arriver à procurer ces attributs ont drastiquement changé dans le dernier siècle. Nous fournissons un aperçu de chiffres rudimentaires célèbres dans l'annexe A.

Dans ce qui suit la présente section, nous formulons mathématiquement la sécurité de différents schémas cryptographiques. Il convient donc de présenter les différentes définitions et concepts permettant de décrire rigoureusement la sécurité d'un chiffre. En particulier, nous parlons brièvement des principes fondamentaux de la cryptographie moderne et donnons les définitions et propriétés mathématiques comme base pour l'expression de la sécurité formelle. Une grande partie des notions sont tirées du livre d'introduction à la cryptographie par Katz et Lindell [46]. En complément, des éléments de la sécurité parfaite définie selon Claude Shannon [87] sont disponibles dans l'annexe A.2.

1.1.1 Notations

Afin d'aborder la suite, nous exposons les différentes notations utilisées en vue de décrire des notions par voie de symboles. Cela allège énormément l'écriture et permet des formulations succinctes.

Une valeur m pouvant être représentée comme une chaîne de bits (0 ou 1) est dénoté par $m \in \{0,1\}^*$. La notation $\{0,1\}^*$ désigne le produit cartésien répété un nombre indéfini de fois sur $\{0,1\}$. On écrit $|m|$ afin de signifier le nombre de bits de m.

Un algorithme est dénoté par une lettre majuscule en police calligraphiée ou bien

par un mot en police linéale :

$$\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \ldots \quad \text{ou} \quad \mathsf{SHA\text{-}1}, \mathsf{MD5}, \mathsf{KDF}, \ldots$$

Quand un algorithme $\mathcal{G}$ génère une sortie déterministe a, on écrit $a \leftarrow \mathcal{G}$. Dans le cas où la sortie produite est indistinguable d'un élément choisi uniformément dans l'ensemble de sortie, alors on écrit plutôt $a \leftarrow_{\$} \mathcal{G}$. De façon similaire, un nombre b choisi uniformément dans un ensemble B est dénoté par $b \in_{\$} B$.

Un *schéma cryptographique*[1] est un tuple d'algorithmes et d'ensembles des textes clairs, des textes chiffrés et des clefs admises par les algorithmes. On note un schéma cryptographique par Π. Lorsqu'on parle d'un *schéma de chiffrement (ou un chiffre) à clef secrète*, on écrit plus précisément :

$$\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$$

où $\mathsf{Gen}, \mathsf{Enc}$ et Dec sont des algorithmes définis de façon abstraite comme suit :

— Gen : un algorithme probabiliste générant une clef k choisie dans une distribution statistique donnée.

— Enc : un algorithme prenant en entrée une clef k et un message $m \in \{0,1\}^*$, et produisant un cryptogramme $c \in \{0,1\}^*$. On dénote par $c = \mathsf{Enc}_k(m)$, le chiffrement de m avec la clef k.

— Dec : Algorithme prenant en entrée une clef k et un cryptogramme $c \in \{0,1\}^*$, et retournant un texte clair $m \in \{0,1\}^*$. On dénote par $m = \mathsf{Dec}_k(c)$ le déchiffrement de c avec la clef k.

Un chiffre Π doit satisfaire le critère de validité suivant :

$$m = \mathsf{Dec}_k(\mathsf{Enc}_k(m)).$$

1. Les termes « système cryptographique » ou « cryptosystème » sont parfois aussi utilisés.

Autrement dit, déchiffrer un cryptogramme donné produit le texte clair associé. Un algorithme simulant un adversaire (ou attaquant) d'un schéma est dénoté généralement par $\mathcal{A}$.

Un protocole de clavardage est normalement décrit dans un texte contenant potentiellement des passages centrés pour désigner des messages ou une opération donnée à effectuer. Par ailleurs, lorsqu'un usager A envoie un message $m \in \{0,1\}^*$ à son interlocuteur B, on note cela par :

$$A \to B : m$$

Pour signifier que le message est chiffré par une clef $k \in \{0,1\}^*$, on utilise l'écriture suivante :

$$A \to B : \{m\}_k$$

Le contenu à droite des deux-points est une liste dont les éléments sont séparés par des virgules (il en va de même pour le contenu chiffré). Par exemple, un message complet pourrait avoir l'allure de « $K, \{\sigma, t, m\}_k$ » où K, σ, t, m et k sont toutes des chaînes de caractères de longueurs arbitraires.

1.1.2 Garanties de sécurité et modèle d'adversaire

En considérant les différentes attaques réalisables sur les chiffres rudimentaires mentionnés au début de la section 1.1, il est légitime de se poser la question de comment bien capturer l'idée de sécurité. Selon la perspective de ceux ayant fait la découverte du chiffrement par substitution, l'idée que celui-ci soit robuste et qu'on puisse s'y fier est naturelle puisque l'espace des substitutions possibles est trop grand pour être exploré en entier dans un temps raisonnable. C'est en identifiant (avec surprise) une attaque sur ce chiffre qu'on perd soudainement confiance en celui-ci. Contrairement à auparavant, le diagnostic stipulant qu'un schéma n'est pas sûr ne devrait pas se faire avec surprise. En effet, on devrait

pouvoir démontrer qu'un chiffre est sûr suivant un ensemble d'hypothèses. En l'absence de preuve, la sûreté d'un schéma devrait être *de facto* mise en doute. Il est donc primordial de définir ce qu'est la sûreté d'un chiffre. Cette définition doit être sans ambiguïté et c'est ainsi qu'on distingue la cryptographie moderne de la cryptographie précontemporaine.

> Si le but qu'on cherche à atteindre n'est pas clair, comment peut-on possiblement savoir quand (ou si) on l'a atteint ?
>
> — Katz et Lindell [46]

Une attaque comme celle sur le chiffrement par substitution indique que le simple fait qu'un chiffre ait un espace de clefs très grand ne suffit pas pour qualifier la sûreté d'un chiffre. Il est donc nécessaire de considérer une caractéristique supplémentaire à respecter.

On discerne normalement deux composants formant une définition de sûreté : les garanties de sécurité et le modèle d'adversaire. Le premier énumère les différentes actions irréalisables par l'adversaire, et le second décrit ce que l'adversaire peut effectuer comme actions. Commençons par analyser la première composante.

Une première suggestion afin de définir la garantie de sécurité pourrait être que *de retrouver la clef utilisée pour (dé)chiffrer est impossible*. Cela semble intuitivement satisfaisant, mais on remarque qu'il est trivial de définir un chiffre pour lequel la clef ne pourrait être révélée, mais où le chiffre ne produit pas une sortie sûre. Par exemple, un schéma tel que $\mathsf{Enc}_k(m) = m$ n'exposerait effectivement jamais k, mais le résultat du chiffre n'est clairement pas protégé.

Une seconde suggestion pourrait être *qu'un adversaire ne peut récupérer dans son entièreté le texte clair associé à un cryptogramme*. Cette proposition comporte

déjà une touche d'amélioration, mais reste toujours très faible, car un adversaire serait donc permis de retrouver potentiellement une partie du texte en clair.

Finalement, la définition admise de sûreté d'un chiffre en mots est quelque chose comme *l'impossibilité pour un adversaire d'apprendre de l'information additionnelle sur le texte clair étant donné son cryptogramme associé*. Évidemment, cette définition est peu utile pour une démonstration formelle puisqu'elle n'est pas écrite dans des termes mathématiques. On y revient dans les sections suivantes.

Pour ce qui est du modèle d'adversaire, on en considère plusieurs. Les voici :

— **Attaque à cryptogramme connu** : l'adversaire a accès au(x) cryptogramme(s) et tente de tirer de l'information au sujet du (ou des) texte(s) clair(s) associé(s).

— **Attaque à texte clair connu** : l'adversaire connaît une (ou plusieurs) paires de $(m_i, \mathsf{Enc}_k(m_i))$ chiffrée(s) par une clef k et tente d'apprendre de l'information sur un (ou plusieurs) autre(s) texte(s) clair(s) associés à son (ou leur) cryptogramme et chiffré(s) par la même clef.

— **Attaque à texte clair choisi** : l'adversaire peut obtenir le(s) cryptogramme(s) associé(s) à un (ou plusieurs) texte(s) clair(s) de son choix.

— **Attaque à cryptogramme choisi** : l'adversaire peut obtenir le(s) texte(s) clair(s) associé(s) à un (ou plusieurs) cryptogramme(s) de son choix. L'adversaire cherche à obtenir de l'information sur un texte clair d'un autre cryptogramme qu'il ne peut pas déchiffrer.

Principe fondamental et approches

Dans un premier temps, lors de la révolution qu'a connue la cryptographie contemporaine, on a vu l'adoption du principe de Kerckhoffs [47] :

> Il faut un système remplissant certaines conditions exceptionnelles, conditions que je résumerai sous les six chefs suivants :
>
> 1. Le système doit être matériellement, sinon mathématiquement indéchiffrable ;
> 2. Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi ;
> 3. La clef doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants ;
>
> [...]
>
> — Kerckhoffs [47]

Les trois desiderata énoncés ci-haut sont normalement paraphrasés par le fait que *la sécurité d'un chiffre ne doit pas reposer sur son caractère secret, mais plutôt sur la garantie de garder la clef de chiffrement utilisée secrète.*

Plus tard, Shannon [87] suivit avec l'idée de sécurité parfaite d'un schéma cryptographique. Sa définition peut être phrasée comme *l'incapacité d'apprendre une quelconque information sur un message clair en observant son cryptogramme correspondant.*

Bien que l'idée de sécurité parfaite puisse paraître attrayante, très peu de chiffres satisfont à cette définition. On compte entre autres le masque jetable parmi ces chiffres et celui-ci n'est pas utilisable en pratique. La cryptographie contemporaine se base donc sur deux techniques, soient la méthode concrète et l'approche asymptotique. Nous définissons l'approche asymptotique et adoptons celle-ci pour le contenu du mémoire.

1.1.3 Sécurité asymptotique

L'approche est essentiellement composée de concepts trouvant leur origine dans la théorie de la complexité. Cela permet de relaxer les définitions strictes telles que la sécurité parfaite et d'ainsi construire des schémas satisfaisant nos définitions. En particulier, un schéma de chiffrement prend en entrée un paramètre de sécurité dénoté par n. Ce paramètre est aussi utile afin de définir la configuration des participants. Il est notamment connu de l'adversaire. On voit le temps d'exécution et la probabilité de réussite de l'adversaire comme fonction de ce paramètre.

Un *algorithme efficace* A est un algorithme probabiliste s'exécutant en temps polynomial en n, c'est-à-dire qu'il existe un polynôme p tel que l'algorithme s'exécute en temps $p(n)$ avec n le paramètre de sécurité. Formellement :

Définition 1.1.1 (Algorithme TPP)

Soit A un algorithme. S'il existe un polynôme p tel que $\forall x \in \{0,1\}^*$ le calcul de $A(x)$ se termine en $p(|x|)$ étapes, alors A est appelé un *algorithme temps probabiliste-polynomiale (TPP)*.

Pour un algorithme probabiliste, on peut calculer sa probabilité de succès dans un scénario donné. Une probabilité de succès est dite *négligeable* si celle-ci est bornée par l'inverse d'un polynôme en n. On décrit formellement l'idée de *fonction négligeable* dans la définition 1.1.2.

Définition 1.1.2 (Fonction négligeable)

Une fonction $f : \mathbb{N} \to \mathbb{R}^+$ est *négligeable* si pour tout polynôme p, il existe N tel que $\forall n > N$,

$$f(n) < \frac{1}{p(n)}.$$

Remarque 1.1.1. On dénote typiquement une fonction négligeable par negl. $\Diamond$

Exemple 1.1.1. *Considérons les fonctions* $f(n) = n, g(n) = \frac{1}{n\log_2(n)}$ *et* $h(n) = 2^{-n}$. *Clairement,* f *n'est pas négligeable puisqu'elle est croissante. Ensuite, bien que* g *décroît, elle ne décroît pas aussi rapidement que* $1/p(n)$ *pour tout polynôme,* p *car :*

$$\begin{aligned}\lim_{n\to\infty} \frac{a_1 n + a_0}{n\log_2(n)} &= \lim_{n\to\infty} \frac{a_1}{1 \cdot \log_2(n) + \frac{n}{n\ln(2)}} \\ &= \lim_{n\to\infty} \frac{a_1}{\log_2(e \cdot n)} \\ &= 0.\end{aligned}$$

Ce faisant, p *croît forcément strictement moins vite que* $n\log_2(n)$ *lorsque* p *est de degré 1, c'est-à-dire que* $\forall c \in \mathbb{N}, \exists N \in \mathbb{N}, \forall n > N$ *:*

$$\begin{aligned} & p(n) < c \cdot n\log_2(n) \\ \implies g(n) = \frac{1}{n\log_2(n)} &< \frac{1}{p(n)}\end{aligned}$$

Sinon si $d > 1$ *:*

$$\begin{aligned}&\lim_{n\to\infty} \frac{a_d n^d + a_{d-1}n^{d-1} + \ldots + a_1 n + a_0}{n\log_2(n)} \\ &= \lim_{n\to\infty} \frac{da_d n^{d-1} + (d-1)a_{d-1}n^{d-2} + \ldots + 2a_2 n + a_1}{\log_2(n) + 1} && \textit{l'Hôpital} \\ &= \lim_{n\to\infty} \frac{d(d-1)a_d n^{d-2} + (d-1)(d-2)a_{d-1}n^{d-3} + \ldots + 2a_2}{1/n} && \textit{l'Hôpital} \\ &\to \infty\end{aligned}$$

Ainsi, il existe un polynôme tel que $\forall N \in \mathbb{N}, \exists n > N$ *où* $g(n) \geq 1/p(n)$. *Par conséquent,* g *n'est pas négligeable. Finalement, puisque* h *est décroissante et que son inverse croît nécessairement plus rapidement que tout polynôme, c'est-à-dire que :*

$$\lim_{n\to\infty} \frac{a_d n^d + a_{d-1}n^{d-1} + \ldots + a_1 n + a_0}{2^n} = 0 \quad \forall d \in \mathbb{N}$$

en effectuant la règle de l'Hôpital d *fois, alors on a bien que* $\exists N \in \mathbb{N}$ *tel que* $h(n) < 1/p(n)$. *En d'autres termes,* h *est une fonction négligeable.* $\Diamond$

Remarque 1.1.2. Au cours du reste du document, on considère que $\log(n) \stackrel{\text{déf}}{=} \log_2(n)$.

◊

Proposition 1.1.1

Soit $\mathsf{negl}_1, \mathsf{negl}_2$ des fonctions négligeables. Alors :

— $\mathsf{negl}_1(n) + \mathsf{negl}_2(n)$ est négligeable.

— pour tout polynôme p positif, $p(n) \cdot \mathsf{negl}_1(n)$ est négligeable.

Ces notions étant maintenant définies, on peut donc proposer un gabarit pour la définition de sécurité d'un schéma cryptographique. Un schéma est considéré sûr si pour tout adversaire TPP, la probabilité que celui-ci réussisse une attaque sur le schéma est *négligeable*.

Dans le but de de correctement exprimer le « succès (ou l'échec) d'une attaque », on conçoit une *expérience*. Ce concept permet de capturer formellement les capacités et hypothèses en rapport à l'adversaire. Les étapes à suivre afin de fournir un contexte approprié à l'adversaire sont subséquemment décrites. Finalement, on indique la nature du résultat de l'expérience, c'est-à-dire quels sont les cas où l'adversaire « gagne » et les cas où il échoue. Par exemple, considérons l'expérience classique suivante présentée dans [46] : un adversaire $\mathcal{A}$ spécifie premièrement deux messages $m_0, m_1 \in \mathcal{M}$ ($\mathcal{M} \subset \{0,1\}^*$ est l'espace des messages, voir l'annexe A.2). Un de ces deux messages est choisi uniformément aléatoirement et est chiffré avec une clef elle aussi choisit aléatoirement. On communique à ce moment-là le résultat du chiffrement à $\mathcal{A}$. L'algorithme $\mathcal{A}$ retourne alors sa conjecture à savoir lequel des deux messages entre m_0 et m_1 lui a été donné au premier abord. $\mathcal{A}$ réussit l'expérience si sa conjecture est vérifiée. Formellement, on écrit cette expérience, qu'on nomme $\mathsf{PrivK}^{\mathsf{esp}}_{\mathcal{A},\Pi}(n)$, comme dans l'encadré suivant.

Expérience $\mathsf{PrivK}^{\mathsf{esp}}_{\mathcal{A},\Pi}(n)$

Soit $\mathcal{A}$ un algorithme TPP, $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ un schéma de chiffrement et n le paramètre de sécurité.

1. Suivant l'entrée, 1^n, $\mathcal{A}$ retourne une paire de messages $m_0, m_1 \in \mathcal{M}$.
2. On génère une clef $k \in \mathcal{K}$ par l'appel de $k \leftarrow_{\$} \mathsf{Gen}(1^n)$ et on prend $b \in_{\$} \{0,1\}$. On calcule alors $c \leftarrow \mathsf{Enc}_k(m_b)$ et acheminons celui-ci à $\mathcal{A}$.
3. $\mathcal{A}$ retourne $b' \in \{0,1\}$.

Le résultat de l'expérience est défini comme 1 si $b = b'$ et 0 sinon.

Cette approche permet d'écrire de manière succincte et dans un formalisme convenable la sécurité d'un schéma. Afin d'accorder son plein sens à l'expérience décrite ci-haut, on donne maintenant une conception d'un schéma produisant un résultat *indistinguable en présence d'espions* dans la définition 1.1.3.

Définition 1.1.3 (Chiffrement indistinguable en présence d'espion)

Soit $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ un schéma de chiffrement, n un paramètre de sécurité et $\mathcal{A}$ un algorithme TPP. Le schéma Π produit une sortie *indistinguable en présence d'espion* si pour tout $\mathcal{A}$, il existe une fonction négligeable negl telle que :

$$\mathbb{P}[\mathsf{PrivK}^{\mathsf{esp}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n).$$

Remarque 1.1.3. Intuitivement, l'algorithme $\mathcal{A}$ peut facilement au moins satisfaire à $\mathbb{P}[\mathsf{PrivK}^{\mathsf{esp}}_{\mathcal{A},\Pi}(n) = 1] = 1/2$. En effet, $\mathcal{A}$ n'aurait qu'à tirer un bit au hasard. ◊

Remarque 1.1.4. Cette définition n'est pas destinée à offrir toutes les notions nécessaires pour exprimer la sécurité des schémas de chiffrement utilisés dans de ce document. L'ensemble des notions abordées dans ne demandent pas plus de démontrer cela particulier. Nous n'analysons cette définition qu'à titre d'intro-

duction au concept de sécurité asymptotique. ◊

Comme le dit la remarque 1.1.3, l'adversaire n'est pas tenu de faire une conjecture fausse dans une majorité de cas. La définition de sécurité demande cependant que $\mathcal{A}$ n'ait pas *d'avantage* significatif. L'avantage est la quantité $\mathbb{P}[\mathsf{PrivK}^{\mathsf{esp}}_{\mathcal{A},\Pi}(n) = 1] - 1/2$ et selon la définition 1.1.3, cette quantité doit être négligeable. Cette définition est un exemple classique où l'adversaire est soumis au problème de distinguer entre deux scénarios. On qualifie ce type de problème de *décisionnel*. Il est pertinent de remarquer que l'adversaire n'est pas tenu de déchiffrer (calculer) c pour réussir, mais seulement de distinguer si $c = \mathsf{Enc}(m_0)$ ou $c = \mathsf{Enc}(m_1)$. Ce faisant, un chiffre satisfaisant à une telle définition est donc une proposition plus forte que celle demandant qu'un adversaire calcule m_b. On qualifie ce dernier type de problème de *calculatoire*. En général, un problème décisionnel, disons D se réduit (au sens de la théorie de la complexité) au problème calculatoire correspondant, disons C. On écrit alors $D \preceq C$. Cette réduction est équivalente à dire que *si D est difficile, alors C l'est aussi*. Les propositions de la sections suivante concrétisent cette idée de réduction sur laquelle repose la sécurité d'un schéma cryptographique indispensable, pour de multiples systèmes bien connus, que nous explicitons dans la section 1.1.5.

1.1.4 Problèmes Diffie-Hellman

Nous abordons maintenant trois problèmes cryptographiques centraux à la bonne compréhension du chapitre 3. Cette section est basée sur les concepts d'arithmétique modulaire et les groupes cycliques. Le lecteur peut consulter l'annexe B pour y trouver les bases mathématiques aidant à la lecture de ce qui suit.

Nous considérons un algorithme TPP (voir la définition 1.1.1), dénoté $\mathcal{G}$, générant la configuration d'un groupe cyclique $\mathbb{G}$ d'ordre q suivant qu'on lui fournit 1^n en

entrée telle que n est la taille en bits de q. Nous considérons de ce groupe cyclique, un générateur $g \in \mathbb{G}$.

Remarque 1.1.5. On suppose l'existence d'un algorithme TPP pouvant calculer l'opération dans $\mathbb{G}$ ainsi que pour tirer un élément $h \in \mathbb{G}$ uniformément au hasard. Cette dernière opération correspond directement à choisir $x \in \mathbb{Z}_q$ et calculer $h = g^x$. ◊

Soit $h_0 \in \mathbb{G}$ tel que $h_0 = g^{x_0}$ pour x_0 un entier quelconque de $\mathbb{Z}_q$, alors on appelle x_0 le *logarithme discret* de h_0 en base g. On écrit formellement que $x_0 = \log_g(h_0)$. Notons que pour $x' \in \mathbb{Z}$, alors on a tout de même $[x' \bmod q] = \log_g h$ (voir la définition B.1.1). Le logarithme discret satisfait quelques propriétés fondamentales du logarithme usuel :

$$\begin{cases} \log_g(0) = 1 & \text{où 1 est l'identité dans } \mathbb{G}; \\ \log_g(h^r) = [r \log_g(h) \bmod q]; & \text{pour } r \in \mathbb{Z}, h \in \mathbb{G} \\ \log(h_1, h_2) = [\log_g(h_1) + \log_g(h_2) \bmod q] & \text{pour } h_1, h_2 \in \mathbb{G} \end{cases}$$

On appelle le *problème du calcul du logarithme discret (CLD)* le calcul de $\log_g(h)$ pour $h \in_\$ \mathbb{G}$. Notez que « $\in_\$$ » signifie que h est choisi uniformément au hasard dans $\mathbb{G}$. Ci-après, nous définissons l'expérience $\mathsf{LogD}_{\mathcal{A},\mathcal{G}}(n)$.

Expérience $\mathsf{LogD}_{\mathcal{A},\mathcal{G}}(n)$

Soit un algorithme de génération de groupe cyclique $\mathcal{G}$, $\mathcal{A}$ un algorithme TPP et n un paramètre de sécurité.

1. Lancer $\mathcal{G}(1^n)$ pour obtenir $(\mathbb{G}, q, g)$ un tuple de configuration d'un groupe cyclique $\mathbb{G}$ d'ordre q et $g \in \mathbb{G}$ un générateur de ce groupe.
2. Choisir $h \in_\$ \mathbb{G}$.
3. On donne à $\mathcal{A}$ le tuple $(\mathbb{G}, q, g, h)$. Celui-ci retourne $x \in \mathbb{Z}_q$.

Le résultat de l'expérience est définit comme 1 si $x = \log_g(h)$ et 0 sinon.

Définition 1.1.4 (Difficulté du CLD)

Le *problème du logarithme discret* est difficile par rapport à $\mathcal{G}$ si pour tout algorithme TPP $\mathcal{A}$, il existe une fonction négligeable negl telle que :

$$\mathbb{P}[\mathsf{LogD}_{\mathcal{A},\mathcal{G}}(n) = 1] \leq \mathsf{negl}(n).$$

On appelle *l'hypothèse CLD*, la supposition qu'il existe un algorithme $\mathcal{G}$ satisfaisant à la définition 1.1.4. Reliés à cette question sont les problèmes de Diffie-Hellman. Ces problèmes ne sont pas démontrés comme équivalents au CLD, mais ils se réduisent (en termes de difficulté) à celui-ci. On discerne deux problèmes principaux : le problème de calcul de Diffie-Hellman (CDH) et le problème de décision de Diffie-Hellman (DDH).

Soit $\mathbb{G}$ un groupe cyclique et $g \in \mathbb{G}$ un générateur de ce groupe. Pour $h_1, h_2 \in \mathbb{G}$, on définit $\mathsf{DH}_g(h_1, h_2) \overset{\text{déf}}{=} g^{\log_g h_1 \cdot \log_g h_2}$. En particulier, écrit comme suit $h_1 = g^{x_1}, h_2 = g^{x_2}$, on a donc :

$$\mathsf{DH}_g(h_1, h_2) = g^{x_1 x_2} = h_1^{x_2} = h_2^{x_1}.$$

Le problème CDH est donc de calculer $\mathsf{DH}(h_1, h_2)$ pour $h_1, h_2 \in_\$ \mathbb{G}$. Plus formellement, on définit l'expérience $\mathsf{CDH}_{\mathcal{A},\mathcal{G}}(n)$ très similairement à l'expérience

$\mathsf{LogD}_{\mathcal{A},\mathcal{G}}(n)$:

Expérience $\mathsf{CDH}_{\mathcal{A},\mathcal{G}}(n)$

Soit un algorithme de génération de groupe cyclique $\mathcal{G}$, $\mathcal{A}$ un algorithme TPP et n un paramètre de sécurité.

- Lancer $\mathcal{G}(1^n)$ pour obtenir $(\mathbb{G}, q, g)$ un tuple de configuration d'un groupe cyclique $\mathbb{G}$ d'ordre q et $g \in \mathbb{G}$ un générateur de ce groupe.
- Choisir $x_1, x_2 \in_\$ \mathbb{Z}_q$ et calculer respectivement $h_1 = g^{x_1}, h_2 = g^{x_2}$.
- On donne à $\mathcal{A}$ le tuple $(\mathbb{G}, q, g, h_1, h_2)$. Celui-ci retourne $h_3 \in \mathbb{G}$.

Le résultat de l'expérience est définit comme 1 si $h_3 \equiv g^{x_1 x_2} \mod q$.

Définition 1.1.5 (Difficulté du CDH)

Le *problème CDH* est difficile par rapport à $\mathcal{G}$ si pour tout algorithme TPP $\mathcal{A}$, il existe une fonction négligeable negl telle que :

$$\mathbb{P}[\mathsf{CDH}_{\mathcal{A},\mathcal{G}}(n) = 1] \leq \mathsf{negl}(n).$$

L'hypothèse CDH est donc la supposition qu'il existe un algorithme TPP $\mathcal{G}$ satisfaisant à la définition 1.1.5. On ne sait pas si le problème CLD se réduit au CDH, cependant il est trivial de montrer la réciproque, c'est-à-dire que CDH $\preceq$ CLD. En effet, soit $h_1, h_2 \in \mathbb{G}$. Si on sait résoudre le CLD, alors on sait calculer $x_1 = \log_g h_1$, $x_2 = \log_g h_2$. Dans ce cas, on a trouve facilement la réponse au problème CDH en évaluant $h_1^{x_2}$ (ou encore $h_2^{x_1}$).

Soit $h' \in \mathbb{G}$. Le *problème DDH* demande de distinguer entre le cas où $\mathsf{DH}_g(h_1, h_2) = h'$ ou bien celui où h' est simplement un élément choisi uniformément dans $\mathbb{G}$. Plus concrètement, on définit l'expérience $\mathsf{DDH}_{\mathcal{A},\mathcal{G}}(n)$ ci-après :

Expérience $\mathsf{DDH}_{\mathcal{A},\mathcal{G}}(n)$

Soit un algorithme de génération de groupe cyclique $\mathcal{G}$, $\mathcal{A}$ un algorithme TPP et n un paramètre de sécurité.

- Lancer $\mathcal{G}(1^n)$ pour obtenir $(\mathbb{G}, q, g)$ un tuple de configuration d'un groupe cyclique $\mathbb{G}$ d'ordre q et $g \in \mathbb{G}$ un générateur de ce groupe.
- Choisir $x_\alpha, x_\beta, x_1 \in_\$ \mathbb{Z}_q$ et calculer respectivement $h_\alpha = g^{x_\alpha}$, $h_\beta = g^{x_\beta}$, $h_0 = g^{x_\alpha x_\beta}$ et $h_1 = g^{x_1}$.
- On choisit $b \in_\$ \{0, 1\}$ de façon uniformément aléatoire.
- On donne à $\mathcal{A}$ le tuple $(\mathbb{G}, q, g, h_\alpha, h_\beta, h_b)$. Celui-ci retourne $b' \in \{0, 1\}$.

Le résultat de l'expérience est définit comme 1 si $b = b'$ et 0 sinon.

Définition 1.1.6 (Difficulté de DDH)

Le *problème DDH* est difficile par rapport à $\mathcal{G}$ si pour tout algorithme TPP $\mathcal{A}$, il existe une fonction négligeable negl telle que :

$$\mathbb{P}[\mathsf{DDH}_{\mathcal{A},\mathcal{G}}(n) = 1] \leq \mathsf{negl}(n).$$

Finalement, on sait que DDH $\preceq$ CLD, mais encore une fois la réciproque n'est pas pour autant vraie. *L'hypothèse DDH* est analogue aux deux autres mentionnées plutôt. Aujourd'hui, ce problème est considéré difficile et constitue l'hypothèse la plus forte des trois. Cependant, pour que la supposition tienne, un choix judicieux d'un groupe doit être fait afin de donner un contexte optimal. En particulier, le problème CLD est considéré plus difficile dans des groupes d'ordre premier. Selon le corollaire B.2.2, tous les éléments d'un tel groupe (sauf l'identité) sont des générateurs. Ce faisant, cela simplifie la recherche d'un générateur pour $\mathcal{G}$. Enfin, un groupe d'ordre premier implique que $\mathsf{DH}(h_1, h_2)$ est « près d'être uniforme » [46] pour h_1, h_2 choisis uniformément.

Katz et Lindell [46] mentionnent une stratégie pour trouver un groupe préservant la confiance en l'hypothèse DDH. Comme point de départ, on considère $\mathbb{Z}_p^*$ avec p premier. Selon le théorème B.2.3, $(\mathbb{Z}_p^*, \cdot)$ est un groupe cyclique d'ordre $p-1$. Bien que le problème CLD est difficile dans de tels groupes, ce n'est pas le cas pour le problème DDH pour $p > 3$ puisque l'ordre de $\mathbb{Z}_p^*$ n'est jamais premier. Pour un groupe cyclique donné $\mathbb{Z}_p^*$ avec p et $q = \frac{p-1}{r}$ tous deux premiers, le groupe $\mathbb{G}$ définit ci-après est un sous-groupe de $\mathbb{Z}_p^*$, cyclique et d'ordre q :

$$\mathbb{G} \stackrel{\text{déf}}{=} \left\{ [h^r \bmod p] \mid h \in \mathbb{Z}_p^* \right\}$$

Cette astuce constitue donc une notion pertinente que peut utiliser un algorithme $\mathcal{G}$ de génération de groupe cyclique préservant ainsi la confiance en l'hypothèse DDH. Nous utilisons, dans la section suivante, les idées vues jusqu'ici afin de donner la description d'un protocole incontournable aux pratiques de la cryptographie moderne.

1.1.5 Échange Diffie-Hellman

Nous abordons maintenant une notion centrale aux concepts explicités dans le chapitre 3. L'idée *d'échange* est celle de secrets menant à au calcul d'une information commune entre deux partis. Le protocole d'échange de clef que nous décrivons ici est particulier en ce qu'il réussit au travers d'un canal de communication publique, c'est-à-dire qu'un adversaire a accès à tous les messages échangés entre les deux partis. Cependant, nous faisons l'hypothèse que l'adversaire est seulement passif (autrement, il faut utiliser des astuces que nous n'explicitons pas ici comme [39, 54]).

L'idée de pouvoir échanger sur un canal public et dériver une information commune que personne d'autre des deux partis impliqués ne connaît est semblable à l'idée de deux personnes qui s'échangent à haute voix dans une pièce des nombres

pour alors faire un calcul chacun de leur côté et finalement connaître une information que personne d'autre de présent dans la pièce serait capable de déduire malgré qu'ils aient entendu les mêmes messages échangés entre les deux partis. Le principe général va comme suit : soit deux partis communiquant, disons Alice et Bob, (i) ceux-ci calculent un secret à eux chacun de leur côté ; (ii) suivant qu'ils s'accordent sur le paramètre de sécurité 1^n (normalement conventionnel), les participants exécutent le protocole ensemble ; (iii) finalement, ils dérivent respectivement $k_A, k_B \in \{0,1\}^n$. La condition finale est que $k_A = k_B$. Par simplicité, on écrit, alors simplement k. Un protocole d'échange de clef est sûr si k est ne peut être deviner par l'adversaire. Formellement, on demande que l'adversaire ne puisse pas distinguer k d'une chaîne de caractères uniformément aléatoire dans $\{0,1\}^n$.

Expérience $\mathsf{KE}^{\mathsf{esp}}_{\mathcal{A},\Pi}(n)$

Soit $\mathcal{A}$ un algorithme TPP et n un paramètre de sécurité et Π un protocole d'échange de clefs.

1. Les deux partis exécutent Π et produisent trans une transcription de la conversation et k, la clef produite par chacun des participants.
2. On choisit $b \in_{\$} \{0,1\}$ et posons :

$$\hat{k} = \begin{cases} k & \text{si } b = 0 \\ k' \in_{\$} \{0,1\}^n & \text{sinon} \end{cases}$$

3. L'adversaire $\mathcal{A}$ se fait transmettre $\hat{k}$ et retourne $b' \in \{0,1\}$.

Le résultat de l'expérience est définit comme 1 si $b' = b$ et 0 sinon.

L'adversaire a donc tous les messages échangés entre Alice et Bob lorsqu'il obtient trans. Il a réussi l'expérience s'il distingue $\hat{k}$ d'un patron de bits uniforme. On dit que le protocole Π est sûr si l'adversaire ne peut réussir avec une probabilité

supérieure à $1/2$.

Définition 1.1.7 (Sûreté d'un échange de clefs)

Un protocole d'échange de clef Π est sûr en présence d'espions si pour tout algorithme TPP $\mathcal{A}$, il existe une fonction négligeable telle que :

$$\mathbb{P}[\mathsf{KE}^{\mathsf{esp}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2}\mathsf{negl}(n).$$

Nous définissons maintenant le protocole d'échange Diffie-Hellman.

Protocole Diffie-Hellman

Soit $\mathcal{G}$ un algorithme TPP qui, suivant l'entrée 1^n, génère la description d'un groupe cyclique $\mathbb{G}$ d'ordre q (avec $||q|| = n$) et un générateur $g \in \mathbb{G}$.

1. Alice calcule $(\mathbb{G}, q, g) \leftarrow \mathcal{G}(1^n)$;
2. Elle choisit $x \in_\$ \mathbb{Z}_q$ et calcule $h_A = g^x$;
3. Elle envoie (G, q, g, h_A) à Bob ;
4. Bob reçoit le tuple (G, q, g, h_A) et choisit $y \in_\$ \mathbb{Z}_q$. Il calcule alors $h_B = g^y$ et envoie h_B à Alice. Finalement, il calcule $k_B = h_A^y$.
5. Alice reçoit h_B et peut maintenant calculer $k_A = h_B^x$.

Remarque 1.1.6. On voit tout de suite que

$$k_B = h_A^y = (g^x)^y = g^{xy} = (g^y)^x = h_B^x = k_A.$$

Ce faisant, les deux partis obtiennent une même clef à la fin de l'exécution. $\Diamond$

Selon les hypothèses de la section 1.1.4, on voit vite que l'exigence minimale est celle du CLD. En effet, si le celle-ci ne tient pas, il est facile de briser ce protocole puisqu'on peut calculer les secrets de chacun de participants à partir des

informations se trouvant dans trans. Il en est de même pour l'hypothèse CDH. Cependant, ces deux seules hypothèses ne satisfont pas la définition 1.1.7. En effet, il est nécessaire que le résultat du protocole, la clef k, soit *indistinguable* d'un patron de bits uniforme. On remarque que cela correspond directement à *l'hypothèse DDH* (voir la définition 1.1.6). Jusqu'ici, les concepts couverts se limitaient au chiffrement. Dans ce qui suit, nous introduisons le concept d'authentification brièvement décrit au début du chapitre.

1.1.6 Authentification

Nous mentionons brièvement le concept élémentaire de code d'authentification de message (CAM) permettant de valider la provenance de messages échangés par des interlocuteurs. Nous attardons ensuite plus profondément sur les schémas de signature pour en faire la définition formelle puisqu'elle est centrale aux propositions finales du chapitre 3.

Code d'authentification de message

Un schéma du type *code d'authentification* permet de vérifier l'intégrité d'un message, c'est-à-dire qu'il n'a pas été modifié entre le moment où il a été écrit par son auteur et reçu par son destinataire. La stratégie repose sur l'utilisation d'un schéma de chiffrement à clef secrète. Ce faisant, les interlocuteurs doivent établir un secret commun par une voie sécurisée (voir la section 1.1.5 pour un exemple d'un tel système). L'idée va comme suit :

(i) l'émmetteur, disons Alice, d'un message m calcule un *jeton* t par l'intermédiaire de l'algorithme Cam en fonction du message et d'une clef secrète k connue entre les deux utilisateur. Plus formellement, elle génère $t \leftarrow \mathsf{Cam}_k(m)$;

(ii) Alice envoie m et t à l'autre participant, disons Bob ;

(iii) Bob reçoit la paire et peut vérifier que le jeton est valide en vérifiant que $t \stackrel{?}{=} \mathsf{Cam}_k(m)$.

Cette approche peut normalement se concrétiser par l'utilisation d'une clef secrète utilisée pour chiffrer, disons k. On dérive normalement une clef k' à partir de celle-ci de telle sorte que k' ne révèle aucune information sur k. On utilise normalement une fonction de hachage cryptographique [2] pour cela : $k \leftarrow \mathsf{hash}(k')$. Ce modèle marche bien dans plusieurs cas, mais on voit au chapitre 3 que d'autres formules permettent des propriétés différentes que les CAMs ne peuvent pas fournir. Nous n'abordons pas la sécurité des CAMs puisqu'elle n'est pas centrale au mémoire (voir [46] pour cela). Nous enchaînons avec une description formelle du concept derrière les idées alternatives dans la section suivante.

Schéma de signature

Un *schéma de signature* est analogue au schéma de CAM. Il ne dépend cependant pas d'une clef secrète, mais plutôt d'une paire de clefs $(\mathsf{sk}, \mathsf{pk})$ qu'on nomme respectivement *clef privée* et *clef publique*.

---

2. Par exemple, une fonction de la famille SHA-2.

Définition 1.1.8 (Schéma de signature)

Un *schéma de signature* consiste en un tuple $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ d'algorithmes TPP tel que :

— L'algorithme Gen est appelé *algorithme de génération de clef* et prend un paramètre de sécurité 1^n en entrée et retourne une paire de clefs $(\mathsf{sk}, \mathsf{pk})$.

— L'algorithme Sign est appelé *algorithme de signature* et prend en entrée une clef privée sk, un message m et retourne une signature σ. On dénote ce procédé par $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m)$.

— L'algorithme déterministe Vrfy est appelé *algorithme de vérification* et prend en entrée une clef publique pk, un message m, une signature σ et renvoit $b \in \{0, 1\}$. Si $b = 1$, alors la signature est *valide*, sinon elle est *invalide*. On dénote la vréfication d'un tel processus par $\mathsf{Vrfy}_{\mathsf{pk}}(m, \sigma)$.

On utilise normalement un schéma de signature de la façon suivante. Un premier usager, disons Alice, génère une paire de clefs $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^n)$ et partage pk avec ses correspondants par un moyen sécurisé. Ceci peut se faire par l'affichage de la clef sur un site web personnel [3] par exemple ou bien en personne comme c'est la tradition dans les communautés utilisant GPG [4]. Quand Alice souhaite envoyer un message signé à Bob, elle calcule $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m)$ et envoie à Bob la paire (m, σ). Une fois que Bob reçoit le message, il peut vérifier l'authenticité du message m en vérifiant si $\mathsf{Vrfy}_{\mathsf{pk}}(m, \sigma) \stackrel{?}{=} 1$. Dans le cas où la validation réussit, alors cela assure à Bob qu'Alice a bel et bien écrit le message m.

Cette description formelle et abstraite d'un schéma de signature permet de définir une expérience cryptographique et la propriété associée pour établir une notion

3. Un site web sécurisé par le protocole HTTPS.

4. https://fr.wikipedia.org/wiki/Kealorsy_signing_party

de sécurité. Le lecteur peut consulter [46] pour y trouver ces informations.

Le schéma de signature bien connu nommé ElGamal utilise les concepts vues dans la section 1.1.4. Nous terminons ici les théories de la cryptographie. Dans la prochaine section, nous passons aux éléments des réseaux distribués pour bien aborder le concept de table de hachage distribuée.

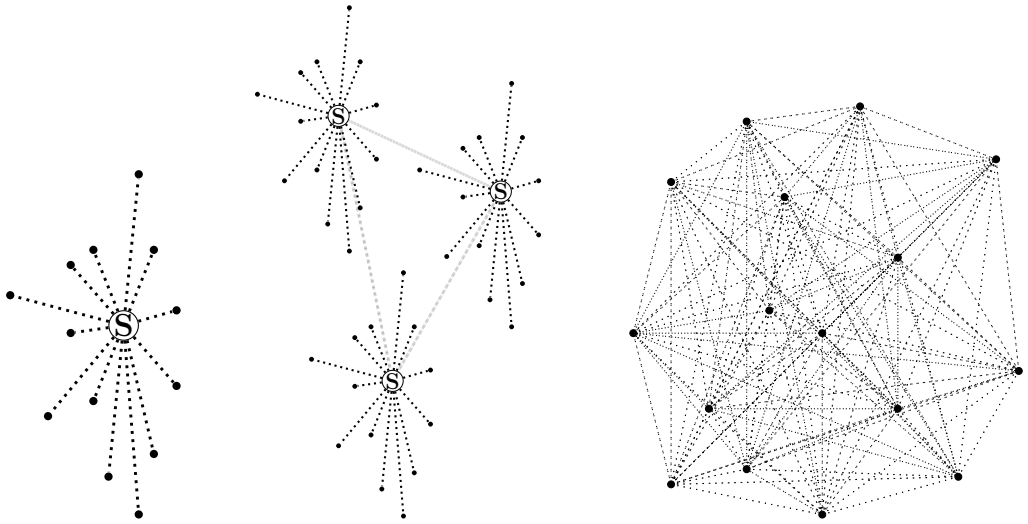
1.2 Réseaux distribués

Un réseau est un ensemble d'équipements communiquant entre eux et formant ainsi un système. Les éléments de ce système sont appelés des *nœuds*. Un nœud peut émettre une requête à un autre et éventuellement recevoir une réponse. Les réseaux soutiennent les applications de communication que nous utilisons de nos jours ont une incidence très significative sur de nombreux aspects des communications. En plus de la protection de la vie privée, on peut distinguer les trois caractéristiques fondamentales suivantes :

- la *cohérence* : capacité d'un réseau à fournir une même réponse à une même requête par deux nœuds différents du réseau ;
- la *disponibilité* : garantie que toutes les requêtes reçoivent une réponse ;
- et la *tolérance au partitionnement* : la coupure d'un nombre inférieur au nombre total de nœuds du réseau n'empêche pas le système de répondre.

L'expression centrale de ces concepts est exprimée dans le théorème CDP [31] (aussi connu sous l'acronyme « CAP ») conjecturé par Eric Brewer et ensuite exposé lors de la conférence *Symposium on Principles of Distributed Computing* (PODC) [13] à l'année 2000 menant alors à la preuve de celui-ci. Ce théorème stipule qu'avec la nécessité de tolérer le partitionnement du réseau, un protocole devra favoriser soit la cohérence ou soit la disponibilité.

Exemple 1.2.1. *Une application de messagerie permet l'envoi et l'affichage de*



(a) Centralisé (b) Décentralisé (c) Distribué

FIGURE 1.1: Types de réseaux. Le symbole Ⓢ représente un serveur.

message entre utilisateurs. Lorsque le système sous-jacent est partitionné, les utilisateurs des différentes partitions s'échangeant des messages (disponibilité du système garantie) ne voient pas une même liste de messages dans un même ordre (cohérence incertaine). Sinon, si le système réagit plutôt en empêchant l'échange de messages (indisponibilité), alors les utilisateurs sont garantis de ne pas désynchroniser l'état de l'historique de messages (cohérence). ◊

Ce résultat exprime très bien la grandeur du défi des experts dans la construction du réseau Internet qui obéit à ce théorème. Cette plateforme a ensuite donné naissance à une multitude d'applications qui ont été construites de façon telle à former des réseaux se posant par-dessus Internet.

Définition 1.2.1 (Réseau de recouvrement)

Un *réseau de recouvrement* est un réseau virtuel qui se forme en se superposant à un (ou plusieurs) autre(s) réseau(x) physique(s). Ce dernier est appelé *réseau sous-jacent*.

Référence(s) : [30, 57]

Remarque 1.2.1. Le réseau de recouvrement est donc formé d'un sous-ensemble de nœuds du réseau sous-jacent. ◊

Exemple 1.2.2. *Le réseau Simple Mail Transfer Protocol (SMTP) acheminant les messages des utilisateurs des services de courriel est un réseau de recouvrement. Son réseau sous-jacent est Internet.* ◊

La Toile, qui comprend entre autres les services HTTP des grands fournisseurs très bien connus tels que Facebook, YouTube et Google, est un réseau de recouvrement. Aujourd'hui, la structure de ce réseau est considérée comme standard, mais pas pour autant dépourvue de défaillance. Contrairement au réseau Internet, la Toile est *centralisée* (figure 1.1a), c'est-à-dire qu'une requête ne peut s'acheminer qu'à un nœud spécial appelé « serveur ». Cette caractéristique s'oppose à celle de la *décentralisation* (figure 1.1b) où plusieurs nœuds spéciaux composent le système. Finalement, une décentralisation totale mène à un modèle qu'on nomme *distribué* (figure 1.1c) où toute requête peut être adressée à n'importe quel nœud du réseau. Dans ce document, nous nous intéressons spécifiquement aux réseaux distribués.

Bien que la Toile soit un modèle qui s'impose aujourd'hui comme le standard le plus répandu pour soutenir les applications d'utilisateurs finaux, de nombreux projets récents prônent un retour à l'architecture décentralisée et distribuée [7, 9, 11, 20, 80]. Ci-après, des descriptions sommaires de certains d'entre eux.

Diaspora [9] est un réseau social décentralisé. Sur cette plateforme, les utilisateurs choisissent un fournisseur (serveur, dénoté par « pod ») comme on choisit

un fournisseur pour un service courriel. Tous les utilisateurs, clients de tous les fournisseurs confondus, communiquent dans un même réseau. Il est facile d'ajouter un fournisseur personnel au réseau et ainsi avoir un contrôle accru.

IPFS [7], un acronyme pour « interplanetary filesystem », est un système de fichier distribué visant à remplacer HTTP comme structure sous-jacente à la Toile. Il offre exactement les mêmes opérations qu'un système de fichier, c'est-à-dire *l'ajout/suppression* de fichier [5]. Sa particularité est que lorsqu'un fichier est lu par un nœud, ce deuxième sert alors de source pour ce premier. On assigne à chaque fichier est attribué un identifiant unique permettant de retrouver celui-ci par l'exécution d'un procédé de routage dans un réseau distribué du type table de hachage distribuée (THD) (plus de détail dans la section **??**).

Tox [20] est une plateforme de communication audio/vidéo et clavardage fonctionnant sur un réseau distribué du type THD. Ce projet libre entièrement soutenu par sa communauté vise à donner un moyen de communication sûr bout en bout sans la nécessité d'intermédiaire pour communiquer. Ce modèle est en compétition avec celui de Matrix [64] qui adopte plutôt une structure décentralisée (à plusieurs serveurs) comme dans le cas de Diaspora. Le protocole Matrix peut être utilisé par différentes applications d'origine variée telles que Riot, [80] car son protocole est entièrement libre.

Syncthing [11] est une plateforme de partage de fichiers distribuée permettant l'échange de répertoires entre différents appareils d'un individu ou entre plusieurs d'entre eux par un simplement glissement de fichiers sous la racine d'un partage (un chemin dans le système partagé par Syncthing). Ce programme permet entre autres une redondance de données pour garantir une sécurité des données en cas de défaillance, une synergie complète entre différents appareils pour par exemple

5. Un répertoire est un fichier.

générer une configuration logicielle redondante sur différentes machines ou encore simplement partager des fichiers quelconques comme des photos entre différents utilisateurs. Il est pertinent de mentionner que la découverte d'appareil fonctionne, quant à elle, grâce à un protocole centralisé. Il fait présentement l'objet d'étude chez les concepteurs de logiciel de Syncthing de trouver une solution complètement distribuée (par exemple par l'utilisation de THDs). De plus, il ne faut pas confondre IPFS avec Syncthing ; le premier ne se veut pas une application d'utilisateur final, mais plutôt un système sur lequel bâtir des applications ; alors que Syncthing est seulement une application d'utilisateur final.

Dans les prochaines sections, nous détaillons les concepts fondamentaux sur lesquels les différents projets cité plus haut se basent, c'est-à-dire les tables de hachage distribuées.

1.3 Tables de hachage distribuées

Les tables de hachages distribuées jouent de façon récurrente un rôle pertinent dans l'architecture de différents logiciels communiquant sur un réseau comme discuté dans la section 1.2. Dans cette section-ci, nous définissons en quoi consiste exactement les THDs. Nous prenons aussi l'occasion de discuter de quelques modèles particuliers comme Pastry [82], Chord [90] et Kademlia [65]. Nous réserverons une section entière à ce dernier qui consiste en une base élémentaire du chapitre 2.

Selon Fayçal [30] et Lua et collab. [57], une *table de hachage distribuée (THD)* est un réseau de recouvrement pair-à-pair accomplissant le stockage de valeurs structurées comme des paires de la forme (K, V) avec $K \in \{0,1\}^m$ l'identifiant de la valeur arbitraire $V \in \{0,1\}^*$. Le paramètre m est le nombre de bits d'une image par la fonction de hachage cryptographique de la table de hachage. Les données sont distribuées plutôt que répliquées à l'échelle du réseau. Ce faisant

chaque nœud à lui seul n'a qu'un sous-ensemble restreint de l'ensemble total des données.

Les THDs possèdent une interface canonique simple qui se résume aux deux fonctions `get` et `put`. Ces fonctions accomplissent respectivement l'obtention et l'insertion d'une valeur.

1.3.1 Routage

Le réseau de recouvrement que constitue une THD est du type *structuré*, en ce sens qu'une opération sur le réseau est effectuée selon un algorithme particulier que tous les nœuds connaissent et valident selon un ensemble de critères donnés afin de discriminer les *bons messages* des *mauvais*. En particulier, les THDs sont structurées de telle façon à ce que l'acheminement d'un message destiné à un nœud donné succède à une recherche de celui-ci par un processus qu'on nomme *routage*. Le routage est propre aux modèles et a une incidence considérable sur la complexité des instructions canoniques *get/put*. Le routage est normalement associé à une *métrique* qu'on emploie afin de calculer la distance entre différents nœuds. De cela, émane la structure sur l'espace des identifiants des nœuds, formant ainsi l'organisation du réseau. Par exemple, Kademlia utilise la métrique du « ou-exclusif » sur les identifiants des nœuds, permettant donc d'ordonner ceux-ci par leur distance avec un identifiant donné.

1.3.2 Pastry

Rowstron et Druschel [82] ont conçu leur modèle de telle sorte que chaque nœud est attribué un identifiant I d'une longueur de 128 bits qu'il présente aux autres nœuds afin de se positionner dans le réseau relativement aux autres. L'identifiant d'un nœud est supposé être déterminé de manière uniformément aléatoire sur l'espace

des clefs de 0 à $2^{128}-1$. Les identifiants sont analysés par chiffres de b bits dans la base 2^b. Le nombre b est un paramètre dont la valeur est conventionnellement 4. Chaque nœud connaît un entourage L de nœuds avec un préfixe commun à leur identifiant ordonné numériquement ($|L|$ est typiquement 16 ou 32).

Chaque nœud maintient trois listes de nœuds ordonnés selon des critères différents. La première liste est la *table de routage* contenant les informations de certains nœuds rencontrés dans le réseau. Ces informations sont ordonnées par la longueur du préfixe en commun avec l'identifiant du nœud courant. La ligne i contient 2^b nœuds ayant respectivement pour identifiant selon la colonne j la valeur $I_{ij} = \mathsf{Préf}_i(I) \cdot j \cdot r_{ij}$ où $\cdot$ représente la concaténation. La chaîne r_{ij} est une valeur quelconque de $128-i-1$ bits, propre à chaque nœud (c'est le reste de leur identifiant).

La seconde liste est l'ensemble des nœuds les plus proches du nœud courant selon une métrique donnée (le nombre de sauts dans le réseau IP pour acheminer un message p. ex.). La dernière contient la liste des nœuds environnants selon la proximité numérique sur les identifiants.

Le routage vers une clef K donnée est effectué selon une recherche des nœuds ayant le préfixe le plus long en commun avec K pour ensuite compléter la recherche dans la liste L. Le préfixe est analysé selon les chiffres (de b bits). Dans chaque case de cette table, plusieurs nœuds rencontrés dans le réseau peuvent être utilisés. Par conséquent, parmi tous les nœuds satisfaisant au préfixe donné, les 2^b nœuds les plus proches selon la métrique de la deuxième liste sont choisis. Les instructions *get/put* sont exécutées en $\mathcal{O}(\log N)$ étapes où N est le nombre total de nœuds dans le réseau. Finalement, les valeurs sont stockées sur plusieurs nœuds de façon redondante de sorte à garantir un plus haut niveau de disponibilité et de tolérance aux erreurs.

1.3.3 Chord

ChordStoica et collab. [90] utilisent une approche différente de Pastry et Kademlia (présenté à la section 1.3.4) à plusieurs égards tout en présentant certaines similarités. Le réseau est configuré avec un paramètre m spécifiant la grandeur de l'espace des clefs circulaire $E = [0, 2^m)$. Chaque nœud est identifié par un nombre dans l'intervalle E dérivé d'une image par la fonction de hachage SHA-1.

Une requête pour une clef K est acheminée à un nœud n dont l'identifiant $n.id$ satisfait à

$$\forall n' \in N,\ n'.id \notin (K, n.id),$$

c'est-à-dire que n est le nœud ayant l'identifiant le plus proche de K dans le sens horaire du cycle modulo. On appelle un tel nœud le *nœud successeur* de K. Les auteurs fournissent deux algorithmes de recherche dans l'espace des clefs. Le premier est une approche naïve qui s'ajoute en complément à la seconde, plus évoluée. L'approche naïve est celle où tous les nœuds connaissent leur successeur et ainsi permettent de compléter une recherche linéaire sur l'espace des clefs. Évidemment, cela n'est pas efficace et peut très bien s'optimiser. La seconde approche vise justement à cela en ajoutant la restriction où chaque nœud possède une table nommée la *table de doigts* ne contenant pas plus de m lignes. Ces lignes comportent l'information sur le successeur de l'identifiant $[n.id + 2^i \mod 2^m], \forall i \in [m]$. Cette table permet de réduire la complexité de l'algorithme de recherche de $\mathcal{O}(N)$ à $\mathcal{O}(\log N)$. La procédure résultante commence donc par une recherche sur la table des doigts et complète la recherche par une recherche linéaire dans la région la plus rapprochée de l'identifiant recherché.

1.3.4 Kademlia

Maymounkov et Mazieres [65] proposent un système se démarquant considérablement de Pastry [82] et Chord [90]. Bien que Pastry emploie une approche par préfixe comme Kademlia, le premier utilise plus précisément deux métriques différentes afin de compléter la recherche, engendrant donc certaines discontinuités, ce qui complexifie le modèle et l'analyse dans un contexte du pire cas algorithmique. Ce qui démarque principalement Kademlia de ses alternatives est sa simplicité. La procédure de routage se fait en une seule phase et ne nécessite qu'une seule même opération à exécuter de façon itérative la *distance (ou métrique) XOR*.

Tout nœud se fait attribuer un identifiant de 160 bits (en calculant une valeur aléatoire à l'aide de la fonction de hachage SHA-1 par exemple). La structure du réseau est un arbre binaire comme le montre la figure 1.2. La position d'un nœud est déterminée par le plus court préfixe unique de son propre identifiant. Dans l'exemple de la figure, le nœud représenté par un cercle rempli possède un identifiant dont le préfixe unique le plus court est 100. Pour tous les nœuds, l'arbre est divisé en une suite de sous-arbres (encerclés dans la figure 1.2) ne contenant pas le nœud courant. Le protocole est conçu de telle manière que tous les nœuds connaissent au moins un nœud dans chaque sous-arbre, de sorte que les nœuds forment un graphe connexe, c'est-à-dire que tout nœud peut éventuellement communiquer avec tout autre nœud.

Les nœuds sont découverts dans le réseau grâce à un algorithme de recherche de complexité $\mathcal{O}(\log N)$ convergeant en accumulant successivement de l'information sur les nœuds les plus « proches » de l'identifiant recherché par une série de requêtes pour d'autres nœuds encore plus proches. Ce processus est répété jusqu'à

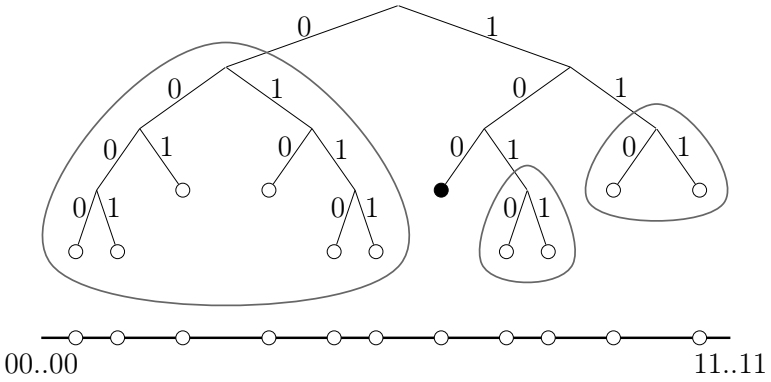


FIGURE 1.2: Kademlia : arbre binaire[6]. Le symbole « ● » dénote le nœud dont l'identifiant est `100...` dans l'arbre. La position respective de chacun des nœuds dans l'espace des clefs est dénotée par le symbole « ○ » placé vis-à-vis sur la ligne.

convergence vers le nœud le plus près de l'identifiant recherché.

MÉTRIQUE XOR

Afin de calculer la distance entre différents nœuds et identifiants, pour finalement déterminer le nœud le « plus près » d'un identifiant donné, les auteurs proposent d'utiliser une métrique $d : \{0,1\}^{160} \times \{0,1\}^{160} \to \{0,1\}^{160}$ définie comme $d(x,y) = x \oplus y$. On remarque que cette définition est complètement conforme puisque $\forall x, y \in \{0,1\}^{160}$:

$$d(x,x) = 0, \quad [x \neq y \implies d(x,y) > 0] \quad \text{et} \quad d(x,y) = d(y,x)$$

De plus, l'inégalité du triangle est satisfaite par l'opération puisque suivant que $\forall x, y \geq 0$, $x + y \geq x \oplus y$, on a que

$$d(x,y) + d(y,z) \geq d(x,y) \oplus d(y,z) = d(x,z).$$

L'ordre de grandeur de la distance entre deux nœuds est la hauteur du plus petit sous-arbre contenant les deux nœuds en question. Finalement, pour toute paire

6. Cette figure est inspirée des figures trouvées dans l'article de Maymounkov et Mazieres [65].

$x, \delta \in \{0,1\}^{160}$ une seule valeur $y \in \{0,1\}^{160}$ satisfait à $x \oplus y = \delta$. Les auteurs distinguent ce type de métrique en la qualifiant *d'unidirectionnelle*. Ceci permet intuitivement que toute recherche pour une même clef converge le long d'un chemin identique.

Les k-seaux

Un nœud stocke les informations sur les nœuds qu'il croise sous forme de triplets (A, P, I) où A est l'adresse IP du nœud, P est le port UDP sur lequel contacter le nœud et I est l'identifiant du nœud. Ces informations sont classées dans des listes — ordonnées par le temps écoulé depuis le dernier contact — selon leur distance avec l'identifiant du nœud courant. Plus précisément, pour tout $0 \leq i < 160$, les nœuds d'identifiant $I \in \{0,1\}^{160}$ sont classés dans la i^{e} liste si $2^i \leq I \oplus I_c < 2^{i+1}$ où I_c est l'identifiant du nœud courant. On appelle ces listes les k-seaux (ou « k-bucket » en anglais). Pour les petites valeurs de i, les k-seaux sont souvent vides, alors que ceux pour des valeurs de i plus grandes atteingnent k. La valeur k est un paramètre choisi de telle sorte à garantir un haut niveau de fiabilité (les auteurs suggèrent 20).

Opérations

L'opération centrale du protocole est celle faisant la recherche des k nœuds les plus proches d'un identifiant donné I. Cette procédure démarre par l'envoie d'une requête à α nœuds tirés du k-seau le plus près de I recherché. Si le seau contient moins de α nœuds, alors le seau adjacent est utilisé pour combler le nombre (α est un paramètre valant communément 3 [65]). La requête à ces nœuds attend comme retour la liste des k nœuds les plus près de I. Récursivement, le nœud renvoie les mêmes requêtes aux α nouveaux nœuds les plus près de I jusqu'à ce qu'aucun nœud ne retourne d'information sur des nœuds encore plus près. Ainsi, de la liste

des nœuds connus dans les étapes précédentes, les k nœuds les plus près de I sont retournés.

Toutes les autres opérations sont alors basées sur la précédente. Afin de récupérer les valeurs stockées pour un identifiant donné, la recherche décrite dans le paragraphe précédent est effectuée. Enfin, le nœud envoie une requête de valeurs aux k nœuds déduits de l'étape précédente.

De façon similaire, afin de stocker une paire clef/valeur (C, V), l'étape de recherche est premièrement effectuée pour ensuite transmettre une requête de stockage aux k nœuds les plus près de C.

Finalement, Kademlia prévoit une expiration des valeurs après 24 heures afin de prévenir un gonflement de l'espace de stockage sur les nœuds individuels. Ce faisant, dans le but de garantir la présence d'une valeur pour une durée plus longue, le nœud à l'origine du stockage devrait annoncer de nouveau sa valeur périodiquement. Il est à noter que Maymounkov et Mazieres proposent des temps de vie plus longs en fonction du type de valeur comme des certificats cryptographiques par exemple.

Performance

En cas de partitionnement réseau ou tout simplement quand certains nœuds ne sont pas joignables, Kademlia prévoit un mécanisme pour garantir un niveau d'accessibilité élevé. En effet, comme une valeur est stockée de manière redondante sur k nœuds pour chaque publication, il ne suffit que d'un seul nœud sur k soit accessible pour compléter une requête de récupération de données. En prenant pour hypothèse que tout nœud choisit son identifiant de façon uniformément aléatoire dans $\{0,1\}^{160}$, alors, pour tout identifiant donné, un groupe de k nœuds répartis uniformément dans l'ensemble du réseau est responsable de répondre à une

requête donnée. Ce faisant, la probabilité qu'une requête réussisse est indépendante de l'identifiant. Ainsi, en contrôlant k on peut obtenir un taux de succès satisfaisant. Par exemple, l'implémentation Kademlia OpenDHT utilise $k = 8$.

Le chapitre suivant porte sur l'implémentation d'une table de hachage du type Kademlia nommée OpenDHT.

CHAPITRE II

OPENDHT : APPLICATION KADEMLIA

Nous commençons maintenant le volet portant sur les éléments des systèmes distribués. Plus particulièrement, ce chapitre porte sur OpenDHT, une implémentation de table de hachage distribuée Kademlia (voir la section 1.3.4). Une description de la bibliothèque logicielle et de ses composants débute le chapitre. Ensuite, les différentes contributions au projet en termes d'optimisation d'accessibilité et de rendement réseau sont détaillées. Finalement, nous présentons nos travaux réalisés et publiés dans [24].

2.1 OpenDHT

Le projet OpenDHT [1] est une implémentation sous forme de bibliothèque `C++11` du standard Kademlia (détaillé à la section 1.3.4) soumise sous licence GPLv3 (voir l'annexe C). Celle-ci fournit plusieurs utilités particulières, dont l'opération non canonique `listen` qui permet l'écoute des mises à jour de l'espace de stockage pointé par une clef donné. Ensuite, il y a la prise en charge simultanée des protocoles IPv4 et IPv6. Finalement, une couche d'outils cryptographiques est à la disposition du concepteur logiciel sous forme d'une interface de programmation

1. On peut trouver le code source du projet sur Github à l'URL suivante : http://github.com/savoirfairelinux/opendht.

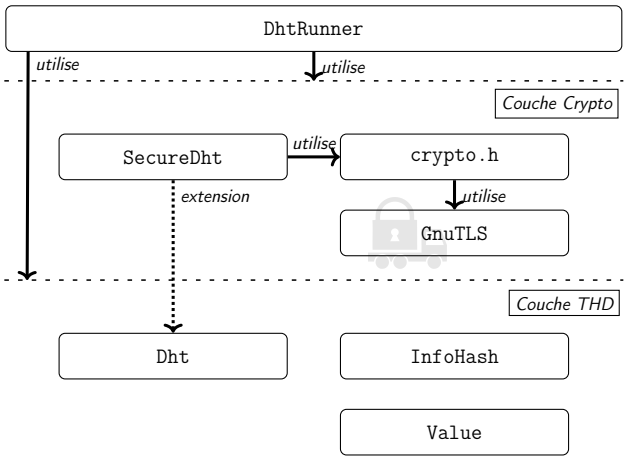


FIGURE 2.1: Interraction entre les composants d'OpenDHT

applicative (IPA) de fonctions de chiffrement symétriques et asymétriques écrites à l'aide de GnuTLS [19]. La figure 2.1 donne une représentation simplifiée de la structure de la bibliothèque et de ses composants.

Le projet est divisé en trois couches principales : l'interface principale, la strate des préoccupations de sécurité et la couche métier. Au sein de cette dernière, on y retrouve les structures de données `Value` et `InfoHash` qui représentent respectivement une valeur entreposée et une clef de stockage. Le sous-composant principal `Dht` contient l'entièreté des routines et structures de la table de hachage distribuée. Les opérations effectuées par ce composant incluent :

— la création et sérialisation des requêtes ;

— la logique de réception des requêtes ;

— le calcul des délais entre chaque envoi de requête ;

— et le calcul des nœuds les plus proches (selon la métrique $\oplus$, voir la section 1.3.4) d'une cible (empreinte) donnée.

À l'interface de cette strate, on trouve les signatures des primitives `get` et `put` en plus de `listen`.

```
size_t Dht::listen(const InfoHash& key, GetCallback cb, Value::Filter filter={}, Query query={});
```

Listage 2.1: Opération `listen` [2]

Ensuite, on gère les préoccupations de sécurité dans le niveau supérieur. L'interface de cette couche offre des fonctionnalités étendues de celles qui sont exposées par le composant inférieur. Par conséquent, lorsque ce dernier retrouve une donnée dans le réseau, il la relaye au niveau supérieur quit déchiffre les données chiffrées et vérifie les signatures, le cas échéant. Évidemment, c'est aussi à ce niveau que le chiffrement et la signature des données s'opèrent pour l'utilisateur de la bibliothèque. Il est à noter que les fonctionnalités de sécurité offertes par l'interface de `crypto.h` sont également accessibles directement sans passer par `SecureDht`.

Finalement, le composant `DhtRunner` est l'interface principale de la bibliothèque. Il supervise l'exécution des requêtes à la bibliothèque placées dans une file en mode asynchrone. Il expose les différentes fonctions de `Dht` et `SecureDht` tout en gérant les préoccupations fondamentales d'un programme connecté comme la prise réseau (*socket* en anglais).

2.1.1 Opération `listen`

Comme décrit précédemment, l'opération `listen` est une particularité d'OpenDHT. Cette procédure permet à un nœud donné de faire la requête d'être tenu informé des valeurs annoncées pour une certaine clef. Son interface d'utilisation est fournie dans le Listage 2.1.

Les arguments sont respectivement une clef de stockage `key`, une fonction de

2. Selon https://github.com/savoirfairelinux/opendht/wiki/API-Overview en date du 14 décembre 2018.

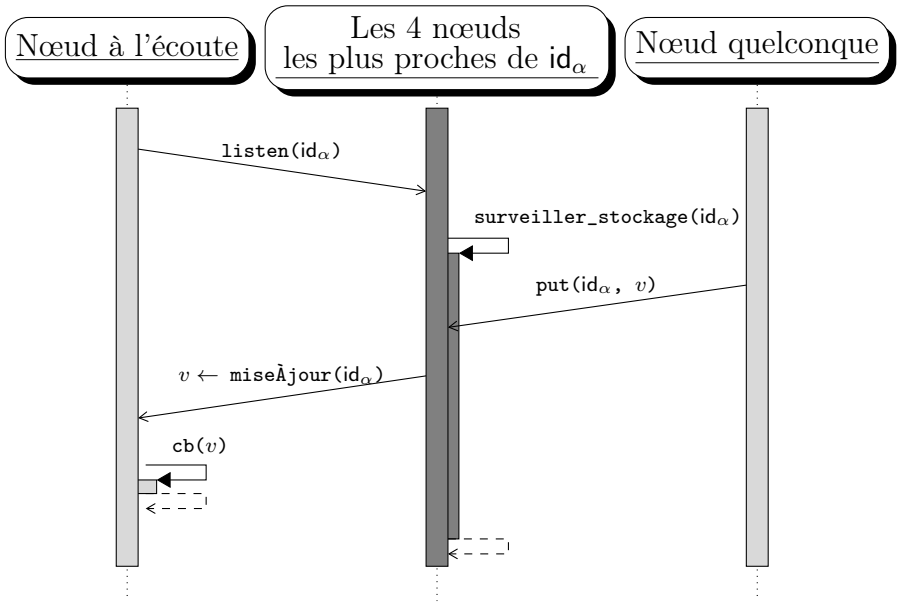


FIGURE 2.2: Séquence des messages échangés suivant une requête `listen`.

rappel `cb` ainsi que les paramètres optionnels `filter` et `query`. Nous verrons plus en détail la nature des troisième et quatrième champs dans la section 2.3. La figure 2.2 illustre les opérations réalisées. Le premier fil d'exécution est celui du nœud à l'origine de la requête ; le second fil représente les quatre nœuds les plus près de la clef id_α ; alors que le dernier est celui d'un nœud quelconque effectuant une requête `put` pour la même clef. Lorsqu'une mise à jour est reçue par le nœud initial, celui-ci invoque la fonction de rappel `cb` mentionnée dans le Listage 2.2 avec comme paramètre la valeur obtenue v.

Ci-après un exemple d'utilisation de cette routine dans le langage `C++`. Une clef aléatoire est générée par l'instruction `InfoHash::getRandom()` et la fonction de rappel affiche toutes les valeurs reçues lors de la mise à jour.

```
auto key = dht::InfoHash::getRandom();
dht::DhtRunner dht {};
dht.run();
dht.listen(key, [](const std::vector<std::shared_ptr<dht::Value>> values) {
    for (auto& v : vals) {
        std::cout << v->toString() << std::endl;
    }
    return true;
});
```

Listage 2.2: Exemple d'utilisation de la primitive `listen`.

2.2 OpenDHT : persistance de données

L'opération canonique `put` (illustrée par la figure 2.3) s'accomplit en deux étapes (voir la section 1.3.4) : (i) recherche des huit (8) nœuds les plus près de la clef (par la métrique $\oplus$) et (ii) envoi de la demande de stockage de la valeur aux huit nœuds trouvés précédemment. Les données entreposées dans le réseau ont une longévité temporelle par défaut de dix (10) minutes. Certaines valeurs spéciales comme les certificats cryptographiques sont stockés pendant une semaine. Pour des valeurs dites « normales », l'émetteur de celles-ci pourra renouveler sa requête de publication de façon périodique pour ainsi augmenter la durée de vie de ses données.

Les huit nœuds interrogés lors de la transmission d'une requête de stockage consistent donc en lieu d'entreposage. Pour rappel, la position de ces nœuds dans la topologie du réseau est relative à la clef associée aux valeurs et la liste des nœuds connus dans la même région.

Considérons le scénario de la figure 2.3. Dénotons la suite finie des nœuds ordonnés par leur identifiant et à proximité de la clef id_α par $N = (n_i)_{i \in [8]}$. Si un nouveau nœud n_0 accédait au réseau alors que $\exists n \in N$ satisfaisant à $\mathsf{id}_{n_0} \oplus \mathsf{id}_\alpha < \mathsf{id}_n \oplus \mathsf{id}_\alpha$,

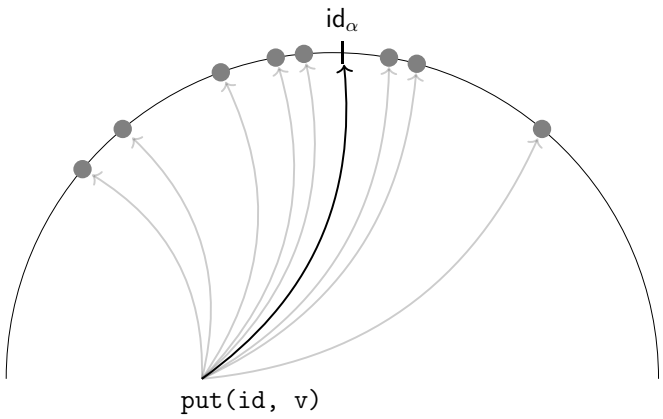


FIGURE 2.3: Opération `put`

alors il se retrouverait dorénavant dans la liste des nœuds les plus près de id_α lors des recherches (voir section 1.3.4). De plus, le nœud $n' \in N$ le plus éloigné de id_α par son identifiant n'apparaîtrait plus dans cette même liste. Ce faisant, on constate que l'arrivée d'un nœud plus près de la cible retire donc un nœud possédant déjà les valeurs emmagasinées relatives à la clef id_α. La conséquence directe est qu'un nœud de moins stocke les données. Notons que l'effet d'un nœud qui quitte le réseau est le même. Cette variation de l'ensemble de nœuds est appelé « remous » (*churn* en anglais). Le remous a une incidence sur l'accessibilité des données et nuit à la propriété de persistance des données. Maymounkov et Mazieres [65] proposent (1) qu'au moment de l'arrivée d'un nouveau nœud n_0, alors $\forall n \in N$, n envoie une requête de stockage à n_0 pour toutes les valeurs que n_0 devrait entreposer et (2) lorsque $n \in N$ quitte, celui-ci devrait transférer les données qu'il stocke au nouveau nœud qui prend sa place. Bien sûr, cette dernière suggestion n'est pas forcément réalisable puisqu'un nœud est à tout moment vulnérable à une interruption, de ses communications, hors de son contrôle. Cependant, comme mentionné dans la section 1.3.4, Kademlia possède un mécanisme de prévention à cet égard.

Concernant le point (1), nous avons choisi une solution différente où tous les nœuds effectuent une gestion de leur espace de stockage de façon périodique (à chaque dix minutes). Ensuite concernant le point (2), nous avons implémenté la suggestion telle quelle où chaque nœud sortant annonce les valeurs qu'il a avant de quitter. De cette manière, le réseau peut donc assurer la persistance de données même en présence de remous. Selon les auteurs de Kademlia, l'analyse des arrivées et départs des nœuds d'une table de hachage distribuée montre que la remous peut être fréquent et donc engendrer des coûts importants de bande passante lors de l'exécution de procédures naïves comme celles dépeintes précédemment. Une deuxième contribution au projet, décrite à la section 2.3, permet d'appliquer des optimisations majeures (aussi décrites par [65]) et ainsi réduire substantiellement le trafic sur la THD.

2.3 OpenDHT : pagination de valeurs

Comme la THD constitue un système de stockage de données, il est légitime de lui adjoindre des fonctionnalités supplémentaires telles que celles qu'on retrouve dans les systèmes de base de données usuels. Par défaut, une requête `get` engendre l'envoi par les huit nœuds les plus proches de toutes les valeurs qu'ils emmagasinent pour une clef donnée. En lieu et place des valeurs, il peut être désirable de ne récupérer que les identifiants des valeurs. En termes de base de données, cette primitive équivaut naturellement à une instruction `SELECT` sur le champ `id`, l'identifiant d'une valeur. Il va alors de soi d'ajouter au système la primitive `WHERE` correspondant à la discrimination des valeurs THD. Un tel système rend possible la *pagination de valeurs*, c'est-à-dire la scission d'une requête `get` en plusieurs petites requêtes. Cela permet donc un meilleur contrôle du trafic par le programmeur et la sélection plus fine de champs (`SELECT`) ou de valeurs (`WHERE`) par identifiants, types de valeur, l'identifiant de l'émetteur (l'empreinte d'une clef publique) et un

type personnalisé.

L'implémentation de ces idées dans la version 1.0.0 (validation `07e4435a`) s'est traduite par l'écriture des trois composants `Select`, `Where` et `Query`. Il s'agit respectivement des requêtes de base de données `SELECT`, `WHERE` et une dernière structure enveloppant les deux autres afin d'y centraliser la sérialisation de la combinaison des deux premières. L'interface de ces structures est décrite dans les listages 2.3-2.6. Les champs pouvant être sélectionnés sont résumés dans la table 2.1.

```
void get(const InfoHash& k, GetCallback cb, DoneCallback dcb={}, Value::Filter f={}, Query q={});
size_t listen(const InfoHash& key, GetCallback cb, Value::Filter filter={}, Query query={});
void query(const InfoHash& key, QueryCallback cb, DoneCallback dcb={}, Query&& q={});
```

Listage 2.3: Interface de la pagination de valeurs (`DhtRunner`) [3]

```
Select();                          // constructeur 1
Select(const std::string& q_str);  // constructeur 2
Select& field(Value::Field field); // sélection d'un champ
```

Listage 2.4: Interface du composant `Select`.

```
Where();                                // constructeur 1
Where(const std::string& q_str);        // constructeur 2
Where& id(Value::Id id);                // restriction sur id
Where& valueType(ValueType::Id type);   // restriction sur ValueType
Where& owner(InfoHash owner_pk_hash);   // restriction sur OwnerPk
Where& seq(uint16_t seq_no);            // restriction sur Seq
Where& userType(std::string user_type); // restriction sur UserType
```

Listage 2.5: Interface du composant `Where`.

3. Selon https://github.com/savoirfairelinux/opendht/wiki/API-Overview en date du 23 décembre 2018.

TABLE 2.1: Champs d'une valeur THD

Champ	Description
`id`	Numéro identifiant une valeur (≠ clef THD).
`ValueType`	Type de valeur : normale (par défaut), certificat cryptographique, valeur ICE [4].
`OwnerPk`	Empreinte de la clef publique de l'émetteur d'une donnée.
`Seq`	Numéro de séquence d'une valeur (incrémenté lorsqu'une valeur signée est changée).
`UserType`	Chaîne de caractères personnalisée.

```
1 Query(Select s={}, Where w={}, bool none=false); // Constructeur
```

Listage 2.6: Interface du composant `Query`.

OpenDHT met en pratique ces concepts pour chaque requête `get`. Une requête `Query` est premièrement construite avec la primitive `SELECT` :

```
1 auto select_q = std::make_shared<Query>(Select {}.field(Value::Field::Id), Where {});
```

L'instruction ci-haut invoque le constructeur de `Select` suivi d'un appel à la fonction `field` de l'instance précédemment construite permettant ainsi de spécifier `id` comme champ sélectionné. La requête est subséquemment transmise aux 8 nœuds les plus proches de la clef demandée. À la réception de la réponse, une requête `get` par champ `id` obtenu est envoyée avec la primitive `WHERE` :

4. Le protocole « Interactive Connectivity Establishment », https://tools.ietf.org/html/rfc5245

```
auto query_for_vid = std::make_shared<Query>(Select {}, Where {}.id(id));
```

Cette instruction scinde ainsi la requête initiale `get` en plusieurs requêtes individuelles. À la réception des valeurs comme telles, celles-ci sont passées aux couches supérieures, comme décrit dans la section 2.1.

2.3.1 Persistance de données : optimisation

Comme énoncé dans la section 2.2, l'approche naïve pour réaliser la persistance de données engendre une importante hausse de bande passante. Cet effet nuisible peut toutefois être atténué par l'optimisation de l'algorithme de recherche de données en employant une requête renseignant sur les données présentes chez un nœud. Cette suggestion, faite par Maymounkov et Mazieres [65], engendre un coût inférieur à celui du transfert des valeurs comme tel. Il apparaît donc comme évident que la pagination de valeurs permet d'accomplir ce type de manipulation. En effet, depuis la version 1.0.1 (validation `8f2cc937`) , un nœud OpenDHT transmet une requête de sondage `SELECT` pour l'identifiant de la valeur qu'il souhaite publier. Une fois la réponse obtenue, il ne transmet la valeur que dans les cas suivants :

— la valeur ne figure pas déjà dans l'espace de stockage (requête `put`) ;

— la valeur doit être rafraîchie, car le temps de vie de la valeur est quasiment atteint (requête `refresh`[5]) ;

— ou bien le numéro de séquence de la valeur signée (par une clef publique) est inférieur à celui qu'on tente d'annoncer.

Ce patron d'instructions apporte certes des avantages pertinents comme l'optimisation susmentionnée. Toutefois, l'étendue des possibilités acquises par ce mécanisme est relativement limitée. Dans la prochaine section, nous détaillons une

5. Demande au destinataire de garder la valeur (déjà en place) pour dix minutes de plus.

strate complètement indépendante d'OpenDHT qui permet de décupler le nombre d'actions que procure une simple THD.

2.4 Indexation complètement distribuée sur une table de hachage distribuée

Indexer des données consiste à les identifier par un certain mécanisme dans une structure de données de façon à faciliter leur recherche. Ce type de problème a largement été étudié dans le cas de systèmes opérants de manière centralisée [5, 6, 8, 18, 23, 71, 100], faisant émerger des structures d'indexation basées sur les structures de données comme les arbres-B(+), les tableaux de bits, les arbres k-d, les tries et les tables de hachages. Ces techniques permettent donc la prise en charge des requêtes de recherche dans un modèle centralisé tel qu'un moteur de recherche dans un système de fichiers comme celui exposé par un serveur Web comme Apache ou bien pour des moteurs de recherche globaux comme DuckDuckGo, StartPage, Google et Yahoo.

Comme évoqué précédemment, les systèmes centralisés sont prédisposés à la censure, ils ne passent pas à l'échelle, c'est-à-dire que l'accessibilité du système décroît lorsque le nombre de requêtes grandit et ils sont fortement sujets aux attaques par déni de service. En revanche, les tables de hachages distribuées bien connues telles que Chord [91] et Kademlia [65] sont des systèmes passant très bien à l'échelle, réduisant ainsi les diverses préoccupations énoncées. Ces systèmes sont donc d'un intérêt majeur pour combattre les différents problèmes de censure et d'accessibilité comme dans les systèmes VoIP [22, 85], les systèmes de fichiers [21, 25] et bien d'autres.

L'interface canonique d'une THD se ramène à des requêtes pour des correspondances (de clefs) exactes. Par exemple, pour faire une instruction `get` avec le protocole Kademlia, on doit connaître la clef de 160 bits associée aux valeurs

qu'on souhaite récupérer. Cependant, si on recherche ces valeurs, c'est qu'on ne connaît pas la clef d'avance ! Il n'est alors pas possible à première vue de faire des requêtes pour des clefs partielles à la manière d'un moteur de recherche (par exemple chercher pour « le petit chaperon rouge »). La question à savoir s'il peut émerger de tels systèmes par la modification des algorithmes originaux ou bien par l'élaboration de systèmes faisant l'usage habile des THDs avec leur conception actuelle s'est avérée donc pertinente dès le début des années 2000s.

Ramabhadran et collab. [78] suggèrent une structure de données appelée « Prefix Hash Tree (PHT) » pour résoudre ce problème. Il s'agit d'un *trie* (ou *arbre-préfixe*) où l'empreinte d'un certain préfixe d'une valeur constitue la localisation dans le réseau THD pour entreposer sa clef d'indexation. Comme interface, le modèle fournit deux opérations canoniques `insert` et `lookup` analogues respectivement aux instructions `get` et `put` d'une table de hachage distribuée. Puisque le PHT en soi ne répond pas à la préoccupation du stockage multidimensionnel, Tang et collab. [92] étendent cette formule avec un accent sur cet aspect en proposant l'utilisation de la courbe de Lebesgue [55], une courbe de Peano [55, 73, 83] (ou « courbe remplissante »). Cependant, dans les deux cas on y retrouve très peu de détails d'implémentation et les algorithmes fournis suggèrent une structure centralisée reposant au-dessus de la table de hachage distribuée.

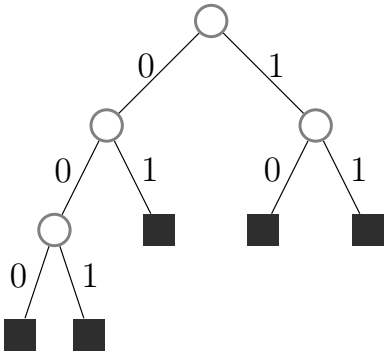
Dans [24], nous soumettons une stratégie augmentant le modèle PHT où l'hypothèse de départ est que la structure d'indexation soit gérée de façon asynchrone sur la table de hachage distribuée par les participants y effectuant des opérations. Une implémentation de ces fonctionnalités est disponible dans le projet OpenDHT et est donc totalement accessible de façon libre.

2.4.1 La structure de trie de hachage

Afin d'aborder notre contribution [24] en détail, nous élaborons premièrement les idées de la structure PHT. Un trie est un arbre dont la position d'un nœud donné définit la clef lui correspondant. Cette position est plus précisément déterminée par un préfixe de la valeur à indexer. Par exemple, si on souhaite indexer la donnée « toto », tout dépendamment le nombre de données déjà indexées dans le trie, la clef « to » pourrait être associée à la donnée. Cette clef représente la position d'une feuille dans un arbre de hauteur 2. Tous les nœuds d'un sous-arbre ont un préfixe en commun avec la racine du sous-arbre. La racine du trie est la chaîne de caractère vide. La structure PHT est un type de trie particulier dont les clefs sont des chaînes de bits $k \in \{0,1\}^D$ pour un certain $D \in \mathbb{N}$. Une telle structure consiste donc en un arbre binaire. On dénote le préfixe de longueur a de la chaîne de caractères k par $\mathsf{Préf}_a(k)$. Ainsi, le chemin menant à une feuille associée à la clef k serait donné par $\ell = \mathsf{Préf}_\alpha(k)$ où $\alpha \leq D$ est la hauteur de la feuille en question. Les clefs d'indexation sont emmagasinées aux feuilles comme on peut le constater dans la figure 2.4. La gestion de la structure de donnée doit se faire en obéissant aux règles suivantes :

1. (*Préfixe universel*) Chaque nœud a soit 0 ou 2 enfants.
2. (*Stockage de clef*) Une clef K est stockée dans une feuille dont l'étiquette est un préfixe de K.
3. (*Scission*) Chaque feuille stocke au plus B clefs.
4. (*Fusion*) Chaque nœud interne contient au moins $B + 1$ clefs dans chacun de ses sous-arbres.

Exemple 2.4.1. *En considérant l'état des clefs indexées dans la figure 2.4, on constate que les clefs* `000000` *et* `000001` *sont placées sous la feuille étiquetée* $\ell_0 = \mathtt{000}$. *La feuille étiquetée* $\ell_1 = \mathtt{001}$ *ne comporte aucune clef indexée.*



Étiquette de la feuille	Clef
000*	000000
	000001
	000011
	000110
	000111
001*	
01*	010000
	010110
10*	100011
	101011
	101010
11*	111011
	111110

FIGURE 2.4: Structure PHT

De par cette figure, il est nécessaire que $B \geq 5$ *puisque la feuille* `000` *renfermant le plus de clefs parmi toutes en contient* 5. *Finalement, les étiquettes* `000`, `001` *et* `01` *ont tous comme préfixe commun* 0. *L'expansion de l'arbre dans ce sous-arbre est expliquée par le nombre de clefs préfixées par* 0 *étant indexées (de par les règles 3 et 4).* ◊

Remarque 2.4.1. Il est conséquemment pertinent de dire que la distribution des données indexées a dans ce cas un impact considérable sur l'équilibre de l'arbre, donc de la performance des opérations d'insertion et de recherche. ◊

La structure de trie de hachage possède deux opérations canoniques `insert` et `lookup` faisant respectivement l'insertion et la découverte de données. Afin de garder une uniformité et de bien discerner les procédures utilisées, on dénote maintenant les algorithmes `get`/`put` respectivement par DHT-LOOKUP et DHT-INSERT.

Lorsqu'on effectue une recherche pour une clef donnée k, donc associée à une certaine étiquette ℓ (dépendant de la distribution des données indexées), l'étape finale d'une recherche est l'interrogation de la table de hachage distribuée par l'appel de DHT-LOOKUP sur l'empreinte de l'étiquette $\mathsf{hash}(\ell)$. L'algorithme don-

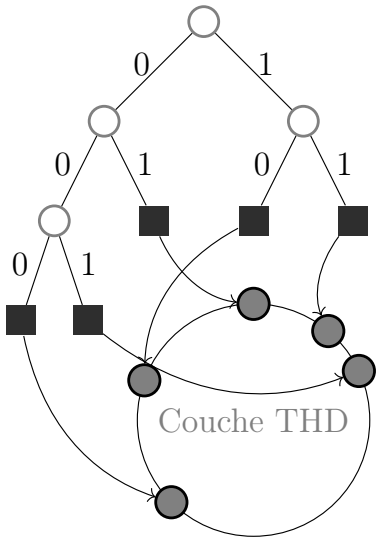


FIGURE 2.5: Recherche/insertion sur la THD

nant l'empreinte de ℓ est déterminé par l'interface de la THD utilisée[6]. Lors d'une insertion, l'algorithme de recherche est premièrement lancé pour finalement exécuter DHT-INSERT($\mathsf{hash}(\ell), v$) où v la structure contenant l'information d'indexation (normalement la clef THD vers la donnée indexée en tant que telle). Ces opérations THD sont illustrées dans la figure 2.5. Notez que la position relative des nœuds sur le cercle n'est pas représentative puisqu'il est probable que leur position réelle soit bien différente considérant qu'il s'agit de clefs fournies par une fonction de hachage cryptographique.

Avec un souci de ne pas surcharger certains nœuds particuliers au début de l'arbre, une fouille binaire est suggérée par Ramabhadran et collab. [78]. Évidemment, la complexité de l'algorithme est donc $\mathcal{O}(\log(n/B))$ où n est le nombre de données introduites dans la structure. Dans la section 2.4.2, nous fournissons des extensions des algorithmes d'insertion et de recherche de Ramabhadran et collab. et nous référons le lecteur à l'article de ces derniers pour connaître les algorithmes

6. Dans le cas d'OpenDHT, il s'agit de SHA1.

originaux.

2.4.2 Indexation distribuée

Comme décrit dans la section précédente, chaque étiquette (ou chemin dans l'arbre PHT) coïncide à une clef dans l'espace des clefs de la THDs, donc cela équivaut conséquemment à un lieu abstrait de stockage dans la THD et cela ensuite associé à huit nœuds du réseau. À un instant donné, on a donc une correspondance entre chaque nœud PHT et huit nœuds chargés d'entreposer des données. Afin de simplifier le langage, on emploie le terme de nœud THD pour désigner les huit nœuds responsables du stockage en question. Nous définissons donc le concept de *canari* comme une valeur particulière stockée par un nœud THD permettant ainsi de délimiter la profondeur de l'arbre PHT. Comme mentionné dans la section 1.3.4, les valeurs stockées dans la table de hachage ont une durée de vie donnée (dépendant de leur type). Cela aura par conséquent un impact pertinent sur la gestion des canaris.

Soit D la taille des clefs d'indexation PHT et $\ell \in \{0,1\}^m$ où $m \leq D$. Si l'appel de DHT-LOOKUP(ℓ) réussit et donne un ensemble de valeurs contenant au moins un canari, alors le nœud est appelé un *nœud PHT*.

Définition 2.4.1 (Feuille)

Si aucun canari n'est retourné par une recherche avec les clefs $\ell \cdot 0$ ou $\ell \cdot 1$, alors on appelle le nœud étiqueté ℓ une *feuille*.

Définition 2.4.2 (Nœud interne)

Un *nœud interne* est un nœud PHT qui n'est pas une feuille, c'est-à-dire que s'il existe deux nœuds PHT étiquetés $\ell \cdot 0$ et $\ell \cdot 1$, alors le nœud étiquette ℓ est un nœud interne.

Algorithme 1 PHT-LOOKUP

```
1: Entrée : Une clef K
2: Sortie : La feuille associée à K
3:
4: bas ← 0
5: haut ← D
6: tant que bas ≤ haut faire
7:     milieu ← (bas + haut)/2
8:     noeud ← DHT-LOOKUP(Préf_milieu(K))
9:     si milieu < D alors
10:        noeud_enfant ← DHT-LOOKUP(Préf_milieu+1(K))
11:    fin si
12:                                  ▷ Le type est déduit par la présence d'un canari
13:    type-de-noeud ← TYPE-NOEUD(noeud.valeurs(), noeud_enfant.valeurs())
14:    si type-de-noeud est « feuille » alors
15:        retourner noeud
16:    sinon
17:        si type-de-noeud est « interne » alors
18:            bas ← milieu + 1
19:        sinon
20:            haut ← milieu − 1
21:        fin si
22:    fin si
23: fin tant que
```

Suivant les définitions 2.4.1 et 2.4.2, les algorithmes PHT-LOOKUP et PHT-INSERT diffèrent forcément de ceux suggérés dans [78]. L'algorithme 1 décrit les étapes pour compléter une opération de fouille dans le PHT. On remarque premièrement que deux recherches THD doivent être effectuées, ceci s'explique par les définitions de feuilles ne possédant aucun enfant.

Remarque 2.4.2. Selon les définitions 2.4.1 et 2.4.2, il est naturel de penser qu'une troisième opération est nécessaire afin de valider la présence d'un canari sur l'autre enfant du nœud étudié. Cependant, l'algorithme d'insertion nous assure que soit aucun des deux nœuds ne stocke un canari, soit les deux en stockent un simultanément. ◊

Algorithme 2 PHT-INSERT

```
1:  Entrée : Une clef K et une valeur v
2:
3:  feuille ← PHT-LOOKUP(K)                          ▷ Peut-être pas la feuille finale
4:  β ← LONGUEUR(feuille.prefix())
5:  si |feuille.valeurs()| < B alors
6:      parent ← DHT-LOOKUP(Préf_{β−1}(K))
7:      soeur ← DHT-LOOKUP(Préf_β(K) ⊕ 0x00...1)
8:      nbrvaleurs ← |feuille.valeurs() + parent.valeurs() + soeur.valeurs()|
9:      si nbrvaleurs < B alors                      ▷ Fusion et insertion
10:         ℓ ← β − 1
11:     sinon                                        ▷ Insertion simple
12:         ℓ ← β
13:     fin si
14: sinon                                            ▷ Scission et insertion
15:     ℓ ← β + 1
16: fin si
17: DHT-INSERT(Préf_ℓ(K), v)
18: UPDATE-CANARY(Préf_ℓ(K))
19: DHT-LISTEN(Préf_{ℓ+1}(K), PHT-INSERT, K, v)
20:                                  ▷ Écouter en cas de nouveaux canaris
```

Ensuite, l'algorithme 2 décrit les étapes pour compléter une opération d'insertion dans l'arbre PHT. La première chose pertinente à noter est que l'opération de recherche faite sur la clef d'indexation (ligne 3) n'est pas finale. En effet, on doit d'abord vérifier le nombre de valeurs se trouvant dans le nœud parent et le nœud sœur afin de déterminer où enfin faire l'opération d'insertion. Le procédé de scission (règle 3) se fait par l'insertion spontanée dans un nœud enfant au niveau inférieur. En revanche, la fusion (règle 4) s'effectue par l'expiration des valeurs au fil du temps. Ce faisant, au moment d'insérer (ligne 17), on veut savoir si une fusion ou bien alors une scission est en cours au niveau des nœuds en question. Pour ce faire, on compte le nombre de valeurs dans chacun des nœuds pour prendre une décision. Une analyse encore plus fine est d'analyser le temps de vie restant des valeurs. L'opération DHT-LISTEN d'OpenDHT est utilisée (ligne 19) afin d'écouter

en cas d'insertion plus bas dans l'arbre. Le cas échéant, l'opération PHT-INSERT sera exécutée à nouveau. Finalement, l'algorithme effectue l'entretien des canaris par l'intermédiaire de l'algorithme UPDATE-CANARY (ligne 18).

Algorithme 3 UPDATE-CANARY

```
1: fonction UPDATE-CANARY(p : mot)
2:     DHT-INSERT(p, canari)                    ▷ Renouvler le canari dans la feuille
3:     DHT-INSERT(p ⊕ 0x00...1, canari)
4:                                   ▷ Renouveler le canari dans le nœud sœur
5:     x ← ALÉATOIRE(0, 1)
6:     si x ≤ P(p) alors          ▷ P(p) est la probabilité de renouveler le parent
7:         UPDATE-CANARY(DÉCALAGEDROITE(p, 1))
8:     fin si                     ▷ Décalage d'un bit donne l'étiquette du parent
9: fin fonction
```

L'algorithme 3 décrit le procédé de renouvellement d'un canari. Si l'indexation est opérée de manière centralisée, alors cette étape est triviale. Dans un modèle distribué comme ce que nous proposons et parce que les valeurs expirent (tout comme les canaris), on doit ajuster en conséquence. Il s'agit d'une stratégie simple et permettant d'assurer la persistance des canaris (tant que l'arbre n'est pas trop déséquilibré).

Soit p, le préfixe pour lequel est exécuté l'algorithme UPDATE-CANARY et $P(p)$ la probabilité que le parent soit renouvelé. On entrevoit deux hypothèses problématiques :

(i) La racine de l'arbre, ou certains nœuds internes ne sont pas actualisés convenablement ;

(ii) La racine et d'autres nœuds internes sont renouvelés trop souvent.

Si on pose $P(p) = 1$, alors le scénario (i) ne se produit pas, mais la racine serait actualisée jusqu'à l'équivalent de la moitié du nombre de nœuds. En effet, le nombre de feuilles d'un arbre est borné par $^{(n+1)}/_{2}$. On imagine bien que ce scénario est problématique. Évidemment, poser $P(p) = 0$ est d'autant pire puisqu'aucun

TABLE 2.2: Borne supérieure pour la probabilité $P(R = 0)$.

a	0	1	2	3	4	5	6	7
$P(R = 0) \leq$	0.7113	0.4225	0.2299	0.1199	0.0613	0.0310	0.0156	0.0078

participant du réseau ne renouvellerait leur parent et l'arbre disparaîtrait. Par conséquent, il est naturel de désirer une valeur de $P(p)$ proche de 1. Dans [24], nous explorons deux stratégies afin d'aborder cette question.

Soit R le nombre de fois que la racine est actualisée. Un premier candidat est $P(p) = 1/2$ ou une valeur un peu plus grande. Dans le cas où l'arbre est parfaitement équilibré, alors la probabilité de ne pas renouveler la racine est :

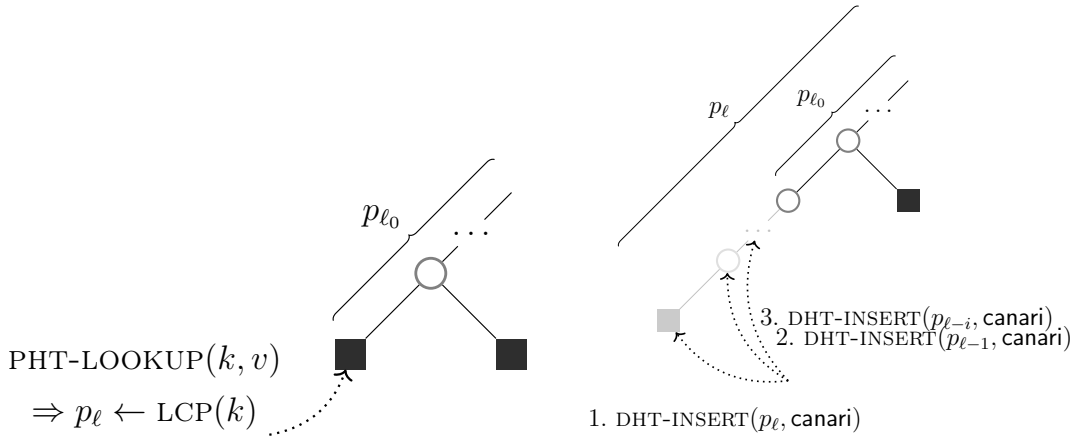
$$P(R = 0) = (1 - P(p)^d)^{2^d} \to 0,$$

où d est la taille de p (en nombre de bits), c'est-à-dire la profondeur dans l'arbre. De plus, la probabilité de renouveler la racine trop souvent est négligeable puisque $R \sim \text{BINOMIALE}(2^d, P(p)^d)$. Ainsi, on a que

$$E[R] = 2^d \times P(p)^d = 2^d \cdot \frac{1}{2^d} = 1.$$

Une seconde stratégie consiste à poser la probabilité en fonction de la taille du préfixe p de telle façon que $P(p) = 1 - 1/2^{|p|+a}$ où $|p|$ dénote la taille du préfixe (en bits) et $a \geq 0$ est un paramètre. Soit K l'ensemble des préfixes engendrant un appel à UPDATE-CANARY, alors :

$$P(\text{ne pas renouveler la racine depuis le préfixe } p) = 1 - \prod_{i=1}^{|p|} P(\mathsf{Préf}_i(p))$$



(a) Le plus long préfixe commun (b) Renouvellement de bas en haut

FIGURE 2.6: Optimisation : scission de bas en haut

et

$$\begin{aligned} P(R=0) &= \prod_{p \in K} P(\text{ne pas renouveler la racine depuis le préfixe } p) \\ &= \prod_{p \in K} \left(1 - \prod_{i=1}^{|p|} P(\mathsf{Préf}_i(p)) \right) \\ &\leq 1 - \prod_{i=1}^{D} \left(1 - \frac{1}{2^{i+a}} \right), \end{aligned}$$

Le pire cas est lorsqu'on ne retrouve qu'une seule clef tout en bas de l'arbre (profondeur D bits). Les valeurs sur cette borne supérieure sont listées dans la table 2.2 en fonction de a et posant $D = 160$ (valeur par défaut pour OpenDHT). Il est à noter que si a est trop grand, la racine sera renouvelée trop souvent, donc ce scénario est aussi à surveiller.

2.4.3 Optimisations

Nous présentons ensuite des optimisations [24] concernant l'algorithme 2 effectuant l'insertion de clef d'index. Notre algorithme réagit à l'arrivée d'autres valeurs

en dessous du lieu d'insertion, c'est-à-dire lorsqu'une scission est en cours, en insérant les données plus bas où se trouvent les nouvelles feuilles. Cependant, comme mentionnée par Ramabhadran et collab. [78], l'implémentation naïve de la règle 3 engendre des comportements indésirables comme la scission en cascade jusqu'au bas de l'arbre. Ce scénario peut se produire lorsque $n \leq B$ clefs indexées sous une même feuille possèdent un long préfixe en commun, le pire cas étant lorsque $n = B$. Afin d'éviter ces problèmes, nous proposons deux optimisations. Premièrement, les doublons ne sont pas comptés dans la somme du nombre des clefs indexées (algorithme 2, ligne 8). Deuxièmement, l'algorithme de scission doit mener au calcul du plus long préfixe en commun entre les clefs présentes dans la feuille d'étiquette ℓ_0 étant sur le point d'être scindée. Le préfixe résultant est donc l'étiquette ℓ du nœud PHT où la scission prendra fin. Par conséquent, comme le suggère la figure 2.6, l'algorithme de scission devrait être modifié afin d'insérer des canaris sur tous les nœuds le long du chemin entre le nœud ℓ et ℓ_0 de bas en haut. Ce faisant, les autres participants en attente (procédure DHT-LISTEN, algorithme 2, ligne 19) seront poussés à exécuter l'algorithme 2 à nouveau où la recherche initiale (3) fournira un premier bon estimé de l'emplacement de la nouvelle feuille évitant ainsi l'insertion répétée en cascade. Il faut noter que l'insertion de canaris pour les nœuds sœurs de chacun des nœuds étiquetés $p_\ell, p_{\ell-1}, \ldots, p_{\ell-i}, \ldots, p_{\ell_0+1}$ est implicite. Cela est nécessaire de par la règle 1.

Finalement, nous proposons l'usage d'une antémémoire pour stocker l'état de l'arbre afin que les opérations subséquentes bénéficient des informations accumulées. Les algorithmes 1 et 2 vont utiliser et maintenir cette antémémoire.

Les efforts décrits dans ce chapitre constituent une première bonne analyse de la question d'indexation dans un modèle distribué. Les optimisations de ce chapitre apportées à OpenDHT sont implémentées depuis l'année 2017 et constituent toujours aujourd'hui la technique employée par la bibliothèque. La prochaine étape

d'approfondissement de ces recherches pourrait s'avérer de considérer les efforts de la communauté du logiciel libre qui tente depuis 2003 de résoudre cette question. L'ensemble des achèvements sont regroupés dans un projet nommé YaCy que Herrmann, Ning, Diaz et Preneel décrivent dans [42].

CHAPITRE III

CLAVARDAGE EN GROUPE SÛR BOUT EN BOUT

Nous abordons maintenant le sujet du clavardage en groupe sûr bout en bout. Nous commençons le chapitre en établissant certaines définitions et la notation utilisée. Nous ajoutons à cela une section sur les différentes propriétés de sécurité conventionnellement désirées pour un système de clavardage respectueux de la vie privée. Ensuite, nous faisons l'état de l'art du clavardage sûr bout en bout en mode pair à pair ainsi qu'en groupe. Finalement, nous terminons la section avec une section détaillant plusieurs propositions pour bâtir un protocole satisfaisant aux propriétés d'authentification niable, la non-transitivité de paternité de message et d'asynchronie.

3.1 Notations et définitions

Nous n'avons pas encore défini le concept de *sécurité d'un système de communication*, comme un système de clavardage, bien qu'on ait couvert la sécurité d'un point de vue cryptographique. Quand on dit « sécurité », on entend la capacité d'un système à préserver diverses propriétés dont nécessairement la confidentialité, l'intégrité et l'authentification (voir les définitions à la section 1.1.6). Lorsque le système est capable de garantir les propriétés désirées, on dit qu'il est *sûr*. La *sécurité bout en bout* est l'exigence que les propriétés de sécurité du système

soient préservées pour tous ses participants **sans** l'existence d'un tiers parti faisant exception et pouvant briser une (ou l'ensemble) des propriétés. Autrement dit, un système sûr bout en bout est un système atteignant toutes les propriétés ci-haut alors que toute entité différente des interlocuteurs d'une conversation sont envisagés comme des ennemis dans le modèle d'adversaire.

Nous définissons maintenant la structure abstraite d'un système de clavardage. En premier lieu, on considère le concept de *canal de communication*. Il s'agit d'un système dont l'interface permet l'écriture et la lecture de messages (voir la table 3.1). On dénote un canal de communication par la lettre $\mathcal{C}$. De façon générale, si Alice transmet sur le canal de communication ouvert avec Bob, alors Bob peut ensuite utiliser ce même canal pour consulter le message qu'Alice a précédemment rédigé. Le canal peut être muni d'une interface cryptographique permettant l'usage de schémas de chiffrement et de signature (voir la section 1.1 pour la notation). Finalement, la dernière composante d'un canal de communication est le transport. Par exemple, celui-ci peut être structuré à la façon des couches Open Systems Interconnection (OSI) avec certaines strates supplémentaires comme le routage requête de stockage de données Kademlia (voir les sections 1.3.4) ou bien une couche de communication du type *client-serveur*. La figure 3.1 illustre le concept.

On considère ensuite un ensemble d'utilisateurs du système de clavardage dénoté par $\mathbb{U}$. Tous les usagers $U_1, U_2, \ldots, U_n$ sont indicés par un nombre. Ils sont les auteurs et les destinataires de signaux (ou messages) transmis entre eux par l'intermédiaire des primitives du système. En particulier, on distingue les procédures fondamentales listées dans la table 3.1.

La communication entre utilisateurs du système est divisée premièrement en plusieurs périodes temporelles appelées *sessions*. Celles-ci se caractérisent par un numéro d'identification unique. Alice peut avoir plusieurs conversations (sessions)

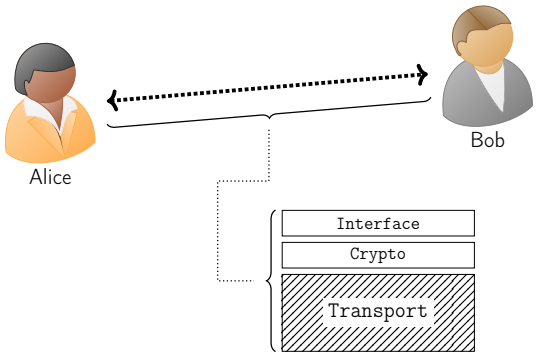


FIGURE 3.1: Canal de communication

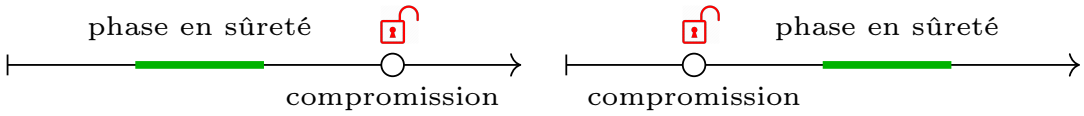
TABLE 3.1: Primitives d'un système de communication

$\textsc{Env}_{\mathcal{C}}(m, U_i)$	Envoi d'un message m à un utilisateur U_i par l'intermédiaire du canal $\mathcal{C}$.
$\textsc{Recv}_{\mathcal{C}}(\cdot)$	Réception d'un message sur un canal $\mathcal{C}$. On dénote la réception d'un message m par $m \leftarrow \textsc{Recv}_{\mathcal{C}}()$.

avec des combinaisons différentes d'interlocuteurs pour des raisons ou buts divers. Ensuite, une session admet plusieurs sous-blocs temporels (ou causaux) nommés *phases*. Durant de ces phases, la liste des participants et les secrets cryptographiques entre eux peuvent changer. Ce faisant, à chaque phase est attribué un *état*, souvent dénoté par π. Par conséquent, nous faisons donc par exemple référence à la clef partagée entre plusieurs membres d'un groupe de communication par $\pi_i.\mathtt{k}$ où π_i est l'état de la phase i. Lorsqu'un adversaire apprend la clef d'une phase i, on dit que π_i est *compromis*.

3.1.1 Propriétés de sécurité

Nous sommes maintenant prêts à fixer plusieurs propriétés de sécurité conventionnellement attribuées aux systèmes de communication. La majorité des définitions



(a) Confidentialité persistante (b) Confidentialité rétroactive

FIGURE 3.2: Confidentialité et compromission

sont traduites directement depuis *SoK: secure messaging* [95] par Unger et collab.

Confidentialité. Assurance qu'un message n'est lisible que par les participants au protocole.

Authentification. Vérification qu'un message a été envoyé par la source réclamant sa paternité.

Confidentialité persistante. Si les secrets de chiffrement [1] pour un message m_i sont dérobés, les clefs respectives associées aux messages m_j ne sont pas compromises pour $j < i$. Voir la figure 3.2a.

Confidentialité rétroactive. Si les secrets de chiffrement [1] pour un message m_i sont subtilisés, les clefs respectives associées aux messages m_j ne sont pas compromises pour $j > i$. Voir la figure 3.2b.

Falsifiabilité. Habileté (d'un parti tiers) de fabriquer de toute pièce une donnée identique à celles possiblement produites par les participants honnêtes du processus.

Validation de destination. : Si le message est accepté par un parti honnête, alors celui-ci est apte à vérifier qu'il faisait bel et bien partie de la liste des destinataires du message.

Déni de paternité. Capacité des participants honnêtes de nier la paternité d'un

1. Inclut la clef du message k ainsi que la clef privée long-terme servant à dériver k.

message.

Déni de participation. Capacité des participants honnêtes de nier la participation à une conversation.

Non-transitivité de paternité. Si un juge est convaincu qu'un message a bien été écrit par Alice, celle-ci n'est tenue responsable d'aucun autre message par transitivité.

Expansion et contraction. Capacité d'ajouter/retirer un usager au groupe sans provoquer le redémarrage du protocole.

Asyncronie. Le protocole est dit *asynchrone* s'il peut être amorcé, s'exécuter et s'arrêter sans la nécessité d'une connexion entre deux pairs au même moment pour toute paire d'utilisateurs. Autrement, le protocole est dit *synchrone*.

3.2 État de l'art

Dans cette section, nous détaillons un ensemble de protocoles connus à ce jour et fournissant une solution au problème de clavardage sûr bout en bout. Ci-après, la chronologie des éléments marquants de la littérature sur la sécurité du clavardage :

Le protocole PGP, une abréviation de *Pretty Good Privacy*, a marqué le début de la littérature sur la messagerie sûre bout en bout et a connu un succès auprès de différentes communautés ayant une connaissance plus pointue des systèmes informatiques que le grand public. Le protocole est conçu pour sécuriser les communications par courriel, mais est couramment exploité à d'autres fins comme l'authentification de contributeurs de paquets pour de multiples distributions GNU/Linux comme Debian, Ubuntu et ArchLinux par exemple. Malheureusement, PGP n'a jamais été déployé à grande échelle pour sa difficulté d'utilisation [104]. Depuis, de multiples efforts ont été fournis afin de bâtir un protocole plus accessible à l'usager moyen, mais surtout avec de meilleures propriétés de sécurité. Borisov,

1986 • PGP (Zimmermann [106])

2004 • OTR (Borisov, Goldberg et Brewer [12])

2009 • mpOTR (Goldberg et collab. [38])

2013 • Axolotl (Marlinspike et Perrin [60])

2013 • GOTR (Liu, Vasserman et Hopper [56])

2015 • SOK (Unger et collab. [95])

2017 • ART (Cohn-Gordon et collab. [17])

Goldberg et Brewer [12] s'inscrivent explicitement dans cette initiative par l'écriture de *Off-the-record communication, or, why not to use PGP*. Depuis, plusieurs ont suivi afin de combler les manques de sécurité [1, 16, 17, 38, 56, 59, 86, 94, 95]. Bien qu'aujourd'hui on puisse considérer la question du clavardage pair-à-pair comme étant plus ou moins réglée, l'extension au clavardage en groupe n'est pas sans défi. Nous listons maintenant les grandes lignes des différents ouvrages visant à répondre à ces questions.

3.2.1 Clavardage pair-à-pair

Nous nous intéressons maintenant à différents protocoles qui implémentent le clavardage pair-à-pair. Deux modèles se démarquent principalement par leur approche : OTR et Signal. L'influence de ces deux ouvrages a été fondamentale pour les travaux qui ont suivi dans les années subséquentes. Nous détaillons ces protocoles dans les prochaines sections.

OTR : Off-the-Record

Borisov, Goldberg et Brewer proposent ce protocole une première fois dans leur article *Off-the-Record Communication, or, Why Not To Use PGP* [12]. Cet article a vivement suscité l'intérêt et est la source d'inspiration principale des travaux qui le suivent. Borisov et collab. ont étudié le cas de PGP [106], un protocole standard basé sur le schéma Rivest Shamir Adleman (RSA) et conçu avant tout pour être utilisé de pair avec les courriels. Les auteurs d'OTR fournissent une argumentation réfutant que les propriétés que PGP apporte à la messagerie, soit la non-répudiation (de participation, de paternité), soient désirées. Ils proposent un tout autre système assurant la confidentialité persistante et le déni de paternité. Les auteurs comparent leur protocole à une conversation se passant dans une pièce dont l'intérieur ne peut être ni espionné ni accédé par un tiers parti. Ils soulèvent alors trois principes à satisfaire en particulier :

— Ce que les participants se disent dans cette pièce *reste entre eux* et aucun d'eux ne peut démontrer à quiconque à l'extérieur que l'un ait dit quoi que ce soit à l'autre.

— Chacun peut reconnaître qui il *a en face* de lui ;

— Chacun peut identifier qui *parle* à tout moment ;

Les auteurs proposent l'utilisation du protocole d'échange de clefs Diffie-Hellman (DH) [103]. Ils innovent en effectuant un cycle perpétuel d'échange de clefs DH permettant ainsi le renouvellement des clefs de chiffrement à chaque message. Cette méthode fournit la confidentialité persistante et rétroactive sur les messages.

Soit $\mathbb{G}$ un groupe cyclique (voir la définition B.2.9) pour lequel le problème DDH décisionnel est difficile et $g \in \mathbb{G}$ un générateur de ce groupe. Alice et Bob choisissent tous deux une clef privée $x_A, x_B \in \mathbb{G}$ respectivement et font l'échange

suivant :

$$A \rightarrow B : g^{x_A}$$
$$B \rightarrow A : g^{x_B}$$

On appelle respectivement g^{x_A} et g^{x_B}, les clefs publiques d'Alice et Bob. Ce dialogue leur permet de calculer un même secret partagé $g^{x_A x_B}$ chacun de leur côté. Ce faisant, un adversaire externe n'a aucun moyen de connaître $g^{x_A x_B}$ quand bien même qu'il aurait en sa possession la paire (g^{x_A}, g^{x_B}). Le processus se répète continuellement avant lors de l'envoi de messages. Ci-après une simulation du protocole avec quelques messages échangés :

$$A \rightarrow B : g^{x_1} \tag{3.1}$$
$$B \rightarrow A : g^{y_1}$$
$$A \rightarrow B : g^{x_2}, \{m_1\}_{k_{11}} \tag{3.2}$$
$$B \rightarrow A : g^{y_2}, \{m_2\}_{k_{21}}$$
$$A \rightarrow B : g^{x_3}, \{m_3\}_{k_{22}}$$

où x_i, y_j sont respectivement les secrets générés par Alice et Bob, M_l sont des messages texte et $k_{ij} = H(g^{x_i y_j})$ avec $H : \{0,1\}^* \rightarrow \{0,1\}^\ell$ une fonction de hachage cryptographique produisant une sortie de longueur ℓ (SHA-256 [37] par exemple). Un premier message est envoyé par chacun des pairs afin de calculer une première clef (ligne 3.1). Lors de l'envoi de son message m_1, Alice ajoute une nouvelle clef g^{x_2} (ligne 3.2). De cette façon, au moment de la lecture du message m_1, Bob peut déjà renouveler le secret commun à l'aide de g^{x_2} et ainsi obtenir $g^{x_2 y_1}$. À l'étape suivante, Bob calcule donc $k_{21} = H(g^{x_2 y_1})$ afin de chiffrer son message et transmet à son tour sa nouvelle clef publique g^{y_2}.

Afin d'authentifier les messages, des CAMs sont utilisés (voir la section 1.1.6) et accompagnent chaque cryptogramme. Les messages sont par conséquent plus

précisément de la forme ci-dessous :

$$g^{x_{i+1}}, \{M_r\}_{k_{ij}}, \mathsf{Cam}_{H(k_{ij})}\big(\{g^{x_{i+1}}, \{M_r\}_{k_{ij}}\}\big).$$

Évidemment, les clefs de CAMs $H(k_{ij})$ sont connues par Alice et Bob en tout temps. Ceux-ci déduisent que s'ils reçoivent un message authentifié par la clef de CAM qu'ils n'ont pas envoyé, alors c'est forcément leur correspondant qui en est à l'origine. Les clefs de CAMs $H(k_{ij})$ sont ensuite publiées sur le canal et « oubliées » par Alice et Bob. Ainsi, l'adversaire a entre ses mains l'outil nécessaire pour falsifier des messages authentifiés de la même manière que àles deux participants. Les auteurs proposent aussi d'utiliser un chiffre malléable tel qu'AES en mode CTR afin de permettre à l'adversaire de modifier les cryptogrammes récupérés sur le canal de façon cohérente. Puisque l'adversaire a également en sa possession les clefs CAMs, celui-ci peut donc contrefaire des messages semblant avoir été créés par Alice ou Bob. Cela fournit donc un déni de la paternité des messages par ses émetteurs.

En somme, l'intégrité de message est garantie par les CAMs, la confidentialité et la validation de la destination des messages sont assurées par un chiffrement AES dont la clef est dérivée d'un échange Diffie-Hellman. Le cycle d'échange de clefs DH nous promet la propriété de confidentialité persistante et rétroactive. L'authentification est assurée par un échange de clefs authentifié (SIGMA [51] depuis OTRv2) en combinaison avec les CAMs. Finalement, la falsifiabilité, la publication des clefs de CAM et le chiffrement malléable préservent l'habileté de déni de paternité des messages.

Le protocole a cependant la faiblesse de céder certains attributs dans des contextes asynchrones. En effet, il suffit qu'Alice envoie plus d'un message de suite sans que Bob réponde pour que les propriétés de confidentialité persistante et rétroactive soient perdues jusqu'à la prochaine réplique de Bob (voir [12, section 4.2]). On

peut facilement imaginer des cas d'utilisation comme les courriels ou la messagerie instantanée où notre correspondant n'est pas en mesure de correspondre.

Signal : algorithme du double cliquet

Il s'agit d'un protocole conçu par Marlinspike et collab. en 2013 et auparavant connu sous le nom de *Axolotl Ratchet* [60]. L'algorithme est celui qui sécurise l'application Signal et est maintenant nommé *Double Ratchet Algorithm* [94] ou double cliquet en français. L'algorithme a été formellement analysé par Cohn-Gordon et collab. [16] en 2016.

Le premier cliquet correspond à la procédure d'échange de clefs cycliques présenté dans la section précédente. Marlinspike et Perrin l'appellent le cliquet Diffie-Hellman. Afin de renforcer la confidentialité persistante du cliquet DH dans un contexte asynchrone, les auteurs créent un second cliquet qu'on dit « symétrique ». Celui-ci permet de dériver deux chaînes de clefs indépendantes : une pour le chiffrement (envoi) de message et l'autre pour le déchiffrement (réception). L'algorithme utilisé pour le second cliquet est inspiré du protocole SCIMP [67, 97] qui consiste en une méthode de rafraîchissement de clefs au moyen d'une fonction de dérivation de clef de clef dénotée par KDF (*Key Derivation Function*). Sans entrer dans les subtilités, ces fonctions rétablissent l'entropie d'une chaîne de caractère passée en entrée en produisant une sortie uniforme.

Nous détaillons maintenant l'algorithme du double cliquet. Les deux pairs commencent par faire un échange clef DH :

$$A \to B : g^{x_0}$$

$$B \to A : g^{y_0}$$

Tous deux étiquettent les secrets DH par $k_{ij} = H(g^{x_i y_j})$ où H est une fonction de hachage cryptographique. Ces clefs forment les racines des chaînes de clefs (cliquet

symétrique). Quand Alice expédie un message à Bob, elle génère une nouvelle paire de clefs DH g^{x_1} et déduit le secret subséquent k_{10}. Elle peut alors dériver de cela la clef racine de la chaîne d'envoi $\lambda_{10} = \mathsf{KDF}(k_{10}, k_{00})$. Pour chiffrer le i$^{\text{e}}$ message, Alice utilise la clef d'envoi suivante :

$$\alpha_{1i} = \mathsf{KDF}^i(\lambda_{10}) = \mathsf{KDF}(\mathsf{KDF}(\ldots(\mathsf{KDF}(\lambda_{10})))), \quad i \geq 1.$$

Ces appels successifs de la fonction de dérivation de clef sont ce qu'on appelle le cliquet symétrique et cela a pour effet de garantir une confidentialité persistante entre chaque message d'Alice vers Bob. Cependant, la confidentialité rétroactive n'est pas satisfaite par ce cliquet. En effet, il s'agit d'une faiblesse de SCIMP [97]. Le prochain échange DH rétablit cette propriété suivant le prochain message de Bob. Avant d'acheminer son message, Bob calcule à son tour la clef racine de sa chaîne d'envoi λ_{11} comme $\mathsf{KDF}(k_{11}, \lambda_{10})$. Il chiffre donc le i$^{\text{e}}$ message avec la clef $\beta_{1i} = \mathsf{KDF}^i(\lambda_{11})$. La figure 3.3 illustre le protocole formellement (figure 3.3a) et visuellement (figure 3.3b). La chaîne d'envoi d'Alice est la chaîne de réception de Bob et vice-versa.

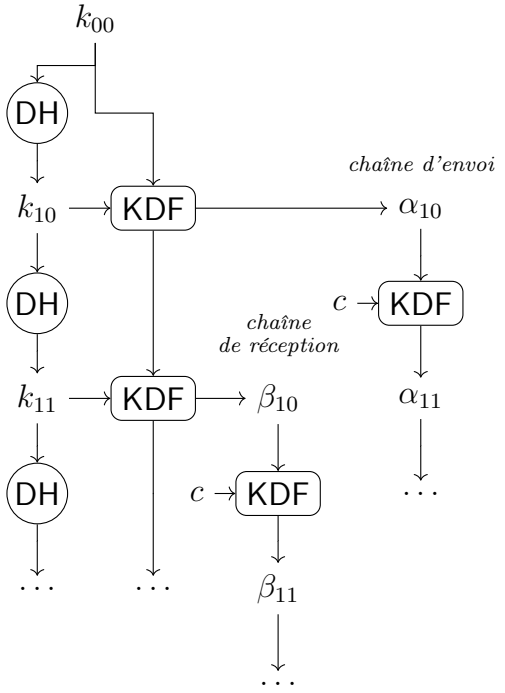
Tant et aussi longtemps qu'Alice n'obtient pas de message de Bob, elle peut continuer à faire avancer son cliquet symétrique en utilisant sa chaîne d'envoi et dérivant donc une nouvelle clef d'envoi à partir de la précédente. Lors d'un échange DH subséquent, elle peut manipuler la nouvelle clef DH afin de répéter le processus et ainsi déduire les nouvelles chaînes d'envoi et de réception.

L'emploi du cliquet symétrique permet de pallier le problème dont souffre OTR en contexte asynchrone. Ici, si Bob ne répond pas, Alice n'a qu'à générer une nouvelle clef sur la chaîne d'envoi pour le nombre de messages qu'elle veut envoyer.

L'authentification des messages en tant que telle est faite par CAMs comme suggéré par OTR. En somme, ce protocole fournit les propriétés suivantes :

$$A \to B : g^{x_1}, \{m_{11}\}_{\alpha_{11}}$$
$$A \to B : g^{x_1}, \{m_{12}\}_{\alpha_{12}}$$
$$A \to B : g^{x_1}, \{m_{13}\}_{\alpha_{13}}$$
$$\dots$$
$$B \to A : g^{y_1}, \{m_{11}\}_{\beta_{11}}$$
$$B \to A : g^{y_1}, \{m_{12}\}_{\beta_{12}}$$
$$B \to A : g^{y_1}, \{m_{13}\}_{\beta_{13}}$$
$$\dots$$
$$A \to B : g^{x_2}, \{m_{21}\}_{\alpha_{21}}$$
$$\dots$$

(a) Communication entre Alice et Bob



(b) Chaînes de dérivation de clefs

FIGURE 3.3: Simulation du protocole du double cliquet — Les M_{ij} (figure 3.3a) sont des messages texte et le symbole c (figure 3.3b) représente une constante arbitraire.

- Confidentialité ;
- Authentification ;
- Validation de destination ;
- Déni de paternité ;
- Non-transitivité de paternité [2] ;
- Confidentialité persistante ;
- Confidentialité rétroactive [2] ;
- Falsifiabilité ;
- Déni de paternité.
- Asynchronie ;

2. Fournit partiellement la propriété.

(a) Signal (b) SilentPhone (c) Messenger (d) iMessage (e) WhatsApp

FIGURE 3.4: Exemples de projets implémentant Signal

En plus d'être utilisé par l'application Signal d'Open Whisper Systems, ce protocole s'est fait récupérer par beaucoup d'autres projets. La figure 3.4 en illustre les logos quelques-uns de ces projets.

3.2.2 Clavardage en groupe

Nous détaillons maintenant les protocoles visant à répondre au besoin de communication en groupe. Ce problème fait l'objet de recherche continue à ce jour et ne constitue pas une tâche triviale. Différentes initiatives ont pavé la route comme mpOTR [38] et une généralisation naïve au groupe de Signal [59]. Nous présentons toutes les solutions, que nous avons étudiées, dans les sections suivantes.

MPOTR : MULTI-PARTY OTR

Dans leur article *Multi-party off-the-record messaging* [38], Goldberg et collab. étendent OTR au cas d'utilisation de groupe. Les auteurs ont cherché à préserver les mêmes propriétés premièrement procurées par OTR : déni de paternité et falsifiabilité. Plus précisément, ils définissent le problème problème fondamental du déni plausible (PFD) stipulant qu'un juge est incapable de distinguer si un message a bel et bien été écrit par un membre du groupe (Alice par exemple), comme le suggère un autre membre du groupe (Bob par exemple). Goldberg et collab. arguent que le déni plausible fourni par les outils cryptographiques ne prouve pas qu'Alice n'a pas écrit un message donné, mais empêchent Bob de démontrer le

contraire. Par conséquent, la situation doit revenir au PFD. Bien qu'OTR ait résolu cette question dans le contexte pair-à-pair, la transposition de cette propriété dans le cas du groupe semble un exercice complexe.

Le protocole s'effectue en trois étapes distinctes, soit la configuration, la communication et la terminaison. La phase de configuration consiste à procurer un identifiant unique de session en fonction de la liste des participants et procéder à un échange de clef de signature niable (ENDCS) (comme MQV [54]). Cette étape ne peut réussir que si tous les usagers parviennent à obtenir une même vision de la liste des membres. Finalement, une négociation de clef en groupe authentifiée (NDCGA) est exécutée afin de munir tous les participants d'une clef symétrique servant au chiffrement. Les clefs de signature précédemment échangées entre tous servent à authentifier les messages de chacun.

La phase de communication succède à celle de configuration. Ce faisant, les membres ont alors une même vue de la composition du groupe et des paramètres cryptographiques. Ils transmettent leurs messages dans le canal de diffusion chiffré et authentifié.

Finalement, la dernière étape consiste à vérifier la transcription de la discussion et à oublier la clef éphémère de chiffrement de la session. Les utilisateurs publient leur clef éphémère de signature pour la session courante de façon à permettre la falsifiabilité de messages. Cette phase doit être appelée lorsqu'un usager quitte ou se joint à la conversation. Une fois la session terminée, si on souhaite continuer de communiquer, on doit repasser par l'étape de configuration pour alors engendrer à nouveau les conditions de communication.

En somme, le protocole ne possède pas les propriétés de contraction et d'expansion à cause de la nécessité de terminer la session lorsqu'on modifie la liste des participants. Ensuite, les propriétés de confidentialité persistante et rétroactive

ne sont pas fortement garanties, car tant que la liste des membres du groupe ne change pas, la clef de session n'est pas renouvelée. Cependant, les propriétés sont préservées d'une session à l'autre. Ainsi, un adversaire apprenant la clef pour un message peut déchiffrer tous les messages de la même session. Les clefs éphémères de signature échangées par un ENDCS permettent l'authentification, le déni de participation et le déni de paternité. Finalement, la non-transitivité de paternité n'est pas atteinte puisqu'une même clef sert pour signer tous les messages lors d'une session.

GOTR : Group OTR

Suivant la publication des protocoles OTR et mpOTR, Liu, Vasserman et Hopper [56] proposent une vision différente ayant pour but de préserver les propriétés de déni plausible et non-transitivité de paternité d'OTR. Ils éliminent le cycle de terminaison et initialisation de la session de mpOTR. Ce faisant, leur protocole vérifie la propriété de contraction et expansion de groupe par l'emploi d'une NDCGA. Cela produit des clefs symétriques qu'ils disent « circulaires » pour chacun des membres et que tous connaissent dès lors que le titulaire fournit une donnée supplémentaire. Ces clefs sont utilisées pour dériver des clefs de CAM. Nous abordons ci-après les étapes du protocole.

Les participants commencent par établir des canaux $\mathcal{C}_{ij}$ pair-à-pair sécurisés et niables afin de transmettre des informations qu'ils peuvent récuser avoir échangées lors du protocole. Ensuite, les participants s'engagent dans un échange de clefs Burmester-Desmedt (BD) [14]. Nous détaillons maintenant les grandes lignes de l'échange BD. Soient $\mathbb{G}$ un groupe cyclique sous les hypothèses DDH, $g \in \mathbb{G}$ un générateur de $\mathbb{G}$, $\mathbb{U}$ l'ensemble des participants. Chaque membre $\mathbb{U}_i \in \mathbb{U}$ produit une clef secrète $r_i \in \mathbb{Z}_p$. Il calcule alors sa clef publique z_i comme :

$$z_i = g^{r_i} \mod p$$

et l'envoie à tous les autres utilisateurs sur les canaux $\mathcal{C}_{ij}$. Ensuite, tous les participants U_i évaluent et publient

$$X_i = (z_{i+1}/z_{i-1})^{r_i} \mod p$$

Finalement, tous les participants U_i obtiennent un secret partagé

$$\kappa = \kappa_i = (z_{i-1})^{nr_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \ldots X_{i-2} \mod p$$

Subséquemment, le protocole entre dans la phase de mise à jour de clef. Il s'agit de la méthode particulière de GOTR basée sur l'emploi des clefs circulaires. Nous exploitons la notation ci-dessous dans ce qui suit :

$$a_{ij} := (a_{ij0}, a_{ij1}),\ a_i = \{a_{ij} \mid \forall U_j \in \mathbb{U} \setminus \{U_i\}\}$$

où $\mathbb{U}$ est l'ensemble des participants. Tous les échanges qui suivent entre U_i et U_j sont faits dans le canal $\mathcal{C}_{ij}$.

L'utilisateur U_i génère deux nœuds virtuels $D_{ijl},\ l \in \{0,1\}$ pour tout j. Postérieurement, U_i évalue r_{ij} et transmet à U_j les *z-valeurs* $z_{ij} = (g^{r_{ij0}}, g^{r_{ij1}})$. Ce dernier stocke les *z-valeurs* de U_i comme $y_{ji} := z_{ij}$. Par la suite, U_i calcule R_{ij} et l'envoie à U_j :

$$R_{ij0} = (z_{ij1}/y_{ij0})^{r_{ij0}} \mod p$$
$$R_{ij1} = (y_{ij1}/z_{ij0})^{r_{ij1}} \mod p$$

De son côté, U_j stocke les *x-valeurs* de U_i comme $V_{ji} := R_{ij}$. En somme, on a :

$$U_i \to U_j : z_{ij}$$
$$U_j \to U_i : z_{ji}$$
$$U_i \to U_j : R_{ij}$$
$$U_j \to U_i : R_{ji}$$

Ainsi, U_i calcule la clef

$$\kappa = \kappa_{ji} = \kappa_{ij} = z_{ij1}^{4r_{ij0}} \cdot \left(\frac{1}{R_{ij0}}\right)^3 \cdot V_{ij0}^2 \cdot V_{ij1} \mod p$$

Cet échange de clefs entre U_i et U_j engendre des canaux pair-à-pair distincts. L'ensemble de nœuds impliqués dans cet échange $\{D_{ijl}, D_{jil} \mid l \in \{0,1\}\}$ est appelé un *flocon*. Ainsi, on nomme la clef déduite par U_i et U_j, la *clef du flocon*. Cette clef peut alors être exploitée pour dériver deux clefs k_{ijl}, $l \in \{0,1\}$ servant respectivement pour le chiffrement et l'authentification. Désignons ce canal par F_{ij}. Dès ce moment, le canal $\mathcal{C}_{ij}$ n'est plus nécessaire.

Finalement, U_i calcule W_i comme suit :

$$W_{ij0} = (y_{ij0}/z_{i(j-1)1})^{r_{ij0}} \mod p$$
$$W_{ij1} = (z_{i(j+1)0}/y_{ij1})^{r_{ij1}} \mod p$$

Ce faisant, U_1 peut donc par exemple inférer la clef circulaire K_1 :

$$z_{131}^{4\cdot5\cdot r_{140}} \cdot W_{140}^{4\cdot5-1} \cdot V_{140}^{4\cdot5-2} \cdot V_{141}^{4\cdot5-3} \cdot W_{141}^{4\cdot5-4} \cdot W_{150}^{4\cdot5-5} \ldots V_{131} \mod p$$

La clef K_i est négociée par tous les usagers, mais U_i est le seul à la connaître de prime abord. Il peut dériver de cette clef deux clefs k_{il}, $l \in \{0,1\}$ servant respectivement pour le chiffrement et l'authentification. On appelle les valeurs $zyWV$, l'ensemble de toutes les *z-valeurs* et *x-valeurs* impliquées dans le calcul d'une clef circulaire. Les valeurs $zyWV$ sont diffusées vers les autres participants de façon à ce qu'ils puissent évaluer la clef K_i de l'utilisateur U_i en faisant une manipulation analogue pour enfin déchiffrer et authentifier les messages expédiés par U_i.

Finalement, la phase de communication décrit les primitives d'envoi et de réception de message. C'est aussi durant cette phase que se fait de manière continuelle la vérification de transcription. L'utilisateur U_i, exploite les clefs de chiffrement

$k_1 := k_{i0}$ et de CAM $k_2 := k_{i1}$ déduites de K_i à l'étape antérieure. Celui-ci échange donc des messages de la forme suivante :

$$\underbrace{\mathsf{sid}, \mathsf{Op}, |\mathbb{U}|, (z_1, X_1), \ldots, (z_{|\mathbb{U}|}, X_{|\mathbb{U}|}), \mathsf{Enc}_{k_1}(m, \ldots 0, e),}_{P} \mathsf{Cam}_{k_2}(P)$$

Le champ Op marque le message comme un message de clavardage ; le champ e est une empreinte des cryptogrammes des messages précédents excluant celui-ci ; les (z_l, X_l) sont les valeurs $zyWV$ nécessaires à l'évaluation de la clef circulaire de l'émetteur du message. Cela permet à un membre du groupe de déduire la clef circulaire de l'émetteur au cas où il l'aurait manquée à l'étape de mise à jour de clef.

Lorsqu'un message de U_i est reçu, l'utilisateur U_j exploite le canal F_{ji} pour valider l'authenticité du titulaire du message. Cela est nécessaire puisque tous les membres du groupe possèdent les valeurs $zyWV$ permettant de falsifier l'authentification d'un message. La vérification par les canaux F_{ji} permet donc de se prémunir contre un adversaire faisant partie du groupe.

En somme, l'authentification, le déni de paternité et la non-transitivité de paternité sont satisfaits par l'utilisation de CAMs ; l'établissement niable de canaux sécurisés permet le déni de participation et la NDCGA permet l'ajout et le retrait de participants sans redémarrer le protocole. Cependant, le modèle n'est pas asynchrone puisqu'un message de mise à jour de clef demande communication avec ses pairs (du flocon). Finalement, les propriétés de confidentialité persistante et rétroactive ne sont pas atteintes lors d'une même session, mais seulement entre les sessions.

Double cliquet (canaux pair-à-pair)

Une première façon d'étendre le double cliquet dans une configuration de groupe est de faire des connexions pair-à-pair $\mathcal{C}_{ij}$ entre tous les membres et d'échanger

tous les messages dans ces canaux [59]. Le protocole X3DH [61], une négociation de clef authentifiée (NDCA), peut être employé afin de démarrer ces connexions, fournissant alors le déni de participation. De plus, cette NDCA est asynchrone. L'authentification est effectuée au moyen de CAMs au travers des canaux $\mathcal{C}_{ij}$, préservant ainsi l'habileté de déni de paternité. Cela a le désavantage d'empêcher la cohérence de transcription parmi les interlocuteurs. L'utilisation du double cliquet [16, 60, 94] permet la non-transitivité de paternité, la confidentialité persistante et rétroactive. Évidemment, dans les cas de partitionnement réseau, les propriétés de non-transitivité et de confidentialité rétroactive peuvent être affaiblies. Il s'agit de conséquences inhérentes au protocole du double cliquet. Comme mentionné par plusieurs [17, 59], cette méthode ne passe pas à l'échelle dû à son nombre de messages en $\mathcal{O}(n)$ envoyés par chaque participant (n est le nombre de participants).

Double cliquet (clefs d'envoi)

Une seconde méthode consiste à employer des *clefs d'envoi* [59, 102]. Il s'agit d'une optimisation qui prend la forme d'un simple schéma d'encapsulation de clefs. Cette méthode est utilisée par les applications Signal et WhatsApp. Les canaux pair-à-pair décrits dans la section précédente sont mis en place de la même manière. Plutôt que d'envoyer leur message dans les canaux $\mathcal{C}_{ij}$, ils créent de nouveaux canaux $\mathcal{C}_\ell$ pour $\ell = 1, 2, \ldots, n$ avec n le nombre de participants. Pour envoyer un message, le membre U_ℓ chiffre ce premier avec une clef k (la clef d'envoi) et écrit le cryptogramme correspondant sur le canal $\mathcal{C}_\ell$. Au même moment, il transmet k dans les canaux $\mathcal{C}_{ij}$. Formellement, l'échange ci-dessous survient :

$$\begin{aligned} \forall U_i \in \mathbb{U} \setminus \{U_\ell\}\,,\ U_\ell \to U_i : \{k\}_{k_{i\ell}} \\ U_\ell \to \mathcal{C}_\ell : m \end{aligned} \tag{3.3}$$

Notons que l'écriture sur $\mathcal{C}_\ell$ implique implicitement que le message est plutôt

$\{m\}_k$ (vu hors du canal). Ensuite, l'étape 3.3 peut s'exécuter qu'une seule fois par session. C'est en fait ce que Cohn-Gordon et collab. [17] reprochent à cette approche puisque cela empêche d'obtenir une confidentialité persistante et rétroactive systématique. Cependant, cette stratégie permet de réduire la complexité de $\mathcal{O}(n)$ à $\mathcal{O}(1)$ en matière de messages texte envoyés. Comme la taille d'une clef est constante et considérée petite (voir négligeable), cela représente un gain pertinent. L'authentification est faite par clefs de signature éphémères échangées en début de communication après la NDCA (avec X3DH). Les clefs ne sont pas rafraîchies systématiquement, alors la non-transitivité de paternité n'est pas atteinte. Cependant, les propriétés de déni de paternité et de participation sont bien vérifiées. Finalement, l'asynchronie, la contraction et l'expansion sont assurées par X3DH.

SYM-GOTR

Le protocole de Schliep, Vasserman et Hopper [86] est paru relativement peu récemment et n'a donc pas fait l'objet d'une étude assez longue pour en fournir les détails complets. Les grandes lignes ont cependant été considérées et traduites dans cette courte section.

Les auteurs ont enrichi le modèle GOTR en remplaçant la NDCGA de Burmester-Desmedt un échange niable et sécurisé de secrets leur permettant de dériver une clef partagée. Par ce même premier canal, les participants échangent des clefs éphémères de signature afin d'authentifier leurs messages tout au long de la conversation. Lors de l'envoi d'un message, les interlocuteurs renouvellent leur clef éphémère et leur secret préservant la non-transitivité de paternité ainsi que la confidentialité persistante et rétroactive. L'ensemble des propriétés fournies sont listées dans la table 3.2. Il est pertinent de noter que l'atteinte de la non-transitivité de paternité ne se fait qu'au coup d'échange de messages.

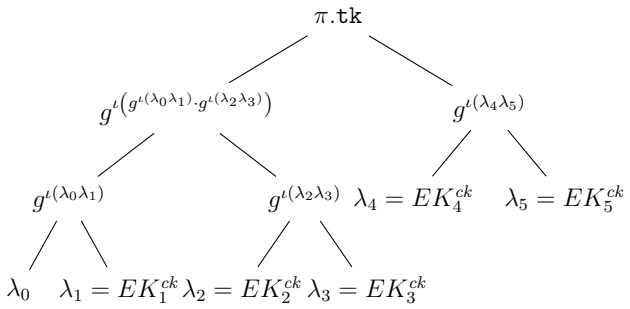


FIGURE 3.5: Exemple de calcul de clef par l'initiateur du groupe

ART : ASYNCHRONOUS RATCHETING TREE

Ce protocole, écrit par Cohn-Gordon et collab. [17], utilise une négociation de clef en groupe (NDCG) inspirée de KPT [48, 49], une approche basée sur structure d'arbre binaire portant le nom de leurs créateurs Kim et collab. La négociation de clef est faite de façon asynchrone suivie d'un cliquet symétrique similairement à SCIMP, c'est-à-dire qu'il s'agit d'une chaîne d'appels de fonction de dérivation de clef. La stratégie globale est relativement comparable à celle du double cliquet, mais transposée dans un contexte de groupe.

Soit $\mathbb{U}$ l'ensemble des participants au protocole. Pour tout usager $U_i \in \mathbb{U}$, U_i calcule d'avance un grand nombre de paires de clefs DH dénotées par $(ek_{i\ell}, EK_{i\ell})$. On appelle ces clefs des « pré-clefs ». De plus, chaque membre U_i possède une paire de clefs d'identité longue durée notée (ik_i, IK_i). Le concept de « pré-clef » a été introduit par Marlinspike [58] et permet d'atteindre la confidentialité persistante en mode asynchrone. En particulier, tout utilisateur peut donc récupérer une paire de clefs éphémères de U_i (sur un point d'échange persistant) et faire un échange DH avec une clef éphémère à eux sans communiquer avec U_i.

Afin de configurer le groupe, l'organisateur récupère une clef éphémère publique EK_i de U_i, $\forall U_i \in \mathbb{U}$ en interrogeant une structure de donnée intermédiaire à

laquelle l'initiateur n'a pas d'obligation de faire confiance (cela pourrait être une table de hachage distribuée par exemple). Ensuite, l'initiateur génère une paire de clefs particulière nommée « clef de configuration » et dénotée par (ck, CK). Pour tout $U_i \in \mathbb{U}$, le créateur calcule la clef placée au niveau de la feuille référant à l'utilisateur U_i dans l'arbre comme $\lambda_i = EK_i^{CK}$. La figure 3.5 montre le procédé. Ici, $\iota : \mathbb{G} \to \mathbb{Z}$ est une injection sur les entiers depuis les éléments du groupe cyclique $\mathbb{G}$. Puisque l'initiateur a calculé lui-même toutes les valeurs λ_i, il est en mesure d'évaluer toutes les valeurs intermédiaires jusqu'à la racine de l'arbre, la clef du groupe $\pi.\texttt{tk}$. Comme il s'agit de la clef de groupe initiale (phase 0), on note celle-ci $\pi_0.\texttt{tk}$. Finalement, afin de calculer la clef de chiffrement comme tel, on dérive une première fois $\pi_1.\texttt{sk} = \mathsf{KDF}(\pi_0.\texttt{sk}, \pi_0.\texttt{tk})$ où on a arbitrairement $\pi_0.\texttt{sk} = 0$.

Définition 3.2.1 (Co-chemin)

Soit A un arbre binaire avec r sa racine et C_f l'ensemble des nœuds le long du chemin de r jusqu'à une feuille f. Le co-chemin de f est l'ensemble

$$\{s \in A \mid \text{s est un nœud frère de } c \in C_f\}.$$

Sans jamais communiquer avec aucun des autres membres, le créateur du groupe a réussi à dériver une clef de chiffrement pour le groupe en atteignant la confidentialité persistante de la clef $\pi_1.\texttt{sk}$. Afin de conclure cette étape et ainsi permettre aux autres utilisateurs de se joindre au groupe, l'initiateur envoie par un canal sécurisé l'information sur l'arbre à laquelle il joint une signature produite à l'aide de sa clef d'identité privée. Par exemple, U_1 recevrait :

$$1, IK_0, IK_2, \ldots, IK_5, CK, g^{g^{\iota(\lambda_2\lambda_3)}}, g^{g^{\iota(\lambda_4\lambda_5)}}, g^{\lambda_0} \tag{3.4}$$

Le message ci-haut consiste donc respectivement en l'identifiant de l'utilisateur U_1, les clefs d'identités des autres participants, la clef publique de démarrage et

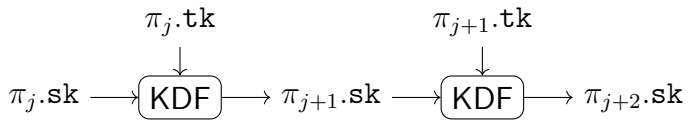


FIGURE 3.6: Dérivation des clefs

l'ensemble des parties publiques des clefs générées le long du co-chemin de λ_1 (voir définition 3.2.1).

Ensuite, lorsque U_i reçoit le premier message venant de l'initiateur, il peut donc calculer π_1.sk à son tour. Ci-après une simulation de conversation entre U_1, U_2 et U_3 (toujours dans le contexte de la figure 3.5) :

$$U_1 \to \mathbb{U} : \left\{ g^{\lambda'_1}, g^{g^{\iota(\lambda_0\lambda'_1)}}, g^{\iota\left(g^{\iota(\lambda_0\lambda'_1)} g^{\iota(\lambda_2\lambda_3)}\right)}, m \right\}_{\pi_1.\mathtt{sk}}$$

$$U_2 \to \mathbb{U} : \left\{ g^{\lambda'_2}, g^{g^{\iota(\lambda'_2\lambda_3)}}, g^{\iota\left(g^{\iota(\lambda_0\lambda'_1)} g^{\iota(\lambda'_2\lambda_3)}\right)}, m \right\}_{\pi_2.\mathtt{sk}}$$

$$U_3 \to \mathbb{U} : \left\{ g^{\lambda'_3}, g^{g^{\iota(\lambda'_2\lambda'_3)}}, g^{\iota\left(g^{\iota(\lambda_0\lambda'_1)} g^{\iota(\lambda'_2\lambda'_3)}\right)}, m \right\}_{\pi_3.\mathtt{sk}}$$

Comme le montre la figure 3.6, π.tk est renégocié afin de produire une clef π.sk. Cela est engendré par le renouvellement de la clef des clefs λ à chaque message. En effet, ci-haut, chacun des utilisateurs envoie la liste des clefs publiques générées sur le nouveau chemin engendré par la clef λ chacun.

En somme, cette stratégie s'opère en mode asynchrone et assure la confidentialité persistante en plus de la confidentialité rétroactive. La contraction et l'expansion du groupe peut se faire aisément, même si Cohn-Gordon et collab. [17] n'en font pas mention. Cependant, rien n'est garanti sur les autres propriétés puisque l'authentification a été catégorisée comme hors propos dès le départ.

COMPARATIF DES PROTOCOLES

Plusieurs tentatives ont paru dans les dernières années afin de répondre au pro-

TABLE 3.2: Comparaison des protocoles de clavardage en groupe [95]

Schéma	Exemple	Propriétés								
		Authentication	Confid. persistente	Confid. rétroactive	Non-transit. de pat.	Déni de paternité	Déni de participation	Asynchronie	Contration	Expansion
NDCGA niable+Valid. de transcription à la terminaison†	mpOTR	●	◐	◐	-	●	●	-	-	-
Clefs circulaires+Valid. de transcription par messages†	GOTR	●	◐	◐	●	●	●	-	●	●
NDCGA+Rotation secrets & clefs éphémères†	SYM-GOTR	●	●	●	●	●	●	-	●	●
Double cliquet (cannaux pairs à pairs)+X3DH+clefs préliminaires*	iMessage	●	●	●	◐	●	●	●	●	●
Double cliquet (clefs d'envoi)+X3DH+clefs préliminaires*	Signal, WhatsApp	●	●	◐	-	●	●	●	●	●
NDCGA par KPT+Déviration de clefs†	ART	-	●	●	-	-	-	●	●	●

● = fournit la propriété ; ◐ = fournit partiellement la propriété ; - = ne fournit pas la propriété ;
†soutenu par une publication académique ; *implémenté dans une application utilisateur.

blème de clavardage en groupe sûr bout en bout. Dans la présente section de ce document, nous avons fait un survol de quelques-uns d'entre les candidats préférables, et ce au meilleur de nos connaissances actuelles. La table 3.2 résume les propriétés pleinement atteintes (●), partiellement atteintes (◐) et manquantes (-) par chacun des différents schémas détaillés précédemment.

3.3 Clavardage sûr bout en bout avec authentification niable, non-transitive et asynchone

Nous présentons maintenant un enrichissement du protocole ART [17] (tel que décrit dans la section 3.2). En lien avec le chapitre 2, soulignons qu'il était initialement prévu de joindre la spécification sous-jacente du transport des messages dans un réseau de THD. Ceci ajoutait donc la dimension d'une difficulté supplémentaire et bien connue, soit le problème des généraux byzantins [53]. Ce problème est résolu par plusieurs modèles tels que les chaînes de bloc [2, 69, 105]. Cependant, cette approche engendre plusieurs désavantages qui ne sont pas encore clairement possible d'éviter comme les frais de transaction. Ce faisant, il s'en

suit majoritairement l'impossibilité de conduire des microtransactions, car le coût excède le gain. Un système, pour lequel le processus de démarrage jusqu'à l'aboutissement d'une conversation munie de toutes ses fonctionnalités est nécessaire, n'est pas totalement compatible avec les limitations qu'imposent les chaînes de bloc. D'autres solutions comme les CRDTs [74, 88], plutôt basées sur l'approche de Leslie Lamport [52], sont aussi envisageables. Au final, l'étude de cette problématique constitue en soi un effort de grande envergure et n'a conséquemment pas pu s'ajouter à notre démarche de développement d'un protocole de clavardage sûr bout en bout.

Comme la table 3.2 le démontre, ART ne procure pas l'authentification, le déni de paternité, le déni de participation et la non-transitivité de paternité. Nous fournissons donc dans cette section-ci un protocole complet comblant ces manques.

Selon le contenu de la section 3.2, on connaît deux méthodes permettant d'atteindre le déni de paternité, soient l'usage de CAMs dans le monde pair-à-pair oconnexionsu bien l'emploi de clefs éphémères dans la configuration de groupe. Il n'est pas possible d'utiliser les CAMs en groupe puisque les autres participants du groupe peuvent falsifier des messages à notre sous notre « identité ». Afin d'éviter ce phénomène, GOTR ajoutait une vérification d'intégrité de la transcription de la conversation en plus des CAMs, ce qui complexifie et alourdit le protocole en comparaison à l'usage d'un schéma de clefs publiques éphémères. De plus, cela empêcherait d'atteindre l'asynchronie du protocole.

De façon à ce que la non-transitivité de paternité soit satisfaite en configuration asynchrone, il est nécessaire que chaque clef éphémère ne soit employée qu'une seule fois pour signer un message. Une stratégie intuitive pour y arriver est de transmettre de nouvelles clefs lors de l'envoi de chaque message. C'est ce pour quoi Schliep, Vasserman et Hopper [86] optent dans SYM-GOTR. Cependant, le

désavantage est que cela engendre un trafic de $\mathcal{O}(n)$ messages (1 par nœuds). On peut imaginer un procédé allant comme suit : (i) Tous les participants génèrent une paire de clefs éphémères. (ii) Tous échangent leur clef publique entre eux par un canal sécurisé et niable. (iii) Dans le but d'authentifier les messages subséquents, tous utiliseraient une fonction rendant possible de dériver une prochaine clef privée afin de fournir une signature à l'aide d'une clef différente à chaque fois. Cette fonction devrait nécessairement être couplée à une seconde fonction qui permettrait aux pairs de calculer la clef publique correspondante de chacun de leurs interlocuteurs. Au meilleur de nos connaissances, aucun schéma de la sorte n'a été proposé dans la littérature sur le clavardage en groupe sûr bout en bout. Nous suggérons donc quatre pistes de solutions afin d'y arriver.

3.3.1 Définitions

Nous prenons maintenant le temps d'établir différentes définitions fondamentales et nécessaires à la construction de nos idées. Puisque jusqu'ici les outils cryptographiques n'ont été utilisés qu'en mode boîte noire, nous invoquons à cet instant le concept de schéma de signature (voir la section 1.1.6) qu'on dénote par $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$. En exécutant, $\mathsf{Gen}(1^n)$ on obtient une paire de clefs $(\mathsf{sk}, \mathsf{pk})$ valide. Nous cernons la notion de validité d'une paire de clefs dans la définition 3.3.1.

Définition 3.3.1 (Paire de clefs valides)

Soient $\mathsf{sk}, \mathsf{pk} \in \{0,1\}^*$, $m \in \{0,1\}^*$ et $\sigma = \mathsf{Sign}_{\mathsf{sk}}(m)$. Une paire de clefs $(\mathsf{sk}, \mathsf{pk})$ est dite *valide* si

$$\mathsf{Vrfy}_{\mathsf{pk}}(m, \sigma) = 1.$$

Soit $s \in \{0,1\}^*$ un paramètre. Étant donné $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^n)$, on cherche une

fonction décrite par deux fonctions $f_s, F_s : \{0,1\}^* \to \{0,1\}^*$ de la façon suivante :

$$\mathsf{Drv}_s(\mathsf{sk}, \mathsf{pk}) \stackrel{\text{déf}}{=} (f_s(\mathsf{sk}), F_s(\mathsf{pk})) \tag{3.5}$$

On présente l'expérience cryptographique (voir la section 1.1.3) suivante mettant en jeu la sécurité de Drv_s défini par l'équation (3.5).

Expérience Liaison$_{\mathcal{A},\Pi}(n)$

Soit $\mathcal{A}$ un algorithme TPP.

1. Suivant l'entrée 1^n, on génère $(\mathsf{sk}, \mathsf{pk})$ et $(\mathsf{sk}', \mathsf{pk}')$ par l'exécution de $\mathsf{Gen}(1^n)$.
2. On choisit $s \in_\$ \{0,1\}^*$ et on calcule

   $$(\mathsf{sk}'', \mathsf{pk}'') = \mathsf{Drv}_s(\mathsf{sk}, \mathsf{pk}).$$

   On *choisit uniformément* $b \in_\$ \{0,1\}$ et on assigne $\mathsf{pk}_b = pk''$ et $pk_{b\oplus 1} = pk'$.
3. $\mathcal{A}$ se fait donner $1^n, \mathsf{pk}, \mathsf{pk}_0$ et retourne :

   $$b' = \begin{cases} 1 & \text{si } \mathsf{pk}_0 \text{ apparaît uniforme} \\ 0 & \text{sinon} \end{cases}$$

Le résultat de l'expérience est défini comme 1 si $b' = b$ et 0 sinon.

Étant donné le tuple $(\mathsf{sk}, \mathsf{pk}, \mathsf{pk}_0)$, cette expérience met donc au défi un adversaire afin qu'il distingue si on lui a fourni un pk_0 comme un résultat de Drv_s ou bien un résultat de $\mathsf{Gen}(1^n)$.

Définition 3.3.2 (Préservation de la validité)

Soit $(\mathsf{sk}, \mathsf{pk}) \leftarrow_{\$} \mathsf{Gen}\left(1^{\ell}\right)$. Pour tout $s \in \{0,1\}^*$, $\mathsf{Drv}_s(\mathsf{sk}, \mathsf{pk})$ est une paire de clefs valide.

Définition 3.3.3 (Non-transitivité de paternité)

La fonction Drv_s produit une sortie *non transitive* si pour tout algorithme TPP $\mathcal{A}$, il existe une fonction négligeable negl telle que

$$\mathbb{P}\left[\mathsf{Liaison}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(n).$$

Suivant les deux dernières définitions, nous pouvons donc définir maintenant la nature complète de Drv_s. On cherche une fonction Drv_s telle que les définitions 3.3.2 et 3.3.3 sont respectées.

3.3.2 Approche sous l'hypothèse DDH

La sécurité de schémas cryptographiques repose sur des problèmes de calcul difficiles. Deux problèmes particuliers sont pertinents dans notre contexte : le problème du CLD ainsi que du problème DDH (voir la section 1.1.4). Par exemple, le schéma de signature ElGamal est sûr sous l'hypothèse que ces deux problèmes sont difficiles. À noter que le problème DDH est une proposition suffisante car plus restrictive que celle faite sur le CLD. Soit un groupe cyclique $\mathbb{G}$ d'ordre p pour lequel l'hypothèse DDH n'est pas démontrée fausse et $g \in \mathbb{G}$ un générateur. Notre stratégie repose plus particulièrement sur un schéma de signature à clef publique dont une paire de clefs $(\mathsf{sk}, \mathsf{pk})$ est telle que $\mathsf{sk} \in \mathbb{Z}_p$ et $\mathsf{pk} = g^{\mathsf{sk}}$.

Multiplication indistinguable

Étant donné $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^n)$, une première proposition intuitive est de faire usage de la somme dans $\mathbb{Z}_p$ et de la multiplication dans $\mathbb{G}$. Ceci se traduit par

l'équation (3.6).

$$\begin{cases} f_s(\mathsf{sk}) = \mathsf{sk} + s \mod p \\ F_s(\mathsf{pk}) = \mathsf{pk} \cdot g^s \end{cases} \tag{3.6}$$

Conjecture 3.3.1

La construction de Drv_s telle que présentée dans l'équation (3.6) produit une sortie *non transitive*.

Remarque 3.3.1. Même si la conjecture était vérifiée, cette technique ne doit pas être employée sans considération d'un défaut important. En effet, puisque si la même clef s était utilisée pour dériver plus d'une paire de clefs, alors la propriété ne tiendrait plus. Évidemment, il serait ainsi trivial de réussir l'expérience $\mathsf{Liaison}_{\mathcal{A},\Pi}(n)$. Par exemple, soit $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^n)$ et $s \in \{0,1\}^*$, il s'ensuit que :

$$\frac{g^{\mathsf{sk}+s}}{g^{\mathsf{sk}}} = \frac{g^{\mathsf{sk}+2s}}{g^{\mathsf{sk}+s}} = g^s.$$

On distinguerait donc directement cet événement du cas uniforme. ◊

Exponentiation indistinguable

Étant donné la nature du problème, plusieurs options paraissent pertinentes. C'est le cas de l'approche par exponentiation. Celle-ci est traduite dans l'équation (3.7).

$$\begin{cases} f_s(\mathsf{sk}) = \mathsf{sk} \cdot s \mod p \\ F_s(\mathsf{pk}) = \mathsf{pk}^s \end{cases} \tag{3.7}$$

Conjecture 3.3.2

La construction de Drv_s telle que présentée dans l'équation (3.6) produit une sortie *non transitive*.

Exponentiation double indistinguable

Finalement, une troisième stratégie est directement inspirée de l'algorithme de générateur pseudo-aléatoire de Blum et Micali [10] est celle traduite par l'équation (3.8).

$$\begin{cases} f_s(\mathsf{sk}) = \iota\left(g^{\mathsf{sk}} \cdot g^s\right) \mod p \\ F_s(\mathsf{pk}) = g^{\iota(\mathsf{pk} \cdot g^s)} \end{cases} \tag{3.8}$$

Conjecture 3.3.3

La construction de Drv_s telle que présentée dans l'équation (3.8) produit une sortie *non transitive*.

Au début de la section, nous avons statué l'exigence des hypothèses DDH. Nous pensons que la sécurité fournie par les conjectures 3.3.1, 3.3.2 et 3.3.3 est nécessaire à la difficulté du problème décisionnel de DH. Afin de le démontrer, il serait donc nécessaire de montrer la réduction de $\mathsf{Liaison}_{\mathcal{A},\Pi}(n)$ à DDH. Il n'est cependant pas exclu que le problème DDH se réduise à $\mathsf{Liaison}_{\mathcal{A},\Pi}(n)$, retirant donc la nécessité de l'hypothèse DDH (puisque DDH est considéré comme difficile). Nous sommes toujours actuellement en réflexion sur ce problème et espérons résoudre ce problème dans un futur proche. Malgré l'absence de preuve, nous fournissons, à la section suivante, un protocole s'appuyant sur l'hypothèse que le problème est difficile d'une manière ou d'une autre.

3.3.3 Protocole

Maintenant, nous présentons le protocole de clavardage répondant aux desiderata spécifiés au début de la section 3.3. Les fondements d'ART (voir la section 3.2) ainsi que les notions énoncées dans la section 3.3.1 sont au centre de notre propo-

sition. Puisque ART est en quelque sorte un enrichissement de Signal, il nous a paru naturel de s'en inspirer et y avons emprunté quelques idées. Premièrement, nous décrivons comment les utilisateurs s'authentifient entre eux afin d'échanger des secrets de démarrage. Subséquemment, nous détaillons l'initialisation des clefs éphémères de signature. Il s'ensuit une simulation de conversation afin de bien saisir les concepts énoncés. Finalement, nous terminons la section par une discussion sur la longévité des clefs de signature éphémères.

Établissement de canaux sécurisés

Plusieurs modèles de NDCA sont discutés dans la section 3.2. Nous favorisons X3DH [61] pour sa propriété d'exécution asynchrone et de déni de participation. Cette approche étant celle choisie par Signal, WhatsApp et plusieurs autres, cela donne une base solide dans un contexte de développement applicatif. Une fois l'échange de clefs initial effectué, un canal de communication sécurisé au moyen d'un double cliquet (voir section 3.2) est entretenu. Cela permet un accès à une chaîne de secrets bénéficiant de confidentialité persistante et rétroactive. La nature de ce canal est une communication pair-à-pair longue durée. Par exemple, cela peut être vu comme une faisant part d'une « relation d'amitié » comme on les connaît des conventions terminologiques bien connues des réseaux sociaux. Par conséquent, ce canal n'est pas lié à une seule conversation, mais est plutôt indépendant. En effet, dans le contexte d'une application d'usager final complète où en plus des communications de groupe, les utilisateurs opèrent des communications pair-à-pair, alors le canal en question servirait de canal de communication pair-à-pair. Par conséquent, il est raisonnable de supposer que ce type de canal soit partie intégrante du système. Ce faisant, aucune charge supplémentaire n'est ajoutée jusqu'ici.

Création d'un groupe

Comme décrit à la section 3.2, l'initiateur du groupe doit récupérer une pré-clef pour tous les utilisateurs $U_i \in \mathbb{U}$ et calculer l'arbre de démarrage (figure 3.5, section 3.2). Ensuite, il doit transmettre à tous les participants les données nécessaires à la construction de l'arbre (voir l'équation (3.4)). Notre proposition est de recourir aux canaux pair-à-pair tels que négociés dans la section précédente comme suit :

$$\left\{ \begin{array}{l} i, \\ \mathcal{I} = (IK_0, IK_2, \ldots, IK_{i-1}, IK_{i+1}, \ldots, IK_{n-1}), \\ CK, \\ \mathsf{co\text{-}chemin}(i), \end{array} \right\}_{s_{U'U_i}}$$

Le contenu du message est tel que décrit dans les détails du protocole ART à l'exception que le message est chiffré par $s_{U'U_i}$, c'est-à-dire le secret partagé entre l'organisateur U' et l'usager U_i provenant de la couche du double cliquet au moment de l'envoi du message. Il est pertinent de mentionner que le message ci-haut ne constitue pas une preuve de participation (ou d'invitation) à la conversation puisque n'importe qui pourrait avoir généré le message clair et $s_{U'U_i}$ est un secret niable.

Remarque 3.3.2. Cette stratégie ne demande pas que tous les utilisateurs aient complété la phase de connexion sécurisée entre eux (l'état « d'amitié »). Seul l'initiateur est requis d'être dans cette situation puisqu'il doit envoyer à tous les informations nécessaires pour construire l'arbre. Même si certains membres n'ont pas établi de connexion sécurisée entre eux, la présente étape de configuration peut continuer sans empêcher les autres de commencer à clavarder. ◊

Subséquemment, tous les usagers calculent la structure KPT et dérivent la première clef $\pi_0.\mathtt{sk}$. S'ensuit la génération de paires de clefs éphémères $(\mathsf{sk}_i, \mathsf{pk}_i)$ par

tous les interlocuteurs U_i respectant les critères déterminés dans la section 3.3.1. Maintenant, tous les participants transmettent leur clef publique pk_i en créant un message de la forme ci-dessous :

$$\left\{ \{\mathsf{pk}_i\}_k , \{k\}_{s_{U_iU_j}} , \{k\}_{s_{U_iU_l}} , \dots \right\}_{\pi_{\ell_i}.\mathtt{sk}} \tag{3.9}$$

où k est un secret uniformément choisi et π_{ℓ_i} est l'état de la phase au moment où l'utilisateur U_i envoie son paquet. Il s'agit d'une approche par encapsulation de clefs exploitée afin de minimiser les coûts de bande passante de chaque membre puisque la taille d'une clef publique est normalement plus grande que celle recommandée pour une clef symétrique (selon NIST).

Conversation

Supposons qu'Alice, Bob et Carole ont complété les étapes précédentes et sont maintenant prêts à clavarder au moment où l'état de la phase est π_ℓ. Notons leur paire de clefs éphémères respectives comme suit :

$$(\mathsf{sk}_{A0}, \mathsf{pk}_{A0}), (\mathsf{sk}_{B0}, \mathsf{pk}_{B0}) \text{ and } (\mathsf{sk}_{C0}, \mathsf{pk}_{C0}). \tag{3.10}$$

Alice envoie un premier message m_0 avec une signature $\sigma_0 = \mathsf{Sign}_{sk_{A1}}(m_0)$ après avoir premièrement calculé $\mathsf{sk}_{A1} = f_{\pi_\ell.\mathtt{sk}}(\mathsf{sk}_{A0})$. Le message est chiffré avec $\pi_\ell.\mathtt{sk}$. On a ainsi :

$$A \to B, C : \{m_0, \sigma_0\}_{\pi_\ell.\mathtt{sk}}$$

Bob et Carole connaissent tous deux $\pi_\ell.\mathtt{sk}$, alors ils peuvent dès lors évaluer $\mathsf{pk}_{A1} = F_{\pi_\ell.\mathtt{sk}}(\mathsf{pk}_{A0})$. Ce faisant, ils sont par conséquent capables de confirmer l'authenticité du message m_0 en vérifiant si $\mathsf{Vrfy}_{\mathsf{pk}_{A1}}(\sigma_0, m_0) \stackrel{?}{=} 1$.

Postérieurement, si Bob souhaite répondre, celui-ci déduit la clef de phase subséquente $\pi_{\ell+1}.\mathtt{sk} = \mathsf{KDF}(\pi_\ell.\mathtt{tk}, \pi_\ell.\mathtt{sk})$. Cela lui permet ensuite de dériver une

nouvelle clef éphémère privée $\mathsf{sk}_{B1} = f_{\pi_{\ell+1}.\mathtt{sk}}(\mathsf{sk}_{B0})$. Il procède donc comme Alice et envoie son message authentifié aux autres.

Durée de vie des clefs éphémères

Même si F_s doit produire une sortie non transitive, les clefs sont toujours sujettes à compromission. Ce faisant, afin de maximiser la crédibilité du déni dans une panoplie de circonstances, il est avisé de renouveler les clefs éphémères de démarrage (celles illustrées dans l'équation (3.10)) en annonçant de nouvelles par les canaux pair-à-pair tel que décrit dans l'équation (3.9).

De façon à garantir un plus haut niveau de plausibilité, les usagers peuvent aller jusqu'à publier leur secret (sk_{A0} pour Alice) de façon à ce que le question d'un juge à savoir si une clef éphémère est liée à un utilisateur donné se réduise au problème du PFD tel qu'argué par Goldberg et collab. [38]. Il est cependant nécessaire qu'un certain accusé de réception du dernier message envoyé et authentifié par la clef correspondante soit recueilli pour ne pas perturber la capacité de validation de signature des autres membres du groupe.

Bilan

En somme, les idées que nous avançons dans les sections précédentes aboutissent donc à un protocole complet qui répond à un problème n'ayant pas été encore considéré dans la documentation scientifique que nous avons couvert. L'approche que nous suggérons élimine la nécessité de communiquer pour garantir la non-transitivité de paternité sur les messages. Cette particularité permet certains avantages lorsque la question de bande passante est critique. Il est d'autant plus pertinent de remarquer que cela engendre aussi une diminution des informations pouvant être enregistrées par un espion. L'étape suivante est de fournir des preuves pour les conjectures 3.3.1, 3.3.2 et 3.3.3.

CONCLUSION

Dans ce mémoire, nous avons présenté un aperçu de concepts complémentaires qui appartiennent chacun à des catégories distinctes. En premier lieu, l'exploration des tables de hachages a démontré une vision différente des réseaux usuels sur lesquels sont basées les applications de nos jours. Ce modèle permet de diminuer les risques de censure tout en générant un coût global d'entretien et de fonctionnement minimal. Sa nature à passer à l'échelle pour des réseaux de grandeur arbitraire explique ce phénomène. Nos travaux sur OpenDHT nous ont mené à nous familiariser avec le concept théorique et la découverte de défis de conception substantiels (chapitre 2). En effet, les opérations triviales faites dans le domaine des réseaux du paradigme centralisé deviennent plus complexes dans un contexte distribué. Notamment, l'élaboration d'une stratégie complètement distribuée d'un index reposant sur une THD s'avère un problème difficile. À ce titre, les efforts décrits dans [42] et concrétisés dans un prototype nommé YaCy méritent considération.

En second lieu, l'état de l'art du clavardage sûr bout en bout a suivi et ainsi permis une vue d'ensemble des questions actuellement non résolues et alors que les autres nous servent de base solide. La structure d'ART détaillé au chapitre 3 se démarque des autres méthodes par sa stratégie de démarrage en contexte asynchrone. C'est en s'inspirant des formules existantes qu'on a finalement proposé un protocole étendu du modèle ART en employant des briques fondamentales bien connues, mais d'une manière qui n'a jamais paru dans la documentation scientifique. Nous avons confiance de pouvoir fournir prochainement les preuves des conjectures 3.3.1, 3.3.2 et 3.3.3 sur lesquelles repose notre prototype.

Bien que ces deux concepts différents puissent s'utiliser l'un sans l'autre, il n'en demeure pas moins que la combinaison des deux formes un système hautement respectueux de la vie privée des gens. De plus, celle-ci garantit non moins une protection crédible contre la censure. Bien sûr, ces systèmes ne demeurent pas sans inconvénient. En effet, la question des généraux byzantins (mentionnée au début du chapitre 3) constitue un des principaux obstacles à considérer lors de l'élaboration de tels structures, une difficulté qui n'existe pas dans le contexte centralisé.

D'autres modèles peuvent compléter celui-ci comme démontré par IPFS [7] qui permet le stockage longue durée. Ce projet, visant à remplacer la Toile centralisée telle que nous la connaissons aujourd'hui, procure certaines propriétés supplémentaires et est donc un candidat prometteur pour fournir plusieurs solutions aux problèmes que nous étudions.

Ensuite, des initiatives de la sorte comme Solid [93, 96] offrent une vision similaire. Cette question est analysée sérieusement depuis plusieurs années et aujourd'hui encore. L'apparition de nouvelles briques imposantes telles que les chaînes de blocs (appelées *blockchain* en anglais) laissent présager un avenir où les notions des systèmes distribués seront d'autant plus approfondies et nombreuses.

La sécurité bout en bout est un paradigme bien connu et naturel, mais peu implémenté de nos jours pour des raisons historiques de limitations maintenant révolues. Les récentes tendances, telles que l'adoption des articles 11 et 13 du controversé projet de loi européen [75] incitent à la censure sur Internet. Penser qu'un passage complet vers des modèles sûrs bout en bout des applications d'utilisateurs finaux est urgent apparaît donc comme une proposition légitime. Comme nous l'avons montré dans ce document, la question du clavardage sûr bout en bout n'est pas totalement réglée et demande encore qu'on s'y penche. Cependant, en pratique,

l'apparition de failles de sécurité [81] est rapidement succédée de corrections par les concepteurs. Ce faisant, plusieurs modèles tels que Matrix [62, 63], Signal et WhatsApp sont déjà aujourd'hui démontrés comme fonctionnels.

ANNEXE A

CRYPTOGRAPHIE : ÉLÉMENTS FONDAMENTAUX

Nous introduisons ici des concepts primaires du domaine de la cryptographie. Le contenu repose plus vraisemblablement sur des connaissances historiques que sur une base solide pour aborder la théorie contemporaine. Il va sans dire que les notions de cette annexe visent à préparer superficiellement le lecteur non initié aux idées élémentaires de l'écriture secrète. Nous décrivons, dans la première section, différents chiffres rudimentaires tels que le schéma de Vernam. Nous terminons l'annexe avec une présentation des concepts de la sécurité parfaite de Shannon [87] avec un exemple sur le chiffrement de Playfair.

A.1 Chiffres rudimentaires

Parmi les tout premiers schémas, on retrouve le « chiffrement par substitution ». Une instance particulière de celui-ci est appelée le « chiffre de César » ou « chiffrement par décalage » et est illustrée par la figure A.1. Ce célèbre schéma consiste en un simple décalage arbitraire de lettres d'un alphabet. Par exemple, avec un déphasage de 3 lettres, le cryptogramme du palindrome « La mariée ira mal » est le phrase anacyclique « Od pdulhh lud pdo ». Cette approche d'un décalage par 3 était utilisé afin de communiquer de façon secrète des informations entre l'empereur César et ses généraux. Bien sûr, le chiffre de César n'est pas sûr du tout, car

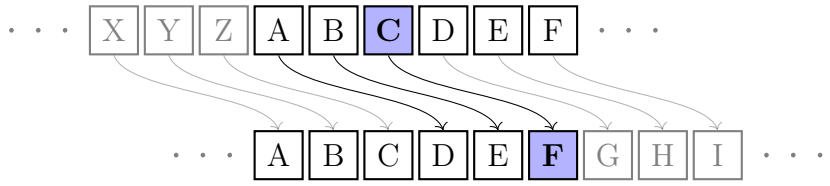


FIGURE A.1: Chiffre de César [1] (décalage de trois lettres).

l'espace des clefs est bien trop petit. En effet, seulement 26 décalages (taille de l'alphabet latin) sont admissibles. Il est dès lors trivial de casser le chiffre en essayant toutes les possibilités. On dit alors que le schéma est sensible aux *attaques par force brute*.

Le cas général du chiffrement par substitution consiste en l'association de chaque lettre de l'alphabet à une unique lettre distincte. L'espace des clefs admissibles passe donc de la totalité des décalages envisageables (la taille de l'alphabet) à l'ensemble de toutes les bijections sur l'alphabet. Ce faisant, la taille de l'espace des clefs n'est plus 26, mais $26 \cdot 25 \cdot 24 \cdot \ldots \cdot 1 = 26! \approx 2^{88}$. Ainsi, le chiffre par substitution n'est pas vulnérable aux attaques par force brute, mais n'est pas pour autant sûr. En effet, il est connu depuis longtemps que ce chiffre peut se casser facilement grâce à une analyse de fréquence de lettres dans le message (Al-Kindi, 801-873 apr. J.-C.). Cela est dû à la caractéristique inhérente des langues qui fait que certaines lettres reviennent plus souvent que d'autres dans le dictionnaire.

En faisant un saut dans le temps jusqu'au début du 20^{e} siècle, on retrouve alors un autre célèbre chiffre : le masque jetable (ou encore « chiffre de Vernam »). Son principe est le suivant : soit m un message composé de ℓ bits et la clef k, une chaîne aléatoire de même longueur. Le cryptogramme c généré par le masque

1. La figure est inspirée d'une image fournie à l'adresse suivante https://fr.wikipedia.org/wiki/Chiffrement_par_d%C3%A9calage

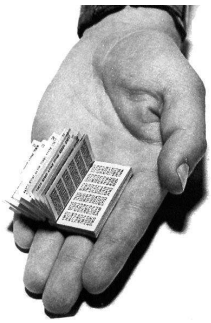


FIGURE A.2: Masque jetable russe (photo prise par un agent du MI5 [2]).

jetable est :

$$c = m \oplus k,$$

où $\oplus$ dénote l'opérateur de disjonction logique exclusive telle que définie ci-après :

$$a, b \in \{0,1\}^{\ell}, \qquad \oplus(a,b) = \Big\| _{i \in [\ell]} \, a_i \oplus' b_i \qquad \text{où} \qquad \begin{array}{c|cc} \oplus' & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array}$$

Remarque A.1.1. L'opérateur $\oplus'$ défini ci-haut est normalement simplement noté par $\oplus$. L'interprétation est évidente suivant que les opérandes sont des bits ou des chaînes de bits. $\Diamond$

Clairement, par la table de Cayley ci-haut, on a que le message déchiffré est obtenu par $c \oplus k = m \oplus k \oplus k = m$. Ce schéma a la particularité de satisfaire à la sécurité parfaite (ou inconditionnelle) établie par Shannon [87], c'est-à-dire que pour toutes valeurs c et m, la notion de c (mais ni de m ni de k) permet un avantage exactement nul pour déterminer m. En d'autres termes, le fait de connaître c ne fournit aucun renseignement sur m (voir la définition A.2.1). Cette définition n'est pas respectée par le chiffrement par substitution par exemple car l'analyse de fréquence de lettres facilite significativement son cassage. Malgré le

2. http://www.ranum.com/security/computer_security/papers/otp-faq/otp.jpg

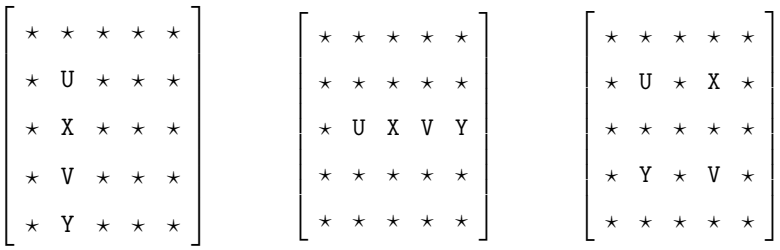
bénéfice incontestable de ce chiffre, il possède un défaut considérable qui le rend non-pratique. Évidemment, il s'agit d'utiliser k pour deux messages m_1, m_2 afin d'apprendre de l'information sur m_1 et m_2 puisqu'étant donné les cryptogrammes $c_1 = m_1 \oplus k$ et $c_2 = m_2 \oplus k$, on a que

$$c_1 \oplus c_2 = m_1 \oplus k \oplus m_2 \oplus k = m_1 \oplus m_2.$$

Cette éventualité permet donc d'associer deux messages ensemble, une faille importante de ce chiffre. En effet, s'il advenait que m_1 soit découvert, alors m_2 pour être directement déduit. Comme illustré dans la figure A.2, ce schéma était tout de même employé par les services secrets russes au cours du 21$^{\text{e}}$ siècle en raison de sa propriété de sécurité parfaite. Cependant, l'usage de ce chiffre demandait une gestion minutieuse des clefs par les agents à l'aide de leur petit calepin pour ainsi éviter d'utiliser deux fois une même clef k pour deux communications différentes.

A.2 Sécurité parfaite

Afin de décrire le concept de *sécurité parfaite* de Shannon [87], on introduit l'espace fini des messages $\mathcal{M}$ avec $|\mathcal{M}| > 1$. On y attribue une variable aléatoire M. Pour $m \in \mathcal{M}$, on dit que $\mathbb{P}[M = m]$ est la probabilité de choisir aléatoirement m dans $\mathcal{M}$. Comme mentionné plutôt, Gen est un algorithme probabiliste retournant une clef k choisie selon une distribution donnée. L'espace de toutes les sorties envisageables de Gen est noté $\mathcal{K}$. Soient K, une variable aléatoire sur l'espace $\mathcal{K}$, et $k \in \mathcal{K}$ on dit donc que $\mathbb{P}[K = k]$ est la probabilité que K prenne une valeur donnée k. On rappelle que pour $m \in \mathcal{M}$ et $k \in \mathcal{K}$, $\mathsf{Enc}_k(m)$ retourne un cryptogramme c. De plus, on dénote par $\mathcal{C}$ l'espace de tous les messages codés possiblement produits par $\mathsf{Enc}_k(m)$ pour toutes valeurs de $k \in \mathcal{K}$ et $m \in \mathcal{M}$. On libelle la probabilité que le cryptogramme ait la valeur $c \in \mathcal{C}$ par $\mathbb{P}[C = c]$. Finalement, Dec admet comme entrée une clef $k \in \mathcal{K}$ et un cryptogramme $c \in \mathcal{C}$ et



(a) U et V dans la même colonne

(b) U et V sur la même ligne

(c) U et V ni sur la même ligne ou colonne

FIGURE A.3: Chiffre de Playfair : $\mathsf{Enc}(\mathtt{UV}) = \mathtt{XY}$

produit un message $m \in \mathcal{M}$. Le critère de validité du chiffre nous dit que

$$\forall m \in \mathcal{M}, \forall k \in \mathcal{K} \quad \mathbb{P}[\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m] = 1.$$

Autrement dit, Dec est un algorithme déterministe puisqu'il produit toujours la même sortie. Ce faisant, on écrit $m \leftarrow \mathsf{Dec}_k(c)$ afin de dénoter un retour non probabiliste. Les lois de distribution des variables aléatoires M, K et C dépendent du chiffre utilisé et des circonstances dans lesquelles celui-ci est utilisé. En particulier, la distribution sur $\mathcal{K}$ est fixée par la distribution sur la sortie de l'algorithme Gen (normalement uniforme). La distribution sur $\mathcal{M}$ relève, elle, du contexte dans lequel est utilisé le chiffre. Il est possible qu'un adversaire apprenne qu'un message m_0 soit choisi avec probabilité $0 \leq p \leq 1$. Par exemple, les espions de force militaires ennemis pourraient être avisés que $\mathbb{P}[M = \mathtt{Attaquer\ !}] = 0.2$ et $\mathbb{P}[M = \mathtt{Battre\ en\ retraite\ !}] = 0.8$. Finalement, la distribution des cryptogrammes sur $\mathcal{C}$ est entièrement déterminée par les distributions sur $\mathcal{K}$ et $\mathcal{M}$, c'est-à-dire qu'étant donné $k \in \mathcal{K}$ et $m \in \mathcal{M}$, on déduit la distribution sur $\mathcal{C}$ par les sorties de $\mathsf{Enc}_k(m)$.

Exemple A.2.1. *Examinons le chiffre de* Playfair *illustré par la figure A.3. On choisit une chaîne de caractères k_0 de longueur $|k_0| \leq 25$ dans l'espace des lettres*

de l'alphabet. Disons $k_0 =$ `playfair` *par exemple. Ensuite, on construit le rectangle* 5×5 *de gauche à droite et de haut en bas avec les lettres de* k_0 *en omettant les doublons et en considérant* $\texttt{j} \overset{\text{déf}}{=} \texttt{i}$*. Le reste du tableau est rempli avec les autres lettres de l'alphabet. On a donc :*

$$k = \begin{bmatrix} \texttt{p} & \texttt{l} & \texttt{a} & \texttt{y} & \texttt{f} \\ \texttt{i} & \texttt{r} & \texttt{b} & \texttt{c} & \texttt{d} \\ \texttt{e} & \texttt{g} & \texttt{h} & \texttt{k} & \texttt{m} \\ \texttt{n} & \texttt{o} & \texttt{q} & \texttt{s} & \texttt{t} \\ \texttt{u} & \texttt{v} & \texttt{w} & \texttt{x} & \texttt{z} \end{bmatrix}$$

Pour encoder $m =$ `parizeau`*, on considère les digrammes un à un. Selon la figure A.3, on a :*

$$\begin{cases} \texttt{pa} \to \texttt{ly} \\ \texttt{ri} \to \texttt{br} \\ \texttt{ze} \to \texttt{um} \\ \texttt{au} \to \texttt{pw} \end{cases} \implies \mathsf{Enc}_k(\texttt{parizeau}) = \texttt{lybrumpw}$$

Ici, l'espace des clefs $\mathcal{K}$ *est l'espace de toutes les tables* 5×5 *telles que chaque lettre ne se répète jamais et où* `j` *est omis. Par conséquent,* $|\mathcal{K}| = 25 \cdot 24 \cdot \ldots \cdot 2 \cdot 1 = 25!$*. En supposant que* $\mathbb{P}[M = \texttt{ab}] = 0.3$ *et que* $\mathbb{P}[M = \texttt{xz}] = 0.7$*, quelle est la probabilité que* $C =$ `cd` *? Soit* $m = m_0 m_1$*, le texte clair destiné à être chiffré. Dans le cas où* m_0 *et* m_1 *sont placées dans la clef* k *telles que dans la figure A.3a, alors on repère deux positions où la substitution* $m_0 m_1 \to c_0 c_1$ *est possible avec* m_0 *fixé dans le tableau. En effet, si on retient la position de* m_0 *au coin supérieur gauche, alors*

on doit forcément avoir une des deux configurations suivantes :

$$\begin{array}{ccc} m_0 & & m_0 \\ c_0 & & c_0 \\ m_1 & \text{ou} & \star \\ c_1 & & m_1 \\ \star & & c_1 \end{array}$$

Cela découle du fait que le caractère d'entrée est envoyé sur celui du dessous et que $m_0 \neq c_1$, $m_1 \neq c_0$. *Cela valant pour toutes les positions, on retrouve donc* $2 \cdot 25 = 50$ *dispositions de la sorte. On déduit la même chose pour l'agencement de la figure A.3b. Ensuite, dans la configuration de la figure A.3c, quand la première lettre est fixée, on remarque que la seconde doit se trouver à l'extérieur de la ligne et la colonne de la première. Cela laisse donc* 16 *cases éventuelles et ce pour toutes les positions du premier caractère, alors on a* $16 \cdot 25 = 400$ *possibilités dans ces conditions. Dans tous ces derniers cas, on considère le nombre de clefs qui satisfait à la configuration. Comme quatre lettres sont déjà fixées, on déduit donc* $21 \cdot 22 \cdot \ldots \cdot 2 \cdot 1 = 21!$ *clefs admissibles pour chacune positions du digramme* $m_0 m_1$. *Ce faisant, on a que :*

$$\mathbb{P}[C = \texttt{cd} \mid M = \texttt{ab}] = \mathbb{P}[C = \texttt{cd} \mid M = \texttt{xz}] = \frac{(50 + 50 + 400) \cdot 21!}{25!}$$

En somme, en utilisant la formule de probabilité totale :

$$\begin{aligned} \mathbb{P}[C = \texttt{cd}] &= \mathbb{P}[M = \texttt{ab}]\mathbb{P}[C = \texttt{cd} \mid M = \texttt{ab}] + \mathbb{P}[M = \texttt{xz}]\mathbb{P}[C = \texttt{cd} \mid M = \texttt{xz}] \\ &= 0.3 \cdot \frac{(50 + 50 + 400) \cdot 21!}{25!} + 0.7 \cdot \frac{(50 + 50 + 400) \cdot 21!}{25!} \\ &= \frac{500}{25 \cdot 24 \cdot 23 \cdot 22} \\ &= \frac{5}{3036} \end{aligned}$$

$\Diamond$

Définition A.2.1 (Sécurité parfaite)

Un schéma de chiffrement $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ avec un espace de messages $\mathcal{M}$ est *parfaitement sûr* si pour toute distribution de probabilité sur $\mathcal{M}$, tout message $m \in \mathcal{M}$ et cryptogramme $c \in \mathcal{C}$ où $\mathbb{P}[C = c] > 0$, on a que

$$\mathbb{P}[M = m \mid C = c] = \mathbb{P}[M = m].$$

Exemple A.2.2. *Le masque jetable (défini à la section A.1) est un chiffre parfait.*

Démonstration. *Premièrement, observons l'équivalence suivante :*

$$\mathbb{P}[M = m \mid C = c] = \frac{\mathbb{P}[C = c \mid M = m]\,\mathbb{P}[M = m]}{\mathbb{P}[C = c]}$$

Maintenant, selon la définition du chiffre (voir section A.1) et les propriétés algébriques de l'opération $\oplus$ (voir section B.2), on sait que pour $m \in \mathcal{M}$ et $c \in \mathcal{C}$ connu, il existe un unique $k \in \mathcal{K}$ tel que $c = k \oplus m = m \oplus k$. Ce faisant, introduisons la variable aléatoire K sur l'espace des clefs $\mathcal{K}$ $\mathbb{P}[C = c \mid M = m] = \mathbb{P}[K = k]$. On peut donc écrire :

$$\mathbb{P}[M = m \mid C = c] = \frac{\mathbb{P}[K = k]\,\mathbb{P}[M = m]}{\mathbb{P}[C = c]}$$

Puisque la définition du schéma oblige que $\mathcal{M} = \mathcal{K} = \mathcal{C}$ et que la clef est choisie uniformément dans $\mathcal{K}$, alors $\mathbb{P}[K = k] = 1/|\mathcal{K}|$. Il en vaut de même pour $\mathbb{P}[C = c]$ du fait que $\oplus$ est une bijection sur entre $\mathcal{M} \times \mathcal{K}$ et $\mathcal{C}$. Conséquemment, on a que $\mathbb{P}[C = c] = 1/|\mathcal{C}|$. Il s'en suit que :

$$\mathbb{P}[M = m \mid C = c] = \frac{|\mathcal{C}|}{|\mathcal{K}|}\mathbb{P}[M = m] = \mathbb{P}[M = m]$$

$\Diamond$

ANNEXE B

MATHÉMATIQUES : NOMBRES ET STRUCTURES

Cette annexe fournit certaines bases mathématiques en théorie des nombres (section B.1) et théorie des groupes (section B.2). L'ensemble des notions décrites dans la première section se rapportent à certains résultats élémentaires de divisibilité et l'arithmétique modulaire. La seconde section couvre, quant à elle, les éléments fondamentaux des groupes, les groupes finis ainsi que les groupes cycliques. L'ensemble des notions sont tirées depuis une synthèse faite par Katz et Lindell dans leur livre *Introduction to Modern Cryptography, Second Edition* [46].

B.1 Théorie des nombres

On note l'ensemble des entiers par $\mathbb{Z}$ et les entiers positifs par $\mathbb{N}$ incluant 0. On note les ensembles sans 0 respectivement par $\mathbb{Z}^*$ et $\mathbb{N}^*$. Soient $a, b \in \mathbb{Z}$, on dit que a divise b s'il existe un entier c tel que $ac = b$. On note la relation $a \mid b$ et a est appelé un diviseur de b. Lorsqu'aucun nombre c existe tel que $ac = b$, on dit que a ne divise pas b et on note $a \nmid b$.

Un diviseur de b différent de 1 et b est appelé un *facteur* de b. Un nombre $p > 1$ qui ne possède pas de facteur est appelé un nombre *premier*. Autrement, p s'appelle nombre *composé*. Le nombre 1 n'est ni un composé, ni un premier.

Proposition B.1.1

Soient $a \in \mathbb{Z}, b \in \mathbb{N}^*$. Il existe deux nombres uniques $q, r \in \mathbb{Z}$ tels que $a = qb + r$ et $0 \leq r < b$.

Cette proposition formalise le principe de la division d'entiers avec un reste. Cela se montre utile lorsqu'on aborde l'arithmétique modulaire un peu plus loin.

Le *plus grand commun diviseur (PGCD)* de deux entiers a et b, dénoté par $\text{pgcd}(a, b)$, est le plus grand entier c tel que $c \mid a$ et $c \mid b$. En particulier, $\text{pgcd}(0, 0)$ est indéfini, $\text{pgcd}(a, 0) = \text{pgcd}(0, a) = a$ et $\text{pgcd}(a, p) \in \{1, p\}$ lorsque p est premier. Si $\text{pgcd}(a, b) = 1$, on dit que a et b sont *relativement entre eux*.

Proposition B.1.2 (Identité de Bézout)

Soient $a, b \in \mathbb{N}^*$. Il existe $X, Y \in \mathbb{Z}$ tels que $Xa + Yb = \text{pgcd}(a, b)$. De plus, $\text{pgcd}(a, b)$ est le plus petit nombre naturel pouvant s'écrire de cette manière.

Nous pouvons maintenant établir les différents concepts d'arithmétique modulaire. Selon la proposition B.1.1, pour $a, b, N \in \mathbb{Z}$, $N > 1$, il existe les nombres $q, r \in \mathbb{Z}$ uniques tels que $a = qN + r$ et $0 \leq r < N$. On dénote par $r = [a \bmod N]$, le *reste de la division* de a par N. Plus généralement, on a la définition suivante.

Définition B.1.1 (Congruence modulo)

Soient $a, b, q, N \in \mathbb{Z}$ et $N > 1$. On dit que a et b sont *congrus modulo* N, et on écrit $a \equiv b \mod N$ si et seulement si

$$a - b = qN$$

De façon équivalente, on écrit $[a \bmod N] = [b \bmod N]$.

La relation $\equiv$ est une relation d'équivalence et s'appelle *équivalence modulo*. Les

propriétés arithmétiques suivantes sont préservées :

$$\begin{cases} a \equiv a' & \mod N, \\ b \equiv b' & \mod N \end{cases} \iff \begin{cases} (a+b) \equiv (a'+b') & \mod N, \\ ab \equiv a'b' & \mod N \end{cases} \tag{B.1}$$

Exemple B.1.1. *Calculons* $[105 \cdot 35 \bmod 5]$. *Selon les propriétés de l'équation* (B.1), *il s'en suit que :*

$$\begin{aligned} [105 \cdot 35 \bmod 5] &= [105 \bmod 5] \cdot [35 \bmod 5] \\ &= [21 \bmod 5] \cdot [7 \bmod 5] \\ &= 1 \cdot 2 \\ &= 2. \end{aligned}$$

$\Diamond$

Toutefois, les propriétés analogues pour la division ne sont pas garanties, c'est-à-dire que $a \equiv a' \mod N$ et $b \equiv b' \mod N$ n'implique pas que ${}^{a}\!/\!_{b} \equiv {}^{a'}\!/\!_{b'} \mod N$. En particulier,

$$ac \equiv bc \mod N \not\Longrightarrow a \equiv b \mod N.$$

Ceci est dû au fait que tous les nombres ne sont pas nécessairement *inversibles*. On souhaite cependant fonctionner avec des nombres possédant un inverses.

Définition B.1.2 (Inversibile modulo)

Soit $a \in \mathbb{Z}, N \in \mathbb{N}$. S'il existe un entier c tel que $ac \equiv 1 \mod N$, on dit que a est *inversible modulo* N.

À cet effet, la proposition B.1.3 permet de déterminer efficacement si un nombre est inversible.

Proposition B.1.3

Soient $a, N \in \mathbb{N}$ tel que $N > 1$. Le nombre a est *inversible modulo* N si et seulement si $\text{pgcd}(a, N) = 1$.

Démonstration. Soient $a, N \in \mathbb{N}$ avec $N > 1$. Supposons que a est inversible modulo N. Alors il existe $b \in \mathbb{Z}$ tel que $ab \equiv 1 \mod N$. Il s'en suit que $ab - 1 = cN$ pour une $c \in \mathbb{Z}$, c'est-à-dire que $ab - 1$ est un multiple de N. On peut réécrire comme :

$$ab - cN = 1$$

Selon la proposition B.1.2, $\text{pgcd}(a, N)$ est le plus petit nombre naturel qui s'écrit de cette manière. On a donc que $\text{pgcd}(a, N) = 1$.

Supposons maintenant que $\text{pgcd}(a, N) = 1$. Par la proposition B.1.2, il existe $X, Y \in \mathbb{Z}$ tels que $Xa + YN = 1$. Ce faisant, on a que

$$\begin{aligned} & Xa + YN \equiv 1 \mod N \\ \implies \quad & Xa \equiv 1 \mod N \qquad \text{puisque } YN \equiv 0 \mod N \end{aligned}$$

On a ainsi que a est inversible modulo N. □

B.2 Théorie des groupes

La théorie des groupes a été initiée par Évariste Galois au XIX^{e} siècle. Celle-ci fournit un formalisme des structures algébriques décrivant les relations entre des éléments par des opérations.

Définition B.2.1 (Structure algébrique)

Une *structure algébrique* est un ensemble non vide d'éléments avec une ou plusieurs opérations binaires définie(s) sur cet ensemble.

Une structure algébrique S munie d'une opération binaire $\circ$ est dénotée par $(S, \circ)$. L'image de $(a, b) \in S^2$ par $\circ$ est dénotée par $\circ(a, b)$ ou plus simplement $a \circ b$.

Définition B.2.2 (Fermeture)

Soit $\circ : S \times S \to S$, une application sur S et $A \subseteq S$. Le sous-ensemble A est *fermé* sous l'opération $\circ$ si $a \circ b \in A$ pour tout $a, b \in A$.

Une structure algébrique $(S, \circ)$, pour laquelle S est fermé sous $\circ$ se nomme *magma*. L'opération $\circ$ est alors nommée la loi de composition interne de S.

Exemple B.2.1. *Le couple* $(\mathbb{N}, +)$ *est un magma. En effet,* $\mathbb{N} \subseteq \mathbb{N}$ *est fermé sous l'addition* $+ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. $\Diamond$

Afin de définir les prochains concepts, nous nous attardons maintenant à certaines propriétés sur les structures algébriques et leurs opérations binaires.

Définition B.2.3 (Associativité)

Soit un magma $(S, \circ)$ et $a, b, c \in S$. L'opération binaire $\circ$ est dite *associative* si

$$(a \circ b) \circ c = a \circ (b \circ c).$$

Exemple B.2.2. *Soit* $S = \{0, 1\}$ *et l'opération binaire* $\oplus$ *soumise à la table de Cayley suivante :*

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

L'opération $\oplus$ *est associative. Pour le démontrer, observons que* $\oplus$ *peut directe-*

ment se définir par $\oplus(a,b) \overset{\text{déf}}{=} [a+b \bmod 2]$. *Par l'équation* (B.1)*, on a :*

$$\begin{aligned}(a \oplus b) \oplus c &= [[a+b \bmod 2] + c \bmod 2] \\ &= [a+b+c \bmod 2] \\ &= [a + [b+c \bmod 2] \bmod 2] \\ &= a \oplus (b \oplus c)\end{aligned}$$

◊

Définition B.2.4 (Commutativité)

Soit un magma $(S, \circ)$ et $a, b, c \in S$. L'opération binaire $\circ$ est dite *commutative* si

$$a \circ b = b \circ a.$$

Exemple B.2.3. *Le magma* $(\{0,1\}, \oplus)$ *défini dans l'exemple B.2.2 est commutatif.* ◊

Un magma avec une loi de composition interne associative s'appelle un *semi-groupe*.

Définition B.2.5 (Élément neutre)

Soit $(S, \circ)$, un semi-groupe et $e \in S$ tel que pour $a \in S$:

$$e \circ a = a \circ e = a.$$

L'élément e s'appelle *élément neutre*.

Un semi-groupe comportant un élément neutre s'appelle un *monoïde*.

Définition B.2.6 (Inverse)

Soit $(S, \circ)$, un monoïde et $a \in S$. S'il existe $a' \in S$ tel que $a \circ a' = e$, alors a' s'appelle *l'inverse* de a.

Il est maintenant possible de définir la notion de groupe.

Définition B.2.7 (Groupe)

Soit $(S, \circ)$, un monoïde tel que $a \in S$ possède un inverse $a' \in S$. La structure $\mathbb{G} = (S, \circ)$ s'appelle un *groupe*. Le nombre d'éléments dans un groupe $|\mathbb{G}|$ s'appelle *l'ordre du groupe*.

Si $\mathbb{G} = (S, \circ)$ est un groupe et que $H \subseteq S$ forme un groupe sous la même opération, alors on dit que $(H, \circ)$ est un sous-groupe de $\mathbb{G}$. Un groupe dont l'opération est commutative s'appelle *groupe abélien*.

Exemple B.2.4. *La structure* $\mathbb{G} = (\{0,1\}, \oplus)$ *définie dans l'exemple B.2.2 est un groupe. Évidemment, d'après la table de Cayley,* $0 \oplus a = a \oplus 0 = a$*, pour tout* $a \in \{0,1\}$*. Par conséquent,* 0 *est l'élément neutre de* $\mathbb{G}$*. De plus, pour* $a \in \{0,1\}$*,* $-a = a$*, c'est-à-dire que* 0 *et* 1 *sont leur propre inverse puisque* $0 \oplus 0 = 0$ *et* $1 \oplus 1 = 0$*. Enfin,* $\mathbb{G}$ *est abélien. En effet,* $0 \oplus 1 = 1 = 1 \oplus 0$*.* ◊

Exemple B.2.5. *Soit* $n \in \mathbb{N}$*. L'ensemble des entiers modulo* n*, dénoté par* $\mathbb{Z}/n\mathbb{Z}$*, muni de l'addition* $+_n$ *définie par* $a +_n b \overset{\text{déf}}{=} [a + b \bmod n]$ *où* $a, b \in \mathbb{Z}$*, est un groupe abélien.* ◊

Par souci de simplicité, on dénote l'opération du groupe $+$ plutôt que $\circ$ lorsqu'on choisit d'adopter la notation *additive*. De plus, pour $a, b \in \mathbb{G}$, $a + (-b)$ s'écrit plus naturellement $a - b$. On signifie aussi l'élément neutre d'un groupe additif par 0. La notation *multiplicative* peut aussi être employée pour libeller les éléments et opérations d'un groupe. Dans ce cas, 1 désigne l'élément neutre et « $\cdot$ » l'opération. L'élément inverse de b est dénoté par b^{-1} ou $\frac{1}{b}$ et on a ainsi $b \cdot b^{-1} = 1$. Lorsqu'on utilise cette notation, on omet parfois d'écrire l'opérateur de multiplication et on obtient donc simplement $bb^{-1} = 1$.

Lemme B.2.1

Soit $\mathbb{G}$ un groupe muni de l'opération de multiplication $\cdot$ et $a, b, c \in \mathbb{G}$. Si $ac = bc$, alors $a = b$. Particulièrement, si $ac = c$, alors a est l'identité dans $\mathbb{G}$.

Démonstration. Puisque $a, b, c \in \mathbb{G}$ et que $\mathbb{G}$ est un groupe, il existe $c^{-1} \in \mathbb{G}$ tel que

$$ac = bc \implies (ac)c^{-1} = (bc)c^{-1} \implies a(cc^{-1}) = b(cc^{-1}) \implies a \cdot 1 = b \cdot 1 \implies a = b$$

□

Lorsqu'une expression devient plus complexe, il est utile d'introduire des notations permettant de simplifier l'écriture sans perte de rigueur. C'est le cas pour les opérations répétées plusieurs fois sur un même élément. Soit $g \in \mathbb{G}$ et $m \in \mathbb{N}$, on a donc :

$$mg \overset{\text{déf}}{=} \underbrace{g + g + \cdots + g}_{m \text{ fois}}$$

Il est pertinent de remarquer que mg n'est qu'une notation et n'est pas l'application de l'opération du groupe entre m et g. Heureusement, certaines propriétés intuitives sont préservées. Soient $m, m' \in \mathbb{N}$ et $g, h \in \mathbb{G}$:

$$(mg) + (m'g) = (m + m')g, \ m(m'g) = (mm')g,$$

$$(mg) + (mh) = m(g + h) \text{ et } 1 \cdot g = g.$$

De façon similaire, on introduit la représentation suivante pour un groupe multiplicatif :

$$g^m \overset{\text{déf}}{=} \underbrace{gg \ldots g}_{m \text{ fois}}$$

On a aussi les règles usuelles sur les exposants :

$$g^m \cdot g^{m'} = g^{m+m'}, \ (g^m)^{m'} = g^{mm'} \text{ et } g^1 = g$$

Si $\mathbb{G}$ est un groupe abélien, alors pour $g, h \in \mathbb{G}$, $g^m \cdot h^m = (gh)^m$.

B.2.1 Groupe fini

Lorsqu'un groupe possède un nombre fini d'éléments, c'est-à-dire que son ordre est un nombre fini, alors on le nomme *groupe fini*. Le théorème B.2.1 montre la caractéristique principale de ce type de groupe.

Théorème B.2.1

Soit $\mathbb{G}$ un groupe fini d'ordre $m = |\mathbb{G}|$. Pour tout élément $g \in \mathbb{G}$, $g^m = 1$.

Le théorème précédent stipule que la simple hypothèse d'un ordre fini engendre des « cycles » dans la multiplication successive d'un élément par lui-même. Cela est d'autant plus clairement formulé dans le corollaire B.2.1.

Corollaire B.2.1

Soit $\mathbb{G}$ un groupe fini avec $m = |\mathbb{G}| > 1$. Alors $\forall g \in \mathbb{G}, x \in \mathbb{Z}$,

$$g^x = g^{[x \bmod m]}$$

Démonstration. Soient $x \in \mathbb{Z}$ et $m = |\mathbb{G}|$. Selon la proposition B.1.1, on peut écrire $x = qm + r$ pour $q, r \in \mathbb{Z}$. Ainsi,

$$g^x = g^{qm+r} = g^{qm} \cdot g^r = \underbrace{(g^m)^q \cdot g^r = 1^q \cdot g^r}_{\text{par thm. B.2.1}} = g^r$$

□

On sait par l'exemple B.2.5 que $\mathbb{Z}/n\mathbb{Z}$ est un groupe sous l'opération d'addition. On peut aussi définir un groupe grâce à $\mathbb{Z}/n\mathbb{Z}$ et la multiplication. Comme tous les éléments d'un groupe doivent avoir un inverse et que 0 ainsi que d'autres éléments de $\mathbb{Z}/n\mathbb{Z}$ n'ont pas d'inverse, on doit retirer de l'ensemble tous les éléments n'étant pas inversibles. La proposition B.1.3 nous signifie qu'un nombre est inversible

modulo n si et seulement si il ne partage aucun facteur avec n, c'est-à-dire que a et n sont relativement entre euxs. Par conséquent, on définit l'ensemble suivant :

$$\mathbb{Z}_n^* = \{a \in \mathbb{Z}/n\mathbb{Z} \mid \text{pgcd}(a, n) = 1\}, \quad \forall n \in \mathbb{N}^*$$

On appelle $\mathbb{Z}_n^*$ *l'ensemble des inversibles modulo* n.

Proposition B.2.1

Soient $N \in \mathbb{N}^*$ et $\mathbb{G} = (\mathbb{Z}_N^*, \cdot)$ tel que pour $a, b \in \mathbb{G}$, $ab \overset{\text{déf}}{=} [ab \bmod N]$. Alors, $\mathbb{G}$ est un groupe.

Démonstration. Afin de démontrer cela, on doit montrer que les axiomes de la théorie des groupes (les Définitions B.2.2 à B.2.6) sont satisfaits. Soient $a, b \in \mathbb{Z}_N^*$.

(i) (Élément neutre) Par hypothèse, $1 \in \mathbb{Z}/N\mathbb{Z}$ et $\text{pgcd}(1, N) = 1$, $\forall N \in \mathbb{N}^*$. Dans ce cas, on a que $1 \in \mathbb{Z}_N^*$ et 1 est clairement l'élément neutre de $\mathbb{G}$.

(ii) (Inverse) *A priori*, pour $a \in \mathbb{Z}_N^*$ on a que $\text{pgcd}(a, N) = 1$. Par la proposition B.1.3, a est inversible.

(iii) (Fermeture) Par définition, on a que $ab = [ab \bmod N]$ est inversible modulo N. Plus particulièrement, $[a^{-1}b^{-1} \bmod N]$ est l'inverse de $[ab \bmod N]$. Par conséquent, $\text{pgcd}([ab \bmod N], N) = 1$, donc on a bien que $ab \in \mathbb{Z}_N^*$.

Les propriétés d'associativité et de commutativité sont héritées directement de la multiplication dans $\mathbb{Z}/N\mathbb{Z}$. □

On dénote l'ordre du groupe $(\mathbb{Z}_N^*, \cdot)$ par $\phi(N) \overset{\text{déf}}{=} |\mathbb{Z}_N^*|$. La fonction ϕ est appelée la *fonction indicatrice d'Euler*. Voici quelques valeurs singulières que prend la fonction : (1) Lorsque N est premier, on voit tout de suite que $\phi(N) = |\mathbb{Z}/N\mathbb{Z}|$, c'est-à-dire $N-1$, puisque tous les nombres sont premiers entre eux. (2) Si $N = pq$ pour p et q des premiers, subséquemment $\phi(N) = (p-1)(q-1)$. Afin de montrer

cela, soient $N_p, N_q, N_{pq} \in \mathbb{N}$, les cardinalités des ensembles de nombres divisibles par p, q et pq, respectivement. Clairement, on a que :

$$\phi(N) = N - 1 - \underbrace{(N_p + N_q - N_{pq})}_{\text{divisble par } p \text{ ou } q}.$$

Afin de déterminer les valeurs respectives de N_p et N_q, on observe premièrement que si $\alpha \in \mathbb{Z}/N\mathbb{Z}$ et que α et N ne sont pas premiers entre eux, alors il partage au moins un facteur avec $N = pq$, c'est-à-dire que $p \mid \alpha$ ou bien $q \mid \alpha$. Cependant, $pq \mid \alpha \Rightarrow \alpha \geq N$, mais on sait que $\alpha < N$. Par conséquent, on voit bien que $N_{pq} = 0$. Ensuite, les éléments divisibles par p sont les $q-1$ éléments $p, 2p, 3p, \ldots,$ et $(q-1)p$. On observe une chose symétrique pour les valeurs divisibles par q. Conséquemment, on a que :

$$\phi(N) = N - 1 - (q - 1 + p - 1) = pq - q - p + 1 = (p-1)(q-1).$$

De façon générale, on a le théorème B.2.2.

Théorème B.2.2

Soit $N = \prod_i p_i^{e_i}$ où $\{p_i\}$ est un ensemble fini de nombres premiers et $e_i \geq 1$. Par conséquent,

$$\phi(N) = \prod_i p_i^{e_i - 1}(p_i - 1).$$

Finalement, on remarque que pour le groupe $(\mathbb{Z}_N^*, \cdot)$, l'application du théorème B.2.1 implique que pour $g \in \mathbb{Z}_N^*$, alors $g^{\phi(N)} \equiv 1 \mod N$. Si N est premier, on obtient ainsi que $g^{N-1} \equiv 1 \mod N$.

B.2.2 Groupe cyclique et générateur

Soit $\mathbb{G}$ un groupe fini d'ordre m. Considérons l'ensemble suivant pour $g \in \mathbb{G}$:

$$\langle g \rangle \overset{\text{déf}}{=} \{g^0, g^1, \ldots\}$$

Remarque B.2.1. $\langle g \rangle$ pourrait être défini différemment pour un groupe infini. $\Diamond$

Selon le théorème B.2.1, on a que $g^m = 1$. Posons $0 < i \leq m$ comme le plus petit entier naturel tel que $g^i = m$, alors on peut réécrire $\langle g \rangle$ de façon équivalente comme :

$$\langle g \rangle = \{g^0, g^1, \ldots, g^{i-1}\}$$

Par conséquent, le nombre d'éléments uniques dans $\langle g \rangle$ est forcément i. On peut démontrer assez aisément que $\langle g \rangle$ est un sous-groupe de $\mathbb{G}$. On appelle donc $\langle g \rangle$ le *sous-groupe de* $\mathbb{G}$ *d'ordre* i *généré par* g. Ceci nous mène à la définition B.2.8.

Définition B.2.8 (Ordre d'un élément)

Soient $\mathbb{G}$ un groupe fini et $g \in \mathbb{G}$. *L'ordre de* g est le plus petit entier naturel, différent de zéro, tel que $g^i = 1$.

La proposition suivante est analogue au corollaire B.2.1.

Proposition B.2.2

Soient $\mathbb{G}$ un groupe fini et $g \in \mathbb{G}$ un élément d'ordre i. Alors, pour tout entier x, on a que $g^x = g^{[x \bmod i]}$

La preuve pour la proposition précédente est identique à celle de B.2.1. De façon encore plus prononcée, on peut démontrer la proposition B.2.3.

Proposition B.2.3

Soient $\mathbb{G}$ un groupe fini et $g \in \mathbb{G}$ un élément d'ordre i. Alors, $g^x = g^y$ si et seulement si $x \equiv y \mod i$.

Démonstration. ($\Leftarrow$) Si $x = y \mod i$, alors $[x \bmod i] = [y \bmod i]$. Ce faisant, par la proposition B.2.2, on a donc :

$$g^x = g^{[x \bmod i]} = g^{[y \bmod i]} = g^y.$$

($\Rightarrow$) Si $g^x = g^y$, on peut en conséquence écrire que $1 = g^{x-y} = g^{[x-y \bmod i]}$. Puisque $[x - y \bmod i] < i$ et que $i \neq 0$ est le plus petit entier naturel tel que $g^i = 1$, on a forcément que $[x - y \bmod i] = 0$. □

Les différents éléments d'un groupe peuvent engendrer différents sous-groupes distincts. En particulier, l'élément neutre engendre toujours le sous-groupe trivial $\langle 1 \rangle = \{1\}$. Définitivement, on voit que $\langle 1 \rangle$ est d'ordre 1. L'autre cas particulier qui vient en tête est celui d'un élément d'ordre m. Ceci est concrètement décrit dans la définition B.2.9.

Définition B.2.9 (Groupe cyclique et générateur)

Soit $\mathbb{G}$ un groupe fini d'ordre m. S'il existe $g \in \mathbb{G}$ un élément d'ordre m, alors $\mathbb{G}$ est appelé *groupe cyclique* et g le *générateur* de $\mathbb{G}$.

Remarque B.2.2. Il s'en suit donc que $\langle g \rangle = \mathbb{G}$. ◊

On appelle un tel groupe « cyclique » puisqu'il est caractérisé par l'existence d'un cycle qui traverse tous les éléments contrairement à un simple groupe fini où aucun cycle ne parcourt tous les éléments.

Exemple B.2.6. *Soit le groupe* $(\mathbb{Z}_4, +)$. *Ce groupe est cyclique et son générateur est* 1. *En effet,*

$$4 \cdot 1 \equiv 0 \mod 4 \quad \text{et} \quad \text{pour tout} \quad 0 < i \leq 4, \; i \cdot 1 \not\equiv 0 \mod 4$$

Le générateur 1 *dans le groupe additif est trivial. En fait, il est un générateur de tous les groupes modulo finis avec l'addition. Cependant, il n'est pas le seul générateur à tout coup. Par exemple, dans le cas de* $\mathbb{Z}_4$, 3 *est aussi un générateur puisque* $\langle 3 \rangle = \{0, 3, 2, 1\}$. *Pas tous les éléments sont des générateurs. Par exemple,* 2 *est d'ordre* 2 *car* $\langle 2 \rangle = \{0, 2\}$. ◊

Proposition B.2.4

Soient $\mathbb{G}$ un groupe fini d'ordre m et g d'ordre i. Alors, $i \mid m$.

Démonstration. Par le théorème B.2.1, on a que $g^m = 1 = g^0$. Selon la proposition B.2.3, on doit donc avoir que $m \equiv 0 \mod i$, c'est-à-dire que $i \mid m$. □

Corollaire B.2.2

Soit $\mathbb{G}$ un groupe d'ordre p. Si p est premier, alors $\mathbb{G}$ est cyclique. De plus, tous les éléments de $\mathbb{G}$, hormis l'identité, sont des générateurs de $\mathbb{G}$.

Démonstration. Selon la proposition B.2.4, 1 et p sont les uniques candidats puisque seuls ces deux éléments divisent le nombre premier p. Comme 1 est d'ordre 1, tous les éléments sont *ipso facto* d'ordre p. Ce faisant, on a le résultat voulu. □

La portée de ce résultat est majeure. En effet, les groupes d'ordre premier jouent un rôle essentiel dans la conception de certains schémas cryptographiques. En outre, le théorème B.2.3 permet de distinguer une autre classe importante de groupes cycliques encore une fois tirant avantage des propriétés des nombres premiers.

Théorème B.2.3

Si p est premier, alors $\mathbb{Z}_p^*$ est un groupe cyclique d'ordre $p - 1$.

ANNEXE C

GPL : GENERAL PUBLIC LICENSE

La licence GPL s'agit d'un texte légal réglementant l'utilisation, le partage et la modification d'un programme où les usagers sont les bénéficiaires des libertés en découlant. La licence est issue du mouvement du logiciel libre qui trouve son origine au milieu des années 80 avec comme acteur principal Richard Matthew Stallmann (RMS). Il est le créateur de la fondation du logiciel libre (*Free Software Foundation*) et du projet GNU visant à fournir un système d'exploitation complet et entièrement libre. L'acronyme est la version courte du nom récursif *GNU's Not Unix*. Ce projet forme maintenant la base, conjointement avec le noyau Linux, d'un nombre très important de structures indispensables de tous les jours [79]. De multiples projets démarrent sur cette base afin d'aboutir à des résultats rapidement, engendrant dès lors par la même occasion l'adoption chez plusieurs collaborateurs, promettant ainsi la longévité de leur produit. Enfin, cette licence s'efforce de garantir la pérennité de quatre (4) libertés fondamentales, soient [33] :

0. la liberté d'utiliser le logiciel à n'importe quelle fin,
1. la liberté de modifier le programme pour répondre à ses besoins,
2. la liberté de redistribuer des copies à ses amis et voisins,
3. la liberté de partager avec d'autres les modifications qu'il a faites.

Ci-après, le texte communément utilisé en entête de fichier pour spécifier la licence

GPLv3.

```
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

La licence intégrale peut-être trouvée à l'adresse suivante :

https://www.gnu.org/licenses/gpl-3.0.fr.html

RÉFÉRENCES

[1] *(n+1)sec*. Anglais. 2015. URL : https://learn.equalit.ie/wiki/Np1sec (visité le 20/04/2017).

[2] Adam Back et collab. « Hashcash-a denial of service counter-measure ». Dans : (2002).

[3] Hari Balakrishnan et collab. « Looking up data in P2P systems ». Anglais. Dans : *Communications of the ACM* 46.2 (2003), p. 43-48.

[4] Alistair P Barros et Marlon Dumas. « The rise of web service ecosystems ». Anglais. Dans : *IT professional* 8.5 (2006), p. 31-37.

[5] Rudolf Bayer. « Symmetric binary B-trees : Data structure and maintenance algorithms ». Anglais. Dans : *Acta informatica* 1.4 (1972), p. 290-306.

[6] Rudolf Bayer. « The universal B-tree for multidimensional indexing : General concepts ». Anglais. Dans : *International Conference on Worldwide Computing and Its Applications*. Springer. 1997, p. 198-209.

[7] Juan Benet. « IPFS-content addressed, versioned, P2P file system ». Anglais. Dans : *arXiv preprint arXiv :1407.3561* (2014).

[8] Jon Louis Bentley. « Multidimensional binary search trees used for associative searching ». Anglais. Dans : *Communications of the ACM* 18.9 (1975), p. 509-517.

[9] Ames Bielenberg et collab. « The growth of diaspora-a decentralized online social network in the wild ». Anglais. Dans : *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*. IEEE. 2012, p. 13-18.

[10] Manuel Blum et Silvio Micali. « How to generate cryptographically strong sequences of pseudorandom bits ». Anglais. Dans : *SIAM journal on Computing* 13.4 (1984), p. 850-864.

[11] Jakob Borg et les autres contributeurs. « Syncthing ». Anglais. Dans : (2013). URL : https://syncthing.net/ (visité le 02/05/2017).

[12] Nikita Borisov, Ian Goldberg et Eric A. Brewer. « Off-the-record communication, or, why not to use PGP ». Anglais. Dans : *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society, WPES 2004, Washington, DC, USA, October 28, 2004*. 2004, p. 77-84. DOI : 10.1145/1029179.1029200. URL : http://doi.acm.org/10.1145/1029179.1029200.

[13] Eric A. Brewer. « Towards robust distributed systems (abstract) ». Anglais. Dans : *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing, July 16-19, 2000, Portland, Oregon, USA*. Sous la dir. de Gil Neiger. ACM, 2000, p. 7. ISBN : 1-58113-183-6. DOI : 10.1145/343477.343502. URL : https://doi.org/10.1145/343477.343502.

[14] Mike Burmester et Yvo Desmedt. « A secure and scalable Group Key Exchange system ». Anglais. Dans : *Inf. Process. Lett.* 94.3 (2005), p. 137-143. DOI : 10.1016/j.ipl.2005.01.003. URL : https://doi.org/10.1016/j.ipl.2005.01.003.

[15] Maria Nicole Cleis et Oliver Diggelmann. « How the Right to Privacy Became a Human Right ». Anglais. Dans : *Human Rights Law Review* 14.3 (juil. 2014), p. 441-458. ISSN : 1461-7781. DOI : 10.1093/hrlr/ngu014. eprint : http://oup.prod.sis.lan/hrlr/article-pdf/14/3/441/1809024/ngu014.pdf. URL : https://dx.doi.org/10.1093/hrlr/ngu014.

[16] Katriel Cohn-Gordon et collab. « A Formal Security Analysis of the Signal Messaging Protocol ». Anglais. Dans : *IACR Cryptology ePrint Archive* 2016 (2016), p. 1013. URL : http://eprint.iacr.org/2016/1013.

[17] Katriel Cohn-Gordon et collab. *On Ends-to-Ends Encryption : Asynchronous Group Messaging with Strong Security Guarantees*. Anglais. Rapp. tech. IACR Cryptology ePrint Archive 2017 (2017), 666. http ://eprint.iacr. org/2017/666, 2017.

[18] Douglas Comer. « Ubiquitous B-tree ». Anglais. Dans : *ACM Computing Surveys (CSUR)* 11.2 (1979), p. 121-137.

[19] Contributeurs de GnuTLS. *The GnuTLS Transport Layer Security Library*. Anglais. 2018. URL : https://gnutls.org/ (visité le 05/12/2018).

[20] Contributeurs de Tox. « A New Kind of Instant Messaging ». Anglais. Dans : (2013). URL : https://tox.chat/ (visité le 02/05/2017).

[21] Contributeurs de ZeroNet. *Web décentralisé utilisant le réseau BitTorrent et la cryptographie Bitcoin*. 2018. URL : https://zeronet.io/ (visité le 23/12/2018).

[22] Contributeurs du projet Tox. *A New Kind of Instant Messaging*. 2018. URL : https://tox.chat/ (visité le 23/12/2018).

[23] Rene De La Briandais. « File searching using variable length keys ». Anglais. Dans : *Papers presented at the the March 3-5, 1959, western joint computer conference*. ACM. 1959, p. 295-298.

[24] Simon Désaulniers et collab. « Fully Distributed Indexing over a Distributed Hash Table ». Anglais. Dans : *Ubiquitous Networking*. Sous la dir. d'Essaid Sabir et collab. Cham : Springer International Publishing, 2017, p. 308-318. ISBN : 978-3-319-68179-5.

[25] David Dias et Juan Benet. « Distributed Web Applications with IPFS, Tutorial ». Anglais. Dans : *International Conference on Web Engineering*. Springer. 2016, p. 616-619.

[26] Discord Inc. *Politique de confidentialité de Discord*. Anglais. Juin 2018. URL : https://discordapp.com/privacy (visité le 12/04/2019).

[27] Renaud Dumont. « La cryptographie informatique ». Dans : (2010).

[28] Équipe « Les Décodeurs ». « Comment communique-t-on en direct avec l'ISS ? » Dans : *Le Monde* (jan. 2017). URL : https://www.lemonde.fr/les-decodeurs/article/2017/01/13/mais-au-fait-comment-communique-t-on-en-direct-avec-l-iss_5062536_4355770.html (visité le 28/04/2019).

[29] Facebook. *Messenger Secret Conversations*. Anglais. Rapp. tech. Facebook, 2016. URL : https://fbnewsroomus.files.wordpress.com/2016/07/secret_conversations_whitepaper-1.pdf.

[30] Marguerite Fayçal. « Routage efficace pour les réseaux pair-à-pair utilisant des tables de hachage distribuées ». Thèse de doct. Télécom ParisTech, 2010.

[31] Armando Fox et Eric A. Brewer. « Harvest, Yield and Scalable Tolerant Systems ». Anglais. Dans : *Proceedings of The Seventh Workshop on Hot Topics in Operating Systems, HotOS-VII, Rio Rico, Arizona, USA, March 28-30, 1999*. Sous la dir. de Peter Druschel. IEEE Computer Society, 1999, p. 174-178. DOI : 10.1109/HOTOS.1999.798396. URL : https://doi.org/10.1109/HOTOS.1999.798396.

[32] Marius François. « Sur Slack, un patron peut lire les messages privés de ses employés ». Dans : *Le Figaro* (mar. 2018). URL : http://www.lefigaro.fr/secteur/high-tech/2018/03/23/32001-20180323ARTFIG00296-sur-slack-un-patron-peut-lire-les-messages-prives-de-ses-employes.php (visité le 12/04/2019).

[33] Free Software Foundation, Inc. *Guide rapide de la GPLv3 - Projet GNU - Free Software Foundation*. Anglais. 2018. URL : https://www.gnu.org/licenses/quick-guide-gplv3.fr.html (visité le 11/12/2018).

[34] Jun Gao et Peter Steenkiste. « An Adaptive Protocol for Efficient Support of Range Queries in DHT-Based Systems ». Anglais. Dans : *12th IEEE International Conference on Network Protocols (ICNP 2004), 5-8 October 2004, Berlin, Germany*. 2004, p. 239-250. DOI : 10.1109/ICNP.2004.1348114. URL : http://doi.ieeecomputersociety.org/10.1109/ICNP.2004.1348114.

[35] Luis Garcés-Erice et collab. « Data Indexing in Peer-to-Peer DHT Networks. » Anglais. Dans : *ICDCS*. IEEE Computer Society, 2004, p. 200-208.

[36] Samuel Gibbs. « Dropbox hack leads to leaking of 68m user passwords on the internet ». Anglais. Dans : *The Guardian* (2016).

[37] Henri Gilbert et Helena Handschuh. « Security analysis of SHA-256 and sisters ». Anglais. Dans : *International workshop on selected areas in cryptography*. Springer. 2003, p. 175-193.

[38] Ian Goldberg et collab. « Multi-party off-the-record messaging ». Anglais. Dans : *Proceedings of the 16th ACM conference on Computer and communications security*. ACM. 2009, p. 358-368.

[39] Shai Halevi et Hugo Krawczyk. « One-Pass HMQV and Asymmetric Key-Wrapping ». Anglais. Dans : *IACR Cryptology ePrint Archive* 2010 (2010), p. 638. URL : http://eprint.iacr.org/2010/638.

[40] Pascal Hérard. *Cybercriminalité : comment l'Europe veut contourner la confidentialité des communications*. 2017. URL : https://information.tv5monde.com/info/cybercriminalite-comment-l-europe-veut-contourner-la-confidentialite-des-communications-198819.

[41] Alex Hern. *UK government can force encryption removal, but fears losing, experts say*. Anglais. 2017. URL : https://www.theguardian.com/technology/2017/mar/29/uk-government-encryption-whatsapp-investigatory-powers-act.

[42] Michael Herrmann et collab. « Description of the yacy distributed web search engine ». Anglais. Dans : *Technical Report, KU Leuven ESAT/COSIC, IMinds* (2014).

[43] Nicolas Hidalgo et collab. « ECHO : Efficient Complex Query over DHT Overlays ». Anglais. Dans : *J. Parallel Distrib. Comput.* 88 (2016), p. 31-45. DOI : 10.1016/j.jpdc.2015.10.007. URL : http://dx.doi.org/10.1016/j.jpdc.2015.10.007.

[44] Scott Hilton. *Dyn Analysis Summary Of Friday October 21 Attack*. Anglais. Oct. 2016. URL : https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/ (visité le 12/04/2019).

[45] *Information technology – Security techniques – Information security management systems – Overview and vocabulary*. Anglais. Standard. ISO 27000 :2018(E). Geneva, CH : International Organization for Standardization, fév. 2018.

[46] Jonathan Katz et Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Anglais. CRC Press, 2014. ISBN : 9781466570269.

[47] Auguste Kerckhoffs. « La cryptographie militaire ». Dans : *Journal des sciences militaires* (1883), p. 5-83.

[48] Yongdae Kim, Adrian Perrig et Gene Tsudik. « Group key agreement efficient in communication ». Anglais. Dans : *IEEE transactions on computers* 53.7 (2004), p. 905-921.

[49] Yongdae Kim, Adrian Perrig et Gene Tsudik. « Tree-based group key agreement ». Anglais. Dans : *ACM Transactions on Information and System Security (TISSEC)* 7.1 (2004), p. 60-96.

[50] David Kravets. *UK prime minister wants backdoors into messaging apps or he'll ban them*. Anglais. 2015. URL : https://arstechnica.com/tech-policy/2015/01/uk-prime-minister-wants-backdoors-into-messaging-apps-or-hell-ban-them/.

[51] Hugo Krawczyk. « SIGMA : The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE-Protocols ». Anglais. Dans : *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*. Sous la dir. de Dan Boneh. T. 2729. Lecture Notes in Computer Science. Springer, 2003, p. 400-425. ISBN : 3-540-40674-3. DOI : 10.

1007/978-3-540-45146-4_24. URL : https://doi.org/10.1007/978-3-540-45146-4_24.

[52] Leslie Lamport. « Time, clocks, and the ordering of events in a distributed system ». Dans : *Communications of the ACM* 21.7 (1978), p. 558-565.

[53] Leslie Lamport, Robert Shostak et Marshall Pease. « The Byzantine generals problem ». Dans : *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4.3 (1982), p. 382-401.

[54] Laurie Law et collab. « An Efficient Protocol for Authenticated Key Agreement ». Anglais. Dans : *Des. Codes Cryptography* 28.2 (2003), p. 119-134.

[55] Henri Leon Lebesgue. *Leçons sur l'Integration et la Recherche des Fonctions Primitives*. Paris, Gauthier-Villars, 1904.

[56] Hong Liu, Eugene Y Vasserman et Nicholas Hopper. « Improved group off-the-record messaging ». Anglais. Dans : *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. ACM. 2013, p. 249-254.

[57] Eng Keong Lua et collab. « A survey and comparison of peer-to-peer overlay network schemes. » Anglais. Dans : *IEEE Communications Surveys and tutorials* 7.1-4 (2005), p. 72-93.

[58] Moxie Marlinspike. *Forward Secrecy for Asynchronous Messages*. Anglais. 2013. URL : https://whispersystems.org/blog/asynchronous-security/ (visité le 19/07/2018).

[59] Moxie Marlinspike. *Private Group Messaging*. Anglais. 2014. URL : https://whispersystems.org/blog/private-groups/ (visité le 29/07/2017).

[60] Moxie Marlinspike et Trevor Perrin. *Axolotl Ratchet*. Anglais. 2013. URL : https://whispersystems.org/blog/advanced-ratcheting/ (visité le 20/04/2017).

[61] Moxie Marlinspike et Trevor Perrin. *The X3DH Key Agreement Protocol*. Anglais. 2016. URL : https://signal.org/docs/specifications/x3dh (visité le 06/08/2018).

[62] Équipe de Matrix. *Megolm group ratchet*. Anglais. 2019. URL : https://git.matrix.org/git/olm/about/docs/megolm.rst (visité le 18/01/2019).

[63] Équipe de Matrix. *Olm : A Cryptographic Ratchet*. Anglais. 2019. URL : http://git.matrix.org/git/olm/about/docs/olm.rst (visité le 18/01/2019).

[64] Matrix.org. *Home | Matrix.org*. Anglais. 2019. URL : https://matrix.org/blog/home/ (visité le 05/02/2019).

[65] Petar Maymounkov et David Mazieres. « Kademlia : A peer-to-peer information system based on the xor metric ». Anglais. Dans : *International Workshop on Peer-to-Peer Systems*. Springer. 2002, p. 53-65.

[66] Peter Mell, Tim Grance et collab. « The NIST definition of cloud computing ». Anglais. Dans : (2011).

[67] Vinnie Moscaritolo, Gary Belvin et Phil Zimmermann. *Silent Circle Instant Messaging Protocol Protocol Specifcation*. Anglais. 2012. URL : https://web.archive.org/web/20150402122917/https://silentcircle.com/sites/default/themes/silentcircle/assets/downloads/SCIMP_paper.pdf (visité le 03/05/2017).

[68] Francesca Musiani et Cécile Méadel. « "Reclaiming the Internet" with distributed architectures : An introduction ». Anglais. Dans : *First Monday* 21.12 (2016). ISSN : 13960466. DOI : 10.5210/fm.v21i12.7101. URL : https://firstmonday.org/ojs/index.php/fm/article/view/7101.

[69] Satoshi Nakamoto et collab. « Bitcoin : A peer-to-peer electronic cash system ». Dans : (2008).

[70] Jason Nieh, S Jae Yang et Naomi Novik. « A comparison of thin-client computing architectures ». Anglais. Dans : (2000).

[71] Patrick E. O'Neil. « Model 204 Architecture and Performance ». Anglais. Dans : *High Performance Transaction Systems, 2nd International Workshop, Asilomar Conference Center, Pacific Grove, California, USA, September 28-30, 1987, Proceedings*. Sous la dir. de Dieter Gawlick, Mark N. Haynie et Andreas Reuter. T. 359. Lecture Notes in Computer Science. Springer, 1987, p. 40-59. ISBN : 3-540-51085-0. DOI : 10.1007/3-540-51085-0_42. URL : https://doi.org/10.1007/3-540-51085-0%5C_42.

[72] Rolf Oppliger. *SSL and TLS : Theory and Practice*. Anglais. Artech House, 2016.

[73] Guiseppe Peano. *Sur une courbe, qui remplit toute une aire plane*. 1890.

[74] Nuno Preguica et collab. « A commutative replicated data type for cooperative editing ». Dans : *2009 29th IEEE International Conference on Distributed Computing Systems*. IEEE. 2009, p. 395-403.

[75] Radio-Canada. « L'Union européenne approuve la controversée réforme du droit d'auteur sur Internet ». Dans : *Radio-Canada* (2019). URL : https://ici.radio-canada.ca/nouvelle/1123640/reforme-droit-auteur-europe-union-europeene-internet-web-article-11-13 (visité le 26/04/2019).

[76] Radio-Canada. « Mark Zuckerberg témoignera devant le Congrès américain ». Dans : *Radio-Canada* (mar. 2018). URL : https://ici.radio-canada.ca/nouvelle/1091780/scandale-facebook-cambridge-analytica (visité le 28/04/2019).

[77] Sriram Ramabhadran et collab. « Prefix Hash Tree : An Indexing Data Structure over Distributed Hash Tables ». Anglais. Dans : (2004).

[78] Sriram Ramabhadran et collab. « Prefix hash tree : An indexing data structure over distributed hash tables ». Anglais. Dans : *Proceedings of the 23rd ACM symposium on principles of distributed computing*. T. 37. 2004.

[79] Red Hat Enterprise Linux Team. *Red Hat continues to lead the Linux server market*. Anglais. 2018. URL : https://www.redhat.com/en/blog/red-hat-continues-lead-linux-server-market (visité le 24/04/2019).

[80] Riot.im. *Riot – open team collaboration*. Anglais. 2017. URL : https://about.riot.im/about-us/ (visité le 15/09/2018).

[81] Paul Rösler, Christian Mainka et Jörg Schwenk. « More is less : on the end-to-end security of group chats in signal, whatsapp, and threema ». Dans : *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2018, p. 415-429.

[82] Antony Rowstron et Peter Druschel. « Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems ». Anglais. Dans : *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer. 2001, p. 329-350.

[83] Hans Sagan. *Space-Filling Curves*. Anglais. 1st. Springer-Verlag, 1997.

[84] Savoir-faire Linux. *A C++11 Distributed Hash Table implementation*. Anglais. http://opendht.net. 2016.

[85] Savoir-faire Linux. *Ring | Ring gives you a full control over your communications and an unmatched level of privacy*. 2018. URL : https://ring.cx/ (visité le 23/12/2018).

[86] Michael Schliep, Eugene Vasserman et Nicholas Hopper. « Consistent Synchronous Group Off-The-Record Messaging with SYM-GOTR ». Anglais. Dans : *Proceedings on Privacy Enhancing Technologies* 2018.3 (2018), p. 181-202.

[87] Claude E Shannon. « Communication theory of secrecy systems ». Anglais. Dans : *Bell system technical journal* 28.4 (1949), p. 656-715.

[88] Marc Shapiro et collab. « Convergent and commutative replicated data types ». Dans : *Bulletin-European Association for Theoretical Computer Science* 104 (2011), p. 67-88.

[89] Guobin Shen et collab. *Distributed Segment Tree : A Unified Architecture to Support Range Query and Cover Query*. Anglais. Rapp. tech. MSR-TR-2007-30. Microsoft Research, mar. 2007, p. 23. URL : http://research.microsoft.com/apps/pubs/default.aspx?id=70419.

[90] Ion Stoica et collab. « Chord : a scalable peer-to-peer lookup protocol for internet applications ». Anglais. Dans : *IEEE/ACM Transactions on Networking (TON)* 11.1 (2003), p. 17-32.

[91] Ion Stoica et collab. « Chord : A scalable peer-to-peer lookup service for internet applications ». Anglais. Dans : *ACM SIGCOMM Computer Communication Review* 31.4 (2001), p. 149-160.

[92] Yuzhe Tang et collab. « m-LIGHT : Indexing Multi-Dimensional Data over DHTs ». Anglais. Dans : *29th IEEE International Conference on Distributed Computing Systems (ICDCS 2009), 22-26 June 2009, Montreal, Québec, Canada*. IEEE Computer Society, 2009, p. 191-198. ISBN : 978-0-7695-3659-0. DOI : 10.1109/ICDCS.2009.30. URL : http://dx.doi.org/10.1109/ICDCS.2009.30.

[93] The Solid Project. *Solid*. Anglais. 2017. URL : https://solid.mit.edu/ (visité le 18/01/2019).

[94] Moxie Marlinspike Trevor Perrin. *The Double Ratchet Algorithm*. Anglais. 2016. URL : https://signal.org/docs/specifications/doubleratchet/ (visité le 26/08/2018).

[95] Nik Unger et collab. « SoK : secure messaging ». Anglais. Dans : *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE. 2015, p. 232-249.

[96] Ruben Verborgh. *Re-decentralizing the Web, for good this time*. Anglais. 2019. URL : https://ruben.verborgh.org/articles/redecentralizing-the-web/ (visité le 18/01/2019).

[97] Sebastian R. Verschoor et Tanja Lange. « (In-)Secure messaging with the Silent Circle instant messaging protocol ». Anglais. Dans : *IACR Cryptology ePrint Archive* 2016 (2016), p. 703. URL : http://eprint.iacr.org/2016/703.

[98] Jane Wakefield. « eBay faces investigations over massive data breach ». Anglais. Dans : *BBC News* (mai 2014).

[99] Nicholas Watt et Patrick Wintour. *David Cameron seeks cooperation of US president over encryption crackdown*. Anglais. 2015. URL : https://www.theguardian.com/uk-news/2015/jan/15/david-cameron-ask-us-barack-obama-help-tracking-islamist-extremists-online.

[100] Peter Weiner. « Linear pattern matching algorithms ». Anglais. Dans : *Switching and Automata Theory, 1973. SWAT'08. IEEE Conference Record of 14th Annual Symposium on*. IEEE. 1973, p. 1-11.

[101] Aaron Weiss. « Computing in the clouds ». Anglais. Dans : *networker* 11.4 (2007), p. 16-25.

[102] WhatsApp. *WhatsApp Encryption Overview*. Anglais. Rapp. tech. WhatsApp, 2016. URL : https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf (visité le 17/06/2018).

[103] Whitfield Diffie et Martin E. Hellman. « New directions in cryptography ». Anglais. Dans : *IEEE Trans. Information Theory* 22.6 (1976), p. 644-654. DOI : 10.1109/TIT.1976.1055638. URL : http://dx.doi.org/10.1109/TIT.1976.1055638.

[104] Alma Whitten et J Doug Tygar. « Why Johnny Can't Encrypt : A Usability Evaluation of PGP 5.0. » Anglais. Dans : *USENIX Security Symposium*. T. 348. 1999.

[105] Gavin Wood et collab. « Ethereum : A secure decentralised generalised transaction ledger ». Dans : *Ethereum project yellow paper* 151 (2014), p. 1-32.

[106] Philip Zimmermann. « A Proposed Standard Format for RSA Cryptosystems ». Anglais. Dans : *IEEE Computer* 19.9 (1986), p. 21-34. DOI : 10.1109/MC.1986.1663326. URL : http://dx.doi.org/10.1109/MC.1986.1663326.

[107] Shoshana Zuboff. *The age of surveillance capitalism : the fight for the future at the new frontier of power*. Profile Books, 2019.

INDEX

Symboles

A