MTU2E

Cryptographie

Conception d'une messagerie sécurisée, basée sur le chiffrement RSA

Marijan Stajic, Thomas Petermann, Cédric Bonda Belo,
Tibô Ferreira de Carvalho
16/01/2026

Résumé opérationnel

De nos jours, avec des applications de messagerie comme WhatsApp, Signal ou Telegram, ainsi que des services comme les transactions bancaires en ligne, la confidentialité des informations est devenue primordiale. Cette préoccupation est loin d'être nouvelle. En effet, depuis plusieurs siècles, la protection des communications sensibles a toujours été un enjeu majeur. Nous échangeons quotidiennement des données sensibles, conversations privées, documents, sans vraiment comprendre les mécanismes qui les protègent, ni les enjeux politiques et sociétaux qui ont façonné le monde du chiffrement numérique.

Pour le grand public, la cryptographie reste un concept abstrait. Même si certaines personnes ont déjà entendu parler de « chiffrement asymétrique » ou de « clés », peu comprennent réellement ce qui se cache derrière ces termes techniques. Comment fonctionne concrètement un système de clés ? Pourquoi les experts parlent-ils de clés « publiques » et « privées » ? Quels défis historiques ont permis d'atteindre le niveau de sécurité dont nous bénéficions, et quelles sont les problématiques actuelles ?

Ce projet explore ces questions à travers une recherche théorique et historique sur les fondements de la cryptographie moderne, ainsi qu'au développement d'une application de messagerie mettant en pratique les principes mathématiques de l'algorithme RSA.

Table des matières

Introduction

Présentation du domaine

Ce travail s'inscrit dans le domaine de la cryptographie, une branche scientifique qui étudie la protection de l'information en garantissant que seules les personnes autorisées y aient accès. Elle vise à assurer trois objectifs fondamentaux qui sont la confidentialité, l'authenticité et l'intégrité des données.

Depuis l'Antiquité, on utilisait déjà la cryptographie avec le chiffrement symétrique. Ce système fonctionnait avec une clé secrète que les deux personnes qui communiquaient devaient connaître. Avec l'arrivée de l'ère informatique, le chiffrement a beaucoup évolué et c'est devenu quelque chose de crucial à l'échelle mondiale.

Dans notre projet, on va surtout se concentrer sur la cryptographie moderne, en particulier le chiffrement asymétrique qui a été créé dans les années 1970. Nous explorons les bases de la cryptographie, comment le chiffrement asymétrique est apparu, comment l'algorithme RSA a été inventé, Comment fonctionne-t-il, à quoi il sert aujourd'hui et quels sont les problèmes actuels autour de cette technologie.

Ce projet s'adresse principalement aux étudiants et enseignants à la recherche d'une ressource introductive au domaine, ainsi qu'à toute personne souhaitant découvrir la cryptographie.

Annonce du projet et présentation des objectifs

Nous avons conçu un système de messagerie chiffrée basé sur l'algorithme RSA, permettant à deux ordinateurs d'échanger des informations en garantissant les trois objectifs principaux des communications.

Les objectifs majeurs de ce projet sont les suivants :

1. Effectuer une recherche historique sur la cryptographie moderne ;
2. Maîtriser les fondements mathématiques de RSA ;
3. Développer l'application de messagerie chiffrée avec implémentation complète du chiffrement RSA ;
4. Déployer le système sur trois appareils interconnectés et valider le bon fonctionnement des échanges sécurisés.

Méthode

Disciplines impliquées

Les deux disciplines sélectionnées sont les mathématiques et l'histoire.

L'algorithme RSA étant une application directe des mathématiques, cette discipline s'imposait naturellement comme branche majeure du projet. L'histoire, branche mineure du travail, permet de contextualiser et de situer le sujet dans son évolution temporelle.

Outils spécifiques utilisés pour chaque discipline

D'un point de vue mathématique, nous avons eu recours à plusieurs notions. Notamment, avec la théorie des nombres premiers pour la génération des clés, les propriétés de l'arithmétique modulaire pour le chiffrement et le déchiffrement des messages, les algorithmes d'Euclide simple et étendu qui sont au cœur du fonctionnement de RSA, ainsi que des concepts élémentaires tels que le PGCD.

Concernant l'histoire, nous avons dû appliquer une méthodologie solide pour la recherche de sources et d'informations. En effet, il a été essentiel de nous documenter dans un premier temps sur les fondamentaux de la cryptographie, avant de nous attaquer à l'ère de la cryptographie moderne. Nous avons principalement utilisé un ouvrage retraçant l'histoire du chiffrement à clé publique, que nous avons complété par des sources secondaires afin d'acquérir une compréhension approfondie du sujet, ainsi que par des sources primaires telles que des rapports et documents d'époque.

Gestion du projet

Définition des objectifs et planification

Pour la définition des objectifs, nous n'avons pas utilisé de méthode comme la méthode SMART. Nous avons plutôt adopté une approche pragmatique en procédant par étapes successives. Dans un premier temps, nous avons réfléchi au domaine qui nous intéressait (la cryptographie), puis nous avons affiné notre focus vers un sujet précis (le chiffrement à clé publique à l'ère du numérique). À partir de là, nous avons déterminé le produit que nous souhaitions concevoir (une messagerie chiffrée) et identifié les disciplines pertinentes en fonction de nos compétences.

Cette démarche nous a permis de définir les différents jalons présents dans notre planification, qui sont les fondations mathématiques et historiques du chiffrement, le développement logiciel, l'électronique, ainsi que les systèmes et l'intégration. Ceci nous a permis de définir clairement les quatre objectifs généraux cités dans la section « Annonce du projet et présentation des objectifs ».

Une fois ces jalons établis, nous avons créé des objectifs plus précis pour chacune de ces phases, permettant ainsi une progression structurée du projet.

Le premier objectif spécifique comportait notamment la compréhension des divers systèmes de chiffrement, l'étude et la formalisation mathématique ainsi que la recherche et la synthèse historique. Au niveau de la planification, il s'agissait des premières tâches à réaliser, jusqu'aux vacances d'octobre, afin de pouvoir démarrer le développement du produit sur de bonnes bases. Bien entendu, ces parties ont continué à être améliorées tout au long du projet.

Un autre objectif marquait le début d'une nouvelle étape : créer l'architecture de l'application. Il fallait réfléchir à comment le système allait fonctionner dans son ensemble pour que tout soit cohérent techniquement et pour savoir comment avancer dans le développement.

S'ensuivait le développement de l'application. Le but était de transformer l'algorithme en code informatique et de créer le système de messagerie sécurisée.

En même temps, on devait aussi s'occuper de la conception des boîtiers et installer les systèmes d'exploitation pour y mettre une première version de l'application. On voulait avoir cette première version fonctionnelle pour mi-novembre.

Une fois la première version réalisée, nous avons poursuivi les travaux en nous concentrant principalement sur trois axes principaux qui sont le développement pour améliorer l'application, l'électronique pour créer des boîtiers au design plus abouti, ainsi que l'intégration du système pour optimiser son fonctionnement, jusqu'à l'obtention du produit final.

La planification, tout comme la définition des objectifs, repose donc sur une approche de gestion de projet relativement classique, avec une durée définie, des jalons clairs et des échéances imposées.

Plateforme

Nous avons employé diverses plateformes, qui ont fait office d'outils pour la réalisation de ce travail.

L'outil central de notre organisation a été Notion, plateforme de collaboration et de gestion. Cette solution nous a permis de centraliser l'ensemble des aspects organisationnels du projet, comme notamment le journal de bord, la planification avec la gestion des échéances, une liste de tâches distribuées entre les membres de l'équipe, le suivi du nombre d'heures consacrées à chaque activité, un inventaire du matériel nécessaire, ainsi que diverses statistiques liées à l'avancement du projet. Cette application a facilité la coordination au sein du groupe tout au long du travail, et a été partagée avec notre coach afin qu'il puisse suivre en temps réel l'évolution de notre projet.

Pour structurer nos recherches, nous avons employé Zotero, un gestionnaire de références bibliographiques permettant de répertorier les différentes sources et de les intégrer directement dans notre bibliographie selon le format approprié. Cet outil nous a également facilité la citation des auteurs dans le corps du texte.

Nous avons également utilisé d'autres applications comme Miro pour créer des cartes mentales ou encore Obsidian pour structurer nos idées issues de séances de brainstorming, mais ces outils sont restés secondaires et ont été moins exploités dans le cadre du projet.

Analyse et gestion des risques

L'analyse des risques a été réalisée en début de projet, comme l'imposait le calendrier des rendus de ce travail. À partir des jalons et des objectifs spécifiques définis au préalable dans notre planification, nous avons énuméré ces derniers, résumé en quoi consistait chacun, identifié le responsable de chaque partie et anticipé les éventuelles difficultés en cas de problème.

Ainsi, face aux imprévus, nous disposions de solutions adaptées, tout en tenant compte du degré d'importance de chaque risque défini à l'avance, nous permettant donc d'agir à temps en cas de nécessité.

Implication des membres

Le projet portant sur la cryptographie à l'ère du numérique, notre équipe compte trois personnes titulaires d'un CFC en informatique. De plus, avec une personne titulaire d'un CFC d'automaticien, nous avons intégré des défis électroniques au projet afin de lui donner un aspect plus tangible.

Marijan Stajic, spécialisé dans les systèmes informatiques, s'est surtout occupé de concevoir l'architecture de l'application. Comme il a également des compétences dans le développement, il a participé à l'écriture du code. Pour les matières qu'on a choisies, il s'est chargé des maths et de la partie historique.

Thomas Petermann, développeur informatique de profession, était responsable du développement de la messagerie chiffrée. Il a travaillé avec Marijan pour concevoir le logiciel et s'est aussi occupé de la partie mathématique.

Cédric Bonda Belo, automaticien de profession, était en charge de toute la partie électronique et de la conception des boîtiers. Il a aussi travaillé avec Marijan et Thomas pour installer et intégrer le logiciel sur les appareils. En plus, il s'est occupé des maths et de la recherche historique.

Tibô Ferreira De Carvalho est spécialisé en sécurité informatique. Il s'est principalement occupé de la partie histoire du projet et a aidé pour la conception de l'application.

Résultats et analyse

Histoire

La cryptographie : fondamentaux et origines

La cryptographie est une branche de la cryptologie, une science qui englobe tout ce qui touche aux codes secrets (Wikipédia - Cryptologie, 2025). Utilisée depuis l'Antiquité, elle a énormément évolué au fil des siècles. L'objectif de la cryptographie est de protéger les messages en garantissant que seuls les destinataires autorisés peuvent les lire. Elle assure ainsi les trois principes fondamentaux suivants :

- La confidentialité (garder le secret)
- L’authenticité (vérifier l’identité de l’auteur)
- L’intégrité (le message n’a pas été modifié)

Les premiers pas de la cryptographie remontent à l'Antiquité, en Égypte ancienne (vers 1900 av. J.-C.), avec des hiéroglyphes modifiés puis par la scytale (vers 400 av. J.-C.), un bâton autour duquel on enroulait une bande de cuir pour y inscrire des messages secrets (Wikipédia - Cryptographie, 2025).

De nombreuses méthodes de plus en plus complexes ont ensuite vu le jour, notamment avec le chiffre de César (vers 50 av. J.-C.) et son principe de décalage des lettres d'un certain nombre de positions dans l'alphabet. Plus tard, pendant la Seconde Guerre mondiale, la machine Enigma illustre cette évolution, cet appareil électromécanique, utilisé principalement par l'Allemagne nazie, chiffrait les messages grâce à un système de rotors tournants (Wikipédia - Enigma, 2024).

Cependant, même si la cryptographie s'est complexifiée au fil du temps (du simple décalage de lettres jusqu'aux rotors tournants, ce qui rendait ces différentes méthodes de plus en plus difficiles à casser), toutes partageaient la même faiblesse, elles reposaient sur du chiffrement symétrique.

Le principe du chiffrement symétrique repose sur l'utilisation d'une même clé secrète pour chiffrer et déchiffrer les messages (Wikipédia - Cryptographie symétrique, 2025). Par exemple, pour la scytale, il suffisait d'utiliser un bâton du même diamètre. Pour le chiffre de César, il fallait connaître le nombre de décalages de lettres appliqué. Et pour Enigma, il s'agissait de configurer les rotors exactement de la même manière. Tous ces éléments constituent ce que l'on considère comme étant la « clé secrète » (CEA Recherche, 2024).

Le problème majeur de ce système réside dans l'utilisation d'une clé identique des deux côtés, et si cette dernière était interceptée, il devenait alors facilement possible de déchiffrer tous les messages échangés.

Le chiffrement asymétrique, pilier du numérique

Cette problématique avec le chiffrement symétrique donne naissance en 1976 à un nouveau concept, le chiffrement asymétrique (autrement dit la cryptographie à clé publique). Mais alors, quelles sont les différences, et comment une idée purement mathématique, la cryptographie à clé publique, est-elle devenue un pilier fondamental de la sécurité numérique, notamment à travers l'algorithme RSA ?

Pour répondre à cette problématique, nous nous appuierons sur l'ouvrage Histoire des codes secrets de Simon Singh, qui servira de base et de fil conducteur à notre travail de recherche. Ce physicien et journaliste retrace l'évolution de la cryptographie de l'Antiquité à l'ère numérique, en contextualisant notamment l'arrivée de la cryptographie asymétrique et de RSA, tout en expliquant l'ordre chronologique des événements ainsi que les différents enjeux politiques de cette époque.

Après-guerre et l'essor de l'informatique

Pour comprendre l'émergence du chiffrement asymétrique en 1976, il faut d'abord revenir sur le contexte de l'après-guerre et l'essor de l'informatique.

Après la Seconde Guerre mondiale, la cryptographie entre dans une nouvelle ère avec l'arrivée des ordinateurs. Au départ, le cryptage électronique était exclusif aux gouvernements et à l'armée, car c'étaient les seuls à avoir en possession des ordinateurs à cette époque (Singh, 1999, p. 309).

L'une des premières machines était l'ENIAC (Electronic Numerical Integrator and Computer), capable d'effectuer 5000 opérations par seconde. Contrairement aux machines électromécaniques comme Enigma qui utilisaient des rotors physiques, l'ENIAC pouvait simuler l'action de cent rotors grâce à ses circuits électroniques et manipulait directement des nombres binaires, offrant une flexibilité impossible avec des mécanismes mécaniques (Singh, 1999, p. 305-306).

L'invention du transistor en 1947 accélère cette révolution en permettant la miniaturisation de l'informatique. Dans les années 60-70, les ordinateurs deviennent plus puissants et moins coûteux, se répandant ainsi progressivement dans les entreprises (Singh, 1999, p. 309). Cette démocratisation soulève rapidement la question du chiffrement pour le secteur privé.

IBM développe alors Lucifer, un algorithme de chiffrement symétrique. En 1973, le National Bureau of Standards (NBS), l'agence fédérale américaine chargée de développer des standards technologiques, aujourd'hui appelé National Institute of Standards and Technology, recherche un algorithme de chiffrement et repère Lucifer comme candidat. Cependant, le NBS demande à la National Security Agency (NSA), l'agence de renseignement américaine, d'évaluer la sécurité de l'algorithme proposé. La NSA impose alors plusieurs modifications techniques. Lucifer doit être suffisamment fort contre des adversaires ordinaires, mais déchiffrable par ses propres moyens (Burr, 2000, p. 250).

Ces modifications soulèvent immédiatement un questionnement majeur déjà à cette époque. Certains critiques soupçonnent que la NSA a délibérément affaibli, plutôt que renforcé l'algorithme, ou peut-être même introduit une porte dérobée qui permettrait à l'agence de déchiffrer les messages chiffrés (Burr, 2000, p. 251). Autrement dit, la NSA aurait volontairement créé une faille secrète dans le système pour pouvoir espionner les communications civiles. Lucifer est par la suite renommé DES (Data Encryption Standard) et adopté officiellement en novembre 1977. Cet algorithme met en avant la volonté du gouvernement américain de maintenir un certain contrôle sur la cryptographie civile.

Cependant, un autre problème fondamental reste la distribution des clefs. Le chiffrement symétrique, pour rappel, impose que l'expéditeur et le destinataire partagent la même clef secrète, nécessitant un échange préalable sécurisé, un défi considérable qui, selon Singh, « a été le fléau constant des cryptographes au cours de l'histoire » (Singh, 1999, p. 314).

Les pionniers du chiffrement à clé publique

C'est dans ce contexte que Whitfield Diffie, cryptographe indépendant né en 1944, décide de s'attaquer au problème de la distribution des clefs. Singh le décrit comme un personnage hors norme, libre penseur ne servant ni le gouvernement ni aucune grande compagnie (Singh, 1999, p. 315-316).

En 1974, lors d'une conférence au laboratoire d'IBM, Diffie découvre qu'un autre chercheur travaille sur la même question, Martin Hellman, professeur à l'université de Stanford. La rencontre est si importante que Diffie prend immédiatement le volant pour parcourir les 5000 kilomètres qui le séparent de la côte Ouest (Singh, 1999, p. 318).

Un troisième homme les rejoint, Ralph Merkle, chercheur dont le directeur précédent ne croyait pas en la possibilité de résoudre ce problème (Singh, 1999, p. 319).

Le trio se met alors à travailler sur les fonctions à sens unique, des opérations mathématiques faciles à effectuer mais virtuellement impossibles à inverser. Singh illustre dans son livre ce concept avec l'exemple suivant : « Mélanger de la peinture jaune et de la peinture bleue pour obtenir du vert est une fonction à sens unique, puisqu'il est facile de mélanger les peintures et impossible de retrouver les couleurs d'origine » (Singh, 1999, p. 324).

Les premières découvertes et conception

Au printemps 1976, Hellman a une illumination. En à peine une demi-heure de calculs, il prouve que deux interlocuteurs peuvent convenir d'une clef sans jamais se rencontrer (Singh, 1999, p. 336). Le système d'échange de clefs Diffie-Hellman-Merkle qu'il vient d'inventer permet à deux personnes d'établir une clef secrète commune en discutant publiquement, même si quelqu'un écoute leur échange.

Cette découverte est confirmée dans leur rapport officiel « New Directions in Cryptography » (novembre 1976), publié dans les archives académiques de Stanford. Whitfield et Hellman en personne présentent leur « nouveau système de distribution de clés publiques » qui « ne requiert qu'une seule clé à échanger » et où « l'effort cryptanalytique croît exponentiellement par rapport à l'effort des utilisateurs légitimes » (Diffie et Hellman, 1976, p. 649, nous traduisons).

Malgré cette avancée majeure, le système présente des limites. D'une part, les deux personnes doivent impérativement être connectées simultanément pour échanger leurs nombres et établir la clef (Singh, 1999, p. 332-333). D'autre part, Diffie et Hellman reconnaissent eux-mêmes que le « surcoût de transmission élevée empêche le système d'être très utile en pratique » (Diffie et Hellman, 1976, p. 648, nous traduisons), expliquant pourquoi leur invention ne révolutionnera pas immédiatement les communications.

En parallèle, Diffie imagine un peu plus tôt, en 1975, un nouveau type de chiffrement basé sur une clef asymétrique (Singh, 1999, p. 334). Son idée est d'utiliser deux clefs distinctes, l'une pour chiffrer (la clef publique) et l'autre pour déchiffrer (la clef privée). Dans leur rapport, il définit précisément ce système où « le chiffrement et le déchiffrement sont gouvernés par des clés distinctes, E et D, telles que calculer D à partir de E est informatiquement impossible » (Diffie et Hellman, 1976, p. 644, nous traduisons).

Le problème, c'est que Diffie reste bloqué au stade théorique. Il a bien conçu le concept général, mais ne parvient pas à proposer un exemple mathématique concret (Singh, 1999, p. 335). Les auteurs l'admettent d'ailleurs ouvertement dans leur publication une possibilité d'évolution de ce concept : « le problème reste largement ouvert » (Diffie et Hellman, 1976, p. 644, nous traduisons). Face à cette impasse, Diffie décide de publier son concept en 1976, espérant que d'autres chercheurs réussiront là où il a échoué (Singh, 1999, p. 338).

Rivest-Shamir-Adleman

Finalement, c'est deux années plus tard qu'un autre trio de chercheurs y parvient, au Massachusetts Institute of Technology (MIT) (Singh, 1999, p. 338).

En 1977, Ron Rivest, Adi Shamir et Leonard Adleman décident de s'attaquer au concept de Diffie. Les trois hommes se complètent parfaitement, Rivest est quelqu'un de très créatif, Shamir sait identifier rapidement l'essentiel d'un problème, et Adleman possède une rigueur mathématique pour repérer les failles des deux autres (Singh, 1999, p. 339).

Pendant des mois, le trio multiplie les échecs. Puis, c'est dans la nuit du 16 avril 1977 que tout change : « Rivest, Shamir et Adleman passaient la fête de Pâques chez un étudiant, et ils avaient bien arrosé la soirée. Rivest, incapable de fermer l'œil, s'étendit avec un livre de mathématiques » (Singh, 1999, p. 339). Il eut une illumination : « Soudain, les brumes se dissipèrent autour de lui, et il eut une inspiration. Il passa la fin de la nuit à formaliser son idée, et à l'aube il avait rédigé tout un article » (Singh, 1999, p. 339-340).

Le lendemain, Adleman vérifie le travail de Rivest mais ne trouve aucune faille. Le système RSA venait de naître et allait devenir le chiffre le plus influent de la cryptographie moderne (Singh, 1999, p. 340). RSA fut présenté publiquement en février 1978 dans un article du Scientific American.

Dans leur article scientifique publié en février 1978 intitulé « A Method for Obtaining Digital Signatures and Public-Key Cryptosystems », Rivest, Shamir et Adleman du MIT présentent la première implémentation concrète du concept imaginé par Diffie deux ans plus tôt.

Cet article décrit comment « révéler publiquement une clef de chiffrement ne révèle pas pour autant la clef de déchiffrement correspondante » (Rivest, Shamir & Adleman, 1978, p. 1, nous traduisons). Les auteurs démontrent mathématiquement que cette asymétrie résout deux problèmes importants. Elle élimine le besoin de « courriers ou autres moyens sécurisés pour transmettre les clefs », et elle permet de « signer » numériquement des messages de manière infalsifiable (Rivest, Shamir & Adleman, 1978, p. 1, nous traduisons).

L’ensemble de leur solution réside dans le fait que de multiplier deux grands nombres premiers est facile, mais retrouver ces nombres à partir du résultat est extrêmement difficile. Cette asymétrie mathématique crée exactement la fonction à sens unique que Diffie cherchait. Concrètement, chaque personne peut publier librement un grand nombre (sa clef publique) que n'importe qui peut utiliser pour lui envoyer des messages chiffrés, mais seule cette personne connaît les deux nombres premiers secrets qui ont servi à créer ce grand nombre (sa clef privée), ce qui lui permet de déchiffrer.

L'article de Rivest, Shamir et Adleman marque l'aboutissement d'une quête mathématique lancée par Diffie et Hellman en 1976, et répond directement à une problématique que possède le DES « Toutes les méthodes de chiffrement classiques (y compris le standard NBS) souffrent du "problème de distribution des clefs" » (Rivest, Shamir & Adleman, 1978, p. 3, nous traduisons).

Diffie, Hellman et Merkle avaient imaginé le concept, Rivest, Shamir et Adleman venaient de le concrétiser.

GCHQ

L'histoire de la découverte du chiffrement asymétrique cache une autre vérité. Selon le gouvernement britannique, ces concepts auraient été inventés au Government Communications Headquarters (GCHQ) de Cheltenham, l'établissement top secret fondé sur les ruines de Bletchley Park (Singh, 1999, p. 348-349).

Fin 1969, l'armée britannique charge James Ellis de résoudre le problème de distribution des clefs. Ellis découvre le concept de cryptographie à clef publique, plusieurs années avant Diffie. Mais n'étant pas mathématicien, il ne peut trouver de fonction concrète. Pendant trois ans, les meilleurs cerveaux du GCHQ échouent également (Singh, 1999, p. 353).

En septembre 1973, Clifford Cocks, jeune diplômé de Cambridge, rejoint le GCHQ. Six semaines après son arrivée, on lui présente l'idée d'Ellis. Il trouve la solution en une demi-heure, il venait d'inventer RSA, quatre ans avant le trio du MIT (Singh, 1999, p. 354).

Début 1974, Malcolm Williamson rejoint le GCHQ. Il tente de prouver que la découverte de Cocks est fausse. Mais en cherchant l'erreur, il découvre l'échange de clefs Diffie-Hellman-Merkle, deux ans avant Hellman (Singh, 1999, p. 357).

En 1975, Ellis, Cocks et Williamson avaient découvert tous les aspects de la cryptographie à clef publique. Mais le secret défense les obligea au silence (Singh, 1999, p. 355, 357).

Ces découvertes ne furent révélées qu'en décembre 1997 (Singh, 1999, p. 362). Cependant, des rumeurs circulaient déjà depuis les années 1990 entre le GCHQ et la NSA. Lors d'une visite de Whitfield Diffie, qui était impliqué indirectement à la NSA suite à sa découverte, James Ellis lui avait confié « Je ne sais pas combien je devrais en dire. Laissez-moi juste dire que vous en avez fait beaucoup plus que nous » (Buchanan, 2024). Par ces mots, Ellis reconnaissait que si le GCHQ avait découvert les concepts mathématiques en premier, ce sont les chercheurs américains qui avaient compris leur potentiel et les avaient rendus accessibles au monde entier, révolutionnant ainsi le monde de la cryptographie.

Le chiffrement pour tous

Dans les années 80, RSA reste un outil réservé aux grandes organisations. Il offrait une protection contre la surveillance de masse, cependant, le processus de chiffrement RSA demandait une puissance de calcul beaucoup trop importante pour les ordinateurs de l'époque (Singh, 1999, p. 371). Seuls l'État, l'armée et les grandes entreprises possèdent des machines assez puissantes pour l'utiliser efficacement.

C'est là qu'intervient Phil Zimmermann, militant pour la vie privée convaincu que tout le monde a droit à la confidentialité offerte par le chiffrement (Singh, 1999, p. 372). Dans l'article « Why I Wrote PGP », rédigé en 1991 par Zimmermann en personne, il explique les motivations qui l'ont poussé à créer ce logiciel.

Dedans, il défend le droit fondamental à la vie privée « C'est personnel. C'est privé. Et cela ne regarde personne d'autre que vous ». Il ajoute également que deux siècles auparavant, « toutes les conversations étaient privées. Si quelqu'un d'autre était à portée de voix, vous pouviez simplement aller derrière la grange et avoir votre conversation là-bas ». Mais aujourd'hui, « l'e-mail peut être scanné de manière routinière et automatique pour détecter des mots-clés intéressants, à une échelle vaste, sans détection ». Il pose également une question pertinente « Si vous êtes vraiment un citoyen respectueux des lois sans rien à cacher, alors pourquoi n'envoyez-vous pas toujours votre courrier papier sur des cartes postales ? » (Phil Zimmermann, 1991, nous traduisions).

La raison qui l'a poussé à agir et à se lancer dans le développement d'un logiciel accessible à tous, qu'il nomme Pretty Good Privacy, ou PGP, est le Senate Bill 266 de 1991, qui aurait obligé les fabricants à insérer des portes dérobées dans leurs produits pour permettre au gouvernement de lire les messages chiffrés (Phil Zimmermann, 1991).

Son objectif est de créer un produit de chiffrement fonctionnant sur un ordinateur personnel. Pour accélérer le chiffrement, Zimmermann combine RSA (lent mais sûr pour échanger des clefs) avec un chiffrement symétrique. Le message est chiffré rapidement avec un algorithme symétrique, tandis que seule la petite clef symétrique est chiffrée avec RSA (Singh, 1999, p. 373).

En juin de cette même année 1991, Zimmermann diffuse PGP gratuitement sur Internet (Singh, 1999, p. 377). Le succès est immédiat. PGP se répand rapidement et trouve des centaines de milliers d'utilisateurs. Mais cette diffusion attire l'attention des autorités. Zimmermann fait face à deux problèmes juridiques majeurs. D'abord, RSA Data Security Inc. l'accuse de piraterie car RSA est breveté. Ensuite, en février 1993, deux enquêteurs officiels lui rendent visite pour une accusation bien plus grave, qui est son implication dans l'exportation illégale d'armement (Singh, 1999, p. 378-379).

L'affaire PGP déclenche un combat. D'un côté, le FBI et la NSA craignent que les criminels et terroristes utilisent PGP pour préparer leurs coups de manière anonyme. De l'autre, les défenseurs des libertés civiles considèrent le chiffrement comme un droit fondamental pour tous (Singh, 1999, p. 379-383). Le gouvernement américain tente de trouver des compromis, notamment avec une nouvelle technologie de chiffrement où les utilisateurs confieraient une copie de leur clef privée à des autorités de confiance, afin de permettre le contrôle en cas d'extrême nécessité pour des enquêtes. Mais ces initiatives échouent face à l'opposition des défenseurs des libertés (Singh, 1999, p. 388-390).

En 1996, après trois ans d'enquête, le gouvernement abandonne ses poursuites contre Zimmermann. PGP s'est déjà trop répandu sur Internet, et poursuivre son créateur ne changerait rien et risquerait de créer un procès inutile. Zimmermann obtient également un accord avec RSA Data Security Inc. concernant le brevet (Singh, 1999, p. 395).

D'ailleurs, en 1999, Phil Zimmermann a mis à jour son article, où il célèbre sa victoire sur les contrôles d'exportation après des années de lutte en inscrivant « PGP permet aux gens de prendre leur vie privée en main. Il y a eu un besoin social croissant pour cela. C'est pourquoi je l'ai écrit » (Phil Zimmerman, 1991, nous traduisions).

PGP représente ainsi le premier exemple d'utilisation grand public du chiffrement asymétrique, démocratisant la cryptographie forte pour tous.

Le chiffrement d’aujourd’hui

Mais alors, comment le chiffrement asymétrique a-t-il évolué et quelle place prend-il dans nos applications du quotidien ? Et RSA, l'algorithme de chiffrement que nous avons étudié mathématiquement ainsi qu’historiquement et à partir duquel nous avons créé notre produit pour ce TIP, qu'en est-il ?

En fait, Phil Zimmermann, en plus d'avoir réussi à créer un outil de chiffrement avec PGP, a indirectement révolutionné le monde de la cryptographie moderne en apportant cette solution combinant le chiffrement symétrique et asymétrique. Comme expliqué plus haut, pour accélérer le chiffrement, Zimmermann a combiné RSA, un algorithme lent mais sûr pour l’échange de clé, avec un chiffrement symétrique. Le message est donc rapidement chiffré avec un algorithme symétrique, tandis que seule la clé symétrique est chiffrée avec RSA (Singh, 1999, p. 373).

Ceci a donné naissance à la méthode utilisée aujourd'hui, qu'on appelle également le chiffrement hybride (Goffinet, 2021), et qui est utilisée tant sur les applications de messagerie comme WhatsApp ou Telegram, que sur les sites internet avec des protocoles de sécurité comme HTTPS (DIGICERT, 2025).

Concernant l'algorithme RSA, il reste toujours l'un des piliers de la cryptographie asymétrique. En effet, il continue à être largement utilisé, notamment dans des protocoles de sécurité comme les certificats numériques pour les sites web. Cependant, des clés de plus en plus longues sont nécessaires pour maintenir un niveau de sécurité suffisant, ce qui rend RSA de plus en plus lent. Ainsi, de nouveaux algorithmes asymétriques ont vu le jour, venant remplacer ce dernier pour certaines utilisations (courbes elliptiques). De plus, avec l'arrivée prochaine des ordinateurs quantiques, RSA est menacé. En effet, il pourrait être cassé assez rapidement par un ordinateur quantique suffisamment puissant. Malgré tout cela, RSA n'a pas disparu et continue d'être largement utilisé (Developpez.com, 2025).

Pour notre produit de messagerie chiffré, nous avons utilisé uniquement le chiffrement asymétrique, donc RSA, pour créer des clés et chiffrer les messages. Comme expliqué ci-dessus, cette pratique n'est donc pas la plus optimale, en raison de la lenteur de RSA avec les longues chaînes de caractères (messages). Il aurait donc fallu combiner ceci avec un chiffrement symétrique, comme DES, ce qui nous aurait demandé beaucoup plus de temps, car il aurait fallu étudier les mathématiques derrière cet algorithme, ainsi que de mettre en œuvre d’un point de vue développement pour notre application, le tout combiné avec RSA.

Les défis du chiffrement moderne

De nos jours, deux menaces majeures sont présentes. D'une part, comme mentionné plus haut, l'arrivée imminente des ordinateurs quantiques pourrait remettre en cause l'ensemble de notre sécurité numérique. RSA pourrait être cassé en une semaine par un ordinateur quantique suffisamment puissant (Developpez.com, 2025).

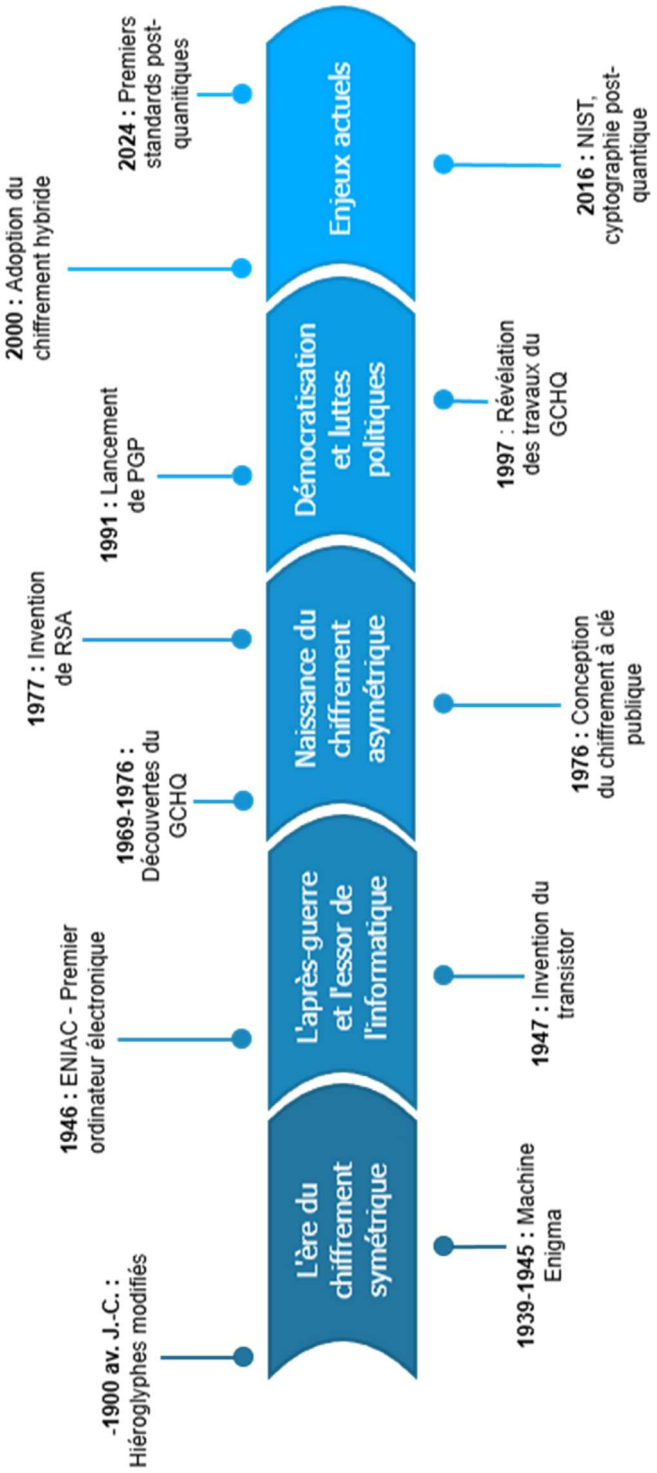
Face à cette menace, le NIST a lancé dès 2016 un concours international pour développer des algorithmes de cryptographie post-quantique, résistants à ces futures machines. Les premiers standards ont été publiés en 2024, mais la transition vers ces nouveaux systèmes représente un défi colossal. En effet, il faudra remplacer RSA dans des milliards d'appareils et de systèmes à travers le monde (NIST, 2024).

D'autre part, il y a toujours les enjeux politiques autour du chiffrement. Comme dans les années 1990 avec l'affaire PGP, les gouvernements cherchent aujourd'hui à contrôler l'accès au chiffrement fort. En Europe, plusieurs propositions législatives visent à affaiblir le chiffrement de bout en bout des applications de messagerie, afin de lutter contre les criminels. Les défenseurs des libertés civiles y voient une atteinte aux droits fondamentaux, rappelant les arguments de Phil Zimmermann il y a plus de trente ans (MACGENERATION, 2025).

Finalement, près de cinquante ans après la révolution du chiffrement asymétrique, la question reste la même. Comment peut-on garantir la sécurité des communications et le respect de la vie privée, tout en répondant aux préoccupations légitimes des États ? L'histoire se répète, mais les enjeux deviennent de plus en plus colossaux, vu que le numérique prend une place bien plus conséquente dans nos vies qu'il y a 40 ans.

Frise chronologique

Ci-dessous, une frise chronologique retracant les éléments les plus marquants de chaque période expliquée tout au long du travail d'histoire, afin d'apporter une vision plus globale sur le travail de recherche effectué.



Mathématique

Définition du chiffrement RSA

En 1977, trois membres du MIT (Massachusetts Institute of Technology, université américaine) ont inventé le chiffrement RSA. Le nom RSA provient d'ailleurs des initiales de leurs noms de famille : Ron Rivest, Adi Shamir et Leonard Adleman.

Ils ont conçu cet algorithme en s'appuyant sur les recherches de Whitfield Diffie et Martin Hellman, qui ont posé le principe du chiffrement asymétrique.

Le chiffrement RSA repose donc sur deux clés mathématiquement liées :

- Une clé publique (utilisée pour chiffrer les messages) ;
- Une clé privée (utilisée pour les déchiffrer).

L'algorithme se décompose ainsi en deux parties principales :

1. La génération des clés publiques et privées
2. Le chiffrement et déchiffrement des messages

Le tout s'appuie sur plusieurs notions mathématiques, qui étaient à l'origine perçues comme abstraites : les nombres premiers, le modulo, la fonction d'Euler, la notion d'inverse modulaire, et bien d'autres encore.

Afin de réaliser l'étude mathématique, les principales sources utilisées sont « The Mathematics of the RSA Public-Key Cryptosystem » de Burt Kaliski et « Asymmetric Cryptography Standards » d'Alexandre Duc. Ces documents présentent les fondements de RSA et détaillent les différentes étapes mathématiques de l'algorithme.

Avantages et désavantages

L'un des principaux avantages de l'algorithme RSA est qu'il résout le problème d'échange de clé initial. En effet, on ne communique jamais la clé privée, qui permet de déchiffrer les messages, mais uniquement la clé publique. De plus, RSA permet d'assurer l'authentification en plus de la confidentialité, notamment grâce à la possibilité de signer numériquement les messages. Aujourd'hui, cet algorithme est utilisé dans de nombreux protocoles Internet, tels que les certificats numériques et HTTPS, qui garantissent la sécurité des sites web.

Cependant, le chiffrement RSA présente aussi certaines faiblesses. Pour garantir un bon niveau de sécurité, il faut utiliser des clés très longues, ce qui rend l'algorithme plus lent que les algorithmes symétriques. Ainsi, il n'est pas optimal pour chiffrer directement de gros volumes de données, car cela prendrait trop de temps. De plus, RSA est vulnérable aux ordinateurs quantiques, qui pourraient un jour le casser en factorisant rapidement les grands nombres sur lesquels repose sa sécurité.

Pour ce travail de maturité, nous allons utiliser RSA pour chiffrer les messages, ce qui n'est pas la méthode la plus recommandée en pratique. Idéalement, il aurait fallu étudier un algorithme de chiffrement symétrique, comme DES, et mettre en place un chiffrement hybride combinant RSA et DES, afin d'obtenir une messagerie à la fois solide et efficace sur le plan de la sécurité. Mais par manque de connaissance et de temps, nous allons limiter notre travail sur RSA.

Développement de la messagerie chiffrée

L'objectif de ce travail d'étude et de recherche en mathématiques est de comprendre les mécanismes mathématiques qui sont derrière l'algorithme RSA, afin d'être capables, à notre tour, de créer une application de messagerie chiffrée utilisant RSA.

Dans chacune des parties, en plus des explications théoriques et des exemples, une partie du code sera détaillée. L'intégralité du code est disponible sur le GitHub du projet (voir annexe), qui constitue le dépôt officiel du travail.

Les fichiers de code concernés sont :

- **flask\rsa_keys.py :** contient les fonctions et la logique principale liées à la génération des clés ;
- **flask\rsa_codec.py :** contient les fonctions et la logique principale liées au chiffrement et déchiffrement des messages ;
- **flask\rsa_maths.py :** contient des fonctions utilisées par rsa_keys.py et rsa_codec.py pour compléter le fonctionnement, notamment la génération de nombres premiers et d'autres opérations mathématiques.

Pour les fonctions telles que la génération des nombres premiers, nous n'avons pas développé le code nous-mêmes.

En effet, nous nous sommes appuyés sur du code existant issu de ressources pédagogiques en ligne, notamment du site GeeksforGeeks, pour l'implémentation de l'algorithme RSA. Par ailleurs, nous avons utilisé des bibliothèques Python déjà disponibles afin de simplifier et fiabiliser le développement.

Génération des clés publiques et privées

La génération des clés publique et privée consiste à créer deux clés liées entre elles, qui permettront de chiffrer et de déchiffrer les messages de manière sûre.

Tout au long des étapes décrites ci-dessous, une explication ainsi qu'un exemple mathématique seront présentés.

Étape 1 : Sélection de deux grands nombres premiers p et q

On choisit deux nombres premiers (ici p et q), car ce sont des nombres qui garantissent que leur produit soit difficile à décomposer en facteurs. Ici, p et q sont utilisées pour les calculs de l'algorithme.

Ces deux nombres ne sont pas la clé privée à eux seuls, mais ils permettent de la calculer. C'est pour cela qu'ils doivent rester secrets.

Ils sont également grands, afin de rendre cette décomposition impossible. En effet, Si p et q étaient petits, un ordinateur pourrait rapidement tester toutes les combinaisons pour retrouver les facteurs. C'est pour cette raison que, dans l'algorithme RSA, on choisit deux grands nombres premiers différents, p et q.

> **Qu'est-ce qu'un nombre premier ?**
>
> Un nombre premier est un entier naturel supérieur à 1 qui n'a que deux diviseurs : 1 et lui-même.
>
> Exemple : 2, 3, 5, 7, 11, 13, 17…
>
> **Qu'est-ce que c'est un algorithme ?** Suite d'opérations permettant d'obtenir un résultat, à l'image d'une recette de cuisine ou d'un itinéraire.

Ici, nous allons choisir deux nombres premiers (en l'occurrence des petits, afin de rendre l'exemple plus clair).

$$p = 19$$

$$q = 23$$

Dans un cas pratique, il est important de choisir des nombres suffisamment grands. Aujourd'hui, on cherche des nombres premiers assez grands pour obtenir un modulo n d'au minimum 2048 bits.

> **Qu'est-ce qu'un bit ?**
>
> Un bit est la plus petite unité d'information en informatique (valeur 0 ou 1). Dire qu'un nombre fait 2048 bits signifie qu'il peut être représenté par 2048 chiffres binaires, soit un nombre d'environ 617 chiffres décimaux.

Dans notre code, la génération de p et q se fait aux lignes suivantes :

```
64    p = rsa_math.generate_prime(half_bits)
65    q = rsa_math.generate_prime(half_bits)
66    while q == p:
67        q = rsa_math.generate_prime(half_bits)
```

rsa_keys.py

Ici, les nombres p et q sont générés en appelant la fonction **generate_prime()** du fichier **rsa_maths.py**, chargée de générer des nombres premiers. La taille en bits demandée est divisée par deux afin de générer deux nombres premiers de même taille.

La boucle « while » garantit que p et q sont bien distincts, condition indispensable au bon fonctionnement de l'algorithme.

Étape 2 : Calcul du modulo n

Une fois que p et q sont choisis, on va calculer le nombre n, aussi appelé le modulo.

Ce dernier fait partie de la clé publique et privée, il est utilisé dans tous les calculs de chiffrement et de déchiffrement.

Il est important de comprendre que RSA fonctionne dans un monde modulaire. Cela signifie que lorsque l'on effectue un calcul et que l'on dépasse une certaine limite (ici n), on revient au début.

Par exemple, sur une horloge, lorsqu'il est 10h, et que l'on ajoute 5 heures, on n'arrive pas à 15h, mais à 3h, car après 12, on revient à 1.

> **Qu'est-ce le modulo ?**
>
> Le modulo permet de donner le restant d'une division.
>
> Exemple : $14\ mod\ 5\ =\ 4 \rightarrow 14\ \div\ 5\ =\ 2 \rightarrow$ Il reste donc 4

Les deux nombres étant définis, on va maintenant les multiplier afin d'obtenir le modulo n :

$$n\ =\ p \cdot q\ =\ 19 \cdot 23\ =\ 437$$

On retrouve ce calcul dans notre code :

```
69    n = p * q
```

rsa_keys.py

n est donc la variable qui va venir conserver la valeur de notre modulo n.

Étape 3 : Calcul de la fonction d'Euler φ(n)

Ensuite, nous allons utiliser une fonction appelée la fonction d'Euler, notée $\varphi(n)$.

La fonction d'Euler permet de lier mathématiquement la clé publique et la clé privée. C'est donc grâce à elle que le chiffrement et déchiffrement deviennent des opérations inverses.

Si on applique la fonction d'Euler sur $n = p \cdot q$, on obtient $\phi(n) = (p-1) \cdot (q-1)$

La fonction $\phi(n)$ (phi de n) indique combien de nombres sont premiers avec n , c'est-à-dire combien de nombres entre 1 et n ne partagent aucun facteur commun ensemble.

> **Qu'est-ce qu'une fonction ?**
>
> Une fonction est une règle de calcul qui permet d'associer une valeur d'entrée à une valeur de sortie.
>
> Exemple : $f(x) = x + 2 \rightarrow f(2) = 2 + 2$

Si on reprend la suite de notre exemple, en reprenant les valeurs de p et q et que on applique la fonction d'Euler, on obtient :

$$\phi(n) = (p-1) \cdot (q-1) = (19-1) \cdot (23-1) = 396$$

Cela signifie donc que 120 nombres n'ont pas pour facteur 19 ou 23, et sont donc premiers avec 143.

C'est grâce à elle que l'on peut trouver un exposant d (privé) correspondant à l'exposant e (public), selon la relation mathématique suivante :

$$d \cdot e \equiv 1 \pmod{\phi(n)}$$

Ce calcul est effectué à la ligne suivante :

```
70    phi = (p - 1) * (q - 1)
```

rsa_keys.py

Elle assure que les opérations de chiffrement et de déchiffrement sont inverses.

Étape 4 : Choix de l'exposant public e

Il faudra ensuite déterminer la valeur de l'exposant public notée e

Ce dernier est la partie principale de la clé publique. Il est utilisé pour chiffrer les messages.

> **Qu'est-ce qu'un exposant ?**
>
> L'exposant indique combien de fois on va multiplier un nombre par lui-même.
>
> Exemple : $3^3 = 3 \cdot 3 \cdot 3 = 27$

On choisit un entier e de façon à que $1 < e < \phi(n)$ et que ce soit le PGDC de $\big(e, \phi(n)\big) = 1$. Cela signifie que e et $\phi(n)$ n'ont aucun diviseur commun autre que 1 (ils sont premiers entre eux).

Si e est trop petit ou proche de $\phi(n)$, la sécurité sera vulnérable à des attaques connues. Les valeurs 3, 17, et 65537 sont des valeurs qui sont facilement calculables par un ordinateur. 3 est la plus petite, moins sécurisée. 17, est valeur historique et 65537, est valeur standard actuelle (Medium, 2022).

Pour continuer notre exemple, nous allons prendre la valeur courante suivante :

$$e = 17$$

Dans notre code, nous avons choisi d'utiliser la valeur 65537, que nous avons donc définie directement dans une variable **e** :

```
73    e = 65537
```

rsa_keys.py

Étape 5 : Calcul de l'exposant privé d

Cet exposant privé est la clé de déchiffrement dans RSA. Elle est composée à partir de l'exposant public e et $\phi(n)$ et donne $(d \cdot e) \equiv 1(mod\phi(n))$.

C'est ici qu'intervient la notion d'inverse modulaire ainsi que l'algorithme d'Euclide étendu afin de former l'exposant privé noté d.

Qu'est-ce qu'un inverse modulaire ?

Un inverse modulaire, c'est lorsqu'on multiplie deux nombres ensemble, et que le résultat, pris modulo un certain nombre, donne 1.

Exemple : $7 \cdot 5 \equiv 1 \ (mod\ 7)$

Que-ce que c'est que l'algorithme d'Euclide ?

L'algorithme d'Euclide permet de retrouver le PGDC (Plus Grand Diviseur Commun) de deux nombres en effectuant des divisions jusqu'à obtenir un reste nul.

Pour la suite de l'exemple, nous allons reprendre la valeur de notre exposant public e ainsi que $\phi(n)$, qui sont déjà des nombres valides pour la formation de l'exposant privé d, grâce au fait que leur PGCD est égal à 1.

Dans un premier temps, nous allons réaliser l'algorithme d'Euclide, afin de déterminer les coefficients nécessaires au calcul de l'algorithme d'Euclide étendu, utilisé pour trouver l'inverse modulaire.

$$a = b \cdot q + r$$
$$a > b$$

Nous allons donc utiliser notre exposant public e et modulo $\phi(n)$:

$$396 = 17 \cdot 23 + 5$$
$$17 = 5 \cdot 3 + 2$$
$$5 = 2 \cdot 2 + 1$$
$$2 = 1 \cdot 2 + 0$$

L'objectif de cet algorithme est d'effectuer les divisions successives du dividende par le diviseur, en ne conservant à chaque étape que le reste, jusqu'à obtenir un reste égal à 0.

Maintenant, nous allons maintenant utiliser l'algorithme d'Euclide étendu afin de trouver l'inverse modulaire :

$$a \cdot x + b \cdot y = 1$$

Ici, le coefficient y correspond à l'inverse modulaire de a et b. Autrement dit, y est la valeur qui permet de retrouver l'exposant privé d à partir de l'exposant public e et $\phi(n)$.

Maintenant, nous allons remonter les étapes de l'algorithme d'Euclide classique afin de déterminer progressivement les coefficients x et y. On commence par la dernière ligne non nulle, en isolant le 1 :

$$5 = 2 \cdot 2 + 1$$

$$\Rightarrow 1 = 5 - 2 \cdot 2$$

Ensuite, on remplace le 2 (qui correspond ici au reste précédent) par sa valeur trouvée dans la ligne d'avant, en isolant à nouveau le reste :

$$17 = 5 \cdot 3 + 2$$

$$\Rightarrow 2 = 17 - 5 \cdot 3$$

En remplaçant cette expression dans la première équation, on obtient :

$$\begin{aligned} 1 &= 5 - (17 - 5 \cdot 3) \cdot 2 \\ &= 5 - 2 \cdot 17 + 6 \cdot 5 \\ &= 7 \cdot 5 - 2 \cdot 17 \end{aligned}$$

Enfin, on procède de la même manière pour le 5, en le remplaçant par sa valeur tirée de la première ligne de l'algorithme d'Euclide classique :

$$396 = 17 \cdot 23 + 5$$

$$\Rightarrow 5 = 396 - 17 \cdot 23$$

En remplaçant cette expression, on obtient :

$$1 = 7 \cdot (396 - 17 \cdot 23) - 2 \cdot 17$$
$$= 7 \cdot 396 - 17 \cdot (23 \cdot 7 + 2)$$
$$= 7 \cdot 396 - 17 \cdot 163$$

Maintenant que toutes les substitutions ont été effectuées, nous obtenons la relation finale suivante :

$$1 = a \cdot x + b \cdot y \Rightarrow 1 = 396 \cdot 7 - 17 \cdot 163$$

On sait donc maintenant que la valeur de y, qui permet de retrouver l'exposant privé d, à partir de l'exposant public e et $\phi(n)$, est de 163. On peut donc écrire :

$$-17 \cdot 163 \equiv 1 \pmod{396}$$

Pour obtenir une valeur positive, on peut isoler et ajouter 396 à 163 :

$$d \equiv -163 \pmod{396} \Rightarrow d = -163 + 396 = 233$$

La valeur de notre exposant privé d est donc de 233 :

$$d = 233$$

Au niveau du code, on vérifie dans un premier temps pour le bon fonctionnement de l'app que l'exposant public e ainsi que $\phi(n)$ aient un PGDC égal à 1 :

```
76    if rsa_math.gcd(e, phi) != 1:
77      e = 3
78      while e < phi and rsa_math.gcd(e, phi) != 1:
79          e += 2
80
81      if rsa_math.gcd(e, phi) != 1:
82          raise ValueError("Failed to find a suitable public exponent e.")
```

rsa_keys.py

Si le PGDC de $(e\phi(n))$ n'est pas égale à 1, c'est que l'exposant public choisi n'est pas utilisable avec $\phi(n)$.

On cherche donc un nouvel exposant en partant de 3 (qui n'est pas sécurisé, mais reste valide), puis on incrémente de 2 en 2 afin de rester sur des nombres impairs, jusqu'à que on tombe sur un PGDC qui est bien égale à 1, ou si e dépasse la valeur de $\phi(n)$.

Si cette valeur est dépassée, aucune valeur valide n'a été trouvée, donc une erreur sera en sortie pour signaler un problème.

La fonction pour le PGDC **gcd()** est importée du fichier **rsa_math.py**.

Ensuite, le cacul de l'exposant d se fait à la ligne suivante :

```
85    d = rsa_math.modinv(e, phi)
```

rsa_keys.py

L'exposant privé est calculé en appelant la fonction **modinv()** de **rsa_math.py**.

Cette fonction appelle une autre fonction qui est **extended_gcd()**, qui permet de s'appeler elle-même avec des valeurs de plus en plus petites jusqu'à atteindre un cas simple, puis remonte les résultats étape par étape (comme démontré dans les explications plus haut) :

```
32 def extended_gcd(a: int, b: int):
33     """
34     Extended Euclidean algorithm.
35 
36     Returns a tuple (g, x, y) such that:
37         g = gcd(a, b)
38         a * x + b * y = g
39     """
40     if b == 0:
41         return a, 1, 0
42     g, x1, y1 = extended_gcd(b, a % b)
43     x = y1
44     y = x1 - (a // b) * y1
45     return g, x, y
```

rsa_math.py

Dans un premier temps, Lorsque **b** vaut **0**, on arrête la récursion et on retourne directement les valeurs **(a, 1, 0)**.

Ensuite, tant que **b** n'est pas nul, la fonction s'appelle elle-même en remplaçant a par **b** et **b** par **a % b** (le reste de la division). C'est exactement ce qui se passe dans l'algorithme d'Euclide classique. Une fois que la fonction atteint le cas de base, elle remonte en ajustant progressivement les coefficients **x** et **y** pour obtenir la combinaison finale.

Finalement, la fonction **modinv()** utilise donc **extended_gcd()** pour calculer l'inverse modulaire :

```
48 def modinv(a: int, m: int) -> int:
49     """
50     Compute the modular inverse of a modulo m.
51 
52     Returns x such that:
53         (a * x) % m == 1
54 
55     Raises ValueError if the inverse does not exist
56     (i.e. when gcd(a, m) != 1).
57     """
58     g, x, _ = extended_gcd(a, m)
59     if g != 1:
60         raise ValueError(f"No modular inverse exists for {a} modulo {m}")
61     return x % m
```

rsa_math.py

Elle vient récupérer le coefficient **x** calculé par **extended_gcd()**, et s'assure que le PGCD vaut bien 1, puis elle retourne **x % m**, qui est l'opérateur modulo, pour s'assurer que le résultat est positif.

Étape 6 : Formation des clés

Voici un récapitulatif des clés, en fonction des différents nombres utilisés durant les étapes précédentes :

- **Clé publique :** n, e
- **Clé privée :** n, d

Même si tout le monde connaît $n, e,$ il est quasiment impossible de retrouver d sans connaître p et q, car cela reviendrait à factoriser n, ce qui serait trop long et complexe, même pour une machine.

Ainsi, les clés sont définies dans notre code :

```
87      public_key: PublicKey = (n, e)
88      private_key: PrivateKey = (n, d)
89
90      return public_key, private_key
```

rsa_keys.py

Le (dé)chiffrement des messages

Une fois les clés générées, on peut les utiliser pour chiffrer et déchiffrer des messages.

Dans un premier temps, il est important de comprendre que RSA ne manipule que des entiers. Il est donc nécessaire de convertir les messages en nombres avant de les chiffrer.

Ainsi, le message, noté ici M doit être un entier compris entre 0 et $n - 1$, où n est le modulo (le produit des deux grands nombres premiers p, q généré lors de la création des clés).

Il est nécessaire que le message M soit compris dans cet intervalle, car sinon une partie du message serait coupée et donc perdue.

Étape 1 : Conversion du texte en nombre

Afin de convertir le texte en nombre, nous allons utiliser la méthode ASCII.

Ainsi, chaque caractère du message sera associé à un code numérique correspondant à sa valeur dans la table ASCII (Wikipédia - American Standard Code for Information Interchange, 2025).

Qu'est-ce que c'est la méthode ASCII ?

ASCII est un système de codage relativement ancien, créé pour représenter du texte dans les ordinateurs. Ainsi, chaque caractère (symbole, lettre, numéro, etc.) sera associé à un code numérique.

Exemple :

A → **65**
B → **66**
C → **67**
a → **97**
b → **98**
c → **99**
" " (espace) → **32**

Pour l'exemple de conversion ici seulement, nous allons chiffrer le message suivant : "Hello ETML", en le convertissant en code ASCII :

H	e	l	l	o	" "	E	T	M	L
72	**101**	**108**	**108**	**111**	**32**	**69**	**84**	**77**	**76**

Ainsi, la valeur de M sera la concaténation du résultat obtenu, c'est-à-dire :

$$M = 72101108108111326984776$$

Que signifie "concaténation" ?

C'est le fait de coller plusieurs nombres (ou une chaîne de caractères) les uns à la suite des autres pour n'en former qu'un seul.

Exemple : 72 101 108 → 72101108

Cependant, pour continuer l'exemple pédagogique, nous allons chiffrer sur un nombre très petit afin de rester compatible avec les valeurs des clés générées précédemment ($n = 143$)

Donc, puisque $M < n$, nous pouvons passer à l'étape suivante.

Étape 2 : Chiffrement du message

Une fois le texte M converti en code ASCII ainsi que la concaténation effectuée si nécessaire, nous pouvons chiffrer le message. Pour cela, on va utiliser notre modulo n ainsi que l'exposant public e.

Le modulo, qui pour rappel, est au cœur du RSA. L'exposant public correspond à la clé publique, utilisée pour chiffrer les messages.

Ainsi, et pour continuer l'exemple, on va pouvoir utiliser et appliquer la formule suivante :

$$C = M^e \ mod \ n = 2^{17} \ mod \ 396 = 409$$

Le résultat C contient donc le message chiffré.

> **Quel est le risque si l'exposant public e est trop petit ?**
>
> Si l'exposant public est trop petit, il peut arriver que, lors du chiffrement, le calcul M^e donne une valeur inférieure à n. Dans ce cas, le message peut devenir vulnérable à certaines attaques, qui ne seront pas détaillées ici.
>
> En pratique, c'est aussi la raison pour laquelle on choisit presque toujours un exposant public de $e = 65537$, un nombre premier suffisamment grand pour garantir que M^e soit supérieur à n.

Pour la partie du code, on utilise des tuples pour regrouper les éléments des clés. Un tuple permet de définir plusieurs valeurs ensemble dans une même structure.

```
38    PublicKey = Tuple[int, int]   # (n, e)
39    PrivateKey = Tuple[int, int]  # (n, d)
```

rsa_codec.py

Ici, le modulo n et l'exposant public e forment un tuple représentant la clé publique, tandis que le modulo n et l'exposant privé d forment un tuple représentant la clé privée. Ces valeurs sont récupérés des étapes précédentes, lors de la génération des clés.

Dans le code, le chiffrement du message ce fait aux lignes suivantes :

```
80  def rsa_encrypt(message: str, public_key: PublicKey) -> int:
81      """
82      Encrypt a plaintext message using the RSA public key (n, e).
83
84      The message is converted to an integer using UTF-8 encoding.
85
86      Args:
87          message (str): The plaintext message to encrypt.
88          public_key (tuple): The public key (n, e).
89
90      Returns:
91          int: The ciphertext as an integer.
92      """
93      n, e = public_key
94
95      # Convert the message to an integer (bytes -> int)
96      message_int = int.from_bytes(message.encode("utf-8"), byteorder="big")
97
98      if message_int >= n:
99          raise ValueError("Message too large for the key size.")
100
101     # Perform RSA encryption: c = m^e mod n
102     ciphertext_int = pow(message_int, e, n)
103
104     return ciphertext_int
```

rsa_codec.py

Ici, on extrait le modulo n et l'exposant public e du tuple de la clé publique.

Ensuite, le message est converti en UTF-8 dans notre code (UTF-8 est un standard d'encodage similaire à l'ASCII, qui applique le même principe de correspondance caractère-valeur numérique), puis transformé en un entier unique.

Une vérification est ensuite effectuée pour s'assurer que la valeur numérique du message soit bien inférieure au modulo n. Si ce n'est pas le cas, une erreur est envoyée, car le message est trop long pour être chiffré avec cette clé.

Finalement, on applique la formule mathématique du chiffrement RSA : $C = M^e \ mod \ n$, dont le résultat est stocké dans la variable **ciphertext_int**.

Étape 3 : Déchiffrement du message

Afin de déchiffrer un message, nous reprenons la valeur C et utilisons à nouveau notre modulo n, ainsi que l'exposant privé d.

L'exposant privé correspond à la clé privée. Il doit être associé à la clé publique utilisée lors du chiffrement, c'est-à-dire qu'il doit appartenir à la même paire de clés générée précédemment.

Pour terminer l'exemple, on va pouvoir appliquer :

$$M = C^d \ mod \ n = 409^{233} \ mod \ 437 = 2$$

Pour le déchiffrement du message dans le code :

```
107 def rsa_decrypt(ciphertext: int, private_key: PrivateKey) -> str:
108     """
109     Decrypt an RSA ciphertext integer using the private key (n, d).
110 
111     Args:
112         ciphertext (int): The encrypted message as an integer.
113         private_key (tuple): The private key (n, d).
114 
115     Returns:
116         str: The decrypted plaintext message.
117     """
118     n, d = private_key
119 
120     # Perform RSA decryption: m = c^d mod n
121     message_int = pow(ciphertext, d, n)
122 
123     # Convert the integer back to a UTF-8 string
124     message_bytes = message_int.to_bytes((message_int.bit_length() + 7)d
125     plaintext = message_bytes.decode("utf-8")
126 
127     return plaintext
```

rsa_codec.py

Comme pour le chiffrement, on extrait le modulo n et l'exposant privé d du tuple de la clé publique.

On applique ensuite la formule du déchiffrement RSA afin de retourner le message sous forme entière : $M = C^d \ mod \ n$.

L'entier est reconverti en une séquence de bytes, puis sont finalement décodés en texte UTF-8 pour retrouver le message en claire **plaintext**.

Messagerie sécurisée

Notre produit est le développement d'une messagerie chiffrée fonctionnelle basée sur l'algorithme RSA, permettant à deux ordinateurs d'échanger, interconnectés grâce à des appareils périphériques et serveur de relais, des informations en garantissant trois principes fondamentaux de la sécurité informatique qui sont la confidentialité, l'authenticité et l'intégrité des communications.

Architecture du système

Le système repose sur une architecture à trois appareils interconnectés : un appareil central faisant office de serveur relais et deux appareils périphériques connectés aux ordinateurs des utilisateurs. Ces derniers gèrent l'authentification, établissent la communication et effectuent les opérations cryptographiques via une interface web permettant d'envoyer et de consulter les messages de manière sécurisée.

Plus précisément, voici ce que fait chacun des trois appareils interconnectés :

- Appareil central : Met à disposition l'interface web et assure la transmission des messages chiffrés entre les différents agents sans jamais avoir accès au contenu des messages ni aux clés de chiffrement. Ce serveur agit uniquement comme un relais, garantissant ainsi une architecture de chiffrement de bout en bout.
- Appareil client (périphérique) : Chaque utilisateur dispose d'un périphérique qui donne l'accès à l'interface web et gère l'ensemble des opérations cryptographiques localement (génération de clés, chiffrement et déchiffrement).

Interdisciplinarité du projet

Ce produit constitue une jonction entre les deux disciplines étudiées dans le cadre de ce travail.

Pour la partie mathématiques, l'application utilise tout l'algorithme RSA, de la création des clés publiques et privées jusqu'au chiffrement et déchiffrement des messages. Chaque étape utilise des concepts mathématiques qu'on a étudiés : les nombres premiers, l'arithmétique modulaire, la fonction d'Euler, l'algorithme d'Euclide étendu et l'inverse modulaire. Le code qu'on a écrit transforme directement ces formules mathématiques en quelque chose de concret, ce qui montre bien comment on passe de la théorie à la pratique.

Pour la partie historique, notre messagerie continue ce qui a été découvert et qui a complètement changé la cryptographie moderne. Elle montre comment les idées de Diffie et Hellman, puis celles des trois chercheurs du MIT, ont permis de résoudre le gros problème qu'on avait avant pour distribuer les clés. Notre application met aussi en pratique ce que Phil Zimmermann défendait avec PGP : le droit à la vie privée et au chiffrement pour tout le monde.

Réponse aux objectifs

Le produit final répond pleinement aux quatre objectifs généraux définis en début de projet :

- Recherche historique sur la cryptographie moderne : On a retracé l'évolution du chiffrement depuis l'après-guerre jusqu'à maintenant. On a vu les bases de la cryptographie, les découvertes de Diffie-Hellman-Merkle, l'invention de RSA par les trois chercheurs du MIT, ce que le GCHQ a révélé et comment PGP a démocratisé tout ça.
- Maîtrise des fondements mathématiques de RSA : L'étude de chaque étape de l'algorithme (génération de nombres premiers, calcul du modulo, fonction d'Euler, choix et calcul des exposants public et privé) nous a permis de comprendre les maths qui constituent l'algorithme.
- Développement de l'application de messagerie chiffrée : L'implémentation complète de RSA en Python, incluant les fonctions de génération de clés (rsa_keys.py), de chiffrement et déchiffrement (rsa_codec.py), ainsi que les opérations mathématiques nécessaires (rsa_maths.py), démontre qu'on a réussi à passer de la théorie à la pratique.
- Déploiement sur trois appareils interconnectés : Le système a été installé avec succès sur le matériel prévu, validant que les échanges sécurisés entre utilisateurs fonctionnent correctement.

Analyse critique de la gestion du projet

Cette analyse s'appuie sur notre journal de bord, notre planification, nos évaluations intermédiaires ainsi que l'expérience et le ressenti vécus tout au long de ce projet. Nous retraçons l'atteinte des objectifs, les aspects organisationnels et méthodologiques, puis le bilan des acquis individuels et collectifs.

Atteinte des objectifs

Nos objectifs fondamentaux étant la recherche et une étude historique sur RSA, la maîtrise des fondements mathématiques de l'algorithme ainsi que le développement et l'intégration sur un système d'appareils interconnectés d'une messagerie de chiffrement sécurisée, ont tous été atteints. Le travail prévu a été produit.

Difficultés rencontrées

Nous avions malheureusement un peu laissé l'histoire de côté pour nous concentrer sur le reste, et nous avons eu du mal aussi à comprendre le résultat attendu en histoire. Ceci nous a légèrement mis en péril car nous avons dû le faire au dernier moment et nous avons recommencé à plusieurs reprises.

Nous avons un peu précipité le travail sur le produit ainsi que sur les disciplines. En effet, nous n’avons pas pris le temps de réellement trouver méthode pour organiser le projet, comme poser toutes les idées sur un mindmap, nous l’avons directement fait sur planificateur. De ce fait, notamment pour l’histoire, nous sommes souvent partis dans trop de directions différentes, ce qui nous a fait perdre du temps.

Finalement, le respect de la timeline et des dates prévues n'a pas été totalement respecté. En effet, nous avions mal planifié les dates et les temps qui parfois étaient trop longs, ou même trop courts. Aussi, certaines activités étaient en trop, car nous avons repensé certaines parties du système pendant la phase de réalisation. Les périodes de vacances ont perturbé le rythme de travail, la plupart d'entre nous étant absents en raison de voyages ou d'autres engagements personnels durant ces périodes.

Solutions mises en place

Tout au long du projet, nous avons su trouver des solutions adaptées à ces difficultés. Nous avons redéfini clairement les objectifs en histoire pour éviter d'autres retards, directement avec l'expert.

La coordination assurée par Marijan a permis aux autres membres de se concentrer sur leurs domaines respectifs sans se préoccuper de la cohérence globale. Nous avons également ajusté la planification et priorisé les tâches essentielles, ce qui nous a permis d'avancer malgré les imprévus.

Facteurs de réussite et efficacité

Plusieurs facteurs ont contribué à la réussite du projet. Premièrement, chaque membre possède des compétences dans les domaines clés, donc les mathématiques, le développement, l'électronique ainsi que l'intégration de systèmes.

Deuxièmement, la disponibilité du coach et des experts a joué un rôle important afin de redéfinir le scope du projet et apporter des retours constructifs, nous permettant de rester pertinents.

Enfin, la coordination centralisée et l'organisation efficace ont permis à chacun de se concentrer sur ses responsabilités, ce qui a grandement facilité la réalisation du projet.

Planification et gestion du temps

La planification initiale n'a pas été totalement respectée. Comme décrit dans les difficultés rencontrées, nous avons surestimé certaines activités et jalons, en y allouant ainsi trop d'heures. De plus, nous n'avons pas totalement suivi les dates fixées pour les différentes parties, car il a été difficile de gérer certains imprévus.

Néanmoins, les jalons ainsi que la plupart des activités définis au départ ont tout de même servi de fil conducteur pour la réalisation de ce travail.

Collaboration et moyens

Le projet a été centralisé sur Notion avec un workflow de travail automatisé créer au départ, ce qui a facilité l'organisation et la répartition des tâches. Chaque membre disposait d'un espace dédié à son domaine, ainsi que d'un tableau de bord regroupant agenda, planification, etc. La communication interne était également bonne, les informations, idées et avancées étaient partagées dans un groupe commun.

Le projet étant principalement numérique, nous avons disposé de toutes les ressources nécessaires pour le mener à bien. L'acquisition de ressources supplémentaires, qu'il s'agisse du livre utilisé pour la recherche ou du matériel électronique pour la conception des boîtiers, a été réalisée sans difficulté.

Vécu et apprentissages

Dans l'ensemble, tous les membres ont apprécié travailler sur ce projet. Le domaine de la cryptographie, tant d'un point de vue mathématique, notamment avec l'algorithme RSA, que d'un point de vue historique, est très intéressant.

Chacun des membres a pu apporter une part de son expérience au projet, que ce soit en développement, en électronique ou en conception d'architectures applicatives. Nous avons également eu l'opportunité d'apprendre de nouvelles choses dans ces différents domaines.

Bien que nous ayons rencontré des moments de stress et de baisse de motivation, l'expérience globale reste largement positive.

Le projet nous a permis de renforcer nos compétences techniques, d'améliorer notre organisation et notre gestion du temps, et de travailler efficacement en équipe.

Effets concrets du projet

Notre projet étant une conception d'une application de messagerie chiffrée fonctionnelle, illustrant concrètement l'utilisation de RSA et de la cryptographie moderne pour sécuriser les échanges de messages.

Il nous a permis d'approfondir nos connaissances théoriques et pratiques, de combiner recherche historique et travail technique, et d'acquérir une expérience réelle de gestion de projet et de coordination.

Enfin, c'est un outil utilisable pour démontrer la sécurité des échanges et a contribué à notre développement personnel et professionnel dans le domaine de la cryptographie et de la sécurité.

Acquis, contributions et appréciation collective

Au cours de ce projet, nous avons tous activement contribué à approfondir notre compréhension des fondamentaux de la cryptographie, et plus particulièrement de la cryptographie moderne avec le chiffrement asymétrique. Nous avons également exploré en détail les mathématiques appliquées à l'algorithme RSA, ce qui nous a permis de mieux relier la théorie à la pratique et de comprendre concrètement comment sécuriser des échanges d'informations. Ces connaissances ont été partagées et discutées entre tous les membres, renforçant notre compréhension et notre capacité à expliquer ainsi que d'utiliser ces concepts dans le cadre de notre application de messagerie sécurisée.

Sur le plan des compétences techniques, chacun a pu développer ses aptitudes. Par exemple, Marijan en conception d'architecture logicielle et gestion d'équipe, Thomas en développement d'applications avec la mise en place d'un WebSocket, et Cédric en intégration matérielle avec la réalisation des boîtiers électroniques.

Au-delà des aspects techniques, ce projet nous a enseigné à travailler efficacement en équipe, avec une répartition cohérente des tâches et un suivi rigoureux grâce aux outils mis à disposition (journal de bord, planificateur, repository GitHub, etc.).

Conclusion

Ce projet a été pour nous une vraie expérience d'apprentissage. En effet, au départ, nos connaissances en chiffrement et en cryptographie étaient assez limitées. Progressivement, nous avons retracé l'histoire de cette discipline, particulièrement à l'ère du numérique. L'étude approfondie des fondements mathématiques de RSA nous a permis non seulement de développer nos compétences en mathématiques, mais surtout de les appliquer concrètement dans une application fonctionnelle.

Nous avons ainsi réussi à concevoir un système complet reposant sur trois appareils interconnectés, garantissant la confidentialité, l'authenticité et l'intégrité des communications entre deux utilisateurs. L'architecture mise en place, démontre la faisabilité technique d'une solution de chiffrement de bout en bout. Ce travail d'équipe nous a permis de combiner nos compétences dans différents domaines, mathématiques, programmation, conception matérielle et applicatif ainsi que documentation pour produire un projet cohérent. Le résultat final correspond aux attentes initiales, tant sur le plan technique qu'historique, et nous sommes globalement satisfaits.

Cependant, notre système actuel a une limite importante, il utilise uniquement le chiffrement RSA pour tous les échanges. Cette approche est intéressante pour apprendre, mais elle n'est pas recommandée pour une vraie utilisation à cause de problèmes de performance. En effet, RSA a été créé surtout pour échanger des clés et faire des signatures numériques, mais il n'est pas très efficace pour chiffrer de grandes quantités de données.

Pour une utilisation réelle, il faudrait adopter un système de chiffrement hybride qui combine les avantages de RSA (chiffrement asymétrique) et de DES (chiffrement symétrique). Dans ce système, RSA servirait uniquement à échanger de façon sécurisée les clés du chiffrement symétrique, qui serait ensuite utilisé pour chiffrer tous les messages. Cette méthode offrirait un meilleur équilibre entre sécurité et vitesse, tout en gardant les avantages du chiffrement asymétrique pour gérer les clés.

Ce projet constitue ainsi une base solide qui pourrait évoluer vers une solution de messagerie sécurisée plus robuste et adaptée aux exigences actuelles de la cybersécurité. Au-delà du produit final, ce sont la démarche allant de la théorie mathématique à l'implémentation pratique, la compréhension historique d'un domaine initialement vaste et complexe pour nous, ainsi que le travail d'équipe qui resteront les apports les plus précieux de cette expérience.

Références

SINGH Simon, 1999, Histoire des codes secrets, J.-C. Lattès, ISBN : 2-7096-2048-0

Stanford University, WHITFIELD Diffie et E. HELLMAN Martin, Novembre 1976. New Directions in Cryptography, pp. 644-654. Disponible à l'adresse : https://ee.stanford.edu/~hellman/publications/24.pdf

Massachusetts Institute of Technology, RIVSET Ron, SHAMIR Adi, ADLEMAN Leonard, Février 1978. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. , pp. 1-14. Disponible à l'adresse : https://people.csail.mit.edu/rivest/Rsapaper.pdf

National Institute of Standards and Technology, BURR, William, 2000. Data Encryption Standard. pp. 250-253. Disponible à l'adresse : https://nvlpubs.nist.gov/nistpubs/sp958-lide/250-253.pdf

BUCHANAN Bill, 2024. So Who Invented Public-key Encryption. Disponible à l'adresse : https://medium.com/asecuritysite-when-bob-met-alice/so-who-invented-public-key-encryption-213ceef7759

ZIMMERMAN Philip, 1991. Why I Wrote PGP. Disponible à l'adresse : https://www.philzimmermann.com/EN/essays/WhyIWrotePGP.html

Wikipédia (contributeurs), 2025. Cryptographie. In : *Wikipédia : l'encyclopédie libre*, 2025. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Cryptographie

Wikipédia (contributeurs), 2025. Cryptologie. In : *Wikipédia : l'encyclopédie libre*, 2025. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Cryptologie

Wikipédia (contributeurs), 2025. Enigma (machine). In : *Wikipédia : l'encyclopédie libre*, 2025. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Enigma_(machine)

Wikipédia (contributeurs), 2025. Cryptographie symétrique. In : *Wikipédia : l'encyclopédie libre*, 2025. Disponible à l'adresse : https://fr.wikipedia.org/wiki/Cryptographie_symétrique

CEA Recherche, 2024. Comment ça marche ? – La cryptographie [Vidéo]. In : *YouTube*, 2024. Disponible à l'adresse : https://www.youtube.com/watch?v=vT8q1mXeZ2A

DEVELOPPEZ.COM, 2025. Combien de qubits faut-il pour casser les algorithmes de cryptographie à clé publique ? Le chiffrement RSA à 2048 bits "pourrait être cassé par un ordinateur quantique à 1 million de qubits en une semaine". In : *Quantique Developpez.com*, 27 mai 2025. Disponible à l'adresse : https://quantique.developpez.com/actu/372510/Combien-de-qubits-faut-il-pour-casser-les-algorithmes-de-cryptographie-a-cle-publique-Le-chiffrement-RSA-a-2048-bits-pourrait-etre-casse-par-un-ordinateur-quantique-a-1-million-de-qubits-en-une-semaine/

GOFFINET, François, 2021. Cryptographie asymétrique. In : *Linux Administration*, 5 juin 2021. Disponible à l'adresse : https://linux.goffinet.org/administration/confidentialite/chiffrement-asymetrique

DIGICERT, 2025. What is SSL Cryptography? In : *DigiCert FAQ*. Disponible à l'adresse : https://www.digicert.com/faq/cryptography/what-is-ssl-cryptography

NIST, 2024. NIST Releases First 3 Finalized Post-Quantum Encryption Standards. In : *National Institute of Standards and Technology*, 13 août 2024. Disponible à l'adresse : https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards

MACGENERATION, 2025. ProtectEU : comment l'Union européenne veut affaiblir le chiffrement de bout en bout. In : *MacGeneration*, 7 juillet 2025. Disponible à l'adresse : https://www.macg.co/ailleurs/2025/07/protecteu-comment-lunion-europeenne-veut-affaiblir-le-chiffrement-de-bout-en-bout-302393

GEEKSFORGEEKS, RSA Algorithm in Cryptography. In : *GeeksforGeeks*. Disponible à l'adresse : https://www.geeksforgeeks.org/computer-networks/rsa-algorithm-cryptography

DUC Alexandre, Asymmetric Cryptography Standards. Support de cours, pp. 1-57.

RSA Laboratories, KALISKI Burt, The Mathematics of the RSA Public-Key Cryptosystem. RSA Laboratories, pp. 1-9.

MEDIUM, CRYPTO COMICS, 2018. Crypto Comics: What is 65537? In : *Medium, Coinmonks*, 18 avril 2018. Disponible à l'adresse : https://medium.com/coinmonks/crypto-comics-what-is-65537-5ac867fe61c

Wikipédia (contributeurs), 2025. American Standard Code for Information Interchange. In : *Wikipédia : l'encyclopédie libre*, 2025. Disponible à l'adresse : https://fr.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange

Annexes

L'ensemble de la documentation projet (journal de bord, planification, procès-verbaux) est accessible sur notre espace Notion : https://www.notion.so/27de36da46f7814981baf30640b5d3a6?v=27de36da46f781308215000cf303f192&source=copy_link

Le code source de l'application ainsi que les fichiers de conception des boîtiers sont disponibles sur notre dépôt GitHub : https://github.com/MarijanStajic/cryptoTIP

Lien du prompt utilisé pour la correction orthographique et syntaxique du projet avec Claude.ai (IA) : https://claude.ai/share/fc9ba071-3223-49f6-858a-20a079f84d9b

DUC Alexandre, Asymmetric Cryptography Standards. Support de cours, pp. 1-57. [Document transmis par courriel]

RSA Laboratories, KALISKI Burt, The Mathematics of the RSA Public-Key Cryptosystem. RSA Laboratories, pp. 1-9. [Document transmis par courriel]

Stanford University, WHITFIELD Diffie et E. HELLMAN Martin, Novembre 1976. New Directions in Cryptography, pp. 644-654. [Document annoté transmis par courriel]

Massachusetts Institute of Technology, RIVSET Ron, SHAMIR Adi, ADLEMAN Leonard, Février 1978. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. , pp. 1-14. [Document annoté transmis par courriel]

National Institute of Standards and Technology, BURR, William, 2000. Data Encryption Standard. pp. 250-253. [Document annoté transmis par courriel]