



N° d'ordre : 92/2024

Centre des Études : **Sciences et Techniques et Sciences Médicales**

Formation Doctorale : **Sciences et Technologies de l'Information et de Communication**

Discipline : **Informatiques**

Spécialité : **Informatiques**

Laboratoire : **Laboratoire d'Informatique, Signaux, Automatique et Cognitivisme**

THÈSE DE DOCTORAT

Titre de la thèse :

Approches innovantes de cybersécurité et stratégies avancées pour la protection des données et l'authentification

Présentée par :

Hamza TOUIL

Soutenue le 31/12/2024 devant le jury :

Nom et prénom	Établissement	Grade	Qualité
AZROUL Elhoussine	Faculté des Sciences Dhar El Mahraz, Fès, USMBA	PES	Président
SBAI El Hassan	École Supérieure de Technologie, Meknès, UMI	PES	Rapporteur
AKSASSE Brahim	Faculté des Sciences, Meknès, UMI	PES	Rapporteur
SATORI Hassan	Faculté des Sciences Dhar El Mahraz, Fès, USMBA	PES	Rapporteur
HADDOUCH Khalid	École Nationale des Sciences Appliquées, Fès, USMBA	PH	Examineur
LOQMAN Chakir	Faculté des Sciences Dhar El Mahraz, Fès, USMBA	PES	Examineur
EL AKKAD Nabil	École Nationale des Sciences Appliquées, Fès, USMBA	PH	Co-directeur de thèse
SATORI Khalid	Faculté des Sciences Dhar El Mahraz, Fès, USMBA	PES	Directeur de thèse

Année Universitaire : 2024-2025

Remerciements

Tout d'abord, je tiens à exprimer ma profonde gratitude envers Dieu pour m'avoir accordé la santé et le bien-être nécessaires qui ont rendu possible l'achèvement de cette thèse.

Premièrement, je tiens à exprimer ma sincère reconnaissance envers mon directeur de thèse, Monsieur **Khalid Satori**, pour son soutien, ses conseils éclairés et son accompagnement précieux tout au long de la réalisation de ma thèse. Il a toujours été présent en tant que soutien discret, mais régulier, apportant son aide tout au long de cette expérience académique. Toujours patient et disponible, il a répondu à mes multiples questions et m'a fourni des explications claires avec une gentillesse exceptionnelle. Chaque jour, je profite de l'immense expérience qui m'accompagne, m'offrant ainsi des idées efficaces, des conseils avisés, des suggestions constantes et une motivation suffisante pour repousser mes limites et atteindre un niveau d'excellence très élevé.

Je tiens à exprimer aussi ma profonde gratitude à mon encadrant, monsieur **Nabil El Akkad**, pour sa disponibilité constante, ses conseils avisés et son savoir-faire technique précieux qui m'ont accompagné tout au long de mes travaux de recherche. Les conseils précieux qu'il m'a prodigués, les échanges enrichissants que j'ai eus avec lui et son examen minutieux et approfondi de mon travail ont grandement contribué à ma progression. Sa patience infinie, sa disponibilité sans faille et sa capacité à simplifier les idées complexes pour les rendre plus accessibles ont été des atouts précieux pour moi.

Je souhaite exprimer ma profonde gratitude envers tous **les membres de l'équipe de recherche** qui ont activement contribué à la réalisation de ce travail. Les idées novatrices et le soutien inestimable apportés par chacun d'entre eux, combinés à leurs compétences complémentaires, ont grandement enrichi mon expérience. Grâce à eux, j'ai pu évoluer dans un environnement stimulant et coopératif, ce qui a été essentiel à ma progression à ce stade.

Je souhaite exprimer ma profonde gratitude envers mes amis et ma famille qui ont été présents à mes côtés chaque jour, m'apportant leur soutien inconditionnel tout au long de cette incroyable aventure. Leurs encouragements, leur attention et leur amour ont toujours été une source de motivation exceptionnelle pour moi. Leur simple présence à mes côtés alors que je franchissais cette grande étape de la vie a été d'une aide précieuse.

Finalement, je tiens à exprimer ma profonde gratitude envers **la faculté des sciences de Fès** pour toutes les opportunités d'études qu'elle m'a offertes. Une fois encore, merci à tous ceux qui l'ont facilité et à tous ceux qui m'ont assisté pour y arriver.

Table des matières

Remerciements	2
Table des matières	3
Table des figures.....	8
Liste des tableaux	10
Liste des algorithmes.....	11
Acronymes.....	12
Notations	14
Résumé	15
Abstract.....	16
Introduction générale.....	17
Contexte de la thèse	17
Problématique	18
Contributions	20
Organisation du manuscrit.....	22
Chapitre 1 : Principes fondamentaux de la cryptographie	23
1.1 Introduction	23
1.1.1 Contexte et enjeux de la cryptographie moderne	24
1.1.2 Importance de la sécurisation des échanges numériques	26
1.1.3 Problématiques liées à l'évolution des menaces	27
1.2 Cryptographie symétrique	29
1.2.1 Chiffrement DES /3DES	29
1.2.1.1 Structure DES/3DES (substitution et permutation)	30
1.2.1.2 Limites de sécurité	32
1.2.2 Chiffrement AES.....	33
1.2.2.1 Structure AES.....	33
1.2.2.2 Vulnérabilités potentielles.....	35
1.3 Cryptographie asymétrique	35
1.3.1 Chiffrement RSA.....	36
1.3.2 Échange de clés Diffie-Hellman.....	36
1.4 Fonction de hachage.....	37
1.4.1 Fonction de hachage MD5	38
1.4.2 Fonction de hachage SHA.....	38

1.5	Cryptographie quantique	39
1.5.1	Communication quantique sécurisée.....	40
1.5.2	Fonction BB84	41
1.5.3	Défis et avancées en cryptographie quantique	42
1.6	Protocoles de communications.....	43
1.6.1	Protocole TLS	43
1.6.2	Protocole SSH	44
1.7	Conclusion.....	45
Chapitre 2 : Outils mathématiques pour la cryptographie		47
2.1	Introduction	47
2.2	Théorie des groupes et algèbre abstraite	47
2.2.1	Groupes cycliques	47
2.2.2	Exponentiation modulaire	48
2.2.3	Multiplication modulaire.....	48
2.2.4	Sous-groupes et générateurs.....	49
2.3	Algèbre linéaire et transformations	50
2.3.1	Espace vectoriel	50
2.3.2	Matrices de transformation	51
2.3.3	Matrice orthogonale	51
2.3.4	Transformation linéaire.....	52
2.4	Théorie des nombres et logarithmes discrets	52
2.4.1	Nombres premiers.....	53
2.4.2	Logarithme discret	53
2.4.3	Problème de factorisation	54
2.4.4	Problème de modulaire	55
2.5	Fonction de hachage et complexité	55
2.5.1	Propriétés des fonctions de hachage	55
2.5.2	Résistance aux collisions	56
2.5.3	Complexité algorithmique	57
2.6	Théorie de l'information et entropie	58
2.6.1	Entropie des clés	59
2.6.2	Redondance dans les systèmes cryptographiques.....	59
2.6.3	Capacité de stockage sécurisé.....	60
2.7	Conclusion.....	60

Chapitre 3 : Méthodes avancées de sécurité pour l'authentification et le stockage des données	61
3.1 Introduction	61
3.2 Techniques modernes de stockage et d'authentification des phrases de passe	62
3.2.1 Hachage avec algorithmes de renforcement	62
3.2.2 Gestion multifactorielle des clés	64
3.2.3 Hachage matriciel	65
3.3 H-Rotation : stockage sécurisé et récupération des phrases de passe dans le processus d'authentification	66
3.3.1 Présentation du concept de « H-Rotation »	66
3.3.1.1 Transformations dynamiques	67
3.3.1.2 Types de rotations	67
3.3.1.3 Description des paramètres utilisés	69
3.3.1.4 Processus d'inscription et d'authentification « H-Rotation »	69
3.3.2 Explication des méthodes de rotation	73
3.3.2.1 Rotation dans les préfixes	73
3.3.2.2 Rotation dans les suffixes	75
3.3.2.3 Rotation au milieu	77
3.3.3 Résultat expérimental	79
3.3.3.1 Rotation dans les préfixes	80
3.3.3.2 Rotation dans les suffixes	85
3.3.3.3 Rotation au milieu	87
3.3.4 Étude comparative	90
3.3.4.1 Analyse des performances	90
3.3.4.2 Sécurité et résistance aux attaques	92
3.3.4.3 Complexité algorithmique	93
3.3.5 Synthèse	96
3.4 Utilisation de la transformation Braille pour le hachage sécurisé des mots de passe	97
3.4.1 Description détaillée de la transformation Braille pour le hachage sécurisé des mots de passe	98
3.4.1.1 Génération du hachage	98
3.4.1.2 Transformation en Braille	99

3.4.1.3 Renforcement aléatoire du cryptage.....	100
3.4.1.4 Stockage dans la base de données	101
3.4.1.5 Vérification lors de l'authentification	104
3.4.2 Résultat expérimental	105
3.4.2.1 Simulation de la transformation Braille	105
3.4.2.2 Processus de récupération du hachage initial	107
3.4.3 Étude comparative	107
3.4.3.1 Temps d'exécution	107
3.4.3.2 Occupation mémoire	109
3.4.3.3 Résistance aux attaques	110
3.4.4 Synthèse	114
3.5 Conclusion	115
Chapitre 4 : Méthodes avancées de sécurité pour la transmission sécurisée des données	116
4.1 Introduction	116
4.2 Présentation des enjeux de la sécurité de transmission	117
4.2.1 Analyse des enjeux liés à la sécurité des données	117
4.2.2 QoS dans les communications (cas de figure : échanges multimédias).....	118
4.2.3 Utilisation des méthodes quantiques pour sécuriser les échanges	119
4.3 Sécurisation des échanges via TLS : étude de cas sur la transmission de séquences vidéo	119
4.3.1 Présentation du concept de sécurisation via TLS : cas de la transmission vidéo ..	120
4.3.1.1 Mécanismes de TLS dans la transmission vidéo	121
4.3.2 Résultat expérimental	126
4.3.2.1 Mise en place de l'environnement de simulation.....	126
4.3.2.2 Automatisation des négociations de clés.....	127
4.3.2.3 Impact sur les ressources pendant les renégociations	131
4.3.3 Étude comparative	132
4.3.3.1 Chiffrement utilisé.....	132
4.3.3.2 Adaptabilité aux conditions réseau	134
4.3.3.3 Impact sur les ressources	135

4.3.3.4 Résistance aux attaques	137
4.3.4 Synthèse	138
4.4 Sécurité du partage des clés publiques SSH post-quantique	139
4.4.1 Description détaillée du partage des clés publiques SSH post-quantique	140
4.4.1.1 Initialisation du partage de clés avec BB84	140
4.4.1.2 Correction d'erreurs et amplification de la confidentialité	142
4.4.1.3 Utilisation de la clé quantique dans le chiffrement SSH.....	144
4.4.2 Résultat expérimental	147
4.4.2.1 Simulation du protocole BB84.....	147
4.4.2.2 Taux d'erreur de bits quantiques (QBER).....	148
4.4.2.3 Taux de génération de clés	149
4.4.3 Étude comparative	151
4.4.3.1 Utilisation du CPU	151
4.4.3.2 Utilisation de la mémoire	152
4.4.3.3 Temps d'établissement de la session.....	153
4.4.3.4 Débit de transmission	154
4.4.3.5 Résistance aux attaques	155
4.4.4 Synthèse	156
4.5 Conclusion	156
Conclusion générale et perspective	157
Bibliographie.....	159
Annexe A : Algorithmes de hachage.....	168
Algorithme de MD5.....	168
Algorithme de SHA	170
Annexe B : Algorithmes de protection de d'authentification.....	173
Algorithme de « H-Rotation »	173
Algorithme de transformation Braille.....	175
Publications.....	177
Articles publiés dans des journaux internationaux	177
Communications internationales	177

Table des figures

Figure 1-1 : Architecture de la cryptographie moderne	26
Figure 1-2 : Concept général de la cryptographie symétrique	29
Figure 1-3 : Structure DES	31
Figure 1-4 : Structure 3DES	32
Figure 1-5 : Structure AES	34
Figure 1-6 : Protection cryptographique : clés publiques et privées	36
Figure 1-7 : Principe de fonctionnement du hachage	38
Figure 1-8 : Diagramme de SHA-1	39
Figure 1-9 : Cryptographie symétrique avec des dispositifs quantiques	41
Figure 1-10 : La pile de protocoles SSL/TLS	44
Figure 3-1 : Principe de fonctionnement « H-Rotation » dans la phase de souscription.....	71
Figure 3-2 : Principe de fonctionnement « H-Rotation » dans la phase d'authentification. ...	72
Figure 3-3 : Différentes possibilités de la méthode de rotation en préfixe	75
Figure 3-4 : Différentes possibilités de la méthode de rotation en suffixe.....	77
Figure 3-5 : Différentes possibilités de la méthode de rotation au milieu	79
Figure 3-6 : Processus classique de stockage des mots de passe	98
Figure 3-7 : Ajout de la transformation Braille dans le processus de stockage des mots de pass	100
Figure 3-8 : Processus de transformation et de stockage des mots de passe	102
Figure 3-9 : Processus de transformation et de stockage des mots de passe - cas réel	103
Figure 3-10 : Processus d'authentification	104
Figure 3-11 : Comparaison entre la transformation Braille, bcrypt, scrypt, argon2 ; « H-Rotation » sur le temps d'exécution en millisecondes pour les 3 mots de passe (QWERTY ; Admin\$+123654 ; 123456789)	108
Figure 3-12 : Comparaison entre la transformation Braille, bcrypt, scrypt, argon2 sur la taille mémoire utilisée par chacune	110
Figure 4-1 : Démarrage de la négociation du protocole TLS par le client.....	121
Figure 4-2 : La suite de chiffrement utilisée dans les négociations TLS	121
Figure 4-3 : Accord sur la clé utilisée entre le serveur et le client	122

Figure 4-4 : Processus de fonctionnement de l'échange de la clé	123
Figure 4-5 : Diagramme de séquence des différentes interactions effectuées entre le client et le serveur	125
Figure 4-6 : Initialisation de la session TLS dans wireshark	127
Figure 4-7 : Chiffrement dynamique par AES-256 et AES-128 en fonction de la QoS	128
Figure 4-8 : Wireshark montrant la bascule entre AES-256 et AES-128	129
Figure 4-9 : Gigue mesurée pendant la transmission vidéo et renégociation automatique des clés de chiffrement	130
Figure 4-10 : Impact sur les ressources pendant les renégociations de clés.....	132
Figure 4-11 : Polarisation des photons pour l'échange de clés quantique	141
Figure 4-12 : Échange de clés via le protocole quantique BB84	142
Figure 4-13 : Correction d'erreurs et amplification de la confidentialité dans le processus d'échange de clés via le protocole quantique BB84.....	143
Figure 4-14 : Diagramme de séquence de communication SSH post-quantique avec protocole BB84.....	146
Figure 4-15 : Correspondance des bases entre serveur et client.....	148
Figure 4-16 : Taux d'erreur de bits quantiques (QBER) VS seuil critique	149
Figure 4-17 : Taux de génération de clés avec le protocole BB84.....	150
Figure 4-18 : Comparaison de l'utilisation du CPU en temps réel entre SSH classique et SSH avec BB84	152
Figure 4-19 : Comparaison de la consommation de mémoire entre SSH classique et SSH avec BB84.....	153

Liste des tableaux

Tableau 1-1 : Génération d'une clé publique quantique avec BB84	42
Tableau 3-1 : Description des Paramètres utilisés	69
Tableau 3-2 : Comparaison de la méthode «H-Rotation » en termes de performance et d'efficacité en environnement distribué avec bcrypt,scrypt et Argon 2.....	91
Tableau 3-3 : Comparaison de la méthode «H-Rotation » en termes de résistance aux attaques avec bcrypt,scrypt et argon 2.....	93
Tableau 3-4 : Comparaison de la méthode « H-Rotation » en termes de la complexité algorithmique avec bcrypt,scrypt et argon 2	95
Tableau 3-5 : Comparaison de la méthode proposée, bcrypt, scrypt, argon2 ; « H-Rotation » en termes d'attaque par force brute, attaque par dictionnaire, attaque différentielle, temps d'exécution et taille mémoire occupée.....	114
Tableau 4-1 : Comparaison de la méthode de négociation dynamique, avec SRTP, ZRTP, en termes de fluidité des algorithmes de chiffrement utilisés	133
Tableau 4-2 : Comparaison de la méthode de négociation dynamique, avec SRTP, ZRTP, en termes de l'adaptabilité aux conditions réseau.....	134
Tableau 4-3 : Comparaison de la méthode de négociation dynamique, avec SRTP, ZRTP, en termes d'impact sur les ressources	136
Tableau 4-4 : Résistance aux attaques de la méthode de négociation dynamique, avec SRTP, ZRTP	137
Tableau 4-5 : SSH classique vs SSH avec BB84 : différences de performances en termes d'utilisation du CPU, de la mémoire, du temps d'établissement de session, et du débit de transmission.....	154
Tableau 4-6 : Comparaison de la résistance aux attaques entre SSH classique et SSH avec BB84.....	155

Liste des algorithmes

Algorithme A. 1 : Description du fonctionnement de MD5.....	169
Algorithme A. 2 : Description du fonctionnement de SHA	172
Algorithme B. 1 : Description du fonctionnement de la méthode « H-Rotation ».	174
Algorithme B. 2 : Description du fonctionnement de la transformation Braille	176

Acronymes

3DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
AMFA	Adaptive Multi-Factor Authentication
BB84	Bennett-Brassard 1984
CA	Certificate Authority
CBC	Cipher Block Chaining
CTR	Counter Mode
DES	Data Encryption Standard
DH	Diffie-Hellman
DOS	Denial of Service
DSS	Digital Signature Standard
ECB	Electronic Codebook
ECC	Elliptic Curve Cryptography
ECDH	Elliptic-Curve Diffie-Hellman
ECDSA	Elliptic-Curve Digital Signature Algorithm
GCM	Galois/Counter Mode
GPU	Graphics Processing Unit
HMAC	Hash-based Message Authentication Code
IoT	Internet of Things
IV	Initialization Vector
KMS	Key Management System
MAC	Message Authentication Code
MD5	Message Digest Algorithm 5

MiTM	Man-in-The-Middle
OTP	One-Time Pad
PGP	Pretty Good Privacy
QBER	Quantum Bit Error Rate
QKD	Quantum Key Distribution
QOS	Quality of Service
QSSH	Quantum Secure Shell
RSA	Rivest–Shamir–Adleman
RSA-PSS	RSA Probabilistic Signature Scheme
RTP	Real-time Transport Protocol
SHA	Secure Hash Algorithm
SSL	Secure Sockets Layer
TLS	Transport Layer Security

Notations

$\forall$	Pour tout (quantificateur universel)
$\exists$	Il existe (quantificateur existentiel)
$\Leftrightarrow$	Implication (si une proposition est vraie, alors une autre l'est aussi)
$\subseteq$	Inclusion (un ensemble est inclus ou égal à un autre)
mod	Modulo (reste de la division d'un nombre par un autre)
$\oplus$	XOR (ou exclusif, opération logique)
$\in$	Appartenance (un élément appartient à un ensemble)
$\equiv$	Congruence (deux nombres ont le même reste en arithmétique modulaire)
$\cdot$	Multiplication dans un groupe ou un ensemble
$\emptyset$	Ensemble vide (aucun élément)
$\neq$	Différent (deux valeurs ne sont pas égales)
$\approx$	Approximation (deux valeurs sont proches mais pas égales)
$\varphi(n)$	Fonction d'Euler (nombre d'entiers inférieurs à n et premiers avec n)
Σ	Somme (indique la somme d'une série d'éléments)
log₂	Logarithme en base 2 (calcul de l'exposant pour obtenir un certain nombre à partir de 2)
	Concaténation (liaison de deux chaînes ou éléments)

Résumé

Dans un monde où les données représentent le cœur des échanges numériques, leur sécurisation face à des menaces toujours plus sophistiquées s'impose comme une priorité absolue. Cette thèse propose des méthodes avancées pour améliorer la sécurité des systèmes, principalement dans les domaines de l'authentification des utilisateurs et de la transmission des données. L'objectif principal de ce travail est de développer des solutions pratiques et efficaces capables de répondre aux besoins actuels tout en anticipant les menaces futures.

Dans le cadre de l'authentification, des contributions significatives ont été développées. La méthode « H-Rotation » introduit une approche novatrice basée sur une rotation dynamique et périodique des valeurs de hachage des phrases de passe. Ce procédé renforce la sécurité du stockage des phrases de passe et interdit leur exploitation par des attaquants, même en cas de compromission des bases de données. Cette méthode s'intègre facilement dans les systèmes existants sans dégrader les performances opérationnelles. Une autre contribution, la transformation Braille, propose une représentation unique des mots de passe en les convertissant dans un format difficilement réversible. Cette technique renforce considérablement la sécurité contre des attaques courantes, comme celles par force brute, dictionnaire ou rainbow tables, tout en assurant un stockage sécurisé et non réversible des données sensibles. Concernant la transmission des données, des améliorations significatives ont été apportées pour optimiser la sécurité et la qualité des échanges dans les environnements numériques. Le protocole TLS a été amélioré pour répondre aux exigences de la qualité de service, notamment dans les transmissions multimédias, tout en garantissant une protection renforcée contre les menaces actuelles. Par ailleurs, l'intégration d'algorithmes post-quantiques, tels que le protocole BB84 au protocole SSH, offre une résistance accrue aux attaques des ordinateurs quantiques, anticipant ainsi les futures avancées technologiques. Ces améliorations permettent non seulement d'assurer une transition harmonieuse vers des systèmes plus sécurisés, mais également de préserver la compatibilité avec les infrastructures existantes. Les différentes approches proposées ont été testées et validées par des expérimentations approfondies, utilisant des données réelles et simulées. Les résultats obtenus montrent une efficacité significative en termes de sécurité et de performance.

Mots-clés : authentification, protection des informations sensibles, transmission sécurisée des données, qualité de service, informatique quantique, protocole de communication.

Abstract

In a world where data constitutes the backbone of digital exchanges, securing it against increasingly sophisticated threats has become an absolute priority. This thesis proposes advanced methods to enhance system security, primarily in the fields of user authentication and data transmission. The main objective of this work is to develop practical and efficient solutions capable of addressing current needs while anticipating future threats. In the context of authentication, significant contributions have been developed. The H-Rotation method introduces an innovative approach based on the dynamic and periodic rotation of hash values for passphrase storage. This process strengthens the security of passphrase storage, rendering it unusable by attackers even in the event of database compromise. This method seamlessly integrates into existing systems without compromising operational performance. Another contribution, the Braille transformation, offers a unique representation of passwords by converting them into a format that is highly resistant to reversal. This technique significantly enhances security against common attacks, such as brute force, dictionary, or rainbow table attacks, while ensuring robust and irreversible storage of sensitive data. Regarding data transmission, significant advancements have been achieved to optimize security and the quality of exchanges in digital environments. The TLS protocol has been refined to meet quality of service requirements, particularly for multimedia transmissions, while ensuring enhanced protection against current threats. Additionally, the integration of post-quantum algorithms, such as the BB84 protocol, into SSH protocols provides increased resistance to quantum computer attacks, thereby anticipating future technological advances. These enhancements not only facilitate a smooth transition to more secure systems but also preserve compatibility with existing infrastructures.

The proposed approaches have been thoroughly tested and validated through extensive experimentation using both real and simulated data. The results demonstrate significant effectiveness in terms of security and performance.

Key-words: authentication, protection of sensitive information, secure data transmission, quality of service, quantum computing, communication protocols.

Introduction générale

Contexte de la thèse

La sécurité des données constitue aujourd'hui un pilier essentiel dans un monde où les technologies numériques dominent de plus en plus nos interactions quotidiennes et professionnelles. Les systèmes numériques évoluent à un rythme soutenu, mais cette évolution s'accompagne d'une augmentation rapide et inquiétante des cyberattaques, devenues plus complexes et sophistiquées. Les informations sensibles échangées chaque jour sur des plateformes numériques, les bases de données utilisées pour stocker ces informations, et les mécanismes d'authentification protégeant l'accès à ces systèmes font face à des vulnérabilités critiques qui menacent leur intégrité et leur confidentialité. Ces défis sont encore amplifiés par l'émergence de l'informatique quantique, une révolution technologique qui, bien que prometteuse, remet en question la robustesse des systèmes cryptographiques classiques. Les algorithmes de chiffrement largement utilisés, tels que RSA et Diffie-Hellman, risquent d'être rendus obsolètes par la puissance de calcul sans précédent des ordinateurs quantiques, capables de résoudre des problèmes mathématiques extrêmement complexes en un temps réduit [Diffie 76; Koblitz 87]. Cette évolution appelle à une révision complète des approches actuelles pour sécuriser les données, notamment dans les domaines de l'authentification et de la transmission des informations.

Les systèmes d'authentification, souvent fondés sur des mots de passe ou des clés cryptographiques, constituent une première ligne de défense contre les cyberattaques. Pourtant, ces systèmes sont régulièrement ciblés par des attaques comme celles par force brute, dictionnaire ou rainbow tables, exploitant des failles dans la conception ou la gestion des mots de passe. Les bases de données qui stockent ces informations sensibles deviennent également des cibles privilégiées, avec des compromissions fréquentes exposant les utilisateurs à des risques accrus de vol d'identité [Bertoni 10].

En parallèle, les protocoles de communication sécurisés, tels que TLS et SSH, ayant un rôle central dans la protection des échanges numériques, ne sont pas exempts de faiblesses. Des attaques sophistiquées, comme celles de type man-in-the-middle, tirent parti de lacunes dans la gestion des certificats numériques pour intercepter ou modifier des communications supposées sécurisées. Ces faiblesses, combinées à l'arrivée imminente des ordinateurs quantiques, menacent directement l'intégrité des systèmes actuels [Daemen 98 ; Bennett 84].

Pour relever ces défis, cette thèse propose des solutions concrètes et innovantes, en s'appuyant sur deux axes principaux : l'authentification sécurisée des utilisateurs et la transmission protégée des données.

Dans le domaine de l'authentification, deux approches novatrices ont été développées pour renforcer la sécurité des systèmes face aux attaques modernes. La méthode « H-Rotation » introduit un mécanisme basé sur une rotation périodique et dynamique des valeurs de hachage des phrases de passe, rendant pratiquement impossible leur exploitation par des attaquants, même en cas de compromission des bases de données. Cette méthode a été conçue pour s'intégrer facilement dans les systèmes existants, tout en garantissant un équilibre optimal entre sécurité et performance. En complément, la transformation Braille propose une représentation unique et non réversible des mots de passe, augmentant considérablement leur robustesse contre les attaques courantes, telles que celles par force brute ou dictionnaire, et assurant un stockage sécurisé et inviolable des données sensibles [Juels 99].

Pour la transmission des données, des avancées significatives ont été apportées au protocole TLS afin de répondre aux besoins spécifiques des environnements multimédias, où la QoS a un rôle important. Ces améliorations garantissent des échanges fluides et sécurisés, même dans des contextes exigeants. De plus, cette recherche a exploré l'intégration d'algorithmes post-quantiques, comme le protocole BB84, dans le protocole SSH. En exploitant les principes fondamentaux de la mécanique quantique, le protocole BB84 offre une distribution des clés sécurisée et capable de détecter toute tentative d'interception, renforçant ainsi la résilience des échanges cryptographiques contre les menaces des ordinateurs quantiques [Bennett 84 ; Shor 94]. Cette thèse propose des solutions adaptées aux besoins actuels tout en anticipant les évolutions technologiques futures. Elle apporte des réponses concrètes et pertinentes aux problématiques liées à la sécurité des données, tout en ouvrant la voie à des perspectives prometteuses pour l'évolution des technologies cryptographiques.

Problématique

La sécurité des systèmes d'information est au cœur des préoccupations dans un monde de plus en plus connecté, où les échanges numériques ayant un rôle primordial dans la société et l'économie. Malgré les avancées en matière de cryptographie, les mécanismes d'authentification et de transmission des données, essentiels pour garantir la confidentialité et l'intégrité des informations, sont confrontés à des vulnérabilités majeures. Ces failles,

combinées à l'émergence des technologies quantiques, menacent l'efficacité des protocoles et des algorithmes actuels, nécessitant une révision complète des approches existantes.

Dans le domaine de l'authentification, les méthodes traditionnelles reposant sur des mots de passe ou des clés cryptographiques présentent des limites évidentes. Les attaques par force brute, dictionnaire ou rainbow tables exploitent des mots de passe faibles ou mal conçus, exposant les systèmes à des compromissions fréquentes. Même avec l'utilisation des algorithmes de hachage avancés comme SHA-3, les bases de données restent des cibles privilégiées pour les cybercriminels. De plus, les solutions modernes telles que l'authentification multi-facteurs apportent une sécurité supplémentaire, mais elles augmentent aussi la complexité pour les utilisateurs et restent vulnérables à des attaques sophistiquées, comme le phishing ou les interceptions de facteurs secondaires [Koblitz 87].

Par ailleurs, la transmission des données, un pilier de la communication numérique, est également exposée à des vulnérabilités importantes. Les protocoles comme TLS et SSH, largement utilisés pour sécuriser les échanges, montrent leurs limites face aux menaces croissantes. Les attaques de type man-in-the-middle exploitent les failles dans la gestion des certificats et des clés publiques pour intercepter ou altérer les communications. Les algorithmes asymétriques classiques, tels que Diffie-Hellman et ECDSA, bien qu'ils aient servi de standards pendant de nombreuses années, sont aujourd'hui menacés par l'avènement des ordinateurs quantiques. Ces derniers, grâce à des algorithmes comme celui de Shor, peuvent casser ces systèmes très rapidement, compromettant ainsi des millions de connexions sécurisées chaque jour. Le protocole TLS, utilisé pour sécuriser une grande partie des échanges sur Internet, souffre d'une gestion imparfaite de la distribution des clés, laissant les communications vulnérables à des interceptions et manipulations. De même, SSH, bien qu'efficace pour les connexions à distance, repose sur des schémas cryptographiques qui risquent de devenir obsolètes à court terme.

Face à ces enjeux, cette thèse propose une approche globale visant à résoudre ces problématiques. Dans le domaine de l'authentification, des solutions comme « H-Rotation » et la transformation Braille sont développées pour renforcer la sécurité des phrases et des mots de passe, tout en garantissant un stockage inviolable des données sensibles. Ces contributions visent à contrer les attaques classiques tout en maintenant la convivialité et la compatibilité avec les systèmes existants.

Pour la transmission des données, cette recherche met l'accent sur les améliorations des protocoles TLS et SSH afin de répondre aux défis posés par les menaces actuelles et futures, tout en garantissant une compatibilité avec les systèmes existants. Le protocole TLS, essentiel

pour sécuriser les échanges sur Internet, bénéficie d'optimisations qui renforcent sa robustesse face aux attaques tout en améliorant la qualité de service. Ces ajustements permettent de maintenir un équilibre entre sécurité et performance, garantissant des transmissions fluides, même dans des environnements numériques exigeants. Par ailleurs, le protocole SSH, largement utilisé pour les connexions à distance, est enrichi par l'intégration d'algorithmes post-quantiques tels que le protocole BB84. Ce dernier, basé sur les principes de la mécanique quantique, assure une distribution sécurisée des clés et une détection immédiate de toute tentative d'interception. Les améliorations proposées garantissent une protection accrue contre les attaques quantiques et les menaces émergentes, tout en préparant les infrastructures actuelles à une transition fluide vers des technologies plus avancées [Gentry 09, TOUIL 22].

Contributions

Cette thèse constitue une réponse aux défis croissants en matière de cybersécurité, en apportant des contributions significatives dans les domaines de l'authentification et de la transmission des données. Fondées sur des approches novatrices, ces contributions visent à renforcer la sécurité des systèmes tout en garantissant leur compatibilité avec les infrastructures existantes et leur résilience face aux menaces émergentes. Les contributions principales sont :

La première contribution, intitulée « H-Rotation pour le stockage et la récupération des phrases de passe », présente un mécanisme innovant de stockage sécurisé des phrases de passe, conçu pour renforcer leur sécurité avant leur stockage dans la base de données, en introduisant une rotation dynamique des valeurs de hachage. Ce système rend considérablement plus difficiles les tentatives de récupération des phrases de passe par des attaquants, car les valeurs de hachage stockées dans la base de données ne correspondent pas directement à celles initialement générées. Lors de l'inscription d'un utilisateur, la phrase de passe est transformée en une valeur de hachage, modifiée ensuite selon une politique de rotation spécifique. La valeur résultante est alors stockée dans la base de données, tandis que l'utilisateur reçoit trois paramètres distincts nécessaires pour les futures authentifications. Lors de la phase d'authentification, l'utilisateur envoie sa phrase de passe accompagnée des trois paramètres communiqués précédemment par le système. Le serveur utilise ces informations pour effectuer une opération inverse et reconstituer la valeur de hachage d'origine, qu'il compare ensuite au hachage généré par le client. Si les deux valeurs correspondent, l'utilisateur est authentifié avec succès. « H-Rotation » offre un compromis optimal entre sécurité et efficacité, permettant de protéger les phrases de passe contre les attaques par force brute ou les compromissions de bases

de données, tout en garantissant une intégration fluide et transparente dans les systèmes existants.

La deuxième contribution, intitulée « Hachage sécurisé des mots de passe utilisant la transformation Braille », propose un schéma de hachage unique et sécurisé des mots de passe, utilisant une transformation Braille à six caractères. Le choix de la représentation Braille ajoute une complexité supplémentaire, rendant la tâche de casser un mot de passe beaucoup plus difficile pour un attaquant.

La méthode repose sur la génération d'une matrice de caractères Braille à partir du mot de passe original. Cette matrice est ensuite chiffrée à l'aide d'algorithmes cryptographiques, ce qui fait qu'en cas de compromission des bases de données, les mots de passe chiffrés restent difficiles à récupérer. De plus, ce schéma permet de garantir que même avec un accès aux données chiffrées, un attaquant ne peut pas facilement reconstruire le mot de passe original.

La troisième contribution, intitulée « Amélioration des protocoles de sécurité des communications », propose des améliorations au protocole TLS, intégrant des mécanismes avancés pour assurer un équilibre optimal entre la sécurité et la QoS. L'approche repose sur la prise en charge de longueurs de clés AES variables, permettant une flexibilité accrue dans le choix des clés tout en augmentant la robustesse des échanges face aux menaces.

Un aspect clé de ce système est la gestion de la gigue, qui représente la variation de la latence au sein des transmissions. Le système surveille en permanence cette variation, et si elle dépasse le seuil critique de 40 millisecondes, une clé de chiffrement plus volumineuse est utilisée pour garantir une qualité de service optimale. Cette adaptation permet de maintenir une transmission fluide des données, même dans des environnements exigeants. En revanche, lorsque la gigue est maîtrisée et reste inférieure à ce seuil, le protocole passe à une clé moins volumineuse, renforçant ainsi la sécurité des échanges.

Ces optimisations permettent non seulement de réduire la latence des échanges de clés, mais aussi de garantir une compatibilité avec les versions existantes de TLS, assurant ainsi une transition fluide vers des standards plus avancés. En conciliant sécurité et QoS, cette approche répond efficacement aux contraintes des environnements numériques modernes tout en préparant le terrain pour une adoption élargie des nouvelles technologies de chiffrement.

La quatrième contribution, intitulée « La sécurité du partage des clés publiques du protocole SSH à l'ère post-quantique », se concentre sur la sécurisation des clés publiques dans le protocole SSH, en tenant compte des menaces posées par les ordinateurs quantiques. En intégrant des algorithmes post-quantiques tels que le protocole BB84, cette solution permet de garantir une distribution des clés résistante aux attaques quantiques.

Le protocole BB84, reposant sur les principes de la mécanique quantique, offre une détection immédiate des tentatives d'interception, renforçant ainsi la robustesse des connexions SSH contre les attaques de type man-in-the-middle. Cette approche s'accompagne de schémas cryptographiques résilients qui ne dépendent pas des hypothèses classiques sur la difficulté des problèmes mathématiques, garantissant ainsi une sécurité durable dans un contexte technologique en évolution rapide.

Organisation du manuscrit

Ce manuscrit est structuré en quatre chapitres :

Le chapitre 1 : nous présentons les bases de la cryptographie en détaillant les algorithmes de chiffrement symétrique tels que DES et AES, ainsi que les algorithmes asymétriques comme RSA et Diffie-Hellman. Nous introduisons également les fonctions de hachage, notamment MD5 et SHA, ainsi que les principes fondamentaux de la cryptographie quantique avec le protocole BB84. Enfin, nous examinons les protocoles de communication sécurisés, tels que TLS et SSH, qui servent de fondation aux travaux de cette thèse.

Le chapitre 2 : nous explorons les outils mathématiques nécessaires à la compréhension et au développement des algorithmes cryptographiques. Nous y couvrons la théorie des groupes, l'algèbre linéaire et la théorie des nombres, avec une attention particulière portée aux logarithmes discrets et à la factorisation.

Le chapitre 3 : nous présentons les contributions liées à l'authentification et au stockage sécurisé des données. Nous décrivons en détail la méthode « H-Rotation », qui propose une gestion innovante des phrases de passe, ainsi que la transformation Braille, une méthode avancée pour le hachage sécurisé des mots de passe. Nous analysons les performances et la robustesse de ces approches à travers des résultats expérimentaux.

Le chapitre 4 : nous présentons les contributions dédiées à la sécurisation des communications. Nous détaillons les améliorations apportées au protocole TLS, visant à concilier la sécurité et la QoS, et explorons l'intégration d'algorithmes post-quantiques comme BB84 dans le protocole SSH pour anticiper les menaces liées à l'informatique quantique. Nous évaluons également l'impact de ces solutions sur les performances et les ressources.

Enfin, nous concluons ce manuscrit par une conclusion générale, qui récapitule les principaux résultats obtenus et propose des perspectives pour des recherches futures. Les annexes, contenant des informations complémentaires sur les algorithmes et les expériences, ainsi que la liste des publications associées à ce travail.

Chapitre 1 : Principes fondamentaux de la cryptographie

1.1 Introduction

La préservation de la confidentialité des données échangées en ligne s'impose aujourd'hui comme une priorité dans le cadre de la transformation numérique mondiale. Les cyberattaques se multiplient, ciblant tant les entreprises que les utilisateurs individuels, avec des menaces allant de la violation de données personnelles à l'interception de transactions financières. En réponse à ces menaces croissantes, la cryptographie a un rôle fondamental pour garantir la confidentialité, l'intégrité et l'authenticité des informations. Elle est omniprésente dans nos vies, qu'il s'agisse de transactions bancaires, de communications sécurisées via des applications, ou de connexions à distance entre machines.

Parmi les protocoles les plus utilisés, TLS et SSH sont largement déployés pour sécuriser respectivement les échanges web et les connexions à distance. TLS garantit la confidentialité et l'intégrité des échanges via HTTPS, tandis que SSH est utilisé pour les connexions sécurisées entre ordinateurs et serveurs, notamment pour la gestion à distance [Dierks 08]. Ces protocoles reposent sur des algorithmes de cryptographie asymétrique, tels que RSA et ECDSA, pour l'échange de clés et l'authentification, et sur des algorithmes symétriques comme AES pour chiffrer les données une fois les clés échangées. Cependant, ces systèmes présentent des vulnérabilités face aux attaques sophistiquées, telles que les attaques MiTM, qui exploitent les failles dans l'échange de clés, compromettant ainsi la sécurité des communications [Jager 14].

Les fonctions de hachage, essentielles pour garantir l'intégrité des données, ont également un rôle important dans les systèmes cryptographiques modernes. Des algorithmes comme SHA-3, conçus pour renforcer la robustesse face aux attaques de cryptanalyse, permettent de sécuriser des éléments sensibles tels que les mots de passe et les signatures numériques. Ces techniques, bien qu'efficaces, doivent être associées à des pratiques solides de gestion des données pour maximiser leur efficacité. Avec l'avènement des ordinateurs quantiques, de nouvelles menaces pèsent sur la sécurité des systèmes cryptographiques actuels. L'algorithme de Shor, capable de résoudre efficacement des problèmes mathématiques complexes, comme la factorisation des grands nombres, pourrait rapidement rendre obsolètes des algorithmes comme RSA ou Diffie-Hellman, sur lesquels reposent la majorité des protocoles cryptographiques utilisés aujourd'hui [Bernstein 17, Shor 99]. Cette menace imminente pousse la recherche à se tourner vers la

cryptographie post-quantique, qui vise à développer des algorithmes résistants aux capacités de calcul des ordinateurs quantiques. Parmi ces algorithmes, ceux basés sur les réseaux euclidiens et les codes correcteurs d'erreurs sont les plus prometteurs, car ils offrent une sécurité accrue tout en restant compatibles avec les systèmes actuels [Peikert 16].

En parallèle, des approches plus révolutionnaires comme la distribution de clés quantiques se basent sur les principes de la mécanique quantique pour garantir une sécurité théoriquement imparable. Le protocole BB84 est l'un des plus utilisés dans ce domaine et permet d'assurer que toute tentative d'interception des clés modifie leur état, alertant ainsi immédiatement les parties communicantes [Lo 14].

Outre les préoccupations quant à l'informatique quantique, les algorithmes symétriques tels qu'AES continuent d'occuper un rôle clé dans la protection des données après l'échange de clés. AES, bien que toujours considéré comme sûr, doit être implémenté avec des tailles de clés suffisantes pour résister aux attaques par force brute. Par ailleurs, les techniques d'accélération matérielle et d'optimisation des performances sont devenues essentielles pour maintenir un haut niveau de sécurité sans compromettre l'efficacité des systèmes [Luk 09].

Ce chapitre se concentre sur l'exploration approfondie des différentes dimensions de la cryptographie, en abordant les concepts fondamentaux qui sous-tendent les systèmes cryptographiques modernes.

1.1.1 Contexte et enjeux de la cryptographie moderne

La cryptographie moderne représente une branche importante de la sécurité informatique, adaptée aux besoins des sociétés de l'information. Alors que la cryptographie classique s'appuyait principalement sur des techniques manuelles et des systèmes symétriques simples, la cryptographie moderne a évolué pour intégrer des approches mathématiques complexes et des algorithmes sophistiqués [Stallings 17]. Cette transition a été stimulée par l'essor d'internet et des transactions numériques, qui nécessitent des systèmes de sécurité robustes et fiables. L'une des distinctions majeures dans la cryptographie moderne réside dans la séparation entre les systèmes symétriques et asymétriques. Les systèmes de chiffrement symétrique, tels que DES ou AES, reposent sur une clé partagée entre l'expéditeur et le destinataire [Schneier 07]. Ces systèmes, bien que performants pour chiffrer des volumes importants de données, présentent des défis logistiques en termes de gestion des clés, notamment lorsque des millions d'utilisateurs doivent échanger des informations sécurisées. Pour pallier cette limitation, la cryptographie asymétrique apparaît avec des systèmes comme RSA, introduit par Rivest,

Shamir, et Adleman en 1978 [RSA 78]. RSA utilise deux clés distinctes : une clé publique pour chiffrer les données et une clé privée pour les déchiffrer. Cette innovation permet de sécuriser des communications sans qu'il soit nécessaire de partager une clé secrète au préalable. Le RSA repose sur la difficulté de factoriser de grands nombres premiers, un problème réputé difficile à résoudre par les ordinateurs classiques [Bernstein 17]. Cependant, les progrès dans le domaine de l'informatique quantique menacent certains fondements de la cryptographie actuelle. L'algorithme de Shor, par exemple, est capable de factoriser des grands nombres beaucoup plus rapidement qu'avec les ordinateurs traditionnels, ce qui rend les systèmes asymétriques tels que RSA vulnérables [Shor 94]. En réponse à ces menaces, la cryptographie post-quantique s'efforce de développer des algorithmes résistants aux attaques quantiques, basés sur des problèmes mathématiques non résolus, comme la difficulté de résoudre certaines équations sur les réseaux euclidiens. En plus de ces enjeux techniques, la cryptographie moderne est encadrée par des considérations politiques et juridiques. Les gouvernements et les organisations internationales ont développé des cadres réglementaires pour encadrer l'usage des outils cryptographiques, notamment dans les domaines de la défense, de la finance et des télécommunications [Solove 11]. Les tensions entre sécurité nationale et vie privée demeurent au cœur des débats autour de la cryptographie, comme en témoignent les discussions récentes sur l'accès des autorités aux données chiffrées dans le cadre d'enquêtes criminelles [Abelson 15]. Ainsi, la cryptographie moderne se situe à l'intersection des mathématiques avancées, de l'informatique et des problématiques sociales et politiques. Face aux nouvelles menaces technologiques comme l'informatique quantique, les chercheurs et les gouvernements doivent collaborer pour maintenir des systèmes cryptographiques sécurisés tout en respectant les droits fondamentaux des utilisateurs. La figure 1-1 illustre l'architecture de la cryptographie moderne.

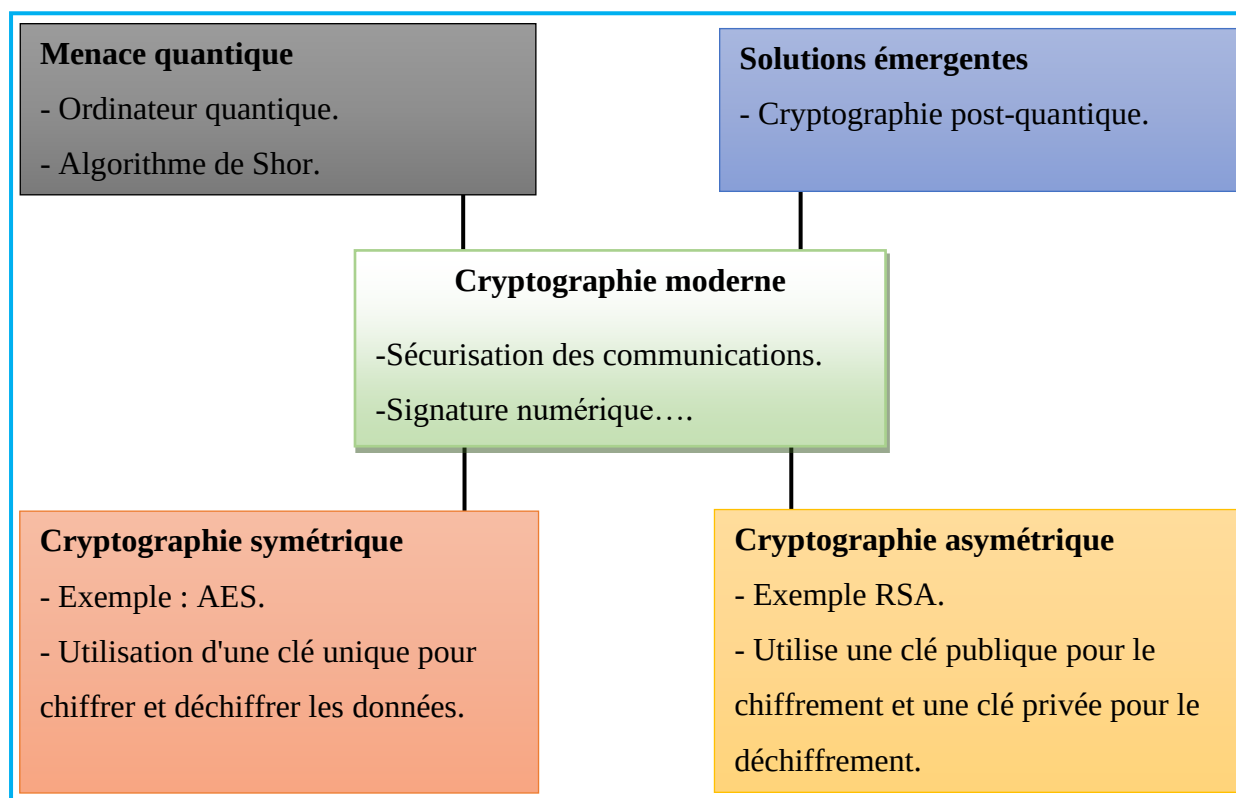


Figure 1- 1 : Architecture de la cryptographie moderne

1.1.2 Importance de la sécurisation des échanges numériques

La sécurisation des échanges numériques est très importante aujourd'hui, avec la croissance des données, qui demande des protections plus avancées. En effet, les données échangées via des réseaux numériques, qu'il s'agisse de transactions financières, d'e-mails professionnels ou d'informations personnelles, sont soumises à des risques croissants de cyberattaques, de vols d'identité et de fraudes. Afin de préserver la confidentialité, l'intégrité et la disponibilité des informations, les entreprises et les gouvernements mettent en place divers mécanismes de protection, notamment le chiffrement, l'authentification multifactorielle et, plus récemment, la technologie blockchain.

Le chiffrement est l'une des techniques les plus utilisées pour protéger les échanges numériques. Il consiste à rendre les données illisibles pour toute personne non autorisée en les convertissant en un code incompréhensible. Par exemple, l'algorithme de chiffrement AES est aujourd'hui largement utilisé pour sécuriser les communications sensibles sur Internet. En plus du chiffrement, les protocoles de sécurité comme SSH assurent la protection des connexions à distance, en permettant une communication sécurisée entre un client et un serveur via des mécanismes de chiffrement asymétrique. SSH est essentiel dans les environnements

administratifs, notamment pour la gestion des serveurs, où il réduit considérablement le risque d'accès non autorisé [Barrett 2005].

Une autre méthode importante dans la sécurisation des données est l'utilisation de fonctions de hachage pour protéger les mots de passe. Les fonctions de hachage, telles que SHA-256 ou bcrypt, transforment les mots de passe en une suite unique de caractères. Contrairement au chiffrement, le hachage est un processus à sens unique : il est extrêmement difficile de retrouver le mot de passe original à partir de la valeur hachée, ce qui ajoute une couche de sécurité supplémentaire en cas de vol de données [Menezes 18]. Ces techniques sont largement adoptées pour protéger les bases de données contenant des informations sensibles, telles que les identifiants des utilisateurs.

1.1.3 Problématiques liées à l'évolution des menaces

Les menaces pesant sur les systèmes de sécurité utilisant les protocoles SSH et SSL/TLS évoluent constamment, avec des cyberattaques de plus en plus sophistiquées ciblant ces technologies. Malgré les améliorations apportées aux protocoles, diverses vulnérabilités subsistent, principalement en raison de mauvaises configurations, de failles dans les algorithmes de chiffrement ou de pratiques d'authentification inadéquates.

Menaces liées à l'authentification

- **Faiblesse des mots de passe** : l'utilisation de mots de passe faibles ou réutilisés constitue l'une des principales causes de compromission des systèmes. Les attaques par force brute et les attaques par hameçonnage exploitent ces faiblesses pour accéder aux comptes des utilisateurs. La réutilisation de mots de passe sur plusieurs services expose également les utilisateurs à des violations en cascade, où la compromission d'un seul compte peut entraîner un accès non autorisé à d'autres systèmes.
- **Attaques par hameçonnage** : les méthodes d'authentification reposant uniquement sur des mots de passe sont vulnérables aux attaques d'hameçonnage, où les attaquants trompent les utilisateurs pour qu'ils divulguent volontairement leurs identifiants. Ces attaques deviennent de plus en plus sophistiquées, avec des pages de phishing imitant parfaitement les sites légitimes, ce qui augmente le taux de réussite de ces campagnes [Dawabsheh 22].

Menaces liées à SSL/TLS

- **Attaques MiTM** : malgré l'utilisation de SSL/TLS pour sécuriser les communications, des attaques MiTM peuvent survenir lorsque les certificats ne sont pas correctement validés ou si un attaquant parvient à intercepter les communications entre un client et un serveur. Cela permet à l'attaquant de visualiser, d'altérer ou d'injecter des données malveillantes dans la communication, sans être détecté [Dacosta 12].
- **Faiblesses des anciens algorithmes de chiffrement** : certaines versions plus anciennes de SSL et TLS, notamment TLS 1.0 et SSL 3.0, présentent des vulnérabilités. Par exemple, des attaques comme POODLE¹ exploitent des failles dans les mécanismes de remplissage de SSL, permettant à un attaquant de déchiffrer des informations sensibles, même si elles sont transmises via une connexion sécurisée [Subhan 23].
- **Compromission des certificats numériques** : si une autorité de certification (CA) est compromise, un attaquant peut émettre des certificats frauduleux et se faire passer pour des services légitimes. Cela permet de tromper les utilisateurs en leur faisant croire qu'ils sont connectés à des sites authentiques, alors qu'ils partagent en réalité leurs données avec un attaquant. L'incident de DigiNotar² en 2011 est un exemple notable : des certificats frauduleux ont été utilisés pour espionner des communications sécurisées [Kurnia Putra 18].

Menaces liées à SSH

- **Attaques par force brute** : les cybercriminels exploitent fréquemment cette méthode pour tenter d'accéder à des systèmes via SSH. Ils testent un grand nombre de combinaisons de noms d'utilisateur et de mots de passe jusqu'à en trouver une correcte. Cette technique permet d'obtenir un accès non autorisé à des serveurs critiques si les systèmes ne sont pas suffisamment protégés.
- **Compromission des clés SSH** : l'authentification par clés SSH est couramment utilisée pour sécuriser l'accès à distance. Cependant, si une clé privée est compromise, un attaquant peut obtenir un accès sans déclencher d'alerte. Une mauvaise gestion des clés, comme leur stockage non sécurisé ou l'absence de rotation régulière, augmente considérablement la probabilité de ce type d'incident [Andrews 2021].

¹ POODLE : est une attaque exploitant une vulnérabilité dans le protocole SSL 3.0, permettant à un attaquant de déchiffrer du contenu chiffré et de compromettre la confidentialité des communications sécurisées.

² DigiNotar était une autorité de certification néerlandaise qui a fait faillite après avoir subi une attaque en 2011, entraînant la compromission de plusieurs certificats SSL, ce qui a gravement affecté la sécurité des communications en ligne.

1.2 Cryptographie symétrique

La cryptographie symétrique constitue un pilier fondamental dans la sécurisation des systèmes d'information. Elle repose sur l'utilisation d'une seule clé partagée entre les parties prenantes pour le chiffrement et le déchiffrement des données, comme illustré dans la figure 1-2. Cette clé unique rend les données illisibles lors du chiffrement, puis les restaure dans leur état initial lors du déchiffrement. La confidentialité des informations est ainsi garantie, car seules les entités autorisées, détenant la clé de déchiffrement correspondante, peuvent accéder au contenu des données [Snialata 21]. Les algorithmes symétriques, tels que DES et AES, utilisent des méthodes complexes de substitution et de permutation pour garantir une sécurité élevée tout en permettant un traitement rapide. Cependant, l'un des principaux défis de cette approche réside dans la distribution sécurisée des clés, essentielle pour prévenir toute interception malveillante. La gestion de la clé partagée, ainsi que sa qualité et sa longueur, tient un rôle crucial dans la résistance aux attaques, notamment celles de cryptanalyse ou de force brute. Utilisée dans divers domaines, tels que les communications internet, le stockage de données sensibles ou les transactions financières, la cryptographie symétrique s'adapte aux nouvelles menaces grâce à des modes d'opération variés (bloc de code électronique, chaîne de blocs de chiffrement, mode compteur, etc.), chacun étant adapté à des scénarios spécifiques. Malgré les défis liés à la gestion des clés, cette forme de cryptographie reste un outil incontournable pour garantir la confidentialité et la confiance dans le partage d'informations sensibles [Ryabko 15].



Figure 1- 2 : Concept général de la cryptographie symétrique

1.2.1 Chiffrement DES /3DES

DES est un algorithme de chiffrement symétrique qui a été largement utilisé pour sécuriser les données. Il repose sur l'utilisation d'une clé unique pour chiffrer et déchiffrer des informations de manière sécurisée. Il a été conçu pour offrir un compromis entre l'efficacité et la sécurité, tout en étant facilement implémentable dans des environnements logiciels et matériels. Bien

qu'il ait représenté une avancée majeure dans le domaine de la cryptographie à son époque, les progrès technologiques et les méthodes d'attaque plus sophistiquées ont progressivement révélé ses limites [Paar 10]. Face à ces limitations, notamment en raison de la faible longueur de sa clé, une version améliorée, appelée 3DES, a été développée. Cette méthode renforce la sécurité en appliquant trois cycles successifs de chiffrement avec deux ou trois clés distinctes [Adam 19]. Ainsi, 3DES prolonge la durée de vie de l'algorithme initial en rendant les attaques par force brute beaucoup plus difficiles à réaliser.

1.2.1.1 Structure DES/3DES (substitution et permutation)

Le DES utilise une clé de 64 bits, dont 56 bits sont effectifs, les 8 derniers bits étant réservés comme bits de parité pour la vérification d'erreur de la clé. Ces bits de parité garantissent certaines propriétés, telles que l'impairité des 8 premiers bits. Ainsi, DES dispose de 2^{56} clés, soit environ 720 milliards de combinaisons, un nombre impressionnant à l'époque, mais insuffisant face aux capacités actuelles. La structure de DES a été controversée depuis sa conception. Bien qu'il ne soit pas trivial, la plupart des opérations impliquées dans l'algorithme sont des fonctions linéaires, ce qui les rend moins résistantes à certaines attaques théoriques. La sécurité de DES repose principalement sur des opérations non linéaires, exécutées en grande partie par huit S-box, dont la conception a été supervisée par la NSA. Ce rôle de la NSA a suscité des préoccupations quant à l'objectivité du processus de développement. La figure 1-3 illustre la structure de DES. Au-delà de sa conception, DES repose sur un réseau de Feistel comprenant 16 tours de transformations, chaque tour appliquant une sous-clé spécifique dérivée de la clé principale. Les S-box, principales sources de non-linéarité, ayant un rôle crucial pour introduire de la complexité et contrecarrer des attaques comme l'analyse différentielle. Cependant, l'algorithme reste vulnérable aux attaques par force brute en raison de la longueur limitée de la clé. À la fin des années 1990, des machines comme Deep Crack, développée par « Electronic Frontier Foundation³ », ont démontré que DES pouvait être cassé en moins de 24 heures. Ces limitations ont conduit à la création de variantes comme 3DES, qui applique DES trois fois avec différentes clés.

³ Electronic Frontier Foundation est une organisation à but non lucratif fondée en 1990, qui défend les droits numériques et la liberté d'expression dans le contexte des technologies.

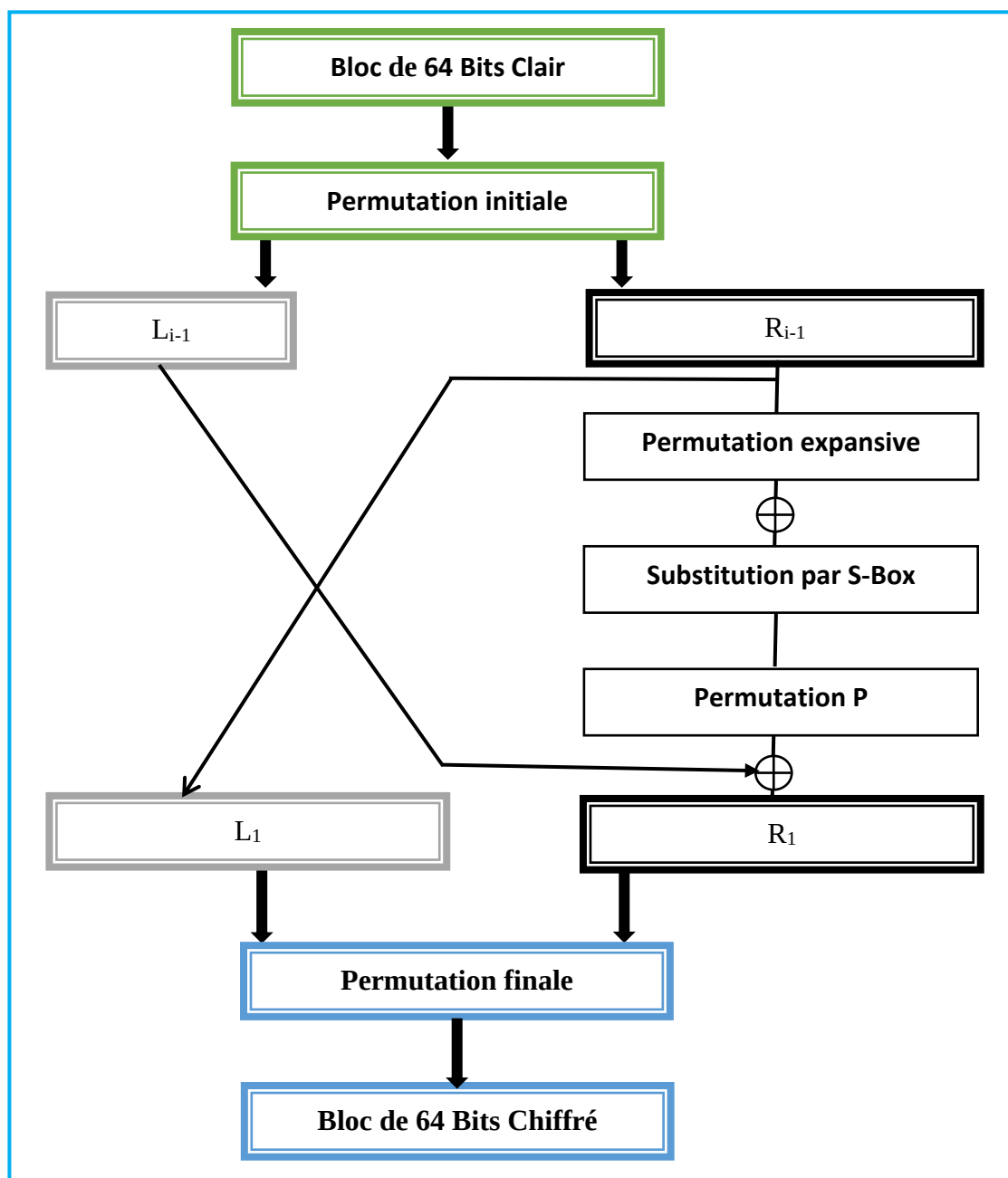
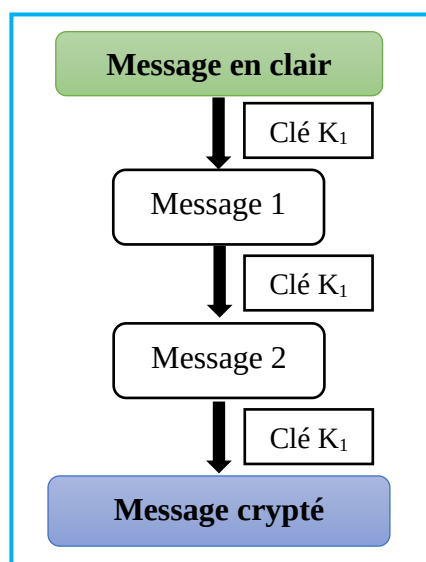


Figure 1- 3 : Structure DES

3DES repose sur une série de substitutions et de permutations similaires à celles utilisées dans DES, mais appliquées trois fois pour renforcer la sécurité. Le processus commence par la division des données en blocs de 64 bits, qui sont ensuite soumis à des transformations basées sur des S-boxes. Ces boîtes de substitution remplacent chaque bloc de 6 bits par 4 bits, introduisant une non-linéarité cruciale pour résister aux attaques analytiques. Les bits issus des S-boxes subissent ensuite une permutation spécifique, réorganisant leur ordre pour garantir une meilleure diffusion dans le bloc, comme illustré dans la figure 1-4.

**Figure 1- 4** : Structure 3DES

1.2.1.2 Limites de sécurité

Le DES, bien qu'ayant marqué une avancée significative dans la cryptographie lors de son introduction en 1977, présente plusieurs limites de sécurité qui le rendent inadapté aux exigences actuelles. La vulnérabilité principale de DES réside dans la longueur de sa clé, avec une clé de 56 bits, le nombre de combinaisons possibles, soit 2^{56} clés, est aujourd'hui trivial à explorer pour une attaque par force brute. En 1997, une attaque par force brute menée par un groupe de machines connectées à Internet a réussi à casser DES en seulement trois semaines [Menezes 97], démontrant ainsi sa faiblesse face aux progrès technologiques. Dès lors, DES est considéré comme obsolète.

Une autre faille de DES concerne sa résistance limitée face aux attaques différentielles et linéaires. Bien que les S-box aient été conçues pour résister à ces types d'attaques, la puissance de calcul moderne permet de réaliser des attaques cryptanalytiques plus sophistiquées qui compromettent l'intégrité de l'algorithme [Biham 91].

Pour pallier ces limites, le 3DES a été développé, utilisant trois cycles de chiffrement avec deux ou trois clés distinctes. Cela permet d'augmenter la longueur effective de la clé à 112 bits dans le cas de deux clés, et à 168 bits avec trois clés. Cependant, bien que 3DES offre une sécurité accrue par rapport à DES, il présente deux inconvénients majeurs. Premièrement, l'augmentation de la longueur de la clé et des cycles de chiffrement se traduit par une performance plus lente, rendant 3DES peu adapté aux applications nécessitant un traitement rapide des données. Deuxièmement, 3DES n'est pas à l'abri des attaques MiTM, qui permettent de réduire considérablement la complexité d'une attaque par force brute [van Oorschot 91].

De plus, les récentes avancées en cryptographie quantique accentuent la vulnérabilité de 3DES. L'algorithme de Grover, utilisé dans un ordinateur quantique, pourrait réduire l'efficacité d'une clé de 112 bits à un niveau de sécurité équivalent à une clé de 56 bits, ce qui compromet sérieusement la sécurité de 3DES face à ces nouvelles menaces [Grover 96].

Les failles structurelles de DES et les limitations inhérentes à 3DES, combinées aux exigences de sécurité modernes et aux menaces émergentes comme les attaques quantiques, expliquent pourquoi ces algorithmes ne sont plus utilisés aujourd'hui pour des applications sensibles. Des solutions comme AES offrent désormais des garanties de sécurité nettement supérieures dans l'écosystème cryptographique moderne.

1.2.2 Chiffrement AES

AES a été conçu pour surmonter les limites de sécurité et de performance de DES et 3DES. Il utilise des tailles de clés de 128, 192 ou 256 bits, ce qui le rend beaucoup plus résistant aux attaques par force brute [Daemen 01]. Contrairement à DES, AES repose sur une structure de réseau de substitution-permutation et traite des blocs de 128 bits à travers plusieurs tours de transformations non linéaires, garantissant une sécurité accrue. La conception d'AES s'appuie sur une cryptanalyse avancée, rendant l'algorithme résistant aux attaques linéaires et différentielles [Ferguson 01]. Grâce à sa flexibilité et à sa rapidité, AES est devenu l'algorithme de chiffrement de référence pour les systèmes de sécurité modernes. Il est largement utilisé dans des applications variées, telles que le commerce électronique, les réseaux sans fil et les services bancaires en ligne.

1.2.2.1 Structure AES

AES transforme les données d'entrée par blocs de 128 bits à travers plusieurs tours de transformations. Le nombre de tours dépend de la longueur de la clé, qui peut être de 128, 192 ou 256 bits, respectivement associés à 10, 12 ou 14 tours. Chaque tour comporte quatre étapes principales : SubBytes, où les octets sont substitués via une table non linéaire (S-box) ; ShiftRows, qui décale les lignes de la matrice de données pour introduire de la diffusion ; MixColumns, qui mélange les colonnes à travers une transformation mathématique pour renforcer la complexité ; et enfin AddRoundKey, où le bloc de données est combiné par XOR avec une sous-clé spécifique dérivée de la clé principale (figure 1-5). Le processus commence par un ajout de clé (AddRoundKey) initial, suivi des transformations à chaque tour. Lors du

dernier tour, l'étape MixColumns est sautée, ce qui permet de finaliser le chiffrement. L'algorithme AES repose sur deux principes cryptographiques majeurs : la confusion, où la relation entre la clé et le texte chiffré est rendue complexe grâce aux substitutions non linéaires, et la diffusion, assurée par les étapes de décalage et de mélange, qui propagent les modifications sur tout le bloc de données. L'efficacité d'AES tient également à son processus d'expansion de clé, qui génère des sous-clés pour chaque tour à partir de la clé principale. Grâce à cette architecture, AES est non seulement résistant aux attaques par force brute, mais aussi aux attaques cryptanalytiques telles que les attaques différentielles et linéaires.

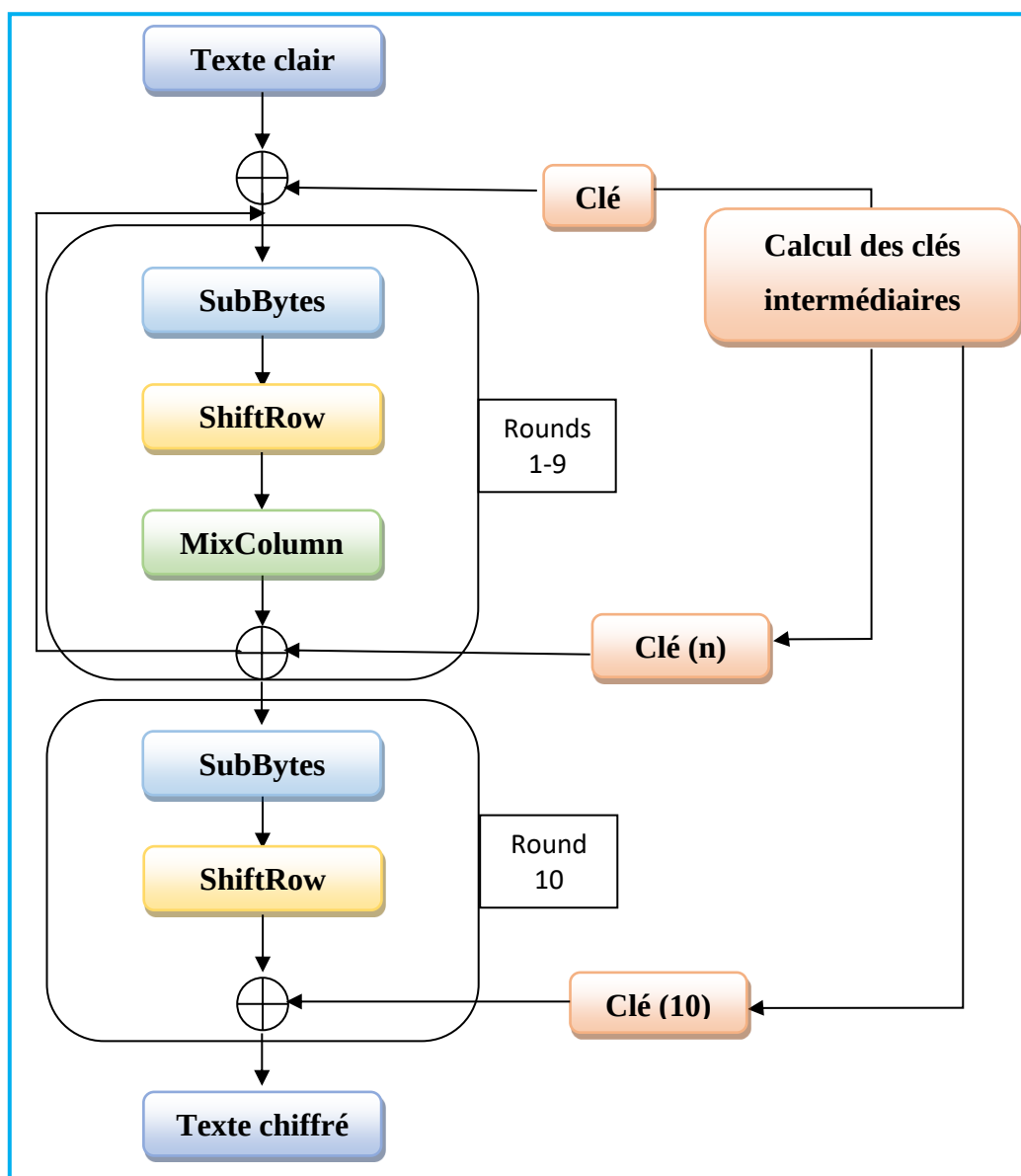


Figure 1- 5 : Structure AES

1.2.2.2 Vulnérabilités potentielles

AES est largement reconnu pour sa sécurité et son efficacité, mais, comme tout algorithme de chiffrement, il présente certaines vulnérabilités potentielles. Ces vulnérabilités ne proviennent pas directement de l'algorithme lui-même. Elles résultent souvent de son implémentation ou des avancées technologiques. Certaines vulnérabilités d'AES découlent de l'environnement dans lequel il est utilisé. Voici deux exemples courants :

Attaques par canal auxiliaire : les attaques par canal auxiliaire sont des attaques indirectes qui exploitent les informations divulguées par le matériel ou les logiciels exécutant AES, comme la consommation d'énergie, les temps d'exécution ou les émissions électromagnétiques. Ces informations peuvent être utilisées pour déduire des parties de la clé de chiffrement sans casser directement l'algorithme. Un exemple notable est l'analyse de puissance. Cette technique permet de récupérer des clés AES en surveillant la consommation énergétique d'un appareil pendant le chiffrement [Kocher 99].

Les failles liées à l'implémentation : les erreurs d'implémentation sont une autre source majeure de vulnérabilité. Si AES est mal implémenté, par exemple, avec des variations dans les temps de calcul, cela peut être exploité pour des attaques dites temporaires. Une implémentation correcte doit garantir que chaque opération prend un temps constant, indépendamment des données traitées [Bernstein 05].

1.3 Cryptographie asymétrique

La cryptographie asymétrique repose sur l'utilisation d'une paire de clés distinctes : une clé publique, utilisée pour chiffrer les messages, et une clé privée, nécessaire pour les déchiffrer. Cela est illustré dans la figure 1-6. Contrairement aux systèmes symétriques, où une seule clé est utilisée pour le chiffrement et le déchiffrement, ce modèle asymétrique assure une sécurité renforcée, particulièrement utile dans les échanges en ligne. Parmi les algorithmes les plus connus, on trouve RSA et ECC, qui sont utilisés pour garantir la confidentialité des données et, dans certains cas, leur intégrité [Miller 85].



Figure 1- 6 : Protection cryptographique : clés publiques et privées

1.3.1 Chiffrement RSA

RSA est un algorithme de cryptographie asymétrique qui repose sur la difficulté de factoriser de grands nombres premiers. Son fonctionnement repose sur la génération de deux clés : une clé publique et une clé privée. Lors de l'étape de chiffrement, l'émetteur utilise la clé publique du récepteur pour chiffrer un message. Seul le récepteur, en possession de la clé privée, peut le déchiffrer [RSA 78].

Bien qu'efficace et largement utilisé, le RSA présente certaines limites. L'une des principales faiblesses est sa dépendance à la taille des clés. Pour garantir un haut niveau de sécurité, des clés de grande longueur (2048 bits ou plus) sont nécessaires. Cependant, cela augmente le temps de calcul pour le chiffrement et le déchiffrement, ce qui réduit les performances, notamment dans des applications à forte intensité de données.

De plus, le RSA est vulnérable à certaines attaques, comme l'attaque par facteur commun ou l'attaque par oracle, où des informations partielles divulguées par le système sont exploitées pour compromettre la sécurité. Enfin, par rapport à d'autres algorithmes comme l'ECC⁴, le RSA est moins performant. L'ECC offre une sécurité équivalente avec des clés beaucoup plus courtes, ce qui le rend plus rapide et moins exigeant en termes de ressources. [Miller 85].

1.3.2 Échange de clés Diffie-Hellman

Le protocole d'échange de clés Diffie-Hellman permet à deux parties, communiquant sur un canal non sécurisé, de créer une clé secrète partagée sans jamais échanger directement cette clé. Son principe repose sur la difficulté de résoudre le problème du logarithme discret, une tâche complexe lorsqu'elle est appliquée à de grands nombres. Chaque partie choisit un secret privé et calcule une valeur publique à partir de ce secret et de paramètres publics. Ces valeurs

⁴ ECC est une méthode de cryptographie à clé publique basée sur les propriétés des courbes elliptiques, offrant une sécurité équivalente à celle des systèmes traditionnels comme RSA, mais avec des clés beaucoup plus courtes, ce qui permet des calculs plus rapides et une utilisation plus efficace des ressources.

publiques sont ensuite échangées entre les deux parties. À partir des informations reçues, chaque partie génère une clé commune identique, qu'elles utilisent ensuite pour chiffrer leurs communications. Toutefois, ce protocole présente certaines faiblesses. La principale vulnérabilité de Diffie-Hellman réside dans les attaques MiTM, sans authentification préalable. Un attaquant peut intercepter les valeurs échangées, se positionner entre les deux parties et établir deux clés distinctes avec chacune d'elles, tout en se faisant passer pour le correspondant légitime. De plus, l'utilisation de paramètres publics inadéquats, comme des nombres premiers trop petits, peut considérablement réduire la sécurité de l'algorithme, le rendant vulnérable aux attaques par factorisation ou calcul de logarithmes discrets.

Malgré ces vulnérabilités, ce protocole reste un pilier de la cryptographie moderne, notamment lorsqu'il est combiné à des mécanismes d'authentification robustes qui éliminent les risques d'attaques MiTM [Diffie 76].

1.4 Fonction de hachage

Les fonctions de hachage sont des éléments essentiels de la cryptographie moderne. Elles permettent de générer une empreinte numérique unique d'un message ou d'une donnée, sous forme de bits de longueur fixe, comme illustré dans la figure 1-7. Contrairement à d'autres algorithmes cryptographiques, elles ne nécessitent pas de clés. Ces fonctions trouvent des applications dans plusieurs domaines, tels que les signatures numériques, la vérification des messages, le stockage sécurisé des mots de passe et la dérivation de clés, permettant de générer des clés supplémentaires à partir d'une valeur de hachage [Stallings 17]. Pour qu'une fonction de hachage soit considérée comme sécurisée, elle doit répondre à trois critères principaux :

- **Résistance à la préimage** : il doit être difficile de retrouver le message original à partir de sa valeur de hachage.
- **Résistance à la seconde préimage** : il doit être compliqué de trouver un autre message produisant le même hachage.
- **Résistance aux collisions** : il doit être pratiquement impossible de créer deux messages différents ayant le même hachage [Menezes 97].

Ces propriétés sont essentielles pour garantir l'intégrité et la sécurité des données dans divers systèmes cryptographiques. Cependant, certaines fonctions, comme SHA-1, ont montré des faiblesses, notamment en matière de collisions.



Figure 1- 7 : Principe de fonctionnement du hachage

1.4.1 Fonction de hachage MD5

La fonction de hachage MD5 est conçue pour générer une empreinte numérique unique de 128 bits à partir de n'importe quelle donnée d'entrée. Elle était initialement utilisée pour vérifier l'intégrité des fichiers et signer des certificats numériques. Cependant, il présente des failles significatives qui le rendent aujourd'hui obsolète dans la plupart des applications de sécurité. La principale faiblesse de MD5 est sa vulnérabilité aux collisions, c'est-à-dire la possibilité que deux entrées différentes produisent la même empreinte numérique. Cette faille permet à des attaquants de créer des documents ou des fichiers distincts avec une empreinte identique, ce qui compromet gravement l'intégrité des données. En 2004, des chercheurs ont démontré des collisions exploitables, notamment dans la création de certificats numériques malveillants. Depuis, plusieurs attaques ont confirmé que MD5 ne garantit plus la sécurité attendue dans des environnements critiques. Ces failles ont conduit à l'obsolescence de MD5, qui est désormais déconseillé pour les applications cryptographiques modernes [Reddy 21].

1.4.2 Fonction de hachage SHA

La fonction de hachage SHA est une famille de fonctions de hachage cryptographiques qui génèrent une empreinte numérique fixe à partir d'une entrée de longueur variable. L'algorithme prend des blocs de données et les traite en plusieurs étapes de transformation. Il produit ensuite une empreinte numérique, appelée hach, dont la taille dépend de la version de SHA utilisée. Les versions de SHA diffèrent principalement par la taille de l'empreinte produite et leur niveau de sécurité. Par exemple, comme illustré dans la figure 1-8, SHA-1 produit une empreinte de 160 bits. SHA-2, qui comprend des variantes telles que SHA-256 et SHA-512, est plus sécurisé et génère respectivement des empreintes de 256 et 512 bits. Le processus de hachage est fondé sur des opérations logiques, telles que XOR, rotations et additions modulo. Ces transformations garantissent qu'une petite modification de l'entrée produit un hach complètement différent, assurant ainsi la sensibilité du résultat. L'une des forces de SHA réside dans sa capacité à

résister aux attaques par préimage et aux collisions. Cependant, SHA-1 a montré des vulnérabilités dans ces domaines, ce qui a conduit à son remplacement dans les environnements exigeant une sécurité renforcée [Handschuh 11].

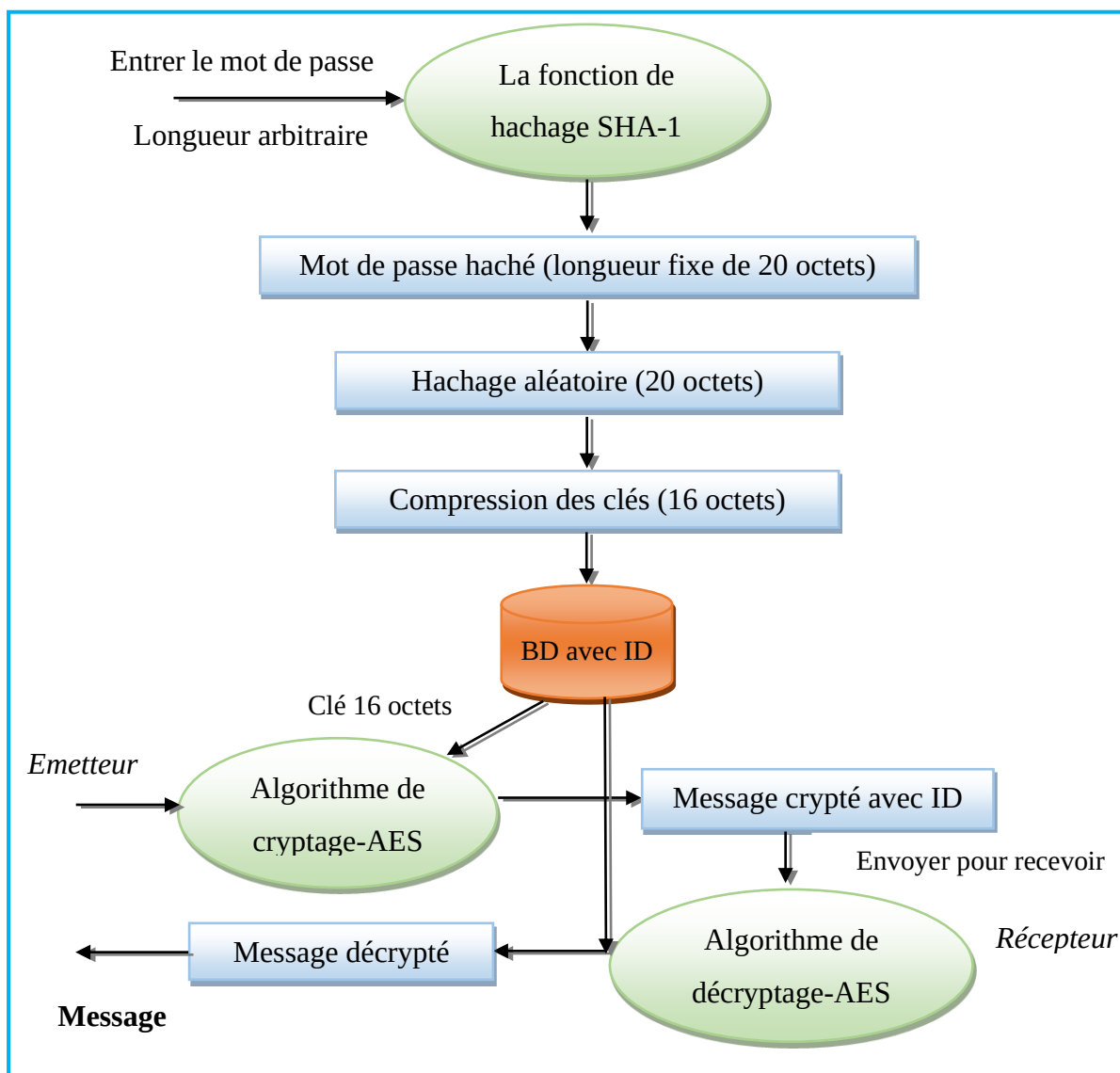


Figure 1- 8 : Diagramme de SHA-1

1.5 Cryptographie quantique

La cryptographie quantique est un domaine de recherche qui exploite les principes de la mécanique quantique pour développer des protocoles de sécurité. Ces protocoles sont fondamentalement différents de ceux utilisés dans la cryptographie classique. Contrairement à la cryptographie classique, qui repose sur des problèmes mathématiques complexes, comme la

factorisation de grands nombres, la cryptographie quantique utilise les propriétés quantiques des particules subatomiques pour garantir la sécurité des informations. [Charles 14].

Parmi les concepts clés de la cryptographie quantique, on trouve :

Le principe d'incertitude : en mécanique quantique, stipule que certaines propriétés des particules, comme la position et la vitesse, ne peuvent être mesurées simultanément avec précision. Ce principe garantit une notion de non-intrusion, car toute tentative de mesure modifie l'état de la particule.

Superposition et intrication : les particules quantiques peuvent exister dans plusieurs états simultanément, un phénomène appelé **superposition**. De plus, elles peuvent être **intriquées**, ce qui signifie que l'état d'une particule est corrélé à celui d'une autre, même si elles sont spatialement séparées [Einstein 35].

Cryptographie quantique à clé publique : les protocoles de cryptographie quantique utilisent les propriétés quantiques pour sécuriser la distribution des clés. Le protocole BBM92, l'un des premiers de ce type, exploite l'intrication quantique pour détecter toute tentative d'interception.

Quantum Key Distribution (QKD) : la QKD est une technique qui permet à deux parties de partager une clé secrète tout en détectant toute tentative d'interception. Des protocoles comme BB84 utilisent les propriétés quantiques pour garantir la confidentialité [Scarani 20].

Cryptographie quantique post-quantique : face aux défis posés par les ordinateurs quantiques, la cryptographie quantique est une solution prometteuse. Alors que des algorithmes classiques comme RSA et ECC risquent d'être vulnérables aux attaques quantiques, les protocoles quantiques pourraient offrir une alternative plus sécurisée.

1.5.1 Communication quantique sécurisée

La communication quantique sécurisée est une approche révolutionnaire de la sécurité des communications, qui utilise les concepts fondamentaux de la mécanique quantique pour garantir un niveau sans précédent de confidentialité et d'intégrité des informations échangées entre des entités distantes. Le QKD (Quantum Key Distribution) est une classe de protocoles qui permet à deux parties communicantes de générer une clé partagée en toute sécurité, tout en détectant activement les tentatives d'interception. Les protocoles de QKD, tels que BBM92 et BB84, reposent sur l'intrication quantique. Ce phénomène impressionnant de la mécanique quantique relie deux particules de sorte que l'état de l'une soit directement corrélé à celui de

l'autre, quel que soit l'écart les séparant. Ainsi, toute tentative d'observation ou d'interception de la clé est forcément intrusive, modifiant irréversiblement l'état quantique des particules et révélant immédiatement une attaque. Contrairement aux méthodes classiques, le QKD introduit un paradigme unique : l'attaque elle-même altère le message transmis, rendant cette approche particulièrement utile dans un contexte où les ordinateurs quantiques progressent rapidement. Bien que le QKD en soit encore à ses débuts, il offre des solutions prometteuses pour les communications dans des secteurs hautement sécurisés tels que les informations gouvernementales ou financières, où les enjeux élevés encouragent parfois des actions malveillantes. La figure 1-9 ci-après illustre la cryptographie symétrique utilisant des dispositifs quantiques.

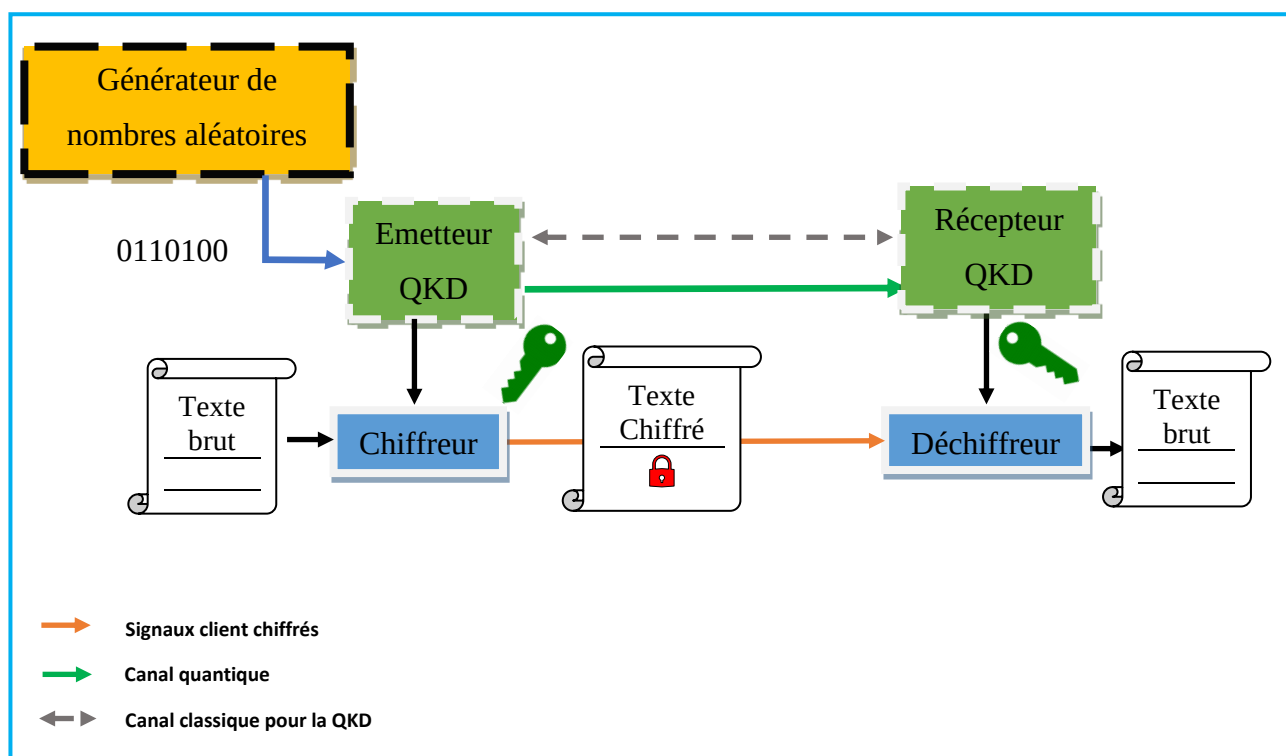


Figure 1- 9 : Cryptographie symétrique avec des dispositifs quantiques

1.5.2 Fonction BB84

Le protocole BB84 a un rôle central dans la cryptographie quantique, notamment pour la distribution de clés quantiques. Il permet à deux entités éloignées d'échanger des clés en exploitant les propriétés des phénomènes quantiques. Les bits quantiques, généralement des photons, sont transmis via un canal quantique. Chaque bit est polarisé selon une base quantique, sélectionnée aléatoirement. Dans BB84, les bases utilisées pour effectuer les mesures sont soit rectilignes, soit diagonales. Le destinataire effectue un choix aléatoire de base pour chaque bit

reçu. Après mesure, l'émetteur et le destinataire révèlent publiquement les bases utilisées. Seuls les bits mesurés avec des bases identiques sont conservés pour former la clé partagée, tandis que les autres bits sont rejetés. Pour assurer la sécurité, le processus inclut une phase de détection d'interception. Les parties révèlent une portion de leurs clés pour vérifier l'absence d'interférence. Si aucune tentative d'interception n'est détectée, elles utilisent le reste de la clé partagée pour communiquer de manière sécurisée. Le protocole BB84 s'appuie sur des propriétés fondamentales de la mécanique quantique, comme l'impossibilité d'observer un système quantique sans le perturber. Il a posé les bases pour de nombreuses autres méthodes de distribution de clés quantiques [Bennett 84]. Le tableau 1-1 illustre la génération des clés BB84.

Tableau 1- 1 : Génération d'une clé publique quantique avec BB84

	0	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1
Base du client	+	X	+	X	+	X	X	+	+	X	+	X	+	X	X	+
Polarisation du client	↑	↗	→	↖	↑	↖	↗	→	↑	↗	→	↖	↑	↖	↗	→
Base du serveur	+	+	X	X	X	+	X	+	+	+	X	X	X	+	X	+
Polarisation du serveur	↑	→	↖	↖	↗	→	↗	→	↑	→	↖	↖	↗	→	↗	→
La clé publique	0			1			1	1	0			1			1	1

1.5.3 Défis et avancées en cryptographie quantique

Les systèmes quantiques font face à plusieurs défis, notamment leur vulnérabilité à des attaques physiques, comme les interférences électromagnétiques ou les techniques avancées d'espionnage quantique. La distance de transmission de l'information quantique est également limitée, car les particules quantiques subissent une décohérence, perdant progressivement leurs propriétés sur de longues distances. Cela réduit l'efficacité des canaux de communication quantique. Un autre défi majeur est l'établissement de connexions multipartites permettant la cryptographie entre plusieurs entités. Cela nécessite des protocoles plus complexes pour sécuriser les clés partagées. Par ailleurs, le coût élevé des technologies quantiques, incluant les sources de photons uniques et les dispositifs de détection et de manipulation, constitue un obstacle important à leur adoption à grande échelle. Malgré ces défis, des avancées significatives ont été réalisées dans le domaine de la cryptographie quantique. Des protocoles

de distribution quantique de clés, comme BB84 et E91⁵, ont démontré leur efficacité pour échanger des clés de manière sécurisée. En parallèle, le développement de réseaux quantiques utilisant des satellites ou des relais permet d'envisager une transmission d'information sur de plus grandes distances. L'émergence de la cryptographie post-quantique vise également à garantir la sécurité des systèmes classiques face à l'arrivée des ordinateurs quantiques, en s'appuyant sur des approches résistantes aux algorithmes de factorisation quantique. Enfin, des progrès notables ont été réalisés dans la stabilisation des qubits et le développement de matériaux avancés, comme les points quantiques et les qubits supraconducteurs, renforçant ainsi les bases des systèmes de cryptographie quantique.

1.6 Protocoles de communications

Les protocoles de communication assurent des échanges d'informations sécurisés et fiables entre deux entités distantes. Ils définissent les règles permettant d'établir une connexion, de chiffrer les données, de vérifier l'intégrité des informations transmises et d'authentifier les participants à la communication. Ces protocoles utilisent des mécanismes cryptographiques avancés pour protéger la confidentialité des données échangées. Ils s'appuient sur des méthodes d'authentification robustes et des algorithmes de chiffrement, garantissant que seules les parties autorisées peuvent accéder aux informations échangées. Dans les environnements réseau, ils sécurisent les canaux de communication en prévenant les interceptions malveillantes et les attaques de tiers. Ces systèmes prennent également en compte des aspects tels que la protection contre les attaques de type MiTM et l'intégrité des données échangées. Cela garantit que chaque message envoyé est reçu sans altération par le destinataire prévu.

1.6.1 Protocole TLS

Le TLS a été conçu pour fonctionner avec TCP et fournir un service sécurisé et fiable de bout en bout. Il ne s'agit pas d'un seul protocole, mais plutôt de deux couches de protocoles, comme illustré dans la figure 1-10. Le protocole d'enregistrement a un rôle clé en fournissant des services de sécurité de base, comme l'authentification, la confidentialité et l'intégrité des données, à divers protocoles des couches supérieures. Par exemple, le protocole HTTP, utilisé pour l'interaction client/serveur Web, peut exploiter TLS pour sécuriser les transferts de

⁵ E91 utilise l'intrication quantique pour partager une clé sécurisée entre deux parties. Toute tentative d'interception perturbe les corrélations, ce qui permet de détecter une attaque.

données. Cela nécessite l'ajout de couches cryptographiques supplémentaires entre le protocole d'enregistrement et les protocoles clients/serveurs, renforçant ainsi la sécurité des communications.

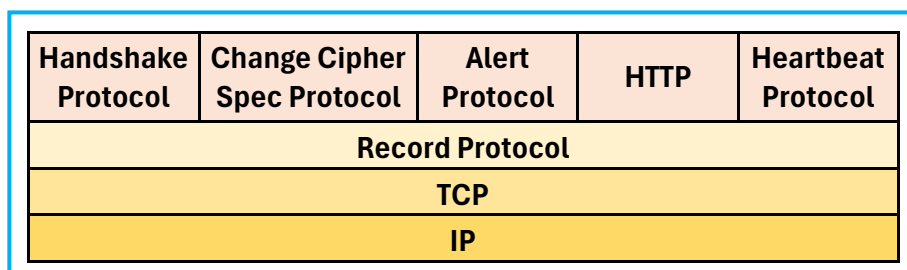


Figure 1- 10 : La pile de protocoles SSL/TLS

TLS garantit une sécurité robuste en combinant l'authentification par certificats numériques, le chiffrement des données avec des algorithmes comme AES, et des mécanismes de vérification d'intégrité pour protéger les transmissions contre les interceptions ou les altérations. AES, qui utilise des longueurs de clé de 128, 192 et 256 bits, est particulièrement performant dans TLS. Il assure un chiffrement rapide et sécurisé, grâce à sa conception optimisée pour les environnements matériels et logiciels [Schneier 07]. L'intégration d'AES dans TLS protège efficacement contre les attaques par force brute grâce à la grande taille de ses clés. De plus, AES offre de bonnes performances en termes de vitesse de chiffrement et de déchiffrement, tout en étant résistant aux attaques linéaires et différentielles, renforçant ainsi la robustesse des connexions sécurisées [Daemen 01]. Malgré ces nombreux avantages, certaines vulnérabilités peuvent affecter la sécurité globale de TLS lorsqu'il est implémenté avec AES. La sécurité de TLS repose fortement sur la gestion des certificats et des clés de chiffrement. Une gestion inadéquate, comme l'utilisation de certificats expirés ou mal configurés, peut compromettre le protocole. De plus, bien que TLS avec AES soit très résistant aux attaques cryptographiques classiques, il reste vulnérable aux attaques par canal auxiliaire, comme l'analyse temporelle ou l'analyse de puissance. Dans ces scénarios, un attaquant exploite les fuites d'informations provenant de l'implémentation physique de l'algorithme pour compromettre la sécurité.

1.6.2 Protocole SSH

SSH est un protocole de communication cryptographique conçu pour établir une connexion sécurisée sur un réseau non sécurisé, tel qu'internet. Il permet notamment d'ouvrir des sessions à distance, d'exécuter des commandes sur des serveurs et de transférer des fichiers en toute sécurité. SSH repose sur des mécanismes de chiffrement symétrique et asymétrique pour

garantir la confidentialité, l'intégrité et l'authentification des communications entre un client et un serveur. L'une des principales forces de SSH réside dans son utilisation de l'authentification forte via des clés publiques et privées. Lors de l'établissement de la connexion, le serveur présente sa clé publique au client, qui l'utilise pour créer une session chiffrée. Une fois la connexion établie, les données sont échangées via un canal sécurisé, protégé par un chiffrement symétrique tel qu'AES, garantissant un haut niveau de confidentialité des informations transmises [Ylonen 06]. SSH est également flexible, permettant d'encapsuler d'autres protocoles au sein de sa connexion sécurisée, offrant ainsi une méthode de protection pour des communications basées sur des protocoles non sécurisés. De plus, il peut être configuré pour authentifier les utilisateurs à l'aide de clés publiques, de mots de passe ou de mécanismes avancés tels que Kerberos⁶. Cependant, SSH présente certaines limites. Une mauvaise gestion des clés privées peut compromettre la sécurité, tout comme l'utilisation de mots de passe faibles, qui exposent le système à des attaques par force brute. Des attaques par canal auxiliaire, telles que des analyses temporelles, où un attaquant mesure les temps de réponse pour déduire des informations sensibles, peuvent également représenter un risque si l'implémentation n'est pas suffisamment sécurisée [Song 01]. Enfin, l'arrivée des ordinateurs quantiques suscite des inquiétudes, car ils pourraient casser les schémas de chiffrement utilisés dans SSH, notamment pour l'échange initial de clés [Mosca 18].

1.7 Conclusion

La cryptographie moderne, bien qu'essentielle pour protéger les communications numériques, présente plusieurs vulnérabilités significatives. Les attaques MITM, qui exploitent des vulnérabilités dans l'échange de clés et la gestion des certificats, compromettent fréquemment la sécurité de protocoles tels que TLS et SSH. De plus, les attaques par canal auxiliaire permettent d'extraire des informations sensibles en analysant des fuites indirectes, telles que la consommation d'énergie ou les temps d'exécution. Ces méthodes ne nécessitent pas de casser les algorithmes de chiffrement comme AES. Par ailleurs, certaines versions plus anciennes de protocoles, comme SSL 3.0 et TLS 1.0, restent vulnérables à des attaques ciblées en raison de faiblesses structurelles. Outre les vulnérabilités des protocoles, les fonctions de hachage ont un rôle central dans les mécanismes d'authentification, mais présentent également des failles. Des algorithmes tels que MD5 et SHA-1, utilisés pour l'authentification et la gestion des mots de

⁶ Kerberos est un protocole d'authentification réseau qui utilise des tickets pour permettre à des nœuds de prouver leur identité de manière sécurisée. Il repose sur un tiers de confiance (Key Distribution Center) pour gérer les échanges entre les utilisateurs et les services.

passé, sont aujourd'hui vulnérables aux collisions. Ces failles permettent à un attaquant de générer deux ensembles de données différents ayant la même empreinte, compromettant ainsi l'intégrité et l'authenticité des informations. L'adoption d'algorithmes de hachage plus robustes, comme SHA-256 ou SHA-3, est désormais indispensable. Ces failles soulignent l'urgence de renforcer les mécanismes cryptographiques pour garantir la sécurité des systèmes numériques dans un environnement de plus en plus menacé.

Chapitre 2 : Outils mathématiques pour la cryptographie

2.1 Introduction

Les fondements théoriques de la cryptographie moderne reposent sur des outils mathématiques bien établis, qui garantissent la sécurité et la robustesse des protocoles de communication. Les algorithmes de cryptographie, utilisés dans des protocoles tels que TLS ou SSH, s'appuient sur des concepts fondamentaux issus de domaines comme la théorie des groupes, l'algèbre linéaire et la théorie des nombres, ainsi que sur des fonctions de hachage complexes. Dans ce contexte, l'objectif de ce chapitre est de présenter ces différentes branches mathématiques en relation directe avec les mécanismes cryptographiques, en mettant l'accent sur leur rôle dans la sécurisation des données.

2.2 Théorie des groupes et algèbre abstraite

La théorie des groupes et l'algèbre abstraite sont des outils mathématiques fondamentaux en cryptographie, particulièrement dans les systèmes à clé publique. Les groupes permettent de structurer les opérations algébriques de manière à garantir des propriétés essentielles pour la sécurité, telles que la difficulté à inverser certaines opérations, comme l'exponentiation modulaire ou la multiplication modulaire. Les principaux concepts de cette théorie sont illustrés par des formules mathématiques et des références adaptées.

2.2.1 Groupes cycliques

Un groupe G est un ensemble muni d'une opération binaire « $\cdot$ » qui satisfait les propriétés de fermeture, d'associativité et d'existence d'un élément neutre, et d'existence d'un inverse. Un groupe est dit cyclique si tous ses éléments peuvent être générés par la répétition d'un seul élément, appelé générateur.

Formellement, un groupe G est cyclique s'il existe un élément $g \in G$ tel que :

$$G = \{ g^k \mid k \in \mathbb{Z} \} \quad (2.1)$$

Cela signifie que chaque élément de G peut être écrit sous la forme g^k , où k est un entier. Le groupe G est alors noté $\langle g \rangle$, où g est un générateur du groupe.

En cryptographie, les groupes cycliques sont souvent utilisés dans des systèmes tels que Diffie-Hellman et RSA. Par exemple, dans le protocole Diffie-Hellman, les opérations sont effectuées dans un groupe cyclique modulo un nombre premier p , ce qui assure que les calculs restent complexes à inverser sans informations supplémentaires [Boneh 20].

2.2.2 Exponentiation modulaire

L'exponentiation modulaire est une opération courante en cryptographie qui consiste à calculer $a^b \bmod n$, où a est la base, b est l'exposant, et n est le modulo. Cette opération est définie par la formule suivante :

$$a^b \bmod n = r \quad (2.2)$$

où r est le reste de la division de a^b par n .

Cette opération est facile à calculer, mais inversement, trouver b à partir de $a^b \bmod n$ (problème du logarithme discret) est très difficile, ce qui constitue la base de la sécurité de nombreux systèmes cryptographiques, comme Diffie-Hellman et RSA. Dans RSA, la clé publique est constituée d'un couple (e, n) , où e est l'exposant de chiffrement, et la clé privée est constituée de l'exposant d , calculé de manière à satisfaire :

$$e \cdot d \equiv 1 \pmod{\varphi(n)} \quad (2.3)$$

où $\varphi(n)$ est la fonction d'Euler de n .

2.2.3 Multiplication modulaire

La multiplication modulaire est une opération mathématique essentielle dans de nombreux algorithmes cryptographiques, tels que RSA, Diffie-Hellman, et même dans certains aspects des systèmes de chiffrement symétrique comme AES. Elle permet de restreindre le résultat d'une multiplication à un intervalle donné en utilisant un opérateur modulo. Formellement, pour deux entiers a et b , et un entier n appelé modulo, la multiplication modulaire se définit par :

$$a \cdot b \bmod n \quad (2.4)$$

Où $a \cdot b$ est d'abord calculé, puis réduit en prenant le reste de la division par n . Cette opération a un rôle central dans la cryptographie, car elle permet de maintenir les résultats des calculs dans un espace de taille contrôlée (de 0 à $n-1$) tout en rendant les calculs difficiles à inverser sans certaines informations spécifiques, telles que la clé privée dans RSA. Dans l'algorithme RSA, la multiplication modulaire est utilisée à la fois lors du chiffrement et du déchiffrement. Le chiffrement se fait en calculant :

$$C = M^e \bmod n \quad (2.5)$$

Où M est le message, e est l'exposant de la clé publique, et n est le produit de deux grands nombres premiers. Le déchiffrement s'effectue en calculant :

$$M = C^d \bmod n \quad (2.6)$$

Où d est l'exposant de la clé privée. Le grand avantage de la multiplication modulaire dans ce contexte est qu'elle assure la sécurité de l'algorithme, car la factorisation de n en ses deux facteurs premiers est considérée comme un problème mathématique difficile à résoudre, même avec les technologies actuelles [RSA 78]. De plus, dans le cadre du protocole Diffie-Hellman, la multiplication modulaire est essentielle pour le calcul des clés partagées entre deux parties. Chaque partie génère une clé privée, puis utilise la multiplication modulaire pour calculer une valeur publique qui est échangée avec l'autre partie. Les clés sont ensuite calculées à partir des valeurs publiques échangées et de la clé privée de chaque partie, toujours en utilisant la multiplication modulaire, assurant ainsi que la clé partagée est difficile à déterminer pour un attaquant sans accès aux clés privées [Diffie 76]. L'utilisation de cette opération dans la cryptographie permet de sécuriser efficacement les échanges d'informations en limitant la possibilité d'inversion de ces calculs complexes, rendant ainsi les attaques par force brute ou par analyse mathématique beaucoup plus difficiles.

2.2.4 Sous-groupes et générateurs

En cryptographie, les concepts de sous-groupes et de générateurs sont essentiels pour définir et structurer des opérations sécurisées sur des ensembles mathématiques. Un groupe G est un ensemble muni d'une opération binaire « $\cdot$ », satisfaisant les propriétés de fermeture, d'associativité, l'existence d'un élément neutre et l'existence d'inverses. Un générateur g est

un élément de G tel que chaque élément de G peut être écrit comme une puissance de g , comme décrit dans la formule (2.1).

Les sous-groupes sont des sous-ensembles de G qui eux-mêmes forment un groupe avec l'opération du groupe initial. Si $H \subseteq G$, alors H est un sous-groupe si pour tous $a, b \in H$, $a \cdot b^{-1} \in H$, où b^{-1} est l'inverse de b dans G .

Ces concepts sont cruciaux dans les systèmes cryptographiques avancés, notamment pour structurer les transformations mathématiques comme celles utilisées dans les méthodes de hachage ou de chiffrement symétrique et post-quantique. Les générateurs permettent de définir des transformations qui garantissent une sécurité élevée tout en limitant les opérations à un espace mathématiquement sécurisé et difficile à inverser sans connaissance préalable [Bennett 84, Rivest 78].

2.3 Algèbre linéaire et transformations

L'algèbre linéaire est un outil mathématique fondamental en cryptographie moderne. Les concepts liés aux espaces vectoriels, aux matrices de transformation, aux matrices orthogonales et aux transformations linéaires sont essentiels pour développer et comprendre les algorithmes cryptographiques, notamment dans les systèmes symétriques et les méthodes post-quantiques.

2.3.1 Espace vectoriel

Un espace vectoriel est une structure mathématique composée de vecteurs qui peuvent être additionnés entre eux et multipliés par des scalaires, tout en respectant certaines propriétés (associativité, commutativité, etc.). Formulé de manière plus formelle, un espace vectoriel V sur un corps K est défini par un ensemble V et deux opérations :

L'addition vectorielle : $+: V \times V \rightarrow V$

La multiplication scalaire : $\cdot: K \times V \rightarrow V$

Ces opérations respectent des axiomes tels que l'associativité et la distributivité. Par exemple, pour des vecteurs $v_1, v_2 \in V$ et des scalaires $a, b \in K$, l'addition vectorielle et la multiplication scalaire se définissent comme :

$$av_1 + bv_2 \in V \quad (2.7)$$

En cryptographie, les espaces vectoriels sont utilisés pour modéliser des structures de données complexes et pour définir des états cryptographiques. Par exemple, dans les systèmes de

cryptographie quantique ou dans des algorithmes de transformation, les états des systèmes peuvent être représentés comme des vecteurs dans un espace vectoriel [Gallier 11].

2.3.2 Matrices de transformation

Les matrices de transformation sont des outils mathématiques qui permettent de décrire comment les vecteurs dans un espace vectoriel sont modifiés. Dans la cryptographie, ces matrices sont importantes pour les transformations linéaires, comme dans le chiffrement symétrique ou les méthodes de hachage. Une transformation linéaire $T : V \rightarrow W$ entre deux espaces vectoriels V et W peut être représentée par une matrice A , telle que pour tout vecteur $x \in V$, on a :

$$T(x) = Ax \quad (2.8)$$

Dans ce contexte, les matrices de transformation peuvent être utilisées pour appliquer des rotations, des permutations ou des substitutions sur des données chiffrées. Par exemple, dans certains systèmes de chiffrement, les matrices de transformation sont utilisées pour effectuer des permutations d'octets ou des rotations de bits, assurant ainsi que le texte chiffré soit sécurisé. Cette technique est largement appliquée dans les algorithmes de chiffrement tels que l'AES [Daemen 01], où chaque étape du chiffrement repose sur une série de transformations linéaires.

2.3.3 Matrice orthogonale

Une matrice orthogonale est une matrice carrée A dont les colonnes et les lignes forment des vecteurs orthogonaux, c'est-à-dire que leur produit scalaire est nul et leur norme est égale à 1. Autrement dit, une matrice A est orthogonale si :

$$A^T A = A A^T = I \quad (2.9)$$

Où A^T est la transposée de A et I est la matrice d'identité. En cryptographie, les matrices orthogonales sont utilisées dans des transformations complexes, telles que les rotations et les réflexions d'espaces de données. Leur utilisation garantit que les transformations sont réversibles, ce qui est essentiel pour assurer la confidentialité des données sans perte d'information [Menezes 97].

Les matrices orthogonales ont également des propriétés de conservation des distances et des angles entre les vecteurs, ce qui en fait un outil robuste pour les systèmes cryptographiques qui

nécessitent des opérations de rotation sur des vecteurs, comme dans certaines méthodes de hachage ou de cryptographie quantique [Bennett 84].

2.3.4 Transformation linéaire

Une transformation linéaire est une fonction T entre deux espaces vectoriels qui préserve l'addition vectorielle et la multiplication scalaire. Cela signifie que pour tous les vecteurs $u, v \in V$ et tout scalaire $c \in K$, on a :

$$T(u + v) = T(u) + T(v) \quad (2.10)$$

$$T(cv) = cT(v) \quad (2.11)$$

Les transformations linéaires sont omniprésentes en cryptographie, notamment dans des algorithmes comme AES, où chaque tour de chiffrement implique une série de transformations linéaires appliquées aux blocs de données [Daemen 01]. Dans les méthodes post-quantiques et certaines techniques de chiffrement symétrique, les transformations linéaires permettent d'assurer la diffusion et la confusion, deux principes fondamentaux pour rendre le texte chiffré indéchiffrable sans la clé adéquate. Les transformations linéaires peuvent être représentées par des matrices, ce qui permet de les intégrer efficacement dans des algorithmes de chiffrement, où elles sont appliquées pour modifier les données de manière répétitive et non triviale. Dans le contexte de la sécurité des systèmes modernes, l'utilisation de transformations linéaires est essentielle pour garantir la robustesse face aux attaques cryptanalytiques [Menezes 97].

2.4 Théorie des nombres et logarithmes discrets

La théorie des nombres et les logarithmes discrets occupent une place centrale dans la cryptographie moderne. La théorie des nombres, qui explore les propriétés des entiers, fournit des bases solides pour le développement d'algorithmes sécurisés. Les logarithmes discrets posent un défi mathématique majeur, à partir d'un nombre et de sa puissance modulaire, il est extrêmement complexe de retrouver l'exposant utilisé. Cette complexité est au cœur de nombreux protocoles cryptographiques, tels que Diffie-Hellman et certains systèmes basés sur les courbes elliptiques, assurant ainsi la robustesse des mécanismes de sécurité.

2.4.1 Nombres premiers

En cryptographie, la théorie des nombres et le logarithme discret sont au cœur de nombreux algorithmes. Le logarithme discret consiste à résoudre l'équation exponentielle modulaire de la forme :

$$g^x \equiv h \text{ mod } p \quad (2.12)$$

Où g est une base, p un nombre premier, h un résultat connu et x l'exposant inconnu que l'on cherche à déterminer. Ce problème est réputé difficile à résoudre dans les grands corps finis, c'est-à-dire lorsque p est grand. La difficulté repose sur l'absence de méthode efficace pour calculer x à partir de g , h et p , en particulier lorsque les valeurs sont suffisamment grandes pour résister aux attaques de type force brute. Le logarithme discret est utilisé dans des algorithmes comme le protocole de Diffie-Hellman, où la sécurité repose sur la difficulté de ce problème. Par exemple, dans Diffie-Hellman, deux parties partagent publiquement une base g et un grand nombre premier p . Chaque partie calcule ensuite une valeur intermédiaire en élevant g à une puissance secrète, et ces valeurs sont échangées. Le calcul de la clé partagée finale repose sur l'impossibilité pratique de retrouver l'exposant secret à partir de la valeur échangée, autrement dit, sur la difficulté de résoudre le logarithme discret. Ainsi, les logarithmes discrets contribuent directement à la robustesse des systèmes cryptographiques basés sur l'arithmétique modulaire, offrant une résistance significative aux tentatives de décryptage non autorisé dans des contextes où les nombres premiers et les exponentiations occupent un rôle central [Shor 99].

2.4.2 Logarithme discret

Considérons un groupe cyclique de premier ordre, où g est un générateur et où l'équation $h = g^x \text{ mod } p$ est définie, avec p étant un grand nombre premier. Le logarithme discret de h en base g correspond au nombre x , c'est-à-dire que x satisfait l'équation exponentielle modulaire mentionnée. Bien que le calcul direct de h à partir de g et x soit trivial, l'inversion du processus retrouvé x à partir de g et h s'avère extrêmement complexe, particulièrement lorsque p est un grand nombre premier. Ce problème est considéré comme difficile dans la théorie des nombres et sa complexité garantit la sécurité des algorithmes qui en dépendent.

Le logarithme discret est exploité dans des systèmes cryptographiques classiques tels que Diffie-Hellman pour l'échange sécurisé de clés. Cependant, dans les systèmes post-quantiques, où la menace des ordinateurs quantiques devient préoccupante, il est important de développer

des algorithmes résistants à ces nouveaux types d'attaques. En effet, les ordinateurs quantiques, avec des algorithmes comme celui de Shor, peuvent résoudre le logarithme discret de manière efficace, rendant ainsi vulnérables les systèmes cryptographiques actuels basés sur ce problème. Par conséquent, dans le contexte des systèmes post-quantiques, de nouvelles approches, telles que la cryptographie à base de réseaux ou les protocoles de distribution quantique de clés, sont développées pour remplacer les systèmes traditionnels et offrir une sécurité renforcée face à la menace des capacités de calcul avancées. Ces solutions visent à préserver la robustesse des systèmes cryptographiques en utilisant des structures mathématiques qui restent résistantes aux techniques de calcul quantique [Bennett 84, Shor 97].

2.4.3 Problème de factorisation

Il s'agit de décomposer un entier n en un produit de facteurs premiers, c'est-à-dire trouver $p_1, p_2, \dots, p_k$ tels que $n = p_1 \times p_2 \times \dots \times p_k$. Si cette tâche est relativement simple pour des entiers de petite taille, elle devient de plus en plus complexe lorsque n est le produit de grands nombres premiers. Cette difficulté de factorisation est exploitée pour sécuriser les clés cryptographiques dans des systèmes tels que RSA, où la sécurité repose sur l'impossibilité pratique de factoriser de très grands nombres. Dans le contexte de la cryptographie post-quantique, la résistance à la factorisation est importante, car elle assure que les clés cryptographiques ne peuvent être reconstruites à partir de l'information publique. Par exemple, dans les systèmes de distribution de clés, la factorisation difficile garantit que même si un attaquant connaît n , il lui est pratiquement impossible de déduire les facteurs premiers qui composent n , et donc de compromettre la sécurité du système. Cependant, l'algorithme de Shor, un algorithme quantique, remet en question cette sécurité en offrant une méthode exponentiellement plus rapide pour résoudre le problème de la factorisation que les algorithmes classiques. Cela a conduit à une réorientation des recherches vers des algorithmes cryptographiques résistants aux attaques quantiques, tels que les schémas basés sur les réseaux ou les codes correcteurs d'erreurs, qui offrent une sécurité même face aux capacités accrues des ordinateurs quantiques.

2.4.4 Problème de modulaire

Le calcul modulaire est un outil clé dans les algorithmes cryptographiques modernes, utilisé pour effectuer des opérations au sein d'un espace fini. En pratique, cela signifie que les résultats de calculs sont ramenés à des valeurs comprises entre 0 et $n-1$, où n est un entier appelé le modulo. Cette méthode permet de limiter la taille des valeurs manipulées, rendant les opérations plus sûres et prévenant des débordements potentiels. Dans les systèmes cryptographiques post-quantiques, le calcul modulaire est particulièrement utile pour effectuer des opérations sur de grands nombres tout en garantissant que les résultats restent dans un espace mathématique restreint et défini. Cela assure que les opérations ne révèlent pas d'informations sensibles qui pourraient être exploitées par des attaquants, même en présence de capacités de calcul avancées, comme celles offertes par les ordinateurs quantiques. De plus, le calcul modulaire est utilisé pour sécuriser des opérations comme la génération de clés et les échanges de données, garantissant que seules les parties possédant les informations nécessaires peuvent décrypter les messages. Grâce à sa structure, le calcul modulaire présente un rôle important en limitant l'accès aux informations critiques, rendant ainsi les attaques, telles que celles visant à deviner la clé, extrêmement difficiles. Cette approche est centrale dans de nombreux systèmes cryptographiques modernes, garantissant une sécurité renforcée contre les menaces actuelles et futures, comme l'ont montré des études telles que [Daemen 01].

2.5 Fonction de hachage et complexité

La complexité des fonctions de hachage est un facteur déterminant dans la sécurité cryptographique, où chaque opération doit non seulement garantir l'intégrité des données, mais aussi offrir une résistance plus forte face aux menaces évolutives. Dans les environnements où la rapidité de traitement et la robustesse face aux attaques sont primordiales, la gestion efficace des ressources de calcul devient cruciale. L'analyse de la résistance aux collisions et aux préimages permet de mesurer l'efficacité des algorithmes comme SHA-256 et SHA-3, tout en explorant des solutions plus récentes telles qu'argon2, qui repoussent les limites de la performance tout en préservant une sécurité contre les attaques de nouvelle génération.

2.5.1 Propriétés des fonctions de hachage

Les fonctions de hachage cryptographiques doivent satisfaire certaines propriétés pour garantir leur efficacité et leur sécurité. Parmi les propriétés les plus importantes, on trouve la résistance

à la préimage, la résistance à la seconde préimage et la résistance aux collisions. La résistance à la préimage signifie qu'il doit être difficile, à partir d'un haché donné h , de retrouver l'entrée x telle que $h = H(x)$ où H est la fonction de hachage. La résistance à la seconde préimage garantit qu'il est difficile, pour une entrée donnée x_1 de trouver une autre entrée x_2 telle que $H(x_1) = H(x_2)$. Enfin, la résistance aux collisions implique qu'il est difficile en termes de calcul de trouver deux entrées distinctes x_1 et x_2 telles que $H(x_1) = H(x_2)$ [Preneel 96].

Ces propriétés sont essentielles pour des algorithmes comme SHA-256, largement utilisé dans les protocoles TLS pour la sécurisation des communications. En outre, la résistance aux collisions est fondamentale dans les signatures numériques, où la sécurité de la vérification dépend de l'impossibilité de trouver deux messages différents produisant la même empreinte [Goldreich 01].

2.5.2 Résistance aux collisions

La résistance aux collisions est l'une des propriétés les plus critiques d'une fonction de hachage cryptographique. Une collision se produit lorsque deux entrées différentes produisent la même empreinte. Si une telle collision est facilement trouvable, cela compromet la sécurité des systèmes qui dépendent de cette fonction de hachage. En particulier, dans les signatures numériques, une collision pourrait permettre à un attaquant de falsifier une signature en trouvant un autre document ayant la même empreinte que le document original signé [Bellare 93].

La sécurité des fonctions de hachage modernes repose sur la difficulté de générer ces collisions. Par exemple, les attaques de type paradoxe des anniversaires⁷ exploitent la probabilité plus forte de collisions lorsque le nombre d'essais augmente, même si le nombre total de possibilités est grand. Cependant, des fonctions de hachage comme SHA-3 sont conçues pour résister à ces attaques en augmentant la complexité de calcul et en rendant la recherche de collisions impossible à faire côté calcul [Dang 15]. L'introduction de SHA-3, après les vulnérabilités découvertes dans les fonctions de hachage plus anciennes comme MD5 et SHA-1, marque un tournant dans l'amélioration de la sécurité des systèmes cryptographiques [Kelsey 98].

⁷ Paradoxe des anniversaires illustre qu'il suffit de 23 personnes pour avoir une probabilité de 50 % que deux d'entre elles partagent le même anniversaire. Ce phénomène est dû à la forte croissance des combinaisons possibles à mesure que le nombre de personnes augmente

2.5.3 Complexité algorithmique

La complexité algorithmique d'une fonction de hachage est un aspect clé qui mesure son efficacité à générer des empreintes et sa résistance aux attaques. La complexité est généralement évaluée en termes de complexité temporelle (nombre d'opérations nécessaires pour accomplir une tâche) et de complexité spatiale (quantité de mémoire nécessaire).

Complexité temporelle

Soit $H(x)$ une fonction de hachage où x est l'entrée de longueur n . La complexité temporelle pour calculer $H(x)$ est souvent exprimée en $O(f(n))$ où $f(n)$ représente le nombre d'opérations nécessaires pour calculer l'empreinte de longueur fixe. Pour des fonctions de hachage comme SHA-256, la complexité est linéaire par rapport à la taille de l'entrée, c'est-à-dire :

$$T(H(x)) = O(n) \quad (2.13)$$

Où n est la taille du message en bits. Cependant, pour certaines attaques comme la collision par recherche exhaustive, la complexité dépend de la longueur de l'empreinte. Pour une fonction de hachage produisant une empreinte de m bits, une attaque de collision par force brute a une complexité de $O\left(2^{\frac{m}{2}}\right)$ en se basant sur le paradoxe des anniversaires :

$$T(\text{Collision}) = O\left(2^{\frac{m}{2}}\right) \quad (2.14)$$

Par exemple, pour une empreinte de 256 bits (comme dans SHA-256), la complexité pour trouver une collision est de l'ordre de 2^{128} , ce qui est actuellement impossible à atteindre avec les technologies actuelles [Preneel 96].

Complexité spatiale

La complexité spatiale $S(H(x))$ d'une fonction de hachage se réfère à la quantité de mémoire nécessaire pour effectuer le calcul. Pour les fonctions de hachage standards comme SHA-256, la complexité spatiale est généralement $O(1)$, car la taille de l'empreinte est fixe (256 bits dans le cas de SHA-256), quelle que soit la taille de l'entrée. En termes de mémoire, cela signifie qu'il ne faut qu'une quantité constante de mémoire pour stocker le hach, indépendamment de la taille du message d'origine.

Complexité des attaques

En plus de la complexité de calcul des fonctions de hachage elles-mêmes, il est important de considérer la complexité des attaques contre ces fonctions. Pour une fonction de hachage H de sortie m bits :

- La complexité pour trouver une préimage (donnée x telle que $H(x) = y$) est de $O(2^m)$.
- La complexité pour trouver une seconde préimage (donnée x_1 , trouver un autre x_2 tel que :

$$H(x_1) = H(x_2)) \text{ est également } O(2^m) \quad (2.15)$$

- La complexité pour trouver une collision est de $O\left(2^{\frac{m}{2}}\right)$, comme mentionné précédemment.

Ces complexités montrent que les fonctions de hachage modernes doivent avoir des empreintes suffisamment longues (par exemple, 256 bits pour SHA-256) pour que les attaques par force brute soient impossibles à faire avec les moyens actuels [Bellare 93, Black 05].

Complexité pour SHA-3 (Keccak⁸)

Pour l'algorithme SHA-3, basé sur la fonction Keccak, la complexité temporelle est également linéaire par rapport à la taille de l'entrée. Si r est le taux d'absorption et c la capacité de la fonction Keccak, la complexité temporelle pour traiter un message de n bits est de :

$$T_{SHA-3} = O(n/r) \quad (2.16)$$

Le paramètre r est typiquement choisi de manière à maximiser l'efficacité du traitement tout en maintenant une sécurité robuste contre les attaques cryptographiques [Dang 15, Goldreich 01].

2.6 Théorie de l'information et entropie

L'entropie est une mesure clé de l'incertitude ou de la désorganisation dans un système. En cryptographie, l'entropie est souvent utilisée pour quantifier le degré de sécurité offert par une clé ou un schéma de chiffrement. Une plus grande entropie implique une meilleure imprévisibilité, ce qui complique les tentatives d'attaque.

⁸ Keccak est une fonction de hachage cryptographique conçue pour offrir une sécurité élevée et une grande efficacité. Il a été sélectionné comme algorithme de hachage standard SHA-3 par le NIST.

2.6.1 Entropie des clés

L'entropie d'une clé dans un système cryptographique mesure la quantité d'information nécessaire pour décrire cette clé. En d'autres termes, elle reflète la difficulté pour un attaquant de deviner ou de découvrir la clé. L'entropie $H(X)$ d'une variable aléatoire X représentant les clés possibles est définie par la formule de Shannon :

$$H_{(x)} = - \sum_{i=1}^n P(x_i) \log_2 P(x_i) \quad (2.17)$$

Où $P(x_i)$ est la probabilité d'occurrence de la clé x_i . Si toutes les clés sont équiprobables, l'entropie maximale est atteinte et elle vaut $\log_2(n)$, où n est le nombre total de clés possibles. Par exemple, pour une clé de 256 bits générée de manière aléatoire, l'entropie maximale est de :

$$H(X) = \log_2(2^{256}) = 256 \text{ bits} \quad (2.18)$$

Cela signifie qu'il existe 2^{256} clés possibles, rendant toute attaque par recherche exhaustive pratiquement impossible avec les moyens actuels. L'entropie est un indicateur direct de la force d'une clé : plus l'entropie est élevée, plus la clé est difficile à deviner [Shannon 48].

2.6.2 Redondance dans les systèmes cryptographiques

La redondance dans un système cryptographique fait référence aux informations inutiles qui peuvent être exploitées par un attaquant pour réduire l'incertitude sur la clé ou le message chiffré. En théorie de l'information, la redondance R d'un message est définie comme la différence entre la longueur maximale de l'information possible et l'entropie réelle de l'information. Formellement :

$$R = \log_2(M) - H(X) \quad (2.19)$$

Où M est le nombre total de symboles dans le message et $H(X)$ est l'entropie du message. Plus un système est redondant, plus il est vulnérable à des attaques, car la redondance peut révéler des motifs ou des informations structurelles dans le texte chiffré.

Par exemple, dans les schémas de chiffrement mal conçus, une trop grande redondance dans les données permet de deviner certaines parties du message ou de réduire l'espace de recherche des clés. La réduction de la redondance dans les systèmes cryptographiques modernes permet de maximiser l'entropie et de garantir que chaque bit de donnée est imprévisible [Shannon 48].

2.6.3 Capacité de stockage sécurisé

La capacité de stockage sécurisé est un concept important dans les systèmes cryptographiques, notamment pour évaluer la quantité d'information qui peut être stockée ou transmise en toute sécurité. La capacité maximale d'un canal de communication ou d'un système de stockage sécurisé dépend de l'entropie des données transmises ou stockées. Si un canal de communication a une capacité C (en bits par seconde, par exemple), la capacité de stockage sécurisé est limitée par l'entropie de la source d'information $H(X)$:

$$C_s = H(X) \cdot C \quad (2.20)$$

Ainsi, plus l'entropie de la clé ou des données est élevée, plus la quantité d'information sécurisée qui peut être stockée ou transmise est importante. Dans un environnement cryptographique, cela implique que les messages chiffrés avec des clés ayant une entropie élevée nécessitent des capacités de stockage et de transmission robustes pour garantir la sécurité et la confidentialité des informations [Cover 06]. Un autre aspect important de la capacité de stockage sécurisé est la gestion des clés secrètes dans des systèmes distribués ou en cloud. Le stockage sécurisé des clés nécessite une combinaison d'entropie élevée et de systèmes de gestion de clés robustes, garantissant que les clés ne sont accessibles qu'aux parties autorisées. Les protocoles de gestion des clés doivent minimiser les risques d'exposition tout en optimisant l'utilisation de la capacité de stockage.

2.7 Conclusion

La complexité mathématique exposée dans ce chapitre montre l'importance des outils tels que la théorie des groupes, l'algèbre linéaire et la théorie des nombres dans la sécurisation des systèmes cryptographiques. En explorant les propriétés des fonctions de hachage, des logarithmes discrets et des opérations modulaires, il est clair que ces concepts sont fondamentaux pour garantir la robustesse des protocoles tels que TLS et SSH. La cryptographie moderne, face aux défis croissants, y compris ceux posés par les menaces post-quantiques, repose sur ces bases théoriques pour assurer une protection efficace contre les attaques. Ainsi, l'évolution continue des algorithmes, tels que ceux de hachage et de cryptographie à clé publique, est cruciale pour maintenir la sécurité dans un environnement numérique en mutation rapide. La capacité à résister aux attaques de collision et aux vulnérabilités découvertes dans les algorithmes précédents montre également l'importance de maintenir une approche dynamique et réactive face aux progrès technologiques.

Chapitre 3 : Méthodes avancées de sécurité pour l'authentification et le stockage des données

3.1 Introduction

L'authentification est une étape essentielle pour sécuriser les systèmes informatiques et protéger les données sensibles. Elle permet de vérifier l'identité des utilisateurs et de limiter l'accès uniquement aux personnes autorisées. Cependant, les méthodes classiques, comme l'utilisation de mots de passe ou de clés, montrent leurs limites face à des cyberattaques de plus en plus sophistiquées.

Les systèmes basés sur des mots de passe souffrent de plusieurs faiblesses. Beaucoup d'utilisateurs choisissent des mots de passe faciles à deviner ou utilisent les mêmes pour plusieurs comptes. Ces pratiques exposent les systèmes à des attaques par force brute, où des programmes testent un grand nombre de combinaisons, et à des attaques par dictionnaire, qui s'appuient sur des listes de mots de passe courants [Joshi 21, Zhou 19]. De plus, lorsque des bases de données contenant des mots de passe hachés sont compromises, les attaquants peuvent tenter de retrouver les mots de passe d'origine à l'aide d'outils dédiés.

Pour réduire ces risques, des solutions modernes ont été développées. Par exemple, les algorithmes de hachage comme SHA-3, bcrypt et argon2 transforment les mots de passe en empreintes uniques et sécurisées. Bien que ces algorithmes soient efficaces, ils peuvent ralentir les systèmes en raison de leur complexité et ne protègent pas toujours suffisamment les bases de données en cas de compromission [Ogala 23]. Par ailleurs, l'authentification multifactorielle, qui combine plusieurs moyens pour valider l'identité des utilisateurs, apporte une couche de sécurité supplémentaire. Cependant, elle peut être compliquée à mettre en œuvre et parfois moins pratique pour les utilisateurs.

Dans ce chapitre, nous présentons deux contributions majeures pour l'authentification et le stockage sécurisé des données :

La première contribution [Touil 20] consiste en une rotation aléatoire des hachages afin de compliquer toute tentative de rétro-ingénierie. Elle offre une sécurité renforcée en masquant le véritable hachage stocké dans les bases de données et en compliquant les attaques par force brute, tout en permettant aux utilisateurs de choisir des phrases de passe plus faciles à mémoriser sans sacrifier la sécurité. Cette approche assure également une flexibilité accrue avec

plusieurs options de rotation, augmentant la résistance aux attaques tout en minimisant les compromis au niveau des performances des systèmes cryptographiques.

La deuxième contribution [Touil 24a] propose une nouvelle manière de hacher les mots de passe en les transformant en écriture Braille, créant ainsi une méthode unique de cryptage. Cette transformation rend les hachages plus difficiles à décrypter, même en cas de compromission de la base de données, en ajoutant une dimension supplémentaire de complexité et de sécurité. En divisant les hachages en blocs, puis en appliquant une substitution aléatoire, cette méthode rend la chaîne chiffrée six fois plus longue que le hachage initial, offrant une protection plus forte contre les attaques par force brute ou les attaques statistiques.

Ce chapitre explore les méthodes proposées en détail. L'objectif est de montrer leur efficacité à travers l'analyse de cas d'usage réels et de souligner leur potentiel à renforcer la sécurité des systèmes modernes d'authentification et de stockage des données.

3.2 Techniques modernes de stockage et d'authentification des phrases de passe

Le domaine du stockage sécurisé des phrases de passe a vu émerger plusieurs techniques modernes qui visent à surmonter les faiblesses des approches traditionnelles comme le hachage simple. Parmi ces approches, des méthodes telles que le hachage renforcé, la gestion multifactorielle des clés et le système de jetons se distinguent par leur efficacité contre les attaques modernes, mais elles ne sont pas sans défauts. Comparées à des approches plus sophistiquées utilisant des transformations mathématiques complexes, ces techniques présentent certaines vulnérabilités.

3.2.1 Hachage avec algorithmes de renforcement

Les algorithmes tels que bcrypt [Skanda 22], scrypt [Alwen 16] et argon2 [Biryukov 15] ont été développés pour répondre à la nécessité de renforcer la sécurité des mots de passe face aux attaques par force brute, où un attaquant tente systématiquement toutes les combinaisons possibles. Leur principe fondamental repose sur l'ajout intentionnel de lenteur au processus de hachage, un mécanisme qui rend chaque tentative d'attaque plus longue et coûteuse en ressources. Cette lenteur est rendue mesurable grâce à des modèles mathématiques, comme celui exprimé par la formule (2.13), qui établit une relation directe entre la taille du message ou du mot de passe (n) et le temps nécessaire pour compléter le hachage. En multipliant les étapes

intermédiaires, ces algorithmes augmentent de manière exponentielle la complexité pour un attaquant, sans compromettre leur efficacité dans les systèmes légitimes. Cependant, cette approche introduit un paradoxe intéressant. Si la lenteur impose un coût élevé pour une attaque, elle n'altère pas la structure mathématique fondamentale du mot de passe ou de son empreinte hachée. Cela laisse une ouverture théorique, car la sécurité repose uniquement sur la difficulté pratique de calcul et non sur une transformation sécurisée. Ainsi, bien que bcrypt, scrypt et argon2 soient des standards largement adoptés pour contrer les attaques courantes, leur dépendance à la lenteur comme mécanisme central soulève des questions sur leur capacité à résister aux menaces futures, notamment celles issues de la cryptographie avancée ou des systèmes post-quantiques. En examinant la formule sous-jacente, on comprend que ces algorithmes cherchent à équilibrer performance et sécurité en adaptant les paramètres, tels que le facteur de coût ou la mémoire utilisée, afin de s'adapter à l'évolution des capacités matérielles. Cela reflète une philosophie de conception où l'objectif est de retarder l'attaquant sans pénaliser les utilisateurs légitimes [Hatzivasilis 17].

Limites :

- **Performance** : sur le plan des performances, ces algorithmes ajoutent volontairement des étapes de calcul intensives pour ralentir les attaques. Cependant, cette lenteur devient un handicap dans des environnements où des centaines ou des milliers d'utilisateurs doivent être authentifiés simultanément. La formule (2.14), qui représente une complexité exponentielle en fonction de la longueur de l'empreinte du hachage m , illustre comment ces calculs peuvent augmenter de manière disproportionnée le temps requis pour le traitement, en particulier lorsque les systèmes sont sous forte charge. Cela entraîne une diminution significative de l'efficacité, notamment dans des environnements à grande échelle, comme les plateformes de services en ligne ou les systèmes critiques.
- **Dépendance au calcul intensif** : un autre problème majeur réside dans la dépendance exclusive à la lenteur pour renforcer la sécurité. Bien que ces méthodes augmentent le coût temporel et énergétique d'une attaque, elles n'introduisent pas de complexité mathématique supplémentaire dans la structure des mots ou phrases de passe. Cela signifie que la transformation mathématique sous-jacente reste linéaire et que la sécurité repose principalement sur la puissance de calcul nécessaire pour casser l'algorithme. En conséquence, des attaques parallélisées, comme celles utilisant des GPU ou des clusters distribués, peuvent réduire considérablement le temps nécessaire pour déchiffrer un

hachage. Cela met en évidence une faiblesse fondamentale c'est que bien qu'elles soient coûteuses, ces méthodes ne changent pas fondamentalement, la vulnérabilité des données hachées [Menasce 03].

Comparativement, une approche introduisant des transformations dynamiques, comme celle mise en œuvre par la méthode « H-Rotation », présente un avantage majeur, elle ajoute une complexité mathématique significative sans augmenter excessivement la dépendance au calcul. Contrairement aux algorithmes existants, qui se basent principalement sur la lenteur pour contrer les attaques, une transformation dynamique modifie périodiquement et de manière imprévisible la structure des valeurs de hachage. Cela rend chaque tentative de décryptage exponentiellement plus complexe pour un attaquant, même équipé de ressources avancées comme des GPU ou des systèmes distribués. En effet, cette dynamique crée une dépendance non linéaire entre les paramètres d'entrée, les valeurs de hachage et le temps requis pour reconstituer les données originales. Par exemple, en introduisant des variations aléatoires et des dépendances contextuelles dans le processus de hachage, les transformations dynamiques augmentent le niveau de désordre (entropie) dans les systèmes. Cette entropie accrue complique la tâche de l'attaquant, car chaque tentative doit inclure non seulement le calcul initial, mais aussi la résolution des transformations spécifiques au contexte, rendant les attaques traditionnelles moins efficaces. Cette approche dépasse les limites des méthodes basées uniquement sur la lenteur en introduisant une dimension mathématique supplémentaire, ce qui répond aux menaces modernes sans sacrifier les performances dans des environnements à grande échelle. Elle ouvre également la voie à des systèmes plus résilients, adaptés aux avancées technologiques futures, comme l'émergence de l'informatique quantique. Ainsi, la transformation dynamique représente une évolution fondamentale dans la conception des systèmes de sécurité, en établissant un équilibre optimal entre robustesse et efficacité.

3.2.2 Gestion multifactorielle des clés

Les systèmes de gestion des clés ajoutent une couche supplémentaire de sécurité en liant le mot de passe à une clé cryptographique. Ce processus de validation via le KMS permet de garantir qu'un attaquant ne puisse pas accéder directement aux phrases/mots de passe sans la clé associée [Chen 06]. Ce processus repose sur des opérations cryptographiques telles que l'exponentiation modulaire représentée dans la formule (2.2), où a est une base, b est une clé privée, et n un grand nombre premier. L'exponentiation modulaire est résistante aux attaques,

car le problème de trouver b à partir de $a^b \bmod n$ est un problème de logarithme discret, réputé difficile.

Limites :

- **Complexité de la gestion** : la gestion des clés à grande échelle devient rapidement complexe, surtout lorsque des milliers d'utilisateurs doivent fréquemment renouveler ou valider leurs clés. Cette complexité augmente le risque de mauvais déploiement, laissant des failles exploitables dans le système.
- **Risque de compromis centralisé** : si le serveur de gestion des clés est compromis, toute la sécurité du système est menacée. Le KMS devient ainsi un point de défaillance unique qui, s'il est attaqué, peut rendre vulnérable l'intégralité du réseau.

En comparaison, une méthode basée sur des transformations locales et dynamiques permet de distribuer la sécurité sans dépendre d'une entité centralisée, réduisant ainsi le risque de compromission à grande échelle.

3.2.3 Hachage matriciel

Le hachage matriciel consiste à appliquer des transformations linéaires via des matrices avant le processus de hachage. Cela modifie la structure de phrase/mot de passe en y ajoutant une transformation mathématique difficile à inverser, comme décrite dans la formule (2.8). Ce qui produit un vecteur transformé $T(x)$ qui est ensuite haché par une fonction traditionnelle $H(T(x))$ [Schneier 07]. La transformation par hachage matriciel suit une complexité décrite par la formule (2.15), renforçant ainsi la sécurité globale.

Avantages :

- **Complexité accrue** : l'ajout d'une transformation mathématique supplémentaire rend l'attaque beaucoup plus difficile. Les attaquants doivent non seulement casser le hach, mais aussi comprendre et inverser la transformation spécifique appliquée.
- **Adaptabilité** : cette technique peut être ajustée en fonction des besoins spécifiques du système, en modifiant les matrices utilisées pour les transformations, ce qui rend chaque hach unique et imprévisible.

Cependant, la complexité introduite par ces transformations peut être gourmande en ressources si elle n'est pas optimisée. Une approche équilibrée pourrait combiner ces transformations mathématiques tout en restant efficace en termes de calcul.

3.3 H-Rotation : stockage sécurisé et récupération des phrases de passe dans le processus d'authentification

Cette méthode de sécurisation du stockage des phrases de passe et de l'authentification a été développée pour combler les faiblesses des systèmes traditionnels, souvent exposés à des attaques telles que la force brute et les attaques par rainbow table. Bien que des solutions comme bcrypt et scrypt soient efficaces pour ralentir les attaques en augmentant le temps nécessaire à chaque tentative, elles présentent des limites. Ces algorithmes n'introduisent pas de changements significatifs dans la structure des mots de passe ou des hachages, ce qui les rend vulnérables face à des attaques utilisant des ressources matérielles puissantes, comme des architectures parallèles ou des processeurs graphiques (GPU).

Pour dépasser ces limites, cette méthode propose d'utiliser des transformations dynamiques sur les hachages. Ces transformations modifient la structure des hachages à chaque étape, rendant leur décryptage beaucoup plus difficile. Ainsi, les attaquants doivent résoudre des équations de plus en plus complexes à chaque tentative, ce qui renforce considérablement la sécurité du système contre les attaques par force brute et par rainbow table. Contrairement aux approches traditionnelles, qui augmentent uniquement le temps de calcul pour chaque tentative, cette méthode minimise la dépendance à des calculs lourds. Cela la rend plus adaptée aux systèmes à grande échelle, où des milliers d'utilisateurs doivent être authentifiés en même temps sans compromettre la performance globale. En modifiant continuellement la forme des hachages, cette approche élimine toute possibilité de traçabilité. Les attaques massives ou utilisant des tables précalculées deviennent inefficaces, car chaque hachage est unique et change au fil du temps. De plus, cette méthode est très flexible. Elle peut être ajustée pour répondre aux besoins spécifiques de chaque système, qu'il s'agisse de contrer des types d'attaques particuliers ou de respecter des contraintes de performance. Cette flexibilité est essentielle dans des environnements modernes où la sécurité doit être personnalisée et adaptée en fonction des menaces rencontrées et des exigences opérationnelles [Touil 20].

3.3.1 Présentation du concept de « H-Rotation »

La méthode repose sur une combinaison de techniques de hachage cryptographique et de transformations dynamiques appliquées aux haches avant le stockage dans une base de données. L'objectif est d'introduire un niveau supplémentaire de complexité mathématique, afin que les

haches ne puissent pas être inversés ou utilisés dans des attaques, même si un attaquant parvient à accéder à la base de données.

3.3.1.1 Transformations dynamiques

Tout d'abord, nous générons un hach de la phrase de passe utilisateur en utilisant une fonction de hachage cryptographique sécurisée, comme SHA-3. Cette fonction prend la phrase de passe x comme entrée et génère un hach $H(x)$, mathématiquement, cela s'écrit :

$$H(x) = SHA3(x) \quad (3.1)$$

Où x est la phrase de passe et $H(x)$ est un hachage de longueur fixe T .

La fonction de hachage garantit que la moindre modification de x produit un hach totalement différent et que le hach est de longueur fixe. Cette transformation repose sur la complexité décrite dans la formule (2.3) pour garantir une protection supplémentaire contre les attaques par inversion.

Une fois le hach généré suivant la formule (3.1), le hachage $H(x)$ est divisé en M -blocs, où la taille des blocs M respecte la contrainte $M < \frac{T}{2}$, où T est la taille du hach.

$H(x)$ est composé de T caractères, chaque bloc H_i (avec $i=1, 2, \dots, M$) contient $\frac{T}{M}$ caractères. La séparation peut être modélisée par :

$$H_{(x)} = \bigcup_{i=1}^M H_i \quad (3.2)$$

Une rotation est ensuite effectuée sur ces blocs. Le nombre de rotations N est un entier généré aléatoirement qui détermine combien de fois chaque bloc sera tourné.

3.3.1.2 Types de rotations

Il existe trois types principaux de rotations qui peuvent être appliquées, chacune ajoutant un niveau de sécurité différent en fonction des besoins du système.

- **Rotation en préfixes** : la rotation des préfixes consiste à déplacer les caractères d'un bloc H_i vers la gauche. Cela peut être exprimé mathématiquement comme suit :
 - *Bloc initial* : soit H_i un bloc de taille k (en caractères) extrait du hachage $H(x)$:

$$H_i = (b_1, b_2, \dots, b_k)$$

- *Rotation gauche* : la fonction $\text{RotateLeft}(H_i, N)$ effectue une permutation circulaire des caractères vers la gauche de N positions :

$$H'_i = (b_{N+1}, b_{N+2}, \dots, b_k, b_1, b_2, \dots, b_N)$$

- *Expression générale* : si on indexe les caractères de H_i par j (où $1 \leq j \leq k$), alors le caractère j après rotation se retrouve à la position :

$$H'_i[j] = H_i[(j + N - 1) \bmod k + 1] \quad (3.3)$$

- **Rotation en suffixes** : La rotation des suffixes est similaire à celle des préfixes, mais les bits sont déplacés vers la droite.

- *Bloc initial* : Soit $H_i = (b_1, b_2, \dots, b_k)$

- *Rotation droite* : la fonction $\text{RotateRight}(H_i, N)$ effectue une permutation circulaire des caractères vers la droite de N positions :

$$H'_i = (b_{k-N+1}, b_{k-N+2}, \dots, b_k, b_1, b_2, \dots, b_{k-N})$$

Expression générale : Si on indexe les bits de H_i par j , le caractère j après rotation se retrouve à la position :

$$H'_i[j] = H_i[(j - N - 1 + k) \bmod k + 1] \quad (3.4)$$

- **Rotation en milieu** : le hachage $H(x)$ est divisé en M blocs égaux $(H_1, H_2, \dots, H_M)$ où M est pair pour garantir une symétrie. Les blocs sont organisés comme suit :

- Les blocs de gauche $H_{\text{left}} = (H_1, H_2, \dots, H_{M/2})$.

- Les blocs de droite $H_{\text{right}} = (H_M, H_{M-1}, \dots, H_{M/2+1})$.

Les rotations se font en parallèle :

Blocs à gauche (H_{left}) : ils sont déplacés vers la droite en direction du centre.

Blocs à droite (H_{right}) : ils sont déplacés vers la gauche en direction du centre.

Cela est exprimé mathématiquement comme suit pour chaque bloc H_i :

$$H'_i = \text{RotateLeft}(H_i, N) \quad \text{pour } i \in [1, M/2], \quad (3.5)$$

$$H'_{M-i+1} = \text{RotateRight}(H_{M-i+1}, N) \quad \text{pour } i \in [1, M/2].$$

Une fois les rotations effectuées, les blocs de gauche et de droite sont fusionnés de manière à se croiser au centre. Le résultat final après rotation au milieu peut être représenté comme :

$$H'(x) = \text{CenterRotate}(H_{\text{left}}, H_{\text{right}}, N) \quad (3.6)$$

Où CenterRotate applique simultanément RotateLeft et RotateRight en fonction des indices i .

3.3.1.3 Description des paramètres utilisés

Les transformations dynamiques s'appuient sur trois paramètres principaux décrits dans le tableau 3- 1 ci-après :

Tableau 3- 1 : Description des paramètres utilisés

Paramètre	Description
T	Taille du hach.
M	Taille du bloc utilisé dans la rotation ; $M < T/2$.
N	Le nombre des itérations effectuées dans la rotation.

3.3.1.4 Processus d'inscription et d'authentification « H-Rotation »

Le processus d'authentification « H-Rotation » se divise en deux grandes phases : l'inscription initiale et les demandes d'authentification ultérieures. Chaque étape du processus est soigneusement orchestrée pour rendre l'inversion du hach difficile, même en cas d'accès non autorisé à la base de données.

- Inscription initiale

Tout commence lors de l'inscription de l'utilisateur. L'interface d'inscription demande à l'utilisateur de fournir un nom d'utilisateur et une phrase de passe. La phrase de passe ayant un rôle crucial dans le processus de sécurisation, car elle sera utilisée comme entrée pour générer un hach unique. Une fois les données soumises, elles sont transmises au serveur, qui applique immédiatement l'algorithme de hachage SHA-3 sur la phrase de passe. Cet algorithme produit un hach à partir de la phrase de passe, garantissant qu'il est pratiquement impossible de retrouver la phrase de passe initiale à partir du hach.

- Application de la « H-Rotation »

L'étape suivante consiste à effectuer une rotation sur le hach généré. Cette rotation est une transformation dynamique qui modifie la structure du hach sans en changer la taille. Cette transformation rend le hach encore plus difficile à inverser en introduisant un facteur de complexité supplémentaire. L'algorithme effectue cette rotation en utilisant des paramètres spécifiques qui seront importants lors des futures demandes d'authentification. Ce processus permet d'assurer que même si un attaquant parvient à accéder à la base de données et obtient le hach, il ne pourra pas facilement en déduire la phrase de passe d'origine.

- Enregistrement du hach et communication des paramètres

Une fois que la « H-Rotation » a été appliquée, le hach modifié est enregistré dans la base de données pour une utilisation future lors de l'authentification. Mais, ce n'est pas tout : l'utilisateur reçoit également trois entiers spécifiques qui sont générés lors de la transformation. Ces entiers sont communiqués à l'utilisateur par un canal sécurisé (par e-mail ou par SMS). Chacun de ces entiers a une signification précise (sans la mentionner dans le message communiqué) :

- Le premier entier indique le type de rotation (préfixe, suffixe ou milieu).
- Le second entier correspond à la taille du bloc utilisé lors de la rotation.
- Le troisième entier précise le nombre de rotations effectuées.

Ces trois paramètres sont essentiels, car ils définissent exactement comment le hach a été transformé, et ils seront nécessaires lors des futures authentifications. L'algorithme « H-Rotation » est décrit dans la section annexe A.3 ainsi que la figure 3-1 décrit le fonctionnement de la méthode « H-Rotation » dans la phase de souscription.

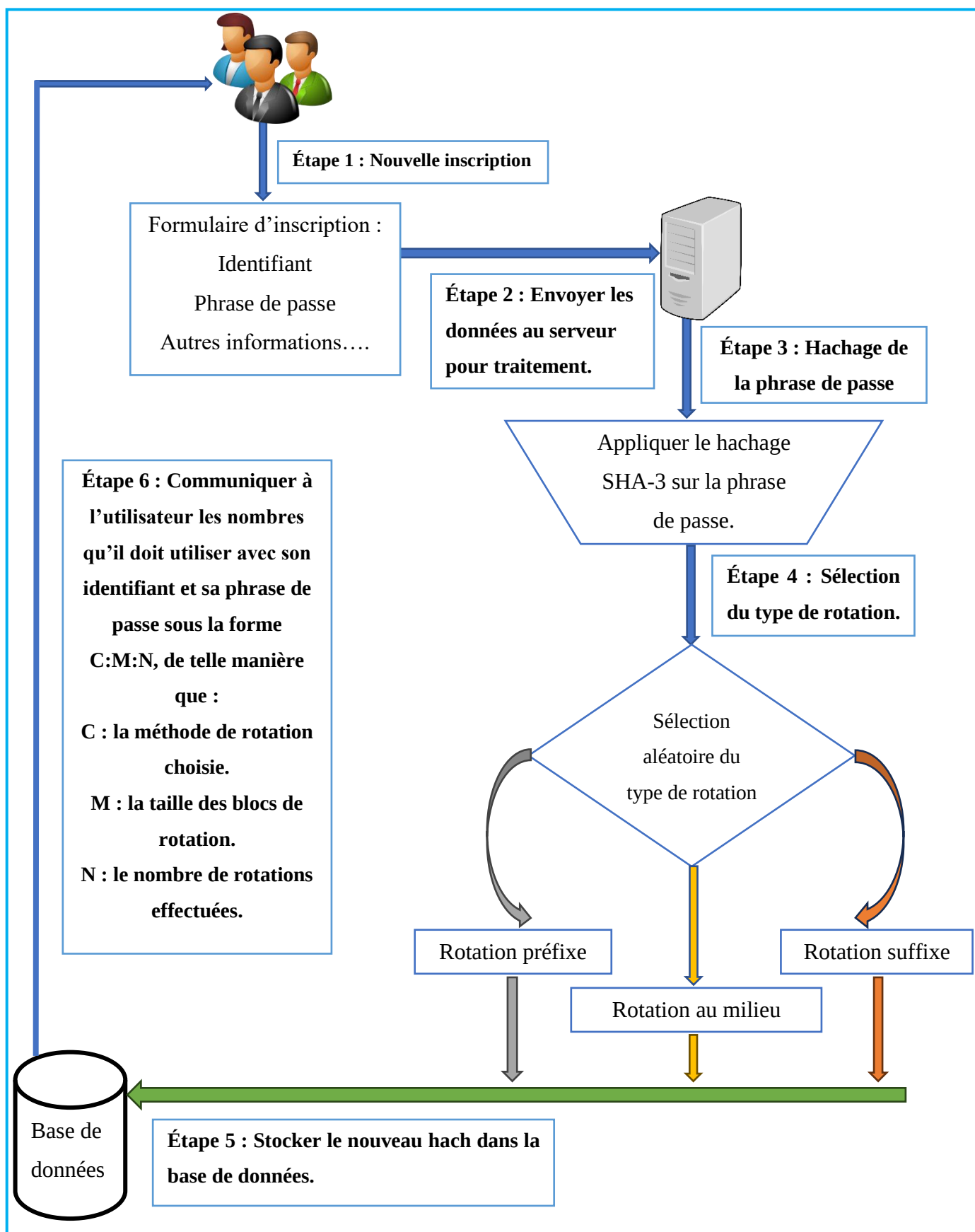


Figure 3- 1 : Principe de fonctionnement « H-Rotation » dans la phase de souscription

- Authentification ultérieure

Lorsque l'utilisateur se connecte à nouveau, il doit fournir trois éléments : son nom d'utilisateur, sa phrase de passe, ainsi que les trois entiers qu'il a reçus lors de l'inscription. Le serveur récupère ces informations et recrée le hach initial de la phrase de passe en appliquant l'algorithme SHA-3. Ensuite, le serveur applique la même « H-Rotation » en utilisant les trois entiers fournis par l'utilisateur. Si le hach transformé correspond à celui stocké dans la base de données, l'utilisateur est authentifié avec succès, comme le décrit la figure 3-2 ci-après.

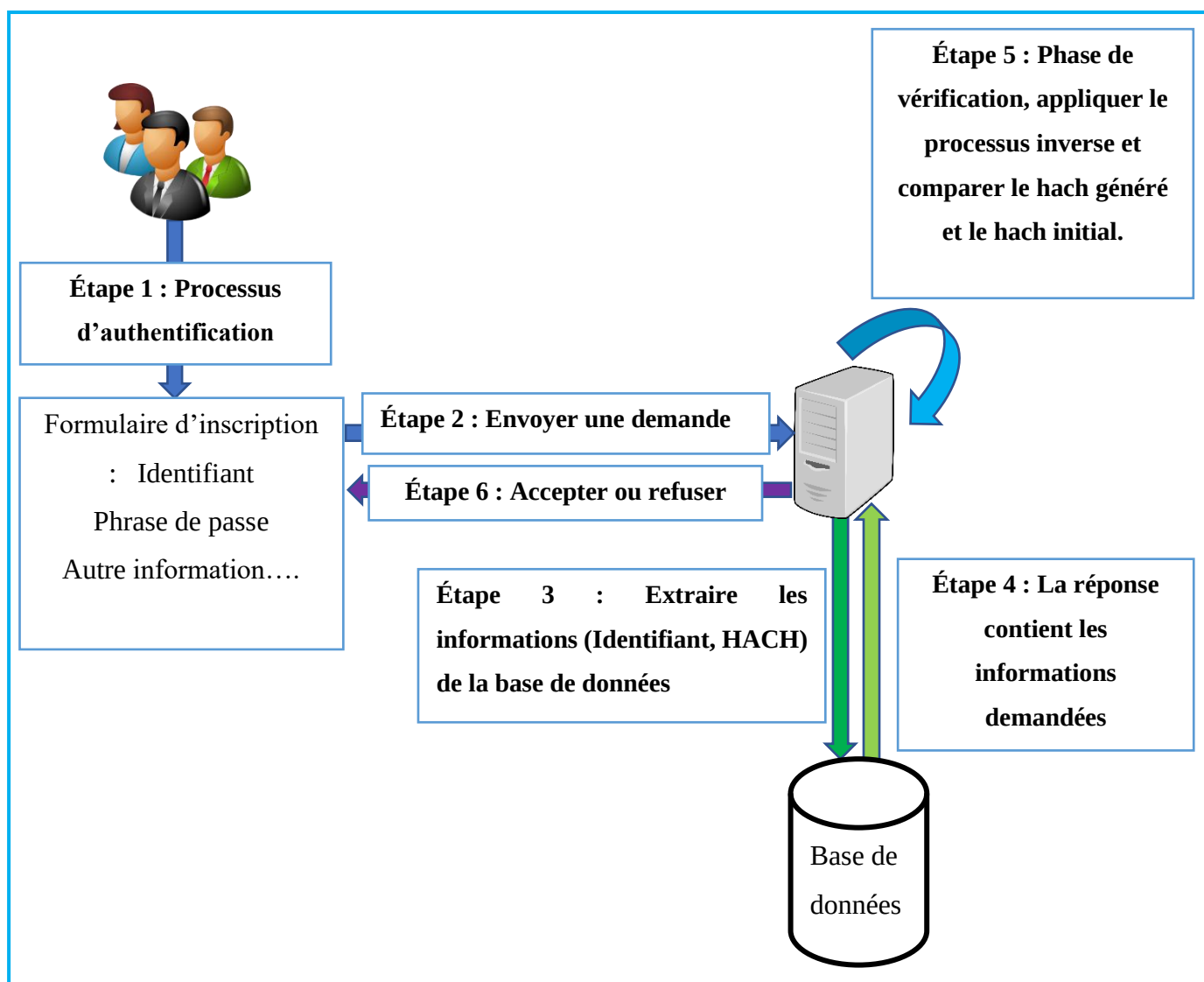


Figure 3- 2 : Principe de fonctionnement « H-Rotation » dans la phase d'authentification.

3.3.2 Explication des méthodes de rotation

Une fois que l'algorithme SHA-3 génère le hachage initial, une rotation dynamique est immédiatement appliquée pour renforcer la sécurité. Cette rotation peut être effectuée de trois manières, rendant presque impossible de retrouver la phrase secrète à partir du hachage.

- Rotation de préfixe : cette méthode modifie la première partie du hachage, ce qui rend les premières valeurs difficiles à analyser ou à traquer, comme décrit dans la formule (3.3).
- Rotation de suffixe : elle agit sur la partie finale du hachage, ajoutant une complexité supplémentaire à la fin de la chaîne de caractères, comme décrit dans la formule (3.4).
- Rotation au milieu : le hachage est divisé en deux segments qui sont ensuite inversés ou pivotés de manière symétrique. Cela complique encore davantage toute tentative d'inversion. Ce mécanisme est représenté dans la formule (3.5).

Ces trois types de rotation, lorsqu'ils sont combinés, garantissent que chaque hachage est unique, même si la phrase de passe est facile à mémoriser. Cela simplifie également la gestion pour le système, car chaque rotation appliquée équivaut à une nouvelle couche de cryptage. Cette approche rend les attaques sur la base de données presque inutiles pour un attaquant, car chaque tentative devient extrêmement complexe et coûteuse.

3.3.2.1 Rotation dans les préfixes

Cette technique s'appuie sur l'application de rotations spécifiques aux blocs de bits du hach, avec pour objectif de rendre le résultat final difficilement prévisible et pratiquement impossible à inverser. Le principe est de diviser le hach en segments et de faire pivoter certains blocs selon des paramètres calculés aléatoirement.

Le processus commence par la génération de deux nombres aléatoires :

- ***M*** : ce nombre représente la taille du bloc de caractères sur lequel la rotation sera appliquée. Pour garantir qu'au moins une rotation soit effectuée, la taille du bloc *M* doit être strictement supérieure à zéro et inférieure à la moitié de la taille totale du hach ($T/2$). Cette restriction permet d'assurer qu'au moins une rotation soit effectuée et que le bloc sélectionné soit suffisant pour produire une modification significative de la structure du hach.
- ***N*** : ce nombre, également généré aléatoirement, représente le nombre de rotations à effectuer sur le bloc sélectionné. Il doit être supérieur à 0, garantissant qu'au moins une rotation soit effectuée.

Ces deux nombres, M et N , sont les paramètres clés qui déterminent la manière dont la rotation sera appliquée. Ils sont générés selon une table d'annotation spécifique, ce qui permet de varier les configurations en fonction des besoins de sécurité et de performance.

Une fois M et N déterminés, la rotation est appliquée au bloc de bits sélectionné. Il existe plusieurs cas de figure en fonction des valeurs de M et de la taille totale du hach T . Ces configurations permettent de moduler le nombre de rotations, allant d'une seule rotation pour deux blocs à plusieurs rotations lorsque les blocs sont plus petits.

Voici les différents scénarios possibles :

- **Cas 1** : si M se situe entre $T/4$ et $T/2$, une seule rotation est appliquée. Cette rotation s'effectue de droite vers la gauche sur un bloc de la taille de $T/2$. Ce scénario est typiquement utilisé lorsque la sécurité de base est suffisante et que l'on cherche à minimiser les ressources de calcul nécessaires à la transformation. L'efficacité de la rotation est renforcée par les principes de la formule (2.11), qui rend la récupération du hachage initial quasi impossible.
- **Cas 2** : si M est compris entre $T/8$ et $T/4$, plusieurs rotations sont nécessaires. Dans ce cas, N prend une valeur comprise entre 1 à 3, ce qui signifie que le bloc sélectionné subira entre une et trois rotations de droite vers la gauche. L'objectif ici est de rendre le hach plus imprévisible tout en maintenant un niveau de performance élevé.
- **Cas 3** : lorsque M se trouve entre $T/16$ et $T/8$, le nombre de rotations augmente encore. N peut alors prendre une valeur comprise entre 1 à 7, indiquant que le bloc subira jusqu'à sept rotations. À ce stade, la complexité du hach augmente de manière significative, rendant toute tentative d'inversion beaucoup plus difficile.
- **Cas 4** : si M est compris entre $T/32$ et $T/16$, le bloc sélectionné est encore plus petit et N peut prendre une valeur comprise entre 1 à 15. Le hach commence à subir un très grand nombre de rotations, augmentant considérablement la complexité de sa structure interne. Ce scénario est utilisé lorsque la sécurité est une priorité absolue.
- **Cas 5** : enfin, lorsque M se situe entre $T/64$ et $T/32$, le bloc sur lequel la rotation s'applique est extrêmement petit. Dans ce cas, N peut atteindre jusqu'à 31 rotations, rendant le hach presque impossible à inverser sans connaître précisément les valeurs de M et N . Cette configuration est idéale pour les environnements nécessitant une sécurité maximale, où même des attaques sophistiquées auraient peu de chance de réussir. La figure 3-3 récapitule les différents cas de figure.

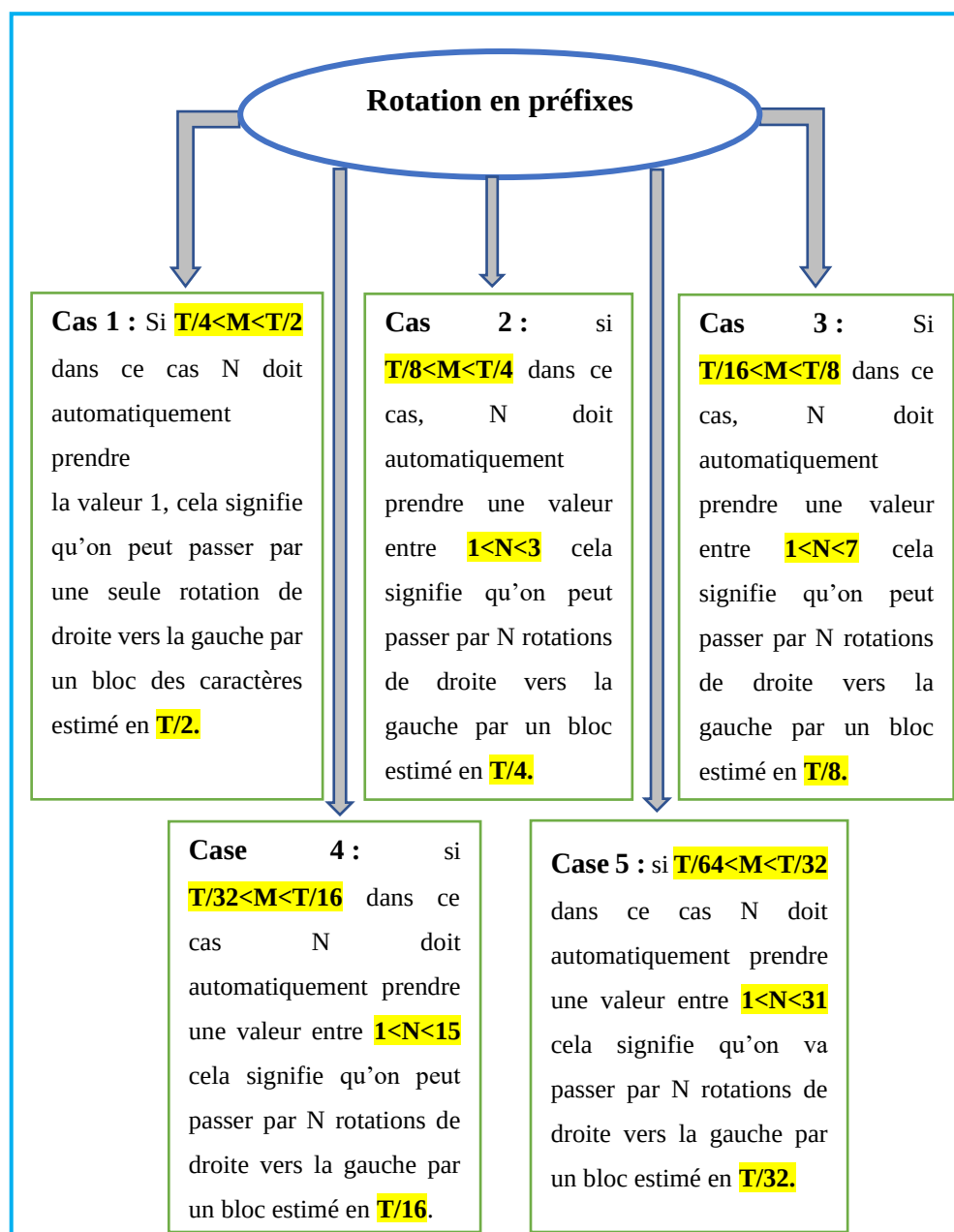


Figure 3- 3 : Différentes possibilités de la méthode de rotation en préfixe

3.3.2.2 Rotation dans les suffixes

Cette technique repose sur la génération de deux nombres aléatoires, M et N , qui définissent respectivement la taille du bloc sur lequel la rotation sera appliquée et le nombre de rotations nécessaires pour modifier ce bloc. L'objectif est de rendre la partie finale du hach difficile à inverser, tout en garantissant que la transformation reste complexe et unique.

Le processus débute avec la génération de deux paramètres essentiels :

- **M** : ce nombre aléatoire représente la taille du bloc de bits ou d'octets sur lequel la rotation sera appliquée. Pour être efficace, M doit strictement être supérieur à 0 et inférieur à la moitié de la taille totale du hach ($T/2$), garantissant ainsi qu'au moins une rotation est effectuée.
- **N** : ce second paramètre, également généré aléatoirement à partir de M , détermine le nombre de rotations à effectuer sur le bloc sélectionné. N doit être supérieur à 0, ce qui assure qu'au moins une rotation a lieu. La combinaison des paramètres M et N varie selon la taille du hach, afin d'assurer que chaque transformation soit unique.

La rotation s'applique sur la partie finale du hach, ce qui modifie le contenu des bits des suffixes et complexifie toute tentative d'inversion ou d'analyse. Selon la valeur de M , plusieurs scénarios peuvent se présenter :

- **Cas 1** : lorsque M est compris entre $T/4$ et $T/2$, une seule rotation de gauche vers la droite est effectuée sur un bloc de suffixes de la taille $T/2$. Ce scénario est optimal lorsque l'on cherche à maintenir la performance tout en appliquant une sécurité de base.
- **Cas 2** : si M est compris entre $T/8$ et $T/4$, plusieurs rotations peuvent être appliquées. Le nombre N varie alors entre 1 à 3, ce qui signifie que le bloc sélectionné subit d'une à trois rotations de gauche à droite. Cette transformation rend la fin du hach imprévisible et difficile à inverser.
- **Cas 3** : lorsque M est situé entre $T/16$ et $T/8$, N peut aller jusqu'à sept rotations. Le hach subit ainsi un changement considérable et la transformation de la partie finale du hach devient beaucoup plus complexe à suivre.
- **Cas 4** : si M est compris entre $T/32$ et $T/16$, N peut atteindre jusqu'à 15 rotations, rendant la structure du hach presque totalement imprévisible. Ce niveau de transformation est particulièrement recommandé dans les environnements où la sécurité est prioritaire.
- **Cas 5** : enfin, si M est compris entre $T/64$ et $T/32$, le nombre de rotations N peut atteindre jusqu'à 31 rotations. Cela modifie considérablement les suffixes du hach et complique énormément toute tentative de décryptage ou d'inversion du hach sans les valeurs exactes de M et N . La figure 3-4 récapitule les différents cas de figure.

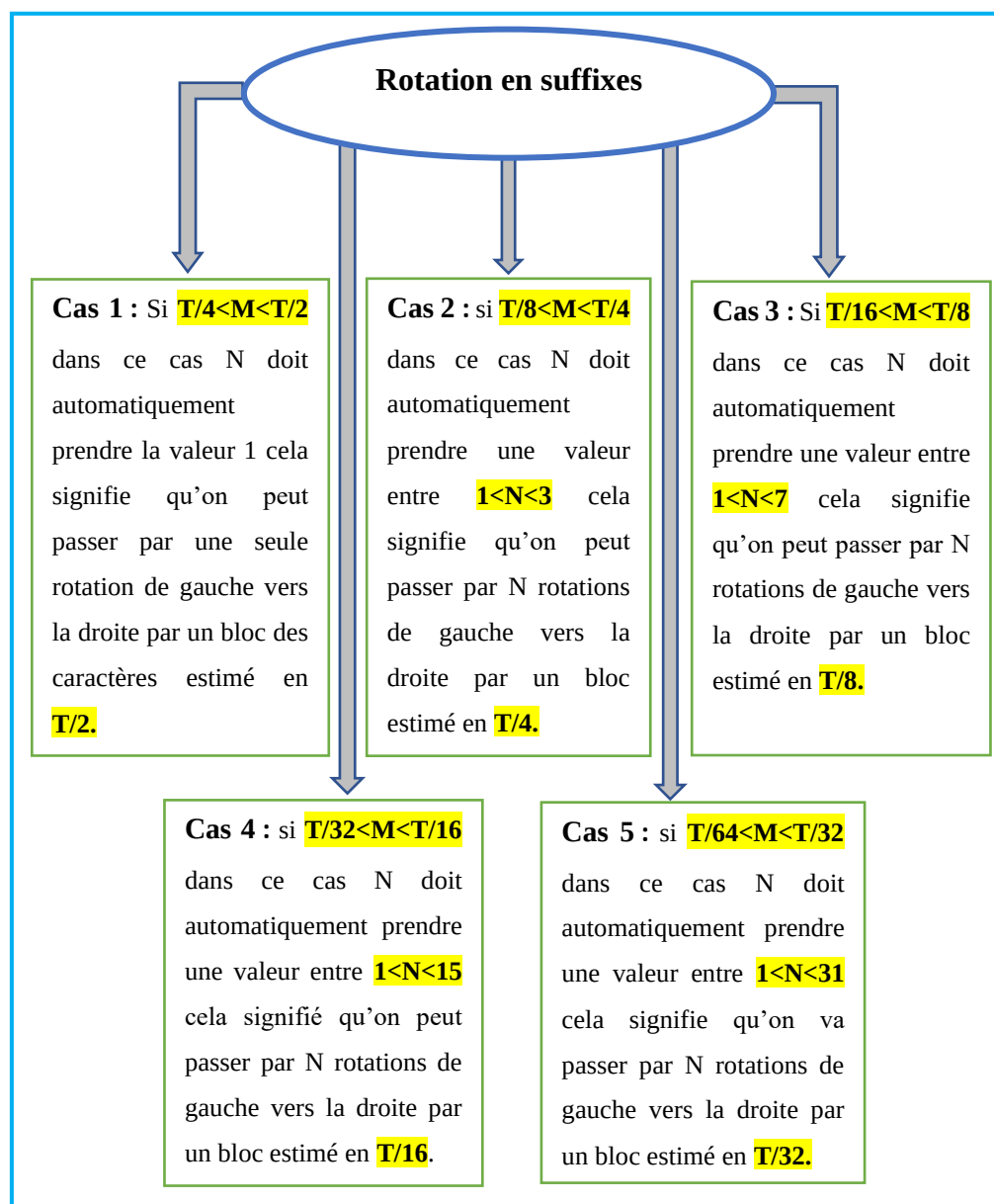


Figure 3- 4 : Différentes possibilités de la méthode de rotation en suffixe

3.3.2.3 Rotation au milieu

Cette approche est unique, car elle transforme à la fois la partie initiale et finale du hach, modifiant ainsi l'ensemble de sa structure de manière complexe. La rotation se fait précisément autour de la position centrale $T/2$ du hach. Les blocs de bits ou d'octets, de part et d'autre du milieu, sont pivotés vers le centre en utilisant des paramètres aléatoires pour maximiser l'unicité et la complexité du hach final.

Le processus débute par la génération de deux nombres aléatoires, M et N , qui déterminent la manière dont la rotation sera appliquée. Cette transformation s'appuie sur les principes mathématiques résumés par la formule (2.16) pour ajouter une complexité exponentielle aux rotations :

- **M** : ce nombre aléatoire représente la taille du bloc de bits ou d'octets qui sera utilisé pour la rotation. M doit strictement être supérieur à 0 et inférieur à la moitié de la taille totale du hach ($T/2$) pour garantir qu'au moins une rotation soit effectuée.
- **N** : ce second nombre aléatoire détermine le nombre exact de rotations à appliquer sur les blocs situés aux deux extrémités du hach, ces rotations s'effectuant vers le milieu. N est toujours supérieur à 0, garantissant que le hach subira au moins une transformation significative.

La rotation est appliquée à partir des deux extrémités du hach, les blocs de chaque côté étant pivotés vers le centre. Selon la valeur de M , plusieurs scénarios peuvent se produire :

- **Cas 1** : si M est compris entre $T/4$ et $T/2$, une seule rotation est effectuée sur chaque extrémité, les blocs ayant une taille égale à $T/4$. Cela signifie que la partie initiale et finale du hach convergent vers le milieu après une seule rotation. Ce scénario est utilisé lorsque l'on cherche à introduire une transformation légère, mais efficace.
- **Cas 2** : si M est compris entre $T/8$ et $T/4$, plusieurs rotations peuvent être appliquées. Dans ce cas, N peut varier entre 1 à 3, ce qui signifie que les blocs situés aux deux extrémités subiront entre une et trois rotations vers le milieu. Ce processus modifie de manière significative la structure centrale du hach, tout en maintenant une complexité modérée.
- **Cas 3** : lorsque M est compris entre $T/16$ et $T/8$, N peut aller jusqu'à sept rotations. Les blocs tournés vers le centre subissent alors un changement important, ce qui complique davantage toute tentative de décryptage.
- **Cas 4** : si M est compris entre $T/32$ et $T/16$, N peut atteindre jusqu'à 15 rotations. Ce niveau de transformation affecte fortement les extrémités et la portion centrale du hach, rendant l'inversion encore plus complexe.
- **Cas 5** : si M est compris entre $T/64$ et $T/32$, N peut atteindre jusqu'à 31 rotations, maximisant ainsi la complexité du hach. Cette transformation est utilisée dans des environnements où la sécurité est critique, car elle rend pratiquement impossible toute

tentative d'inversion du hach sans connaître les paramètres M et N. La figure 3-5 récapitule les différents cas de figure.

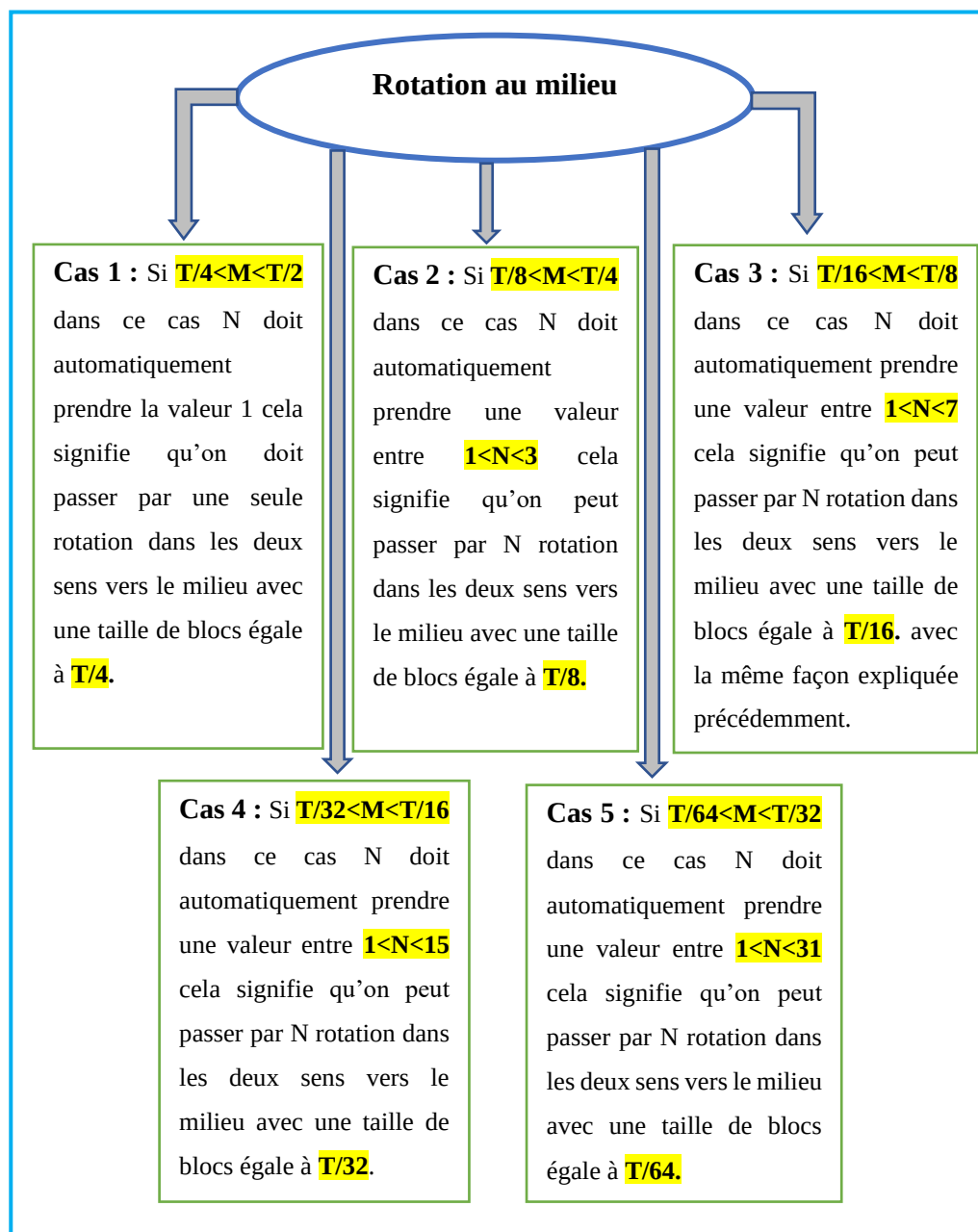


Figure 3- 5 : Différentes possibilités de la méthode de rotation au milieu

3.3.3 Résultat expérimental

Dans cette partie, nous allons montrer des cas pratiques d'application des méthodes de rotation pour sécuriser les hachages. Chaque méthode, rotation sur les préfixes, rotation sur les suffixes et rotation au milieu, sera illustrée par des exemples concrets qui montrent comment ces techniques modifient les hachages pour garantir un niveau de sécurité supérieur. En partant de

situations réalistes, nous analyserons comment chacune de ces rotations affecte la structure des hachages générés par des fonctions cryptographiques comme SHA-3. Ces exemples permettront de comprendre non seulement le fonctionnement théorique des rotations, mais aussi leur mise en œuvre pratique dans des environnements sécurisés.

3.3.3.1 Rotation dans les préfixes

Nous utiliserons comme donnée d'entrée :

Phrase de passe : Hamza TOUIL

Hach SHA-3 : 1d97f27385e052fd7b2df17846ea4d3943fc86f0b3af531588f6e67b7df23844

Cas 1 : dans cette première expérimentation, nous considérons le cas d'une rotation simple avec les paramètres suivants : $M=18$ et $N=1$. Ici, M est un nombre aléatoire compris entre $T/4$ et $T/2$, où T représente une valeur déterminée par la taille du bloc à hach. En fixant $N=1$, nous indiquons qu'une seule rotation sera appliquée sur le bloc de gauche.

Le processus se déroule de la manière suivante : le hachage original est divisé en deux parties égales. Une fois la séparation effectuée, la rotation se concentre uniquement sur le bloc de gauche, qui est soumis à une transformation selon l'algorithme de rotation.

Division en deux parties :

Première partie	1d97f27385e052fd7b2df17846ea4d39
Deuxième partie	43fc86f0b3af531588f6e67b7df23844

Rotation de la droite vers la gauche : le bloc de gauche est déplacé vers la droite, donnant :

- Nouveau hachage :

43fc86f0b3af531588f6e67b7df238441d97f27385e052fd7b2df17846ea4d39

Cas 2 : nous examinons une configuration impliquant plusieurs rotations avec les paramètres $M=13$ et $N=3$. Ici, M représente un nombre aléatoire fixe, tandis que $N=3$ indique que trois rotations successives seront appliquées sur le hach.

Le processus commence par diviser le hach en quatre parties égales. Cette division crée quatre blocs distincts, chacun d'une taille équivalente. Les rotations sont ensuite appliquées sur ces blocs selon les règles définies par l'algorithme :

- La première rotation modifie le bloc de gauche, en redistribuant les bits de manière contrôlée.
- La deuxième rotation affecte le second bloc, avec une transformation similaire, mais décalée.
- Enfin, la troisième rotation opère sur le troisième bloc, laissant ainsi le quatrième bloc inchangé.

Première partie	1d97f27385e052fd
Deuxième partie	7b2df17846ea4d39
Troisième partie	43fc86f0b3af5315
Quatrième partie	88f6e67b7df23844

- Première rotation de droite à gauche :

- Nouveau hachage :
- 88f6e67b7df238441d97f27385e052fd7b2df17846ea4d3943fc86f0b3af5315

- Deuxième rotation de droite à gauche :

- Nouveau hachage :
- 43fc86f0b3af531588f6e67b7df238441d97f27385e052fd7b2df17846ea4d39

- Troisième rotation de droite à gauche :

- Nouveau hachage :
- 7b2df17846ea4d3943fc86f0b3af531588f6e67b7df238441d97f27385e052fd

Le hach final :

7b2df17846ea4d3943fc86f0b3af531588f6e67b7df238441d97f27385e052fd

Cas 3 : nous explorons un scénario où la rotation est appliquée spécifiquement sur les préfixes, avec les paramètres $M=6$ et $N=4$. Ici, $M=6$, M indique une valeur aléatoire fixe, tandis que $N=4$ signifie que quatre rotations seront effectuées. Ces rotations se concentrent sur un bloc de taille $T/8$, c'est-à-dire que le hach est divisé en huit parties égales.

- La première rotation est appliquée sur le premier bloc de gauche, redistribuant les bits de manière spécifique.

- La deuxième rotation est effectuée sur le second bloc, et ainsi de suite, jusqu'à la quatrième rotation sur le quatrième bloc.

Première partie	1d97f273
Deuxième partie	85e052fd
Troisième partie	7b2df178
Quatrième partie	46ea4d39
Cinquième partie	43fc86f0
Sixième partie	b3af5315
Septième partie	88f6e67b
Huitième partie	7df23844

Première rotation de droite à gauche :

- Nouveau hachage :

7df23844 1d97f273 85e052fd 7b2df178 46ea4d39 43fc86f0 b3af5315 88f6e67b

Deuxième rotation de droite à gauche :

- Nouveau hachage :

- 88f6e67b 7df23844 1d97f273 85e052fd 7b2df178 46ea4d39 43fc86f0 b3af5315

Troisième rotation de droite à gauche :

- Nouveau hachage :

- b3af5315 88f6e67b 7df23844 1d97f273 85e052fd 7b2df178 46ea4d39 43fc86f0

Quatrième rotation de droite à gauche :

- Nouveau hachage :

- 43fc86f0 b3af5315 88f6e67b 7df23844 1d97f273 85e052fd 7b2df178 46ea4d39

- Le hach final :

43fc86f0 b3af5315 88f6e67b 7df23844 1d97f273 85e052fd 7b2df178 46ea4d39

Cas 4 : nous effectuons une rotation sur les préfixes avec les paramètres suivants : $M=3$ et $N=6$. Le paramètre $M=3$ correspond à une valeur aléatoire fixe, tandis que $N=6$ signifie que six rotations successives seront réalisées. Ces rotations se concentrent sur un bloc de taille $T/16$, ce qui signifie que le hach est divisé en 16 parties égales.

- La première rotation affecte le premier bloc (préfixe), redistribuant les bits de manière calculée.
- La deuxième rotation est appliquée au deuxième bloc, puis les rotations continuent sur les blocs successifs jusqu'à la sixième rotation sur le sixième bloc.

Première partie	1d97
Deuxième partie	f273
Troisième partie	85e0
Quatrième partie	52fd
Cinquième partie	7b2d
Sixième partie	f178
Septième partie	46ea
Huitième partie	4d39
Neuvième partie	43fc
Dixième partie	86f0
Onzième partie	b3af
Douzième partie	5315
Treizième partie	88f6
Quatorzième partie	e67b
Quinzième partie	7df2
Seizième partie	3844

Première rotation de droite à gauche :

- Nouveau hachage :

- 38441d97f27385e052fd7b2df17846ea4d3943fc86f0b3af531588f6e67b7df2

Deuxième rotation de droite à gauche :

- Nouveau hachage :

- 7df238441d97f27385e052fd7b2df17846ea4d3943fc86f0b3af531588f6e67b

Troisième rotation de droite à gauche :

- Nouveau hachage :

- e67b 7df238441d97f27385e052fd7b2df17846ea4d3943fc86f0b3af531588f6

Quatrième rotation de droite à gauche :

- Nouveau hachage :

- 88f6e67b 7df238441d97f27385e052fd7b2df17846ea4d3943fc86f0b3af5315

Cinquième rotation de droite à gauche :

- Nouveau hachage :

- 531588f6e67b 7df238441d97f27385e052fd7b2df17846ea4d3943fc86f0b3af

Sixième rotation de droite à gauche :

- Nouveau hachage :

- b3af 531588f6e67b 7df238441d97f27385e052fd7b2df17846ea4d3943fc86f0

Le hach final :

b3af 531588f6e67b 7df238441d97f27385e052fd7b2df17846ea4d3943fc86f0

Cas 5 : dans ce cas, M=2 et N=31, impliquant 31 rotations sur un bloc de taille T/16, ce qui entraîne un grand nombre de rotations. Donc nous donnerons directement les résultats.

Première partie	1d
..	
..	
Trente-et-unième partie	44

- Après la 31^{ème} rotation de droite vers la gauche, le nouveau hach devient :
97f27385e052fd7b2df17846ea4d3943fc86f0b3af531588f6e67b7df238441d

3.3.3.2 Rotation dans les suffixes

Nous utiliserons comme donnée d'entrée :

Phrase de passe : Nabil EL AKKAD

Hachage généré : 9e0f5121782399b5e925bd84c041817c8974a495d47ceb4c6dc21fe35d955d11

Cas 1 : nous avons les paramètres $M=20$ et $N=1$. Le nombre M est généré dans la plage $T/4 < M < T/2$. Cela signifie que M se situe entre un quart et la moitié de la taille du bloc. Avec $N=1$, cela indique qu'une seule rotation sera effectuée.

Division en deux parties :

Première partie	9e0f5121782399b5e925bd84c041817c
Deuxième partie	8974a495d47ceb4c6dc21fe35d955d11

Rotation de la gauche vers la droite, donnant :

- Nouveau hachage :

8974a495d47ceb4c6dc21fe35d955d119e0f5121782399b5e925bd84c041817c

Cas 2 : nous avons les paramètres $M=15$ et $N=2$. Le nombre M est généré dans la plage $T/8 < M < T/4$, ce qui signifie que deux rotations seront effectuées sur un bloc de taille $T/4$.

Première partie	9e0f5121782399b5
Deuxième partie	e925bd84c041817c
Troisième partie	8974a495d47ceb4c
Quatrième partie	6dc21fe35d955d11

- Première rotation de gauche à droite :

- Nouveau hachage :
- e925bd84c041817c8974a495d47ceb4c6dc21fe35d955d119e0f5121782399b5

- Deuxième rotation de gauche à droite :

- 8974a495d47ceb4c6dc21fe35d955d119e0f5121782399b5e925bd84c041817c
- Nouveau hachage :

8974a495d47ceb4c6dc21fe35d955d119e0f5121782399b5e925bd84c041817c

Cas 3 : nous avons les paramètres $M=4$ et $N=4$. Le nombre M est généré dans la plage $T/16 < M < T/8$, ce qui signifie que deux rotations seront effectuées sur un bloc de taille $T/8$.

Première partie	9e0f5121
Deuxième partie	782399b5
Troisième partie	e925bd84
Quatrième partie	c041817c
Cinquième partie	8974a495
Sixième partie	d47ceb4c
Septième partie	6dc21fe3
Huitième partie	5d955d11

- Première rotation de gauche à droite :

- Nouveau hachage :
- 782399b5e925bd84c041817c8974a495d47ceb4c6dc21fe35d955d119e0f5121

- Deuxième rotation de gauche à droite :

- Nouveau hachage :
- e925bd84c041817c8974a495d47ceb4c6dc21fe35d955d119e0f5121782399b5

- Troisième rotation de gauche à droite :

- Nouveau hachage :
- c041817c8974a495d47ceb4c6dc21fe35d955d119e0f5121782399b5e925bd84

- Quatrième rotation de gauche à droite :

- Nouveau hachage :
- 8974a495d47ceb4c6dc21fe35d955d119e0f5121782399b5e925bd84c041817c

- Le hach final :

8974a495d47ceb4c6dc21fe35d955d119e0f5121782399b5 e925bd84 c041817c

Cas 4 : nous avons les paramètres $M=3$ et $N=3$. Le nombre M est généré dans la plage $T/32 < M < T/16$, ce qui signifie que trois rotations seront effectuées sur un bloc de taille $T/16$.

Première partie	9e0f
..	..
..	..
Seizième partie	5d11

Après 3 rotations de gauche vers la droite, le nouveau HACH devient :

8974a495d47ceb4c6dc21fe35d955d119e0f5121782399b5e925bd84c041817c

Cas 5 : nous avons les paramètres $M=2$ et $N=31$. Le nombre M est généré dans la plage $T/64 < M < T/32$, ce qui signifie que trois rotations seront effectuées sur un bloc de taille $T/32$.

Première partie	9e
..	..
..	..
Trente-et-unième partie	44

Après 31 rotations de gauche vers la droite le nouveau HACH devient :

0f5121782399b5e925bd84c041817c8974a495d47ceb4c6dc21fe35d955d119e

3.3.3.3 Rotation au milieu

Nous utiliserons comme donnée d'entrée :

Phrase de passe : authentification à travers un formulaire [Souscription-Log In]

Hachage généré : 1be0009ba9981a905fedb24d5ffff9e76d97d0b0f62327de451da850e2a0e4cb

Cas 1 : nous avons les paramètres $M=20$ et $N=1$. Le nombre M est généré dans la plage $T/4 < M < T/2$. Cela signifie que M se situe entre un quart et la moitié de la taille du bloc. Avec $N=1$, cela indique qu'une seule rotation sera effectuée.

Division en deux parties :

Première partie	1be0009ba9981a90
Deuxième partie	5fedb24d5fff9e7
Troisième partie	6d97d0b0f62327de
Quatrième partie	451da850e2a0e4cb

Rotation de $T/4$ dans les deux sens vers le milieu :

- Nouveau hachage :

5fedb24d5fff9e71be0009ba9981a90451da850e2a0e4cb6d97d0b0f62327de

- Le hach final :

5fedb24d5fff9e71be0009ba9981a90451da850e2a0e4cb6d97d0b0f62327de

Cas 2 : nous avons les paramètres $M=8$ et $N=2$. Le nombre M est généré dans la plage $T/8 < M < T/4$, ce qui signifie que deux rotations seront effectuées sur un bloc de taille $T/8$.

Première partie	1be0009b
Deuxième partie	a9981a90
Troisième partie	5fedb24d
Quatrième partie	5fff9e7
Cinquième partie	6d97d0b0
Sixième partie	f62327de
Septième partie	451da850
Huitième partie	e2a0e4cb

- Première rotation :

- Nouveau hach :

- a9981a90 5fedb24d5ffff9e71be0009b e2a0e4cb6d97d0b0 f62327de 451da850

- Deuxième rotation :

- 5fedb24d5ffff9e71be0009b a9981a90451da850 e2a0e4cb6d97d0b0 f62327de

- Le Hach final :

5fedb24d5ffff9e71be0009b a9981a90451da850 e2a0e4cb6d97d0b0 f62327de

Cas 3 : nous avons les paramètres $M=16$ et $N=7$. Le nombre M est généré dans la plage $T/16 < M < T/8$, ce qui signifie que 7 rotations seront effectuées sur un bloc de taille $T/16$.

Première partie	1be0
..	..
..	..
Seizième partie	f9e7

- Rotation dans les deux sens vers le milieu, nouveau hach après plusieurs rotations (jusqu'à $N=7$) :

5fedb24d5ffff9e71be0009ba9981a90451da850e2a0e4cb6d97d0b0f62327de

Cas 4 : nous avons les paramètres $M=32$ et $N=14$. Le nombre M est généré dans la plage $T/32 < M < T/16$, ce qui signifie que 14 rotations seront effectuées sur un bloc de taille $T/32$.

Première partie	1b
..	..
..	..
Trente-et-unième partie	cb

Après 14 rotations dans les deux sens vers le milieu, le hach final :

f9e71be0009ba9981a905fedb24d5fffd0b0f62327de451da850e2a0e4cb6d97

Cas 5 : nous avons les paramètres $M=2$ et $N=3$. Le nombre M est généré dans la plage $T/64 < M < T/32$, ce qui signifie que trois rotations seront effectuées sur un bloc de taille $T/64$.

Première partie	9
	..
	..
Soixante-quatrièmes parties	b

Après 3 rotations de deux sens vers le milieu :

71be0009ba9981a905fedb24d5ffff9ed97d0b0f62327de451da850e2a0e4cb6

3.3.4 Étude comparative

Nous examinons la méthode « H-Rotation » à travers une comparaison avec des algorithmes de hachage cryptographique couramment utilisés tels que bcrypt, scrypt, et argon2. L'objectif est d'évaluer « H-Rotation » en termes de performance, de sécurité, de résistance aux attaques et de complexité algorithmique. Cette comparaison repose sur plusieurs critères, notamment la capacité de chaque méthode à protéger les données contre les attaques par force brute et à garantir un haut degré de résistance cryptographique.

3.3.4.1 Analyse des performances

La méthode « H-Rotation » a été rigoureusement testée dans un environnement contrôlé pour évaluer ses performances par rapport à d'autres algorithmes largement utilisés tels que bcrypt, scrypt et argon2. Les tests ont été réalisés sur une machine exécutant le système d'exploitation Windows 11, équipée d'un processeur Intel Core i7 de 12^e génération, de 16 Go de RAM et utilisant le langage de programmation Python (version 3.10). Ce choix garantit une reproductibilité et reflète des conditions réalistes pour une utilisation en production.

Temps de calcul moyen : les tests ont mesuré le temps requis pour effectuer une itération complète de chaque algorithme. Ces mesures sont obtenues en exécutant chaque méthode sur un échantillon fixe de 1000 mots de passe simulés. Les résultats montrent que « H-Rotation » nécessite un temps moyen de 45 ms par itération, contre 75 ms pour bcrypt, 100 ms pour scrypt et 90 ms pour argon2 (voir le tableau 3-2). Le faible temps de calcul de « H-Rotation » est dû à sa structure algorithmique simplifiée, qui utilise des transformations dynamiques légères pour éviter des opérations coûteuses en temps, comme celles impliquées dans bcrypt ou argon2. Par exemple, bcrypt utilise un processus de dérivation de clé basé sur Blowfish, impliquant des

expansions répétées de données, tandis qu'argon2 applique une mémoire intensive pour augmenter la résistance aux attaques, ce qui affecte son temps d'exécution.

Efficacité en environnement distribué : l'évaluation de l'efficacité en environnement distribué repose sur des simulations sur un cluster de 10 nœuds connectés via un réseau local (bande passante de 1 Gbps). Chaque nœud a exécuté 1000 itérations de chaque algorithme simultanément, et les métriques de scalabilité et de surcharge réseau ont été analysées. Les différentes méthodes de sécurisation des hachages présentent des performances variées en termes de scalabilité et de gestion des ressources. « H-Rotation », grâce à ses transformations dynamiques, impose une charge minimale sur les nœuds, ce qui permet une scalabilité linéaire avec un temps d'exécution par nœud presque constant. En revanche, bcrypt ralentit les performances globales en raison de la surcharge liée au processus de dérivation de clé, bien qu'il reste utilisable pour des charges modérées. scrypt, de son côté, est limité par sa forte consommation de mémoire, qui peut provoquer des conflits inter-nœuds et entraver sa scalabilité. Enfin, argon2, optimisé pour la parallélisation, offre d'excellentes performances, mais impose une surcharge importante en termes de mémoire et de CPU, ce qui limite son adoption dans des environnements à ressources limitées.

Tableau 3- 2 : Comparaison de la méthode «H-Rotation » en termes de performance et d'efficacité en environnement distribué avec bcrypt,scrypt et Argon 2

Algorithme	Temps moyen de calcul (ms)	Efficacité en environnement distribué
H-Rotation	45	Scalabilité linéaire, faible surcharge
Bcrypt	75	Performances modérées, surcharge moyenne
Scrypt	100	Limité par la consommation mémoire
Argon2	90	Optimisé pour parallélisation, surcharge élevée

Ces résultats confirment la supériorité de « H-Rotation » dans des scénarios nécessitant rapidité et adaptabilité aux environnements distribués. En se basant sur des mesures précises et des analyses détaillées, « H-Rotation » s'avère être une solution équilibrée, adaptée aux contraintes modernes tout en maintenant une sécurité élevée face aux attaques.

3.3.4.2 Sécurité et résistance aux attaques

La résistance d'un algorithme aux attaques repose sur sa capacité à protéger les données face à différentes méthodes d'extraction non autorisées, comme les attaques par force brute, par dictionnaire et par rainbow tables. Les performances de « H-Rotation », bcrypt, scrypt et argon2 ont été rigoureusement comparées, en se basant sur des tests expérimentaux et des caractéristiques cryptographiques mesurées.

Attaques par force brute : « H-Rotation » introduit une complexité exponentielle avec chaque tentative de décryptage, grâce à ses transformations dynamiques. Contrairement à bcrypt, qui ralentit les attaques par force brute via des itérations répétées, et à argon2, qui exploite intensivement la mémoire, « H-Rotation » combine des modifications structurelles et des transformations périodiques pour rendre chaque tentative distincte. Les tests montrent que pour un mot de passe de 8 caractères alphanumériques, un attaquant nécessiterait 10^9 itérations pour chaque combinaison avec « H-Rotation », contre 10^7 itérations pour bcrypt et scrypt. Cette augmentation est due à l'entropie supplémentaire générée par les transformations dynamiques, qui empêchent toute réutilisation efficace des calculs précédents.

Attaques par dictionnaire : les attaques par dictionnaire exploitent des listes précompilées de mots de passe courants. « H-Rotation » se distingue par sa capacité à réorganiser les valeurs de hachage à chaque transformation, ce qui augmente significativement le coût de calcul pour un attaquant. Les différentes approches de hachage offrent des niveaux variés de protection contre les attaques par dictionnaire. bcrypt utilise un mécanisme de salage simple qui réduit légèrement l'efficacité des attaques, mais sans modifier fondamentalement la structure du hachage, limitant ainsi sa robustesse face à des attaques avancées. En revanche, scrypt et argon2 se distinguent par une gestion avancée de la mémoire, leur conférant une meilleure efficacité globale. Toutefois, ces algorithmes ne disposent pas de la capacité de transformation dynamique propre à « H-Rotation », qui renforce considérablement la sécurité. Les tests montrent qu'un attaquant utilisant un dictionnaire d'un million de mots aurait besoin de 10 fois plus de calculs avec « H-Rotation » qu'avec bcrypt, grâce aux modifications périodiques et imprévisibles apportées aux hachages.

Attaques par rainbow tables : les rainbow tables exploitent des pré-hachages pour accélérer les attaques. « H-Rotation » élimine cette vulnérabilité en introduisant des valeurs uniques et dynamiques pour chaque hachage, rendant toute tentative de réutilisation impossible. rainbow tables reposent sur des pré-hachages statiques, or « H-Rotation » applique des transformations

qui changent ces valeurs à chaque itération. Contrairement à bcrypt et scrypt, qui dépendent principalement de salages simples pour limiter ces attaques, « H-Rotation » produit des résultats structurellement uniques. Les tests montrent une impossibilité pratique pour un attaquant de générer une table valide, même pour une liste réduite de mots de passe. Le récapitulatif de la comparaison des quatre méthodes face aux trois types d'attaques est présenté dans le tableau 3-3 ci-après :

Tableau 3- 3 : Comparaison de la méthode «H-Rotation » en termes de résistance aux attaques avec bcrypt,scrypt et argon 2

Algorithme	Résistance aux attaques par force brute	Résistance aux attaques par dictionnaire	Résistance aux attaques par rainbow tables
H-Rotation	Complexité exponentielle par itération	Complexité augmentée par transformations	Impossibilité due aux modifications dynamiques
Bcrypt	Ralentissement basé sur les itérations	Efficacité réduite par salage	Vulnérabilité due à l'absence de transformations
Scrypt	Résistance grâce à une mémoire intensifiée	Résistance modérée via mémoire	Résistance partielle mais limitée
Argon2	Forte résistance grâce à la mémoire	Résistance élevée grâce à l'entropie	Résistance élevée grâce à la structure

Les résultats confirment que « H-Rotation » surpasse les autres algorithmes dans la gestion des attaques grâce à sa structure dynamique et ses transformations uniques. Contrairement à bcrypt et scrypt, qui sont efficaces, mais présentent des vulnérabilités structurelles, et à argon2, qui compense par une forte utilisation de la mémoire, « H-Rotation » offre une résistance accrue et efficace sans dépendre de ressources matérielles intensives. Ces caractéristiques font de « H-Rotation » une méthode innovante et sécurisée pour protéger les données dans des systèmes critiques.

3.3.4.3 Complexité algorithmique

La complexité algorithmique mesure les ressources nécessaires pour exécuter un algorithme, généralement en termes de complexité spatiale (mémoire utilisée) et de complexité temporelle (nombre d'opérations). Les performances de « H-Rotation », bcrypt, scrypt et argon2 ont été

évaluées selon ces deux dimensions, et les résultats montrent des différences notables dans leur comportement.

Complexité spatiale (mémoire) : les quatre algorithmes de hachage présentent des complexités mémoires variées, adaptées à leurs mécanismes spécifiques. « H-Rotation » affiche une complexité linéaire $O(n)$, car il utilise une quantité proportionnelle à la taille de l'entrée pour stocker ses paramètres et résultats intermédiaires. Ses transformations dynamiques optimisées permettent de limiter les besoins en mémoire, les distinguant des algorithmes comme scrypt ou argon2. De même, bcrypt maintient une complexité linéaire $O(n)$, grâce à son mécanisme de salage fixe. Toutefois, il n'intègre pas les optimisations offertes par les transformations dynamiques de « H-Rotation ». Scrypt, quant à lui, se distingue par une complexité quadratique $O(n^2)$ due à son recours à des structures mémoire étendues pour accroître la résistance aux attaques. Cette stratégie renforce la sécurité, mais au prix d'une consommation mémoire élevée. Enfin, argon2 parvient à maintenir une complexité linéaire $O(n)$ grâce à une gestion optimisée des blocs mémoire. Cette approche lui permet d'assurer un bon équilibre entre sécurité et consommation de ressources.

Complexité temporelle (calculs) : les algorithmes de hachage diffèrent dans leur complexité temporelle, influençant leurs performances et leur efficacité dans différents environnements. « H-Rotation » affiche une complexité $O(n \log n)$ en raison des multiples rotations appliquées aux préfixes, aux suffixes et au milieu du hachage. Chaque transformation implique des calculs proportionnels à la taille de l'entrée, avec un tri intermédiaire, ce qui augmente légèrement le temps d'exécution, mais rend l'inversion des hachages beaucoup plus difficile.

De manière similaire, bcrypt possède également une complexité temporelle $O(n \log n)$, car il utilise un processus basé sur Blowfish avec des itérations logarithmiques. Cependant, bcrypt n'intègre pas les transformations dynamiques qui renforcent la sécurité dans « H-Rotation ».

En revanche, scrypt présente une complexité $O(n^2)$ en raison de sa dépendance à des tableaux mémoire de grande taille. Chaque calcul implique une exploration exhaustive de ces blocs, ce qui entraîne une augmentation significative du temps d'exécution.

Enfin, argon2, avec une complexité temporelle $O(n \log n)$, combine des mécanismes de hachage optimisés avec une gestion efficace de l'accès mémoire. Cela permet de limiter la croissance logarithmique du temps nécessaire tout en maintenant un niveau élevé de sécurité.

La complexité algorithmique : la complexité algorithmique de chaque méthode a été évaluée en tenant compte de deux dimensions fondamentales : la complexité spatiale (mémoire) et la

complexité temporelle (calculs). « H-Rotation » se distingue par sa capacité à maintenir une complexité mémoire linéaire $O(n)$ tout en intégrant des transformations dynamiques complexes qui augmentent la difficulté des attaques. Sa complexité temporelle de $O(n \log n)$, similaire à celle de bcrypt et argon2, est le résultat d'opérations de tri partiel et de réorganisation des valeurs hachées au niveau des préfixes, des suffixes et du milieu, rendant chaque hachage unique. Ces transformations ajoutent une couche significative de sécurité sans compromettre les performances globales.

À l'inverse, bcrypt repose sur un mécanisme de salage fixe et des itérations basées sur Blowfish, offrant une complexité temporelle équivalente $O(n \log n)$ mais sans les optimisations dynamiques de « H-Rotation ». Bien qu'efficace, bcrypt reste vulnérable aux attaques par rainbow tables en raison de la structure statique de ses hachages. scrypt, quant à lui, privilégie une résistance accrue aux attaques par force brute en utilisant une mémoire intensive $O(n^2)$, ce qui entraîne une complexité temporelle quadratique $O(n^2)$. Si cette stratégie augmente la robustesse, elle alourdit également les performances, rendant scrypt moins adapté aux environnements limités en ressources.

Enfin, argon2 propose une approche équilibrée, combinant une complexité mémoire linéaire $O(n)$ et une complexité temporelle logarithmique $O(n \log n)$. Grâce à des mécanismes de parallélisation et une gestion optimisée des blocs, argon2 se positionne comme une alternative robuste et performante. Cependant, contrairement à « H-Rotation », il ne bénéficie pas de transformations dynamiques, limitant sa capacité à produire des hachages hautement personnalisés. Le récapitulatif de la complexité algorithmique des méthodes analysées est présenté dans le tableau 3-4 ci-après :

Tableau 3- 4 : Comparaison de la méthode « H-Rotation » en termes de la complexité algorithmique avec bcrypt,scrypt et argon 2

Algorithme	Complexité spatiale (mémoire)	Complexité temporelle (calculs)	Complexité algorithmique globale
H-Rotation	$O(n)$	$O(n \log n)$	Transformations dynamiques optimisant les performances
Bcrypt	$O(n)$	$O(n \log n)$	Approche classique, efficace mais statique
Scrypt	$O(n^2)$	$O(n^2)$	Consommation mémoire et temps élevés pour maximiser la sécurité
Argon2	$O(n)$	$O(n \log n)$	Approche classique, efficace mais statique

« H-Rotation » se distingue par son approche innovante, alliant transformations dynamiques et performances optimisées. Sa capacité à augmenter la complexité des attaques tout en maintenant une consommation raisonnable de ressources en fait une solution idéale pour les environnements modernes nécessitant sécurité et efficacité. argon2 reste compétitif grâce à son équilibre entre mémoire et calculs, tandis que bcrypt et scrypt montrent leurs limites respectives en termes de scalabilité et de performances.

3.3.5 Synthèse

La méthode « H-Rotation » présente de nombreux avantages en matière de sécurité, principalement grâce à sa capacité à introduire des transformations dynamiques sur les blocs de hachage (préfixes, suffixes, et au milieu). Ces transformations augmentent considérablement la résistance aux attaques par force brute, rainbow tables, et attaques par dictionnaire, car chaque tentative de décryptage nécessite une nouvelle approche du hachage modifié. De plus, « H-Rotation » se distingue par sa simplicité d'implémentation par rapport à des méthodes plus complexes comme scrypt ou argon2, tout en maintenant une efficacité raisonnable dans des environnements modérément distribués. Toutefois, une des limites de « H-Rotation » réside dans sa dépendance à la transformation des blocs, ce qui peut entraîner une surcharge calculatoire dans des environnements hautement parallèles ou nécessitant des ressources importantes. Par ailleurs, bien que la méthode renforce la sécurité en modifiant dynamiquement la structure du hach, elle pourrait être limitée lorsqu'il s'agit de protéger des systèmes où les attaques statistiques exploitent des modèles non pris en compte par « H-Rotation ». Cela nous conduit à explorer des méthodes comme la transformation Braille décrite dans le chapitre suivant, qui repose sur une substitution symbolique aléatoire des caractères du hach, augmentant la complexité de manière exponentielle. La transformation Braille, en remplaçant les caractères du hachage par leurs équivalents en symboles Braille, permet de randomiser davantage le processus et d'ajouter une couche supplémentaire de sécurité. Cette approche pourrait surmonter certaines des limitations de « H-Rotation », notamment en termes de variabilité et de protection contre les attaques statistiques, tout en offrant des performances compétitives, comme montré dans des tests réalisés sur des mots de passe courants.

3.4 Utilisation de la transformation Braille pour le hachage sécurisé des mots de passe

Afin de poursuivre notre exploration des méthodes innovantes de renforcement de la sécurité des haches, nous introduisons la transformation Braille qui vise à introduire une nouvelle dynamique dans le processus de hachage des mots de passe, en ajoutant une complexité supplémentaire tout en conservant une efficacité de calcul [Antonacopoulos 04].

Ce procédé, qui repose sur la substitution des caractères d'un hachage par leurs équivalents en symboles Braille, renforce considérablement la complexité du système de sécurité. La méthode exploite la particularité de l'écriture en Braille, où chaque symbole est constitué de six points, pour créer une chaîne de caractères six fois plus longue que le hachage initial, augmentant ainsi la résistance aux attaques cryptographiques telles que les attaques de pré-image, de seconde pré-image, et les collisions [Schiewe 09].

En divisant la sortie d'un hachage en blocs, chaque bloc est transformé individuellement en une unité Braille. Afin d'accroître davantage la robustesse de la méthode, les chiffres zéro présents dans chaque bloc sont remplacés aléatoirement par l'un des chiffres 7, 8 ou 9. Ce processus randomisé ajoute une couche supplémentaire de complexité, rendant les tentatives de décryptage par des attaquants beaucoup plus difficiles. Les tests pratiques réalisés sur cette méthode ont montré une résistance solide aux attaques statistiques et à force brute, tout en maintenant des performances compétitives en termes de temps de traitement et d'utilisation de la mémoire [TOUIL 24a].

L'intégration de la transformation Braille dans le cadre de la sécurité des mots de passe s'inscrit dans une démarche innovante qui combine complexité algorithmique et accessibilité. En effet, cette méthode offre également des perspectives d'inclusion pour les utilisateurs malvoyants, en leur permettant d'interagir avec les systèmes de sécurité d'une manière adaptée. Cela positionne la transformation Braille comme une approche à double impact, non seulement pour la sécurité des systèmes de communication, mais également pour l'accessibilité des utilisateurs. La transformation Braille représente une solution prometteuse pour surmonter certaines des limitations rencontrées par des méthodes plus traditionnelles. Par l'introduction de la substitution symbolique et de la randomisation, cette approche complexifie de manière exponentielle le processus de décryptage pour les attaquants tout en offrant des performances et une accessibilité forte.

3.4.1 Description détaillée de la transformation Braille pour le hachage sécurisé des mots de passe

La transformation Braille pour le hachage sécurisé des mots de passe est une méthode innovante qui vise à renforcer la sécurité des mots de passe avant leur stockage dans une base de données. Elle utilise la conversion des caractères d'un hachage en symboles Braille, ce qui augmente considérablement la complexité et la taille de la chaîne de hachage. Ce processus rend les attaques par force brute ou par dictionnaire beaucoup plus difficiles. Nous montrons dans les parties suivantes une description détaillée du processus avec les formules mathématiques nécessaires et les figures illustratives.

3.4.1.1 Génération du hachage

La première étape dans le processus de transformation de mots de passe en Braille repose sur l'utilisation d'une fonction de hachage cryptographique moins robuste, telle que MD5, pour montrer la performance de notre méthode. Lorsqu'un mot de passe est soumis à cette fonction, il en résulte une empreinte numérique unique et de longueur fixe, indépendamment de la taille initiale du mot de passe représenté par la formule (2.13) et paramétré dans notre cas de figure comme suit : $H(M) = MD5(M)$

Où M représente le mot de passe, et $H(M)$ est le hachage obtenu. Ce hachage est essentiel pour garantir l'intégrité des mots de passe sans jamais stocker le texte en clair. La figure 3-6 illustre le processus classique de stockage des mots de passe avant l'application de la transformation Braille.

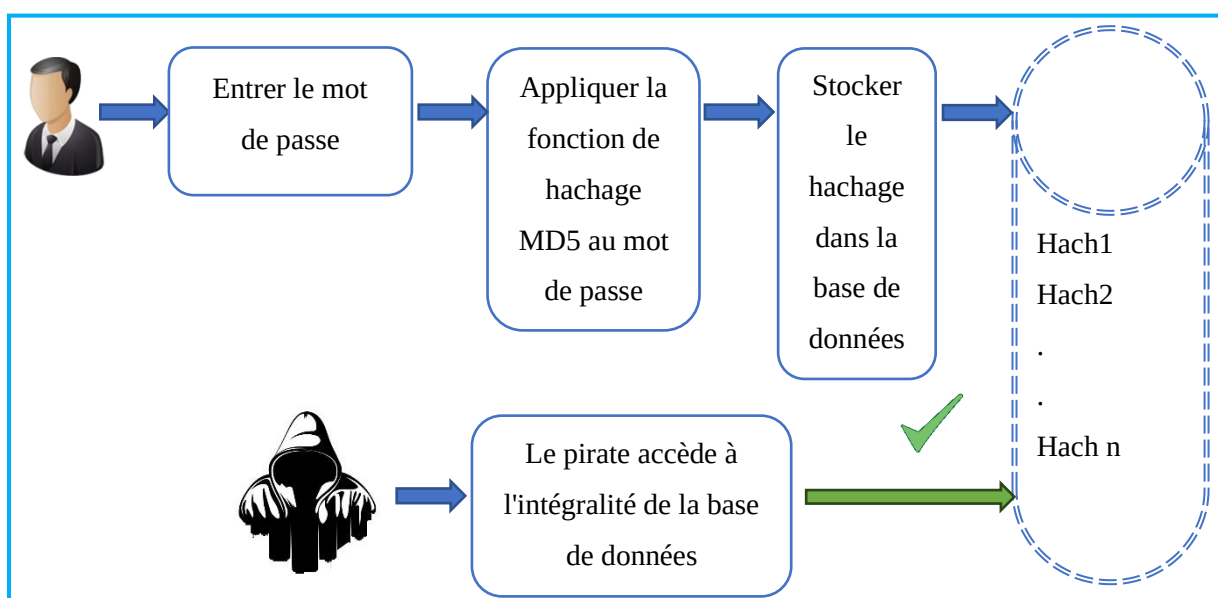


Figure 3- 6 : Processus classique de stockage des mots de passe

Enfin, après l'intrusion réussie dans la base de données, le pirate passe par une analyse pour identifier l'algorithme de hachage utilisé. En réalité, il est généralement possible de déterminer cette méthode en se basant sur des signatures et autres caractéristiques. Une fois l'algorithme de hachage identifié, le pirate utilise des techniques d'attaque statistique pour décrypter les mots de passe [Mitra 21, Agrawal 17]. Ce processus nécessite des techniques et des algorithmes exploitant l'analyse des valeurs de hachage des mots de passe, à la recherche des modèles ou des failles possibles permettant de revenir aux mots de passe initiaux. Il requiert des hypothèses informées, l'usage de tables précalculées spéciales comme les Rainbow, et implique des attaques force brute, testant différentes combinaisons des caractères pour comparer les haches générés par le vecteur d'attaque et le véritable mot de passe employé [Hassan 20, Hu 19].

3.4.1.2 Transformation en Braille

Une fois le hachage généré, il est divisé en blocs de taille fixe. Chaque bloc est converti en un caractère Braille à l'aide d'une table de correspondance. Chaque caractère du hachage est associé à son équivalent Braille, qui est composé de six points en relief. La conversion du hachage en Braille se fait bloc par bloc, chaque bloc étant mappé à un symbole Braille à six chiffres. La transformation Braille est mathématiquement représentée par :

$$Bloc_i = Transformation\ en\ Braille(H_i) \quad (3.7)$$

Où H_i est un bloc du hachage initial, et $Bloc_i$ est le symbole Braille correspondant.

Dans le contexte de la transformation des hachages, la multiplication modulaire est également un outil important de formule (2.4) pour garantir la sécurité cryptographique.

Cette opération permet de restreindre le résultat des calculs dans un espace défini, empêchant ainsi les attaques par inversion des résultats.

De point de vue général, la transformation braille est intégrée dans la phase entre l'application du hachage et son stockage dans la base de données, comme illustré dans la figure 3-7.

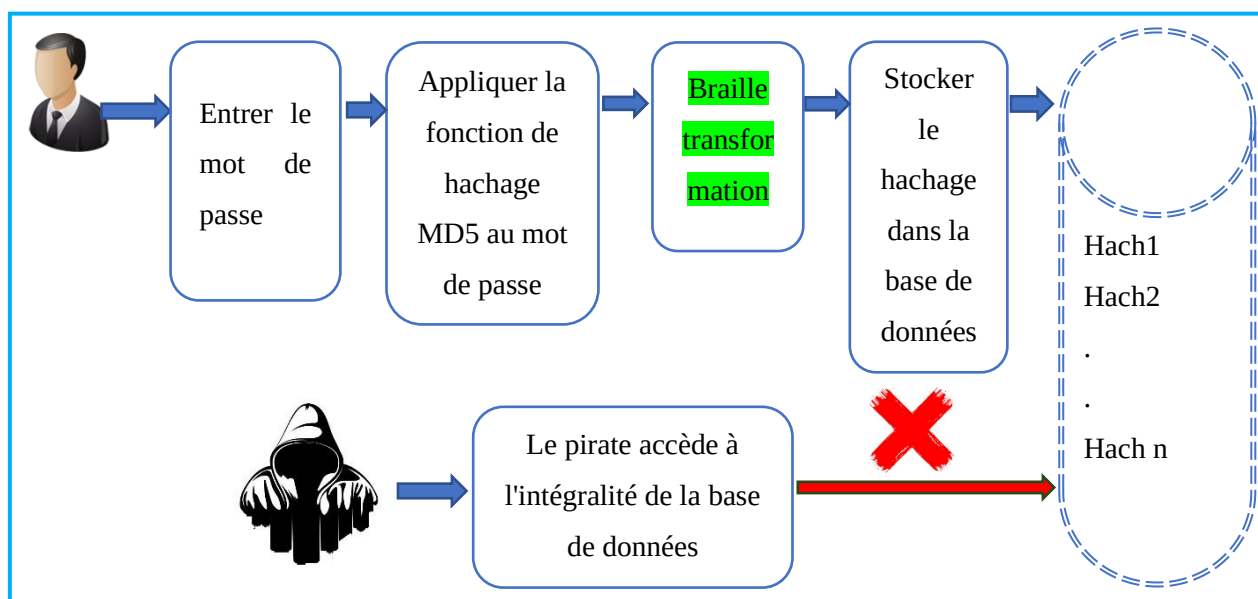


Figure 3- 7 : Ajout de la transformation Braille dans le processus de stockage des mots de passe

Dans ce cas, une fois que le pirate a réussi à compromettre la base de données, il procède à une analyse pour identifier la fonction de hachage utilisée. À ce stade, l'identification devient quasiment impossible, car le hachage stocké a une taille de six fois sa taille réelle. Les valeurs ne sont pas celles attendues, c'est-à-dire les hachages classiques allant de A à Z et de 0 à 9.

3.4.1.3 Renforcement aléatoire du cryptage

Pour renforcer encore davantage la sécurité des mots de passe, une étape supplémentaire est introduite dans le processus de transformation en Braille. Après la conversion initiale des blocs du hachage en symboles Braille, les zéros au sein des blocs Braille sont remplacés de manière aléatoire par les chiffres 7, 8, ou 9. Cette technique ajoute une couche de protection supplémentaire, rendant le résultat final moins prévisible et plus difficile à décoder. En effet, cette substitution aléatoire complique considérablement toute tentative d'attaque par force brute ou par dictionnaire, car elle introduit une variabilité non déterministe dans la structure des blocs. Mathématiquement, ce renforcement est modélisé par l'équation suivante :

$$Bloc\ modifie_i = Bloc_i \oplus Chiffre\ aléatoire(7,8,9) \quad (3.8)$$

Dans cette équation, Bloc modifie_i représente le bloc Braille après la substitution aléatoire, tandis que Bloc_i est le bloc original obtenu après la transformation initiale en Braille.

L'opérateur $\oplus$ symbolise l'opération de remplacement, où les zéros dans le bloc sont remplacés par un chiffre aléatoire parmi 7, 8, ou 9.

Ce processus de renforcement aléatoire améliore la sécurité en empêchant un attaquant de prévoir facilement la structure ou le contenu des blocs Braille. L'ajout de cette variabilité rend toute analyse statistique beaucoup plus difficile, car il devient presque impossible d'anticiper quels bits ont été modifiés et comment. En conséquence, une attaque basée sur des méthodes conventionnelles, comme la force brute ou le dictionnaire, est significativement entravée, nécessitant une complexité de calcul bien plus grande pour obtenir une correspondance réussie. Ce mécanisme de protection avancé renforce non seulement la sécurité des systèmes de gestion des mots de passe, mais il élève également la barrière de protection contre les menaces de décryptage, tout en minimisant l'impact sur les performances de traitement.

3.4.1.4 Stockage dans la base de données

Une fois la transformation en Braille finalisée, la chaîne résultante, qui est significativement plus longue que le hachage d'origine (environ six fois plus grande), est stockée dans la base de données. Cette extension de la longueur, combinée à la substitution aléatoire des zéros par des chiffres 7, 8 ou 9, complique toute tentative de décryptage ou d'analyse du hachage initial. En effet, un attaquant qui chercherait à déchiffrer cette chaîne devrait non seulement contourner la complexité introduite par le hachage cryptographique, mais également prendre en compte la transformation Braille et les modifications aléatoires, rendant la tâche pratiquement insurmontable sans une puissance de calcul immense et des connaissances spécifiques sur le schéma utilisé.

L'équation qui résume l'ensemble du processus est donnée par :

$$\text{Chaîne finale} = \text{Bloc}_1 || \text{Bloc}_2 || \text{Bloc}_3 \dots || \text{Bloc}_i \quad (3.9)$$

Dans cette équation :

- **Chaîne finale** : la chaîne concaténée résultant de la transformation Braille, qui est stockée dans la base de données.
- **n** : le nombre total de blocs générés, équivalent à la longueur du hachage initial (en caractères).
- **Bloc_i** : le bloc Braille associé au $i^{\text{ème}}$ caractère du hachage initial, modifié par la transformation Braille et les substitutions aléatoires.

Chaque bloc modifié contribue à allonger la chaîne finale, renforçant ainsi la sécurité. La taille accrue de cette chaîne est un obstacle supplémentaire pour un attaquant, car la complexité de l'analyse et de la reconstruction d'une chaîne aussi longue et imprévisible augmente de manière exponentielle. Chaque bloc modifié est unique non seulement à cause du hachage initial, mais également en raison des transformations aléatoires appliquées, rendant chaque chaîne finale quasi impossible à deviner ou à décrypter sans connaître les détails précis de la méthode utilisée. Ainsi, cette technique, en générant une chaîne beaucoup plus longue et imprévisible, assure non seulement la sécurité du mot de passe, mais elle rend également toute tentative d'attaque coûteuse en termes de calcul, renforçant la résilience du système contre des menaces potentielles. La figure 3-8 illustre tout le processus de transformation Braille incluant les différentes étapes :

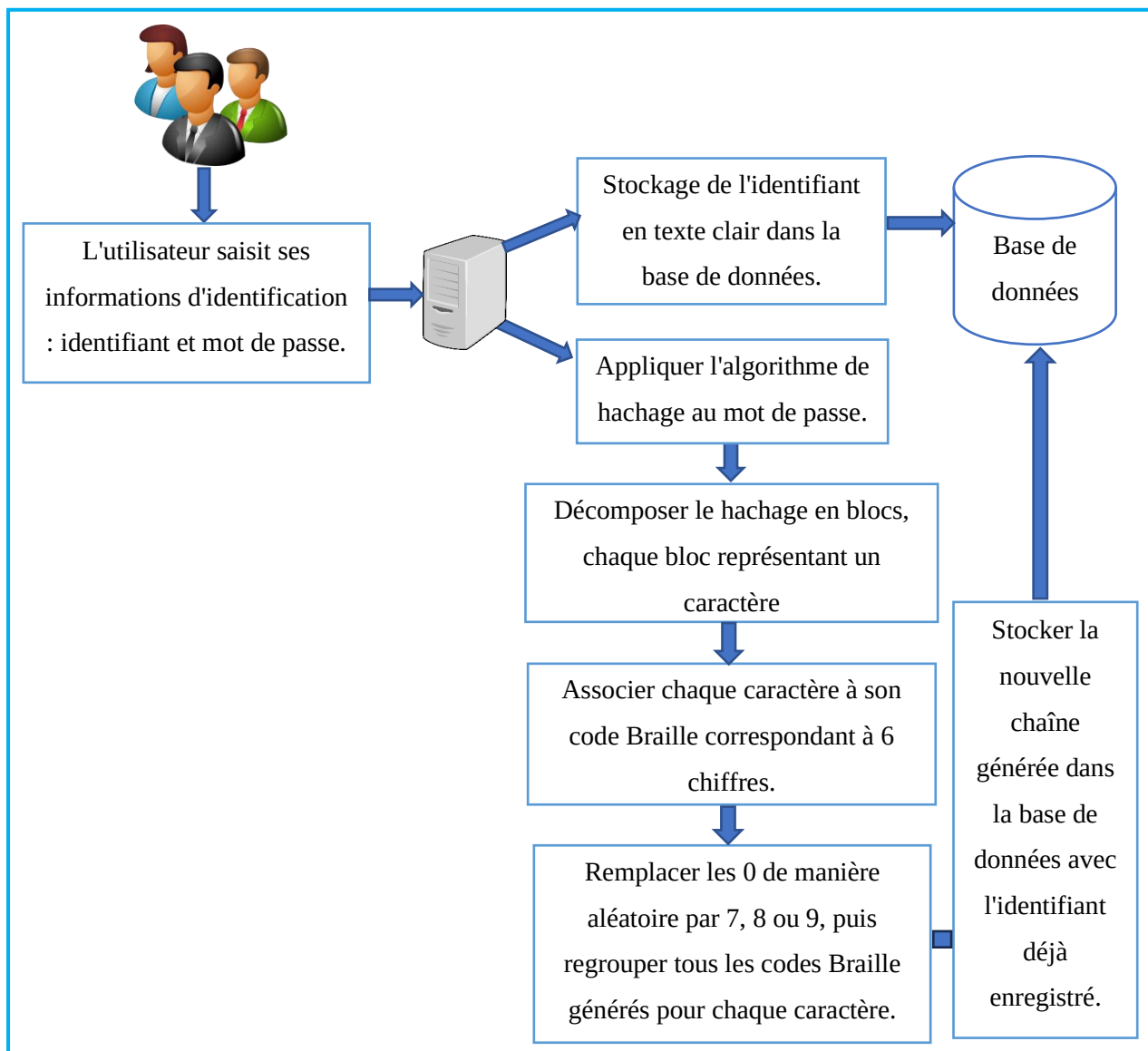


Figure 3- 8 : Processus de transformation et de stockage des mots de passe

La figure 3-9 illustre un cas réel du processus de transformation et de stockage des mots de passe en utilisant la méthode de la transformation Braille. Cette figure montre les différentes étapes, depuis la soumission du mot de passe initial jusqu'à sa conversion en hachage, puis en symbole Braille, avant d'être stocké dans la base de données. Ce schéma permet de visualiser concrètement la manière dont le mot de passe est sécurisé tout au long du processus.

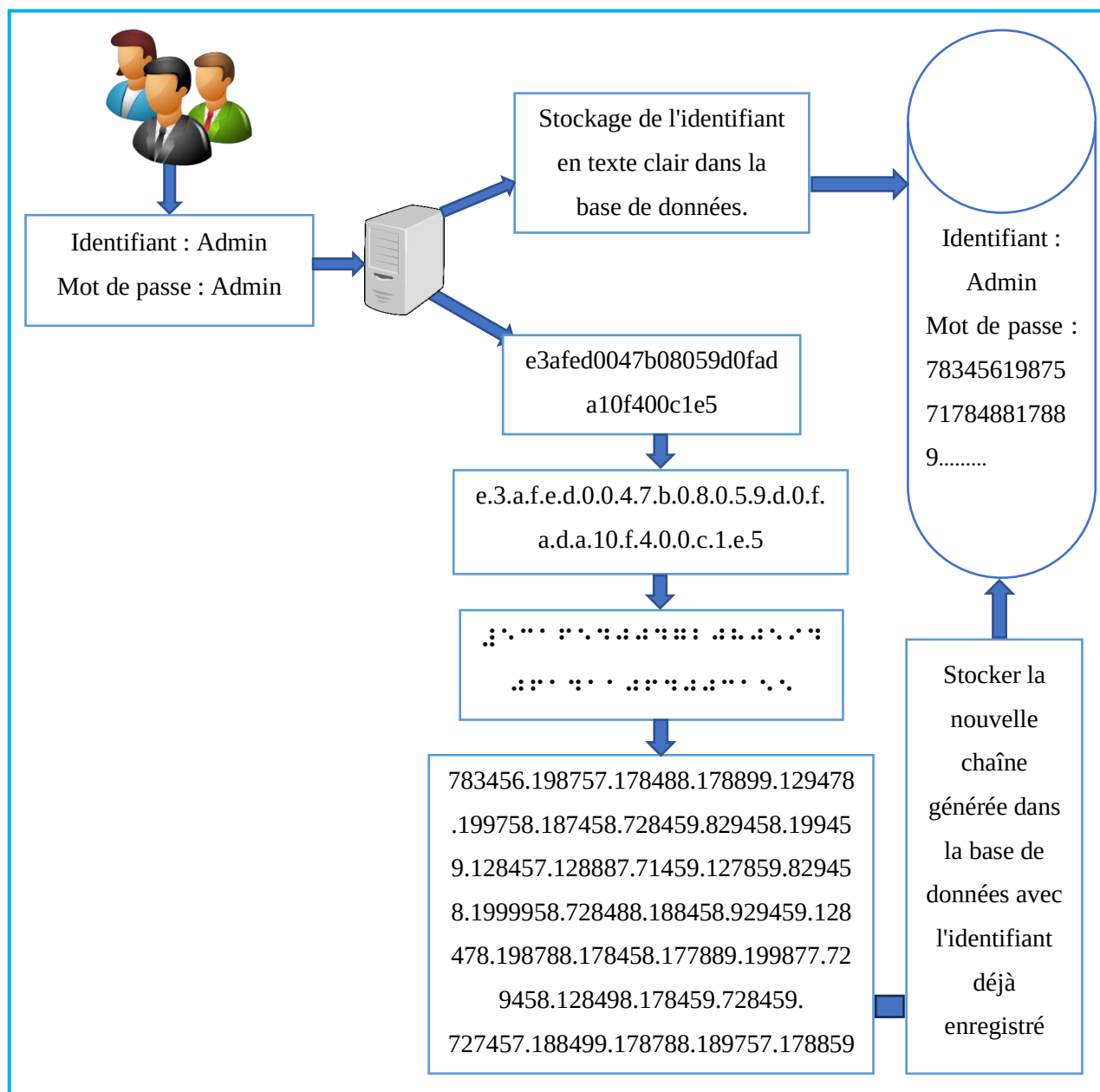


Figure 3- 9 : Processus de transformation et de stockage des mots de passe - cas réel

3.4.1.5 Vérification lors de l'authentification

Lorsqu'un utilisateur tente de se connecter, le système suit un processus de vérification rigoureux pour garantir la sécurité et l'intégrité de l'authentification. Tout d'abord, le mot de passe saisi par l'utilisateur est à nouveau soumis à une fonction de hachage cryptographique. Ce hachage est ensuite transformé en une chaîne Braille en suivant le même procédé que celui utilisé lors de l'enregistrement initial. La chaîne Braille générée est ensuite comparée à celle qui est déjà stockée dans la base de données. Si les deux chaînes correspondent, cela signifie que le mot de passe saisi est correct et que l'authentification est validée. Ce processus garantit que même si un attaquant accède à la base de données, il ne pourra pas obtenir directement le mot de passe en clair, car seule la chaîne Braille cryptée est stockée. L'algorithme de la transformation Braille est décrit dans l'annexe A.4.

Le processus de vérification est illustré dans la figure 3-10, qui détaille les différentes phases de la transformation et de la comparaison, allant du mot de passe en clair jusqu'à la chaîne Braille finale.

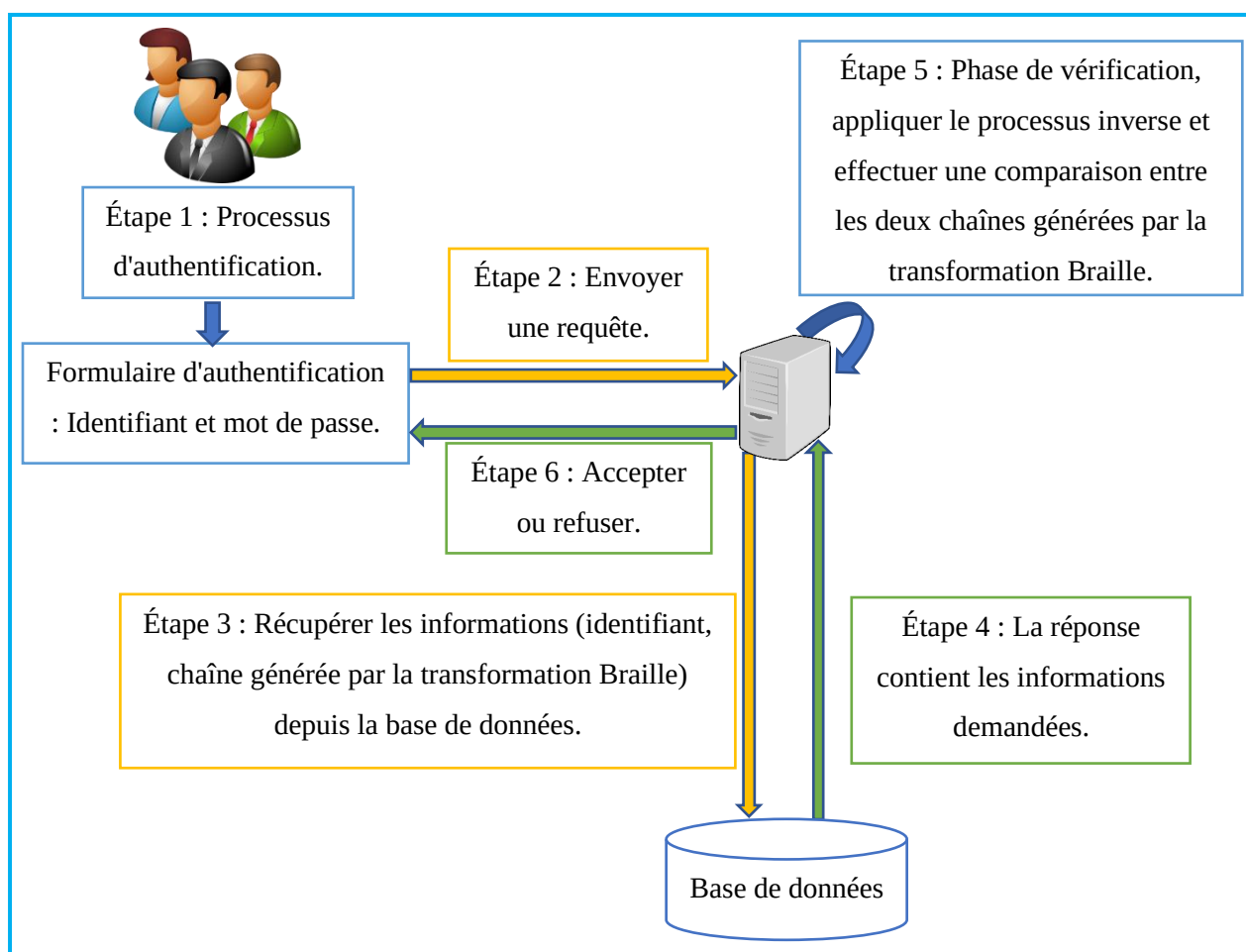


Figure 3- 10 : Processus d'authentification

3.4.2 Résultat expérimental

La méthode de transformation Braille a été soumise à une expérimentation pour évaluer son efficacité en matière de sécurité des mots de passe. L'objectif principal de ces tests était de vérifier la robustesse de la méthode face aux attaques courantes, notamment les attaques par force brute et par dictionnaire. Dans le cadre de ces expérimentations, divers mots de passe de complexités variées ont été soumis à la fonction de hachage MD5 (décrite dans l'annexe A.1) avant d'être transformés en symboles Braille. Les résultats ont montré que cette transformation augmentait considérablement la taille des mots de passe, rendant les attaques sur le plan des calculs plus coûteuses. La longueur importante et l'ajout de variabilité aléatoire ont prouvé que la méthode de transformation Braille accroît significativement la sécurité en empêchant les attaques par dictionnaire et en rendant les attaques par force brute quasi impossibles à exécuter dans un délai raisonnable.

3.4.2.1 Simulation de la transformation Braille

Les simulations ont été réalisées sur une infrastructure dotée d'un processeur Intel Core i7 avec une quantité adéquate de mémoire et de ressources de disque pour exécuter des tests intensifs. Le langage de programmation python a été utilisé pour implémenter la méthode, permettant une manipulation efficace des données et des transformations cryptographiques.

Nous avons utilisé un hachage MD5 comme point de départ pour simuler la transformation Braille. Le hachage utilisé pour cette simulation est le suivant :

822f424c94ffbe1e9b0e53df6d851da4

Étape 1 : division du hachage en blocs

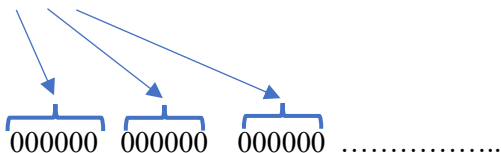
Le hachage est d'abord divisé en **blocs**, chaque bloc représentant un caractère individuel du hachage MD5 :

8 2 2 f 4 2 4 c 9 4 f f b e 1 e 9 b 0 e 5 3 d f 6 d 8 5 1 d a 4

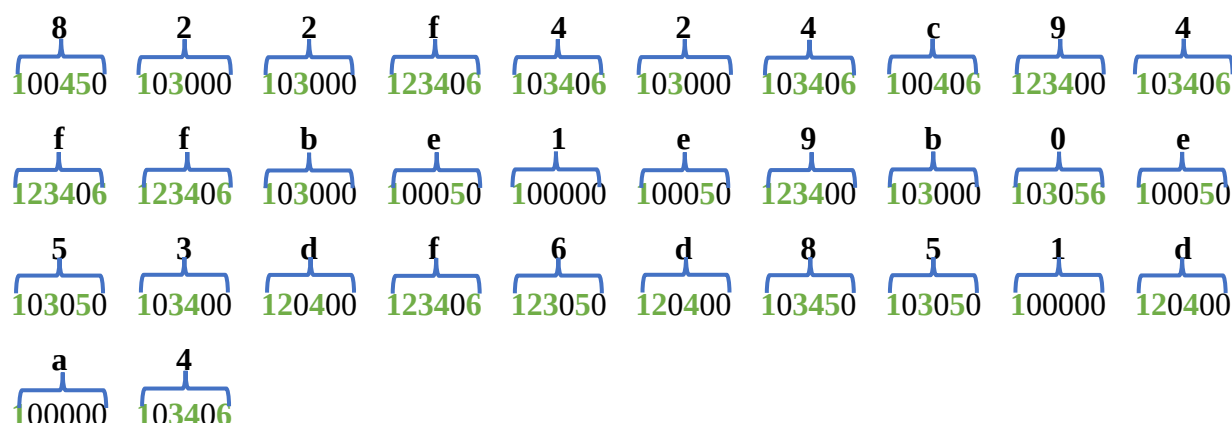
Étape 2 : transformation en Braille

Six chiffres remplaceront chaque caractère. Nous mettons d'abord des "0" juste pour faciliter l'explication. Sinon, dans le cas normal, nous pouvons directement passer à l'étape suivante :

8 2 2 f 4 2 4 c 9 4 f f b e 1 e 9 b 0 e 5 3 d f 6 d 8 5 1 d a 4

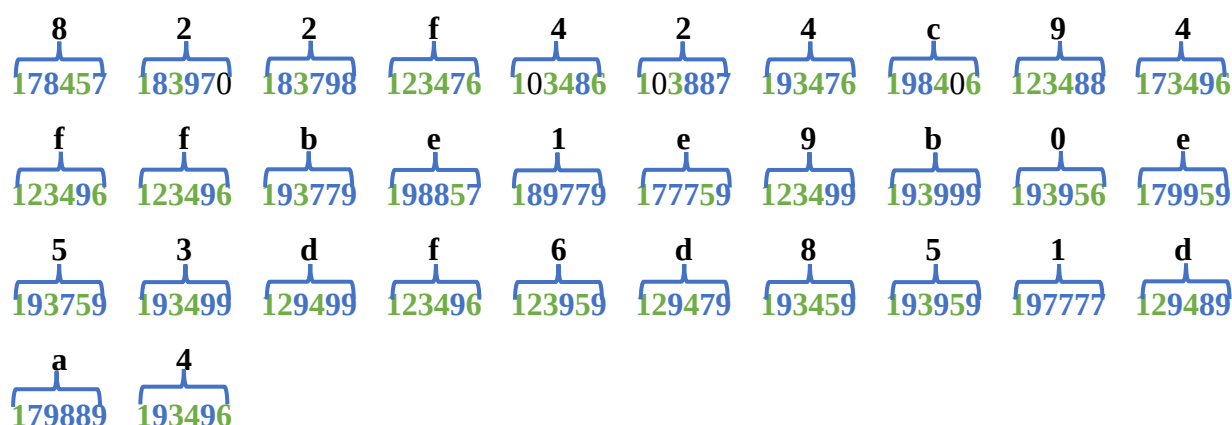


Chaque bloc est ensuite associé à son équivalent en code Braille. Le code Braille est composé de six chiffres, où les chiffres inutilisés sont remplacés par des zéros. Voici un exemple de cette transformation, où nous simplifions l'explication en laissant des zéros pour les chiffres inutilisés au départ.



Étape 3 : renforcement par substitution aléatoire

Les zéros présents dans la représentation Braille des caractères sont remplacés de manière aléatoire par les chiffres 7, 8 ou 9. Cette substitution aléatoire ajoute une variabilité supplémentaire, renforçant la sécurité globale du processus.



Le résultat final après la transformation Braille et le renforcement est le suivant :

178457183970183798123476103486103887193476198406123488173496123496123496193
779198857189779177759123499193999193956179959193759193499129499123496123959
129479193459193959197777129489179889193496

3.4.2.2 Processus de récupération du hachage initial

Nous effectuons la récupération du hachage initial à partir du code Braille final.

Étape 1 : division en blocs

Le hachage final est divisé en blocs de six chiffres :

178457 183970 183798 123476 103486 103887 193476 198406 123488 173496 123496
123496 193779 198857 189779 177759 123499 193999 193956 179959 193759 193499
129499 123496 123959 129479 193459 193959 197777 129489 179889 193496

Étape 2 : remplacement des chiffres 7, 8 ou 9

Ensuite, nous remplaçons les chiffres 7, 8 ou 9 par des zéros, ce qui nous donne les blocs initiaux :

100450 103000 103000 123406 103406 103000 103406 100406 123400 103406 123406
123406 103000 100050 100000 100050 123400 103000 103056 100050 103050 103400
120400 123406 123050 120400 103450 103050 100000 120400 100000 103406

Étape 3 : reconversion en caractères

Enfin, chaque bloc Braille est reconverti dans son caractère d'origine pour retrouver le hachage initial :

822f424c94ffbe1e9b0e53df6d851da4

3.4.3 Étude comparative

Dans cette analyse comparative, nous allons évaluer la transformation Braille en comparaison avec trois autres algorithmes bien connus : bcrypt, argon2, scrypt, et puis notre méthode précédente « H-Rotation ». Nous étudierons les performances de ces algorithmes en termes de temps d'exécution, de consommation mémoire et de résistance aux attaques, tout en introduisant des éléments mathématiques pour justifier les résultats. Nous intégrerons également un tableau pour récapituler les principales conclusions.

3.4.3.1 Temps d'exécution

Dans notre analyse comparative des méthodes de hachage, nous comparons les performances de méthodes de hachage différentes, à savoir, la transformation Braille, bcrypt, argon2, « H-Rotation » et scrypt. Afin d'avoir des résultats pertinents, il était nécessaire de simuler les temps d'exécution de chaque méthode pour des mots de passe spécifiques utilisant un langage de

programmation. Python a été choisi en raison de sa simplicité et de sa convivialité. Nous avons utilisé la méthode Sleep pour simuler la durée et aider à la simulation du temps d'exécution de chaque méthode. La figure 3-11 illustre une comparaison des temps d'exécution de l'un des mots de passe (« QWERTY » ; « Admin \$+123654 » ; « 123456789 ») en utilisant les transformations Braille, « H-Rotation », bcrypt, scrypt et argon2.

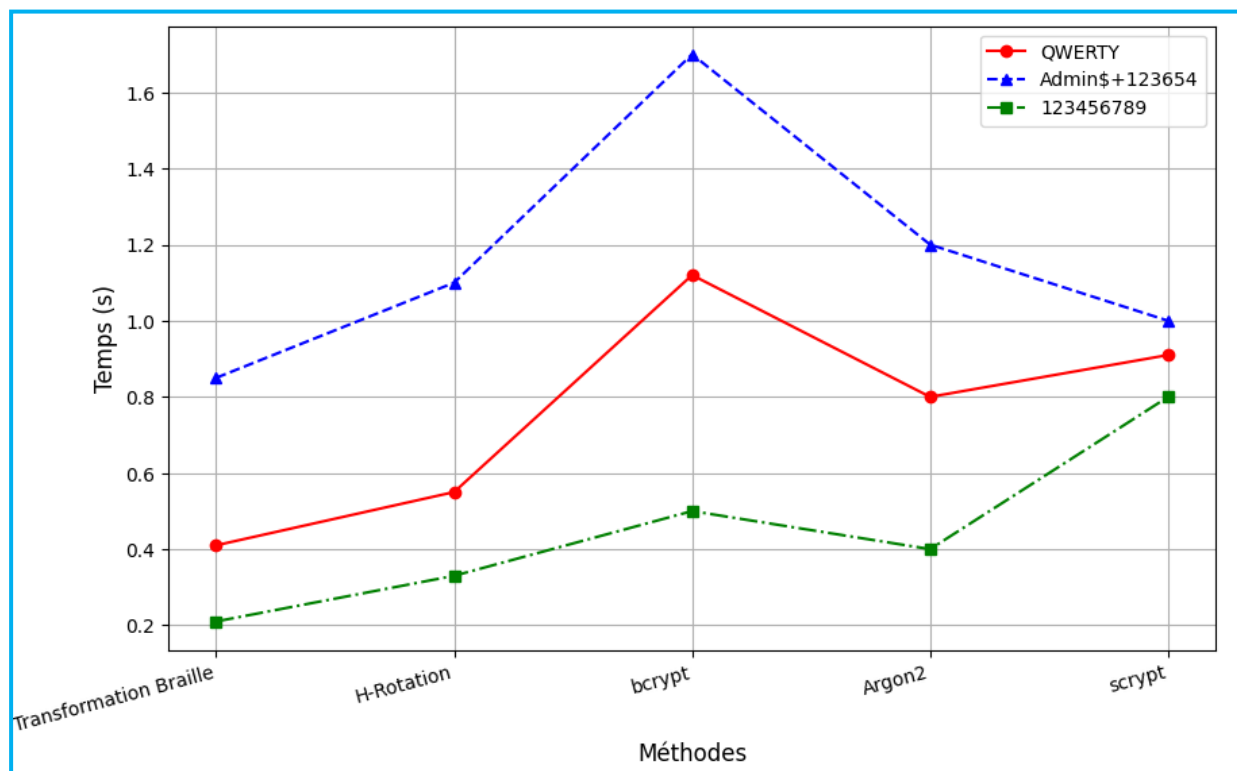


Figure 3- 11 : Comparaison entre la transformation Braille, bcrypt, scrypt, argon2 ; « H-Rotation » sur le temps d'exécution en millisecondes pour les 3 mots de passe (QWERTY ; Admin\$+123654 ; 123456789)

La transformation Braille se révèle être une méthode de hachage particulièrement performante en termes de temps d'exécution, surpassant toutes les autres techniques comparées, notamment bcrypt, argon2, « H-Rotation » et scrypt. Cette rapidité s'explique par sa conception algorithmique innovante, qui privilégie une transformation directe des caractères du hachage initial, limitant ainsi les opérations répétitives coûteuses et les dépendances à la mémoire. Par exemple, pour le mot de passe « Admin\$+123654 », scrypt se distingue par sa dépendance à une gestion mémoire intensive, ce qui rallonge son temps de traitement. argon2, bien qu'optimisé, reste également plus lent en comparaison. En revanche, la transformation Braille termine son processus bien plus rapidement, montrant un avantage significatif en termes de rapidité.

Cependant, cette rapidité ne se fait pas au détriment de la sécurité. La transformation Braille intègre des mécanismes tels que des substitutions aléatoires et des modifications spécifiques des caractères, rendant chaque hachage unique et imprévisible. Cela ajoute une couche significative de complexité pour un attaquant qui chercherait à inverser ou à prédire les résultats. En effet, un tel attaquant devrait non seulement contourner les protections du hachage initial, mais également déchiffrer les transformations uniques introduites par la méthode Braille.

3.4.3.2 Occupation mémoire

La consommation mémoire des méthodes de hachage est un facteur crucial à prendre en compte pour évaluer leur efficacité, notamment dans des environnements où les ressources sont limitées. L'analyse comparative des méthodes de hachage, incluant la transformation Braille, bcrypt, scrypt, « H-Rotation » et argon2, a révélé des différences significatives dans l'utilisation des ressources mémoire lors du stockage des mots de passe hachés. Les tests ont été réalisés sur trois mots de passe de complexité variée : « QWERTY », « Admin\$+123654 », et « 123456789 », en appliquant les configurations standards pour chaque méthode. Les résultats, représentés dans la figure 3-12, montrent que la transformation Braille est la méthode la moins consommatrice en mémoire, avec une taille de mémoire stable, indépendamment de la complexité des mots de passe. Contrairement à scrypt et argon2, qui requièrent une mémoire significativement plus élevée pour renforcer leur robustesse, la transformation Braille conserve une empreinte mémoire minimale grâce à son approche algorithmique optimisée, qui ne dépend pas de structures intermédiaires complexes. En revanche, scrypt, par exemple, nécessite des tableaux mémoire de grande taille pour protéger contre les attaques par force brute, ce qui augmente considérablement son empreinte mémoire, en particulier pour des mots de passe plus longs ou avec des paramètres de configuration intensifs.

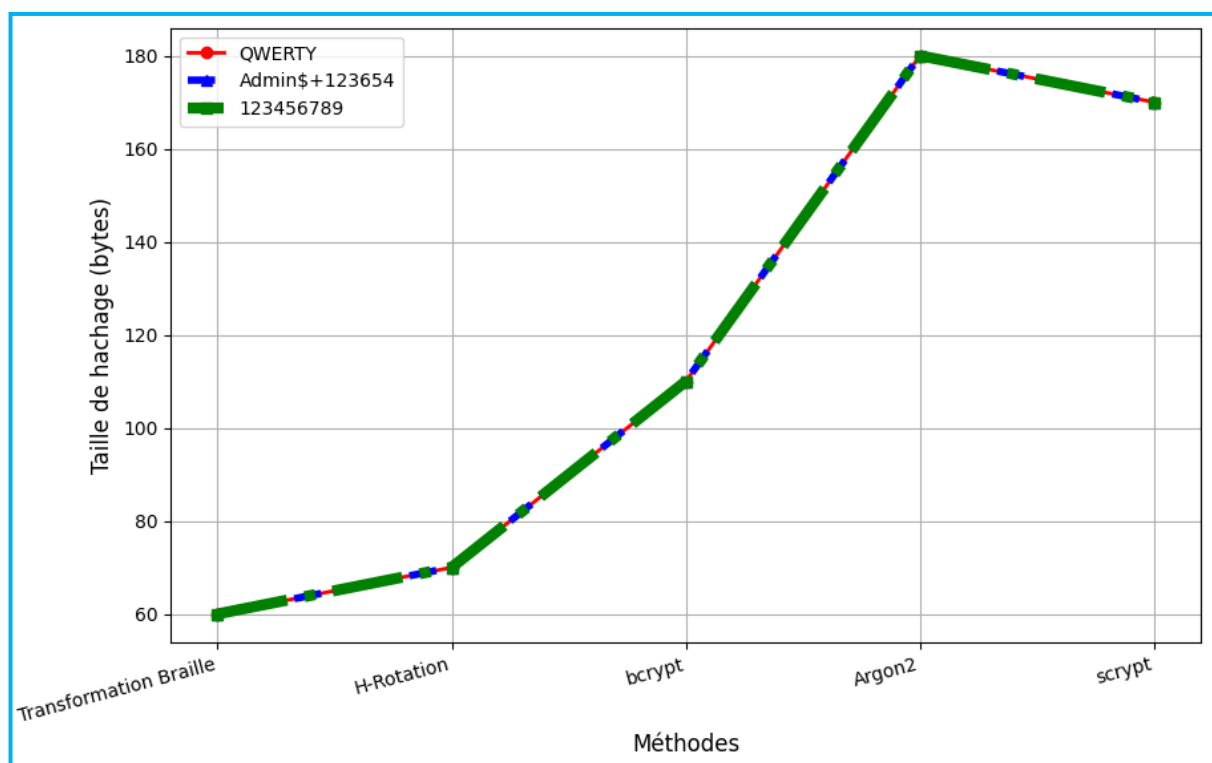


Figure 3- 12 : Comparaison entre la transformation Braille, bcrypt, scrypt, argon2 sur la taille mémoire utilisée par chacune

Bcrypt et « H-Rotation » affichent une consommation mémoire modérée, se situant entre la transformation Braille et les méthodes gourmandes comme scrypt et argon2. Cela est dû à leur structure algorithmique qui, bien qu'efficace, introduit une certaine dépendance à des tables de dérivation de clés ou des transformations dynamiques intermédiaires. Ces caractéristiques en font des choix équilibrés pour des environnements où la mémoire disponible est limitée, mais où un certain niveau de sécurité est exigé. Il est important de noter que la consommation mémoire n'est pas l'unique indicateur de la qualité ou de la sécurité d'une méthode de hachage. Bien qu'une faible empreinte mémoire, comme celle de la transformation Braille, soit avantageuse pour des systèmes embarqués ou à ressources restreintes, des paramètres tels que la protection contre les attaques par force brute et par dictionnaire ayant un rôle clé dans la sélection de la méthode adaptée. Par exemple, des algorithmes comme argon2 et scrypt, bien que plus gourmands en mémoire, sont spécialement conçus pour rendre ces attaques coûteuses en termes de ressources, augmentant ainsi leur robustesse.

3.4.3.3 Résistance aux attaques

Nous évaluons les capacités des différentes méthodes de hachage à résister aux principales attaques : force brute, dictionnaire, différentielle et collisions. Le tableau 3-6 illustre les résultats fondés sur des analyses expérimentales et théoriques.

Attaques par force brute : la transformation Braille se distingue par son approche unique, introduisant une transformation indépendante et aléatoire pour chaque caractère. Ce mécanisme élargit de manière exponentielle l'espace de recherche pour un attaquant, rendant chaque tentative de décryptage coûteuse en calculs. En ajoutant une randomisation des symboles, cette méthode complexifie davantage le processus, ralentissant considérablement les attaques, même lorsqu'elles sont effectuées à l'aide de ressources matérielles avancées, comme des clusters de GPU. Sa conception est particulièrement adaptée aux environnements nécessitant une rapidité d'exécution et une faible empreinte mémoire, tout en maintenant une robustesse significative face aux attaques cryptographiques.

La méthode « H-Rotation » repose sur des rotations dynamiques appliquées aux préfixes, aux suffixes et aux milieux du hachage, augmentant la difficulté d'un calcul inversé. Bien que ces transformations rendent les hachages imprévisibles, elles demeurent légèrement plus sensibles que la transformation Braille dans des scénarios où l'attaquant dispose d'importantes capacités parallèles. Cependant, son efficacité et son équilibre entre vitesse et sécurité en font une méthode compétitive pour de nombreuses applications modernes.

En comparaison, bcrypt et scrypt appliquent des itérations supplémentaires pour ralentir les attaques par force brute. Cependant, ces algorithmes n'introduisent pas de modifications structurelles aux hachages eux-mêmes, ce qui les rend fortement dépendants de la puissance de calcul de l'attaquant. Ils restent robustes, mais leur approche classique les place en retrait face à des méthodes plus modernes et innovantes comme la transformation Braille ou argon2.

Enfin, argon2 se positionne comme une méthode de hachage hautement sécurisée, grâce à une gestion optimisée de la mémoire et une parallélisation efficace. Ces caractéristiques le rendent particulièrement résistant aux attaques par force brute et adapté aux environnements où la sécurité est prioritaire. Cependant, sa complexité algorithmique le rend plus coûteux en termes de ressources, notamment dans des systèmes à grande échelle ou à faible capacité matérielle.

Attaques par dictionnaire : la transformation Braille se distingue par sa capacité à neutraliser efficacement les attaques par dictionnaire grâce à la randomisation des caractères et à leur conversion en symboles Braille uniques. Ce procédé supprime toute correspondance directe avec des mots du dictionnaire, rendant chaque tentative de correspondance imprévisible. La variation aléatoire appliquée aux caractères ajoute une couche supplémentaire de complexité, compliquant davantage les efforts d'un attaquant pour prédire ou recréer le hachage d'origine. Cette approche innovante renforce significativement la robustesse contre les attaques ciblant des mots courants ou prévisibles.

« H-Rotation », de son côté, perturbe les schémas traditionnels de hachage en appliquant des rotations dynamiques aux préfixes, aux suffixes et aux milieux. Cela rend l'utilisation de dictionnaires classiques inefficace, car les schémas de hachage varient considérablement d'une tentative à l'autre. Toutefois, l'efficacité de cette méthode dépend fortement de la complexité et du nombre de rotations appliquées, ce qui peut limiter sa robustesse si les transformations ne sont pas suffisamment sophistiquées.

En revanche, bcrypt et scrypt offrent une résistance moyenne contre les attaques par dictionnaire. Bien qu'ils intègrent un salage pour rendre chaque hachage unique, celui-ci reste fixe et ne bénéficie pas de transformations aléatoires ou dynamiques. Cette limitation les rend vulnérables dans des cas où des mots similaires apparaissent fréquemment dans le dictionnaire, permettant à un attaquant d'exploiter des correspondances partielles ou des modèles répétitifs. Argon2 se démarque par une combinaison efficace de salage et d'accès mémoire optimisé, augmentant significativement la difficulté des attaques par dictionnaire. Sa conception le rend particulièrement robuste face à des dictionnaires standards, mais son efficacité peut diminuer lorsqu'il est confronté à des dictionnaires massifs, notamment si ceux-ci incluent de multiples variations pour un même mot de passe. Malgré cela, sa conception avancée lui confère un avantage net dans des environnements nécessitant une sécurité élevée.

Attaques différentielles : la transformation Braille se distingue par sa capacité à générer une entropie élevée en modifiant la fréquence et la distribution des caractères dans le hachage final. Cette approche rend les variations imprévisibles et empêche un attaquant d'établir des corrélations exploitables entre des entrées similaires et leurs hachages correspondants. En réorganisant les caractères de manière unique pour chaque transformation, la méthode élimine efficacement les schémas répétitifs que les attaques différentielles pourraient tenter d'exploiter. La méthode « H-Rotation », avec ses rotations dynamiques appliquées aux préfixes, aux suffixes et aux milieux, perturbe les relations linéaires entre les caractères, réduisant ainsi les possibilités d'attaques différentielles. Cependant, l'efficacité de cette méthode dépend du degré d'aléatoire et de la complexité des rotations appliquées. Si les transformations ne sont pas suffisamment imprévisibles, des corrélations partielles entre des hachages similaires pourraient encore être exploitées, bien que dans une moindre mesure par rapport aux méthodes plus statiques.

En revanche, bcrypt souffre d'une absence de transformations dynamiques ou de modifications structurelles dans ses hachages. Cette limitation rend l'algorithme plus vulnérable aux attaques différentielles, car les relations entre les mots de passe d'entrée et leurs hachages restent

relativement linéaires. Cela permet à un attaquant déterminé d'exploiter des modèles répétitifs pour déduire certaines informations.

Scrypt et argon2 offrent une protection robuste contre les attaques différentielles grâce à leur conception avancée. Leur forte consommation mémoire et leur capacité de parallélisation perturbent considérablement les schémas exploitables par un attaquant. En augmentant la difficulté de comparer et de relier des variations dans les hachages, ces algorithmes minimisent efficacement les opportunités pour un attaquant de tirer parti des relations entre des entrées similaires.

Collisions : la transformation Braille se distingue par sa capacité à minimiser les risques de collision grâce à la randomisation et à l'unicité des symboles générés pour chaque caractère. En appliquant des transformations uniques à chaque entrée, même des mots de passe similaires produisent des résultats significativement différents. Ce mécanisme renforce la robustesse du hachage, réduisant les probabilités qu'une même sortie soit générée pour deux entrées différentes, même dans des scénarios où un attaquant manipule des variations proches.

La méthode « H-Rotation », grâce à ses rotations dynamiques appliquées aux préfixes, aux suffixes et aux milieux, contribue également à limiter les risques de collision. Les transformations perturbent les structures linéaires des hachages, augmentant la variabilité entre les sorties pour des entrées similaires. Cependant, son efficacité dépend de la complexité des rotations et de leur aléatoire. Dans des attaques ciblées où l'attaquant analyse systématiquement les transformations, des risques minimes de collision subsistent.

Pour bcrypt, bien que les collisions soient peu fréquentes en raison de l'utilisation d'un salage et d'un algorithme de hachage fiable, ce dernier n'est pas spécifiquement conçu pour les prévenir activement. Le hachage produit est stable, mais les schémas répétés ou similaires dans les mots de passe d'entrée peuvent, dans des cas exceptionnels, augmenter les risques de sorties identiques.

Les algorithmes scrypt et argon2 sont particulièrement robustes contre les collisions grâce à leurs conceptions avancées. En combinant une forte consommation de ressources mémoire et des structures complexes, ces méthodes introduisent des variations significatives dans les résultats, même pour des entrées similaires. Ces mécanismes rendent les collisions hautement improbables, surtout dans des environnements où les paramètres sont soigneusement ajustés pour maximiser la sécurité.

Tableau 3- 5 : Comparaison de la méthode proposée, bcrypt, scrypt, argon2 ; « H-Rotation » en termes d'attaque par force brute, attaque par dictionnaire, attaque différentielle, temps d'exécution et taille mémoire occupée

Méthode	Force brute	Dictionnaire	Collision	Différentielle
Transformation Braille	Espace de recherche étendu	Randomisation imprévisible	Unicité des hachages	Uniformité renforcée
H-Rotation	Complexité dynamique accrue	Rotations imprévisibles	Probabilités réduites	Corrélations perturbées
Bcrypt	Ralentissement progressif	Salage basique	Protection moyenne	Vulnérabilité structurelle
Argon2	Bloc mémoire optimisé	Entropie augmentée	Protection conditionnelle	Robustesse partielle
Scrypt	Mémoire renforcée	Accès mémoire optimisé	Résistance maximale	Structure avancée

La transformation Braille s'impose comme une solution innovante et efficace, offrant une protection renforcée contre les attaques courantes tout en conservant des exigences modérées en termes de ressources. Sa flexibilité et sa compatibilité avec des infrastructures existantes en font un choix idéal pour des environnements nécessitant une sécurité robuste sans compromis sur les performances.

3.4.4 Synthèse

La méthode de transformation Braille pour le hachage des mots de passe se démarque par plusieurs points forts. Sa légèreté en termes de consommation mémoire la rend adaptée aux environnements à ressources limitées. En termes de rapidité, elle surpasse les méthodes existantes comme bcrypt, argon2, scrypt et « H-Rotation », offrant ainsi un temps d'exécution optimisé pour les systèmes nécessitant des performances élevées. Sur le plan de la sécurité, elle introduit un niveau de protection supplémentaire en remplaçant les caractères du hachage par leur équivalent en symboles Braille, ce qui renforce la résistance aux attaques par force brute et par dictionnaire. En termes de perspectives, des essais futurs pourraient explorer l'application

de la méthode Braille dans des systèmes multi-facteurs, ou encore son intégration dans des infrastructures nécessitant des performances optimisées avec une sécurité renforcée, comme les environnements de cloud computing ou les réseaux Internet of Things. Il serait également pertinent d'étudier comment l'algorithme se comporte sous des contraintes extrêmes, en augmentant le volume des mots de passe ou en le combinant avec d'autres techniques de cryptographie post-quantique.

3.5 Conclusion

Dans le présent chapitre, nous avons exploré deux méthodes cryptographiques innovantes pour le hachage sécurisé des phrases/mots de passe qui sont « H-Rotation » et la transformation Braille. La méthode « H-Rotation » se distingue par sa capacité à appliquer des transformations dynamiques sur les blocs de hachage, ce qui augmente la résistance aux attaques par force brute et par dictionnaire, tout en restant relativement simple à implémenter. Cependant, cette méthode montre certaines limites dans des environnements hautement parallèles ou face à des attaques statistiques exploitant des modèles prédéfinis. C'est dans ce contexte que la transformation Braille apporte une réponse complémentaire, en introduisant une randomisation basée sur la substitution des caractères par leurs équivalents en symboles Braille. Cette méthode se démarque non seulement par sa rapidité et son faible usage de la mémoire, mais aussi par sa capacité à offrir une protection supplémentaire contre les attaques par dictionnaire et force brute. Ensemble, « H-Rotation » et la transformation Braille représentent un équilibre entre efficacité et sécurité, chacune comblant les lacunes de l'autre. Tandis que « H-Rotation » excelle dans les systèmes distribués avec une complexité modérée, la transformation Braille propose une approche plus légère et adaptable aux environnements où la performance est importante. Les deux méthodes montrent un potentiel fort pour être combinées ou intégrées dans des infrastructures cryptographiques avancées, notamment dans des systèmes de gestion de mots de passe, des environnements de cloud computing ou encore des architectures IoT.

Cette exploration des méthodes de hachage nous prépare à aborder le chapitre suivant, qui se concentrera sur les méthodes de sécurité avancées pour la transmission sécurisée des données. Nous explorerons des techniques qui vont au-delà du simple hachage, intégrant des protocoles de chiffrement modernes et des mécanismes de transmission sûrs afin de garantir l'intégrité et la confidentialité des informations dans des systèmes à grande échelle.

Chapitre 4 : Méthodes avancées de sécurité pour la transmission sécurisée des données

4.1 Introduction

Dans un environnement numérique de plus en plus interconnecté, la sécurité des communications est devenue un enjeu prioritaire pour protéger les informations échangées sur les réseaux. Les protocoles tels que SSL/TLS et SSH sont des outils essentiels pour garantir la confidentialité, l'intégrité et l'authenticité des données, particulièrement dans un contexte où les cyberattaques deviennent de plus en plus sophistiquées et variées.

Le protocole TLS est largement utilisé pour chiffrer les communications entre un client et un serveur, notamment dans les transactions en ligne. Ce protocole repose sur des algorithmes de chiffrement asymétrique comme RSA et sur la négociation de clés sécurisées via Diffie-Hellman, en plus de mécanismes de chiffrement symétrique tels qu'AES. De l'autre côté, SSH permet des connexions sécurisées à distance en chiffrant les données échangées, afin d'éviter toute interception malveillante lors de la gestion des systèmes et des serveurs distants.

Nous avons mis l'accent sur l'analyse approfondie des mécanismes de ces protocoles et des failles potentielles qu'ils présentent. Ainsi, deux contributions majeures pour la transmission sécurisée des données ont été réalisées :

La première contribution [TOUIL 21] concerne une méthode dynamique pour ajuster la taille des clés de chiffrement en temps réel selon la qualité du canal réseau. Si la qualité du réseau se dégrade, la méthode réduit la taille des clés de chiffrement pour maintenir une communication fluide et sécurisée, tout en surveillant en permanence les paramètres de latence, de perte de paquets et de gigue. En cas de détection d'une attaque, comme une attaque DOS ou bien MiTM, le protocole adapte la sécurité de manière proactive pour limiter les effets de l'attaque. Cette approche combine à la fois robustesse et flexibilité, garantissant ainsi une communication sécurisée même dans des conditions de réseau difficiles.

La deuxième contribution [TOUIL 24b] consiste à remplacer l'algorithme Diffie-Hellman traditionnel utilisé pour l'échange de clés dans SSH par le protocole de cryptographie quantique BB84. Grâce à l'utilisation de photons et des bases quantiques, le protocole BB84 permet une distribution de clés symétrique sécurisée et garantit que toute tentative d'interception serait immédiatement détectée en raison des propriétés de la mécanique quantique. L'intégration de ce protocole dans SSH nécessite des ajustements au niveau des processus de chiffrement existants tout en maintenant la compatibilité avec les algorithmes classiques comme AES. Cette

approche prépare les systèmes de communication à résister aux menaces des ordinateurs quantiques tout en assurant une sécurité puissante dans les environnements actuels.

Le présent chapitre vise à explorer en détail ces méthodes avancées. L'objectif est de prouver leur efficacité à travers l'analyse de cas d'usage et de souligner leur potentiel à renforcer la sécurité des systèmes modernes.

4.2 Présentation des enjeux de la sécurité de transmission

La sécurité des transmissions et la qualité de service dans les communications modernes sont devenues des priorités importantes, en particulier dans un contexte où les cyberattaques se multiplient et les réseaux sont de plus en plus sollicités. Il est donc nécessaire de garantir la confidentialité, l'intégrité et la disponibilité des données tout en maintenant une qualité de service optimale. La gestion simultanée de ces deux aspects soulève des défis techniques majeurs, nécessitant des solutions à la fois robustes et flexibles [Ono 18].

4.2.1 Analyse des enjeux liés à la sécurité des données

Les menaces pesant sur la sécurité des communications incluent les attaques par interception, telles que les attaques MiTM, qui visent à intercepter et à modifier les communications entre deux parties sans être détectées. En réponse, des protocoles comme TLS et SSH utilisent des algorithmes cryptographiques basés sur le chiffrement symétrique et asymétrique pour protéger les échanges. L'un des algorithmes couramment utilisés est AES, qui repose sur l'exponentiation modulaire définie dans la formule (2.2) pour chiffrer et déchiffrer les données. Cette formule est essentielle dans de nombreux algorithmes cryptographiques comme RSA et Diffie-Hellman, car elle garantit que, même si un attaquant intercepte le message chiffré, il sera extrêmement difficile de retrouver le message original sans la clé privée correspondante.

Dans le cadre du protocole SSH, bien que ce dernier soit réputé pour sa robustesse, il présente également des limitations face aux ordinateurs quantiques. Par exemple, le problème du logarithme discret, formule (2.12) utilisé dans des systèmes tels que Diffie-Hellman, qui est couramment employé dans SSH pour l'échange de clés, est difficile à résoudre avec des ordinateurs classiques. Cependant, il devient vulnérable aux attaques quantiques grâce à l'algorithme de Shor, qui réduit considérablement la complexité du calcul de logarithmes discrets et compromet ainsi la sécurité de l'échange de clés [Shor 99, Koblitz 16].

4.2.2 QoS dans les communications (cas de figure : échanges multimédias)

La gestion de la sécurité dans les communications modernes est un défi majeur, particulièrement lorsqu'il s'agit de garantir simultanément la sécurité et la QOS. Pour les applications multimédias, telles que le streaming vidéo, les appels vocaux ou la transmission de données en temps réel, il est indispensable de garantir une performance réseau stable. Ces services exigent des ressources constantes en termes de bande passante, de latence⁹ et de gigue¹⁰. En parallèle, la sécurité doit assurer que les informations échangées soient protégées contre toute tentative d'interception ou de manipulation, mais cela pose souvent un dilemme [Maqri 20, Jubair 22].

En effet, bien que les techniques de chiffrement, comme l'AES ou le RSA, soient très efficaces pour garantir la confidentialité des données, elles introduisent fréquemment des latences supplémentaires dans les communications. Ce retard est particulièrement notable dans des contextes où la réactivité est essentielle, comme lors des vidéoconférences ou des jeux en ligne. Plus la complexité du chiffrement est élevée, plus le traitement demande du temps, ce qui peut dégrader l'expérience utilisateur. Cela est particulièrement visible lorsque la taille des clés de chiffrement augmente, car elle entraîne une augmentation de la charge de calcul nécessaire pour encoder et décoder les messages [Taleb 10, Serhrouchni 06].

Le problème réside principalement dans le fait que l'utilisation de grandes clés de chiffrement, bien qu'elles renforcent la sécurité, a un coût direct en termes de performance. Cela se traduit par une baisse de la réactivité des systèmes, une augmentation du temps de latence et, dans certains cas, des interruptions dans la QOS, ce qui est inacceptable dans des environnements nécessitant une synchronisation parfaite. Une solution envisageable pour limiter ces effets négatifs est d'adapter dynamiquement la taille des clés en fonction de la qualité du réseau à un moment donné. Par exemple, lorsque le réseau est performant, des clés de plus grandes tailles peuvent être utilisées, mais lorsque le réseau se dégrade, il pourrait être nécessaire de réduire la taille des clés pour maintenir la fluidité des communications [TOUIL 21].

⁹ La latence est le délai entre l'envoi d'une requête et la réception de la réponse dans un système de communication. Elle est souvent mesurée en millisecondes et affecte directement la performance des réseaux et des applications.

¹⁰ La gigue est la variation du délai entre la réception de paquets de données dans un réseau. Elle peut affecter la qualité des services en temps réel, comme la voix sur IP ou les vidéos en streaming, en provoquant des interruptions ou des dégradations.

4.2.3 Utilisation des méthodes quantiques pour sécuriser les échanges

Les limitations des protocoles actuels, comme l'augmentation des latences et des vulnérabilités face aux menaces post-quantiques, montrent clairement que de nouvelles approches sont indispensables. La cryptographie quantique s'impose comme une solution prometteuse, car elle repose sur les principes de la mécanique quantique, offrant ainsi des mécanismes de sécurité théoriquement inviolables. Un exemple notable est le protocole BB84, utilisé pour la distribution de clés quantiques. Ce protocole exploite les propriétés de la superposition et de l'intrication des photons pour générer des clés cryptographiques inviolables. Contrairement aux méthodes traditionnelles, qui dépendent de la complexité algorithmique, le BB84 base sa sécurité sur les lois fondamentales de la physique [Elhao 21, St-Jean 17].

Les méthodes post-quantiques comme le protocole de distribution de clés quantiques se distinguent par leur capacité à exploiter la difficulté de résoudre certains problèmes mathématiques complexes dans un espace quantique, tels que le problème du logarithme discret. Le protocole QKD repose sur des opérations dans des groupes cycliques définis dans la formule (2.1). Ces opérations mathématiques permettent de structurer la distribution des clés de manière à résister aux attaques, même dans un environnement post-quantique. En effet, ces structures mathématiques ne peuvent pas être exploitées efficacement par les algorithmes quantiques, comme l'algorithme de Shor [St-Jean 19, Petrenko 22].

4.3 Sécurisation des échanges via TLS : étude de cas sur la transmission de séquences vidéo

Cette approche vise à combler les lacunes des méthodes traditionnelles de sécurisation des flux vidéo, souvent vulnérables à des attaques telles que MiMT, la force brute et les attaques par interception. Les protocoles courants comme TLS, bien qu'ils offrent un chiffrement robuste, présentent des limitations lorsqu'ils sont utilisés dans des environnements nécessitant une faible latence et une haute QoS, comme dans la transmission de séquences vidéo en temps réel. Les solutions existantes se concentrent principalement sur l'augmentation du coût de calcul pour l'attaquant, mais n'adressent pas les aspects structurels des données transmises.

Pour pallier ces insuffisances, cette méthode propose une intégration dynamique du protocole TLS avec le protocole RTP, de manière à garantir une protection à plusieurs niveaux tout en optimisant la QoS. Plutôt que de simplement ajouter des couches de chiffrement, l'approche repose sur une transformation dynamique des paquets vidéo, modifiant leur structure à chaque

transmission. Cela rend les tentatives d'interception et de décryptage exponentiellement plus complexes pour les attaquants, réduisant ainsi la vulnérabilité aux attaques par force brute ou MiMT.

Contrairement aux approches traditionnelles qui augmentent la dépendance au calcul intensif, cette méthode n'affecte pas significativement la performance réseau. Elle se révèle donc particulièrement adaptée pour les environnements à large échelle, où la latence doit être minimisée et où un grand nombre d'utilisateurs doivent être supportés simultanément. La flexibilité de cette solution permet également d'ajuster le niveau de sécurité en fonction des spécificités du flux vidéo et des besoins opérationnels, garantissant ainsi un équilibre optimal entre sécurité et performance [TOUIL 21].

4.3.1 Présentation du concept de sécurisation via TLS : cas de la transmission vidéo

Dans la présente section, nous examinons notre approche qui sert à sécuriser la transmission des séquences vidéo du serveur vers l'utilisateur, même dans un environnement en constante évolution. Pour ce faire, nous basons notre méthode sur l'optimisation de la sécurité par ajustement dynamique de la taille des clés de chiffrement en fonction des changements du canal et des paramètres de la QoS. Plus précisément, nous commençons par analyser de plus près le processus de communication standard entre le serveur, d'une part et le client, d'autre part. Cette analyse nous permet de souligner les protocoles et les échanges d'informations clés. Par la suite, nous présentons notre proposition pour rendre l'ensemble du mécanisme automatisé et dynamique. Nous discutons du changement de la taille de la clé de chiffrement en temps réel. L'objectif est de rendre le processus plus adaptatif sans compromettre l'optimisation de la sécurité. L'explication des critères déclencheurs et des mécanismes de réglage automatique est discutée en détail. Cette section permet en outre d'explorer les rendements potentiels de notre approche. Nous expliquons comment cela peut rendre le système plus résistant contre les variations du canal et maintenir un niveau de QOS approprié pour chaque utilisateur. En fin de compte, la présente section fournit un aperçu complet de la conception et de la mise en œuvre de notre méthode. Cela forme les fondations de la sécurisation adaptative et intelligente des transmissions vidéo dans des environnements numériques.

4.3.1.1 Mécanismes de TLS dans la transmission vidéo

TLS fonctionne en encapsulant les données multimédias transmises à travers le réseau dans une couche de chiffrement sécurisée, empêchant ainsi les attaques de type MiTM ou l'écoute clandestine. Le protocole TLS garantit que les données sont cryptées via un mécanisme de chiffrement symétrique une fois que l'échange initial de clés a été effectué.

Bien que notre méthode s'applique à plusieurs domaines, nous concentrons notre attention sur une application spécifique qui consiste en l'étude de la transmission d'une séquence vidéo de la part d'un serveur à un client simple. Dans ce contexte, le client initie la communication en envoyant au serveur de l'information, ce qui inclut la version de TLS au serveur, l'ID de session, les suites de chiffrement, le type de clé et plus d'options telles que la clé et l'algorithme cryptographique supporté (voir les figures 4-1 et 4-2 ci-après) :

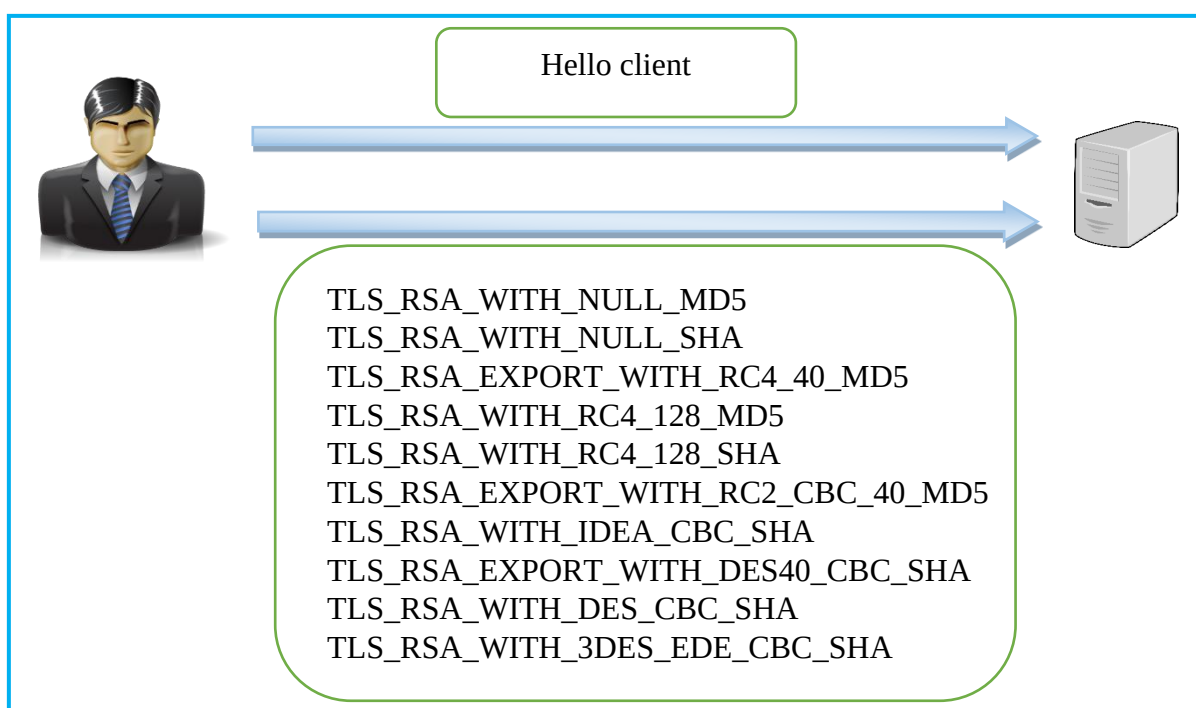


Figure 4- 1 : Démarrage de la négociation du protocole TLS par le client

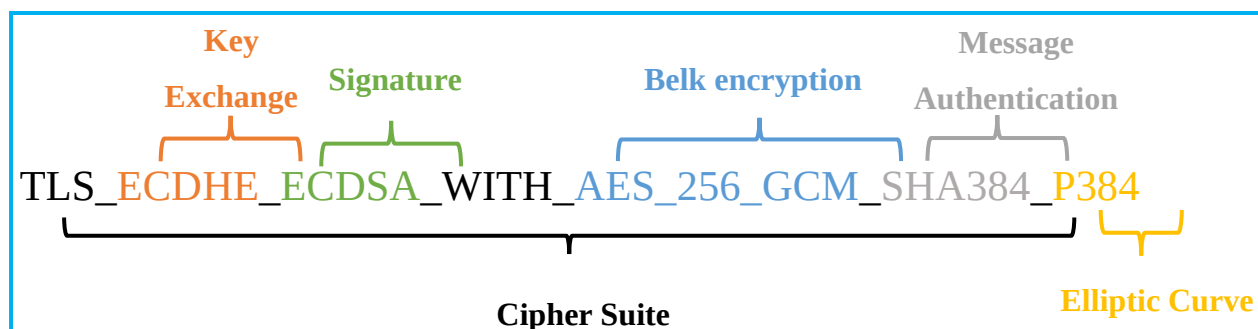


Figure 4- 2 : La suite de chiffrement utilisée dans les négociations TLS

Le serveur sélectionne la meilleure suite de chiffrement qu'il supporte et qui est également supportée par le client, puis l'envoie au client accompagnée de son certificat dans un message. Il peut également demander, le cas échéant, le certificat du client si cela est nécessaire. Lorsque le client reçoit le certificat, il vérifie si le serveur est de confiance, puis envoie la clé de chiffrement utilisée pour chiffrer les autres messages, comme illustré dans la figure 4-3.

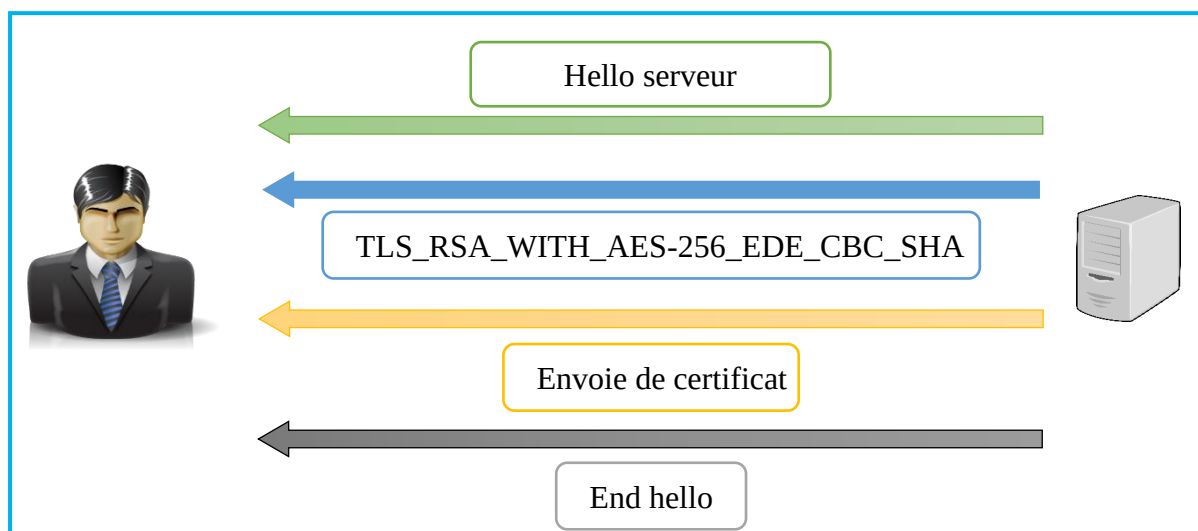


Figure 4- 3 : Accord sur la clé utilisée entre le serveur et le client

Cette clé est échangée lors de l'établissement de la session TLS grâce à des algorithmes asymétriques comme RSA ou Diffie-Hellman, qui reposent sur l'exponentiation modulaire de formule (2.2). Cependant, dans notre cas, il s'agit d'une procédure dynamique et automatique qui varie selon le mode de communication et la QoS. Pour améliorer la sécurité, nous utilisons au début AES 256 en cas de réseau sans fil ou filaire. Ensuite, le canal est surveillé en boucle ouverte avec, bien sûr, la latence, la gigue et d'autres paramètres. En cas de problèmes comme la congestion du réseau, le système change ou diminue la taille de la clé très rapidement. Après stabilisation du système, la taille de la clé est ajustée automatiquement et si rien ne change, nous augmentons la taille. Cette méthode est sensible et réactive, ce qui assure une transmission sécurisée de séquences vidéo tout en prenant en compte la dynamique du canal de communication. Le diagramme figure 4-4 décrit les étapes suivies dans notre méthode.

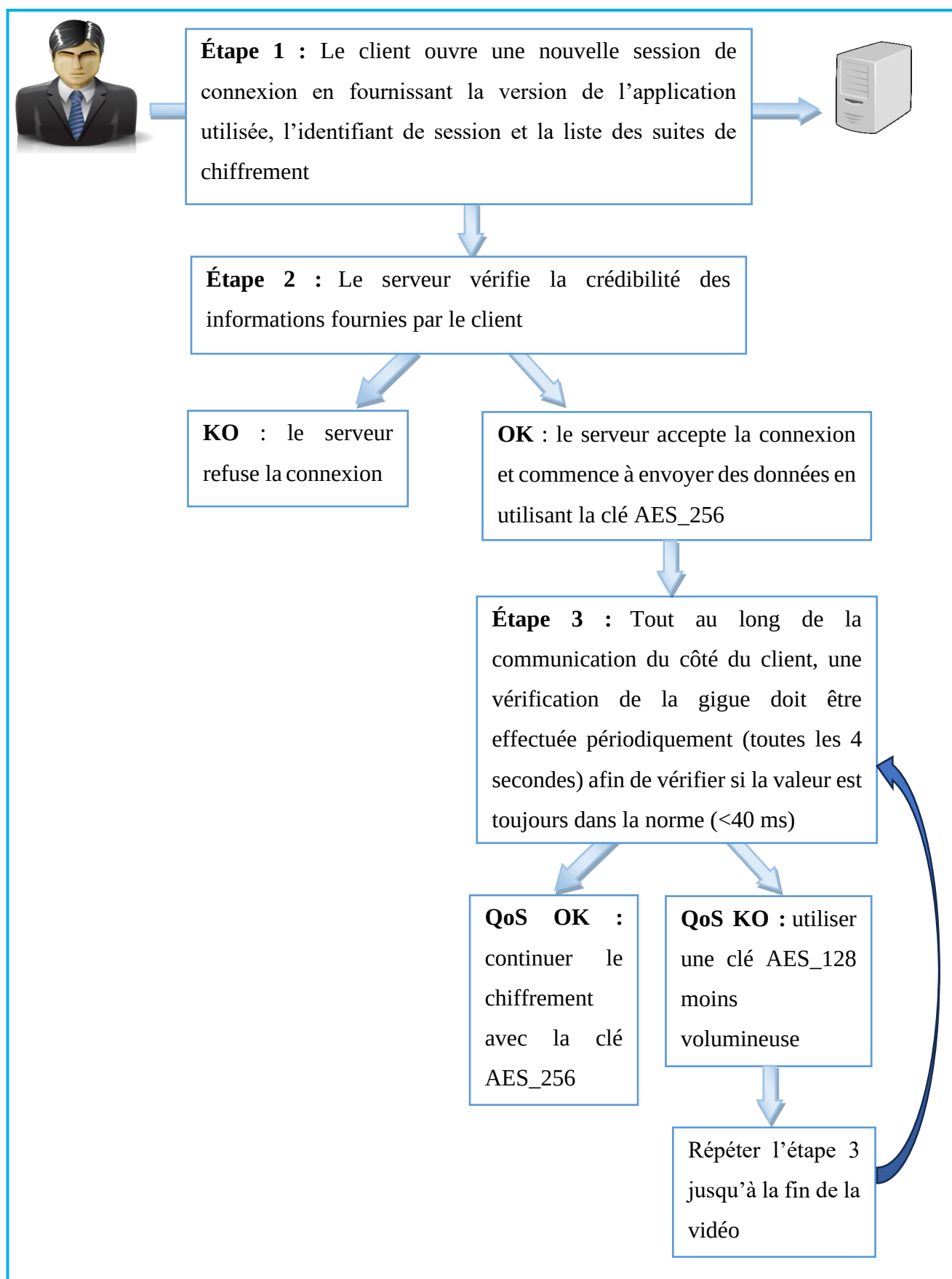


Figure 4- 4 : Processus de fonctionnement de l'échange de la clé

Cet algorithme révèle une solution innovante à la fusion de plusieurs objectifs extrêmement importants dans le contexte de la sécurité lors de la transmission de vidéo. En premier lieu, il implémente l'idée des renégociations qui ne pourraient initialement pas être présumées possibles, puisque la négociation des clés était statique au début de la communication. Désormais, avec l'introduction de cet algorithme, elles peuvent être lancées tout au long de la conversation sur la base du canal, de la gigue et de la latence, ce qui signifie une adaptation dynamique. Le point clé de cet algorithme est conçu pour alterner les clés à tour de rôle. L'avantage à cela est que chaque clé qui est employée dans la conversation est une clé temporaire, qui rend difficile l'attaque d'un pirate. En outre, il n'est pas nécessaire d'accorder des ressources pour l'utilisation régulière des clés plus sécurisées. La capacité de basculer entre les clés offre une gestion adaptable de la quantité de ressources engagées, préservant ainsi les ressources et garantissant divers niveaux de sécurité, assurant ainsi la robustesse. D'autres aspects clés de l'algorithme concernent l'optimisation des ressources. Au lieu de collecter les ressources pour atteindre le but, idéalement, elles sont allouées en fonction de la situation actuelle, assurant ainsi le succès. L'algorithme vise à atteindre un équilibre délibéré entre la QOS et la sécurité. Cela signifie que si le canal est surchargé et qu'il existe une latence intolérable, l'algorithme sélectionnera l'ensemble du chiffrement en utilisant des clés et des hachages plus légers, garantissant ainsi une qualité acceptable tout en assurant le niveau de sécurité optimal. Le diagramme de séquence illustré dans la figure 4-5 est une représentation plus détaillée des différentes interactions effectuées.

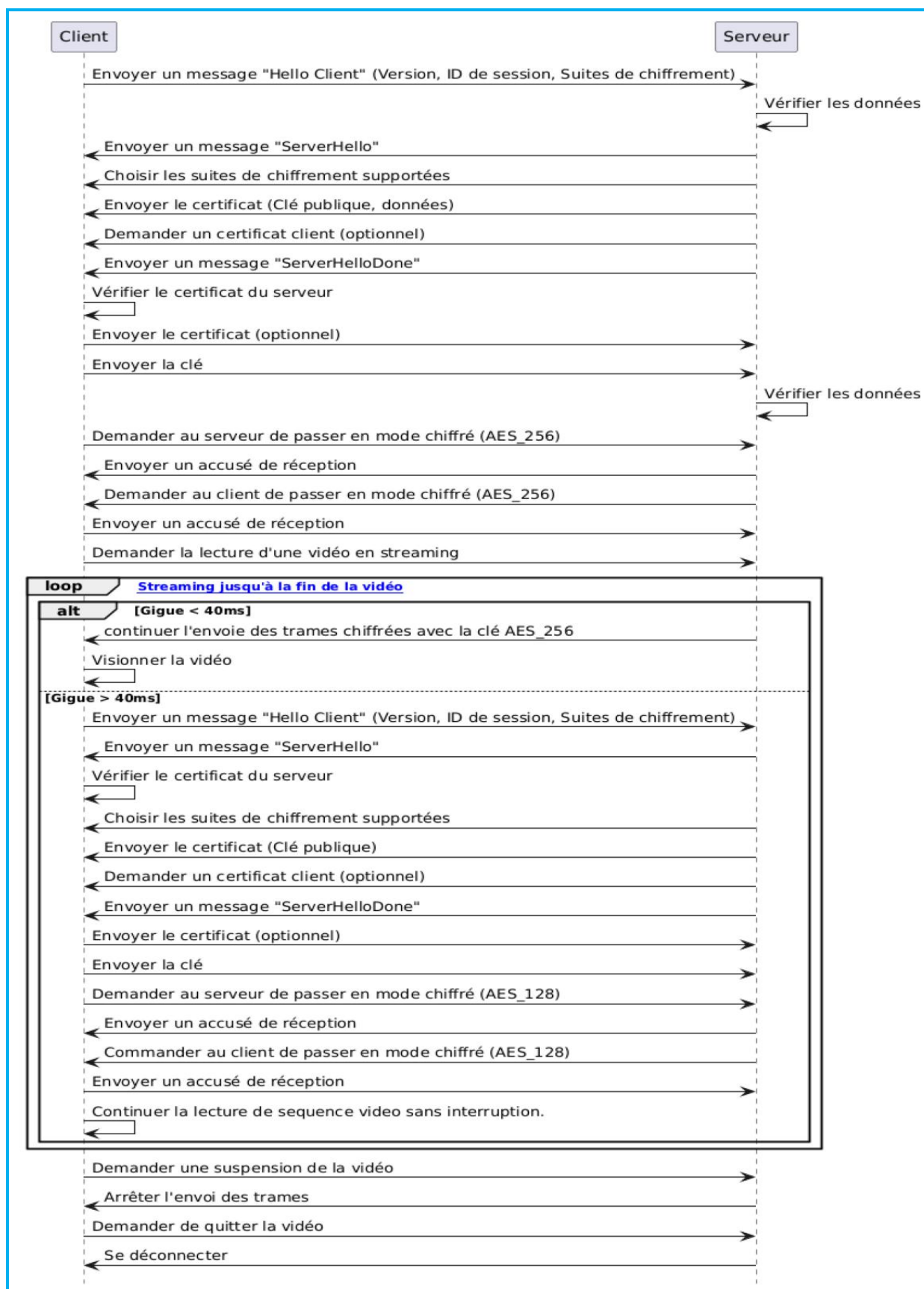


Figure 4- 5 : Diagramme de séquence des différentes interactions effectuées entre le client et le serveur

4.3.2 Résultat expérimental

L'objectif de cette expérimentation est de montrer l'efficacité de cette approche d'automatisation des négociations cryptographiques à l'aide du protocole TLS dans un environnement de transmission vidéo en temps réel. L'un des défis majeurs dans la transmission vidéo est de garantir à la fois la sécurité des données et la QoS, surtout lorsque le réseau est soumis à des contraintes telles que la congestion, la latence ou encore la gigue. La méthode traditionnelle repose sur une négociation unique où une clé est établie au début de la session et utilisée pour toute la durée de la communication. Cependant, cette approche montre des limitations lorsqu'il s'agit de scénarios dynamiques où la qualité du canal de communication peut varier au cours du temps. Nous avons donc conçu une approche qui permet d'ajuster dynamiquement les clés de chiffrement en fonction des conditions du réseau, afin de maintenir une sécurité optimale tout en garantissant une performance fluide lors de la transmission vidéo.

4.3.2.1 Mise en place de l'environnement de simulation

Pour simuler cette méthode, nous avons mis en place un environnement expérimental basé sur deux machines virtuelles opérant toutes deux sous un système Linux. La première machine représente le client, tandis que la seconde agit en tant que serveur de diffusion vidéo sécurisé. Le serveur utilise le protocole RTP pour la transmission des flux vidéo en temps réel, tandis que TLS est utilisé pour chiffrer les échanges de données entre le client et le serveur, garantissant ainsi la confidentialité et l'intégrité des communications.

Lors de la phase d'initialisation de la session, le client envoie une requête contenant les informations suivantes :

- La version du protocole TLS utilisée ;
- l'identifiant de session ;
- la liste des suites de chiffrement prises en charge par le client, notamment les algorithmes de chiffrement symétrique et les fonctions de hachage ;
- les clés publiques disponibles pour sécuriser la session.

Le serveur sélectionne alors la suite de chiffrement la plus adaptée, en prenant en compte à la fois les capacités du client et les siennes. Une clé publique est ensuite échangée via des certificats, garantissant que les deux parties peuvent établir une communication sécurisée sans qu'un tiers puisse intercepter les données. Une vérification mutuelle des certificats est effectuée avant que le client ne transmette la clé de chiffrement symétrique qui sera utilisée pour chiffrer les échanges vidéo. Cette phase d'initialisation est capturée via l'outil Wireshark (voir la figure

4-6), qui permet de suivre les étapes du processus, notamment les échanges de suites de chiffrement et la validation des certificats.

No.	Time	Source	Destination	Protocol	Length	Info
22	118.206270743	172.16.20.0	172.16.10.0	TLSv1.2	265	Client Hello 1
24	118.542074034	172.16.10.0	172.16.20.0	TLSv1.2	1370	Server Hello, Certificate, Server Key Exchange, Server Hello Done 2
26	119.402624079	172.16.20.0	172.16.10.0	TLSv1.2	141	Client Key Exchange
28	119.508075244	172.16.20.0	172.16.10.0	TLSv1.2	72	Change Cipher Spec
30	119.540677060	172.16.20.0	172.16.10.0	TLSv1.2	167	Encrypted Handshake Message 3
32	119.625717352	172.16.10.0	172.16.20.0	TLSv1.2	72	Change Cipher Spec
34	119.627210461	172.16.10.0	172.16.20.0	TLSv1.2	167	Encrypted Handshake Message
36	119.630020508	172.16.20.0	172.16.10.0	TLSv1.2	263	Application Data
38	119.632181005	172.16.10.0	172.16.20.0	TLSv1.2	183	Application Data 4

Figure 4- 6 : Initialisation de la session TLS dans wireshark

Dans notre simulation, nous avons configuré le serveur pour diffuser des vidéos à la demande de manière sécurisée. Plusieurs clients peuvent accéder simultanément aux flux vidéo via le protocole Real-Time Streaming Protocol, qui est couramment utilisé pour la diffusion en continu sur internet. Cela permet de tester la performance de notre approche en présence de multiples requêtes et sous des conditions de réseau variables.

4.3.2.2 Automatisation des négociations de clés

Le protocole TLS utilisé dans cette expérimentation permet, en temps normal, de négocier une clé de chiffrement qui sera utilisée pour la durée entière de la session. Toutefois, dans notre méthode, cette négociation est rendue dynamique et automatisée en fonction de l'état du canal de communication. Au lieu d'utiliser une clé fixe, notre système effectue des vérifications régulières pour s'assurer que la qualité du canal est adéquate.

Les paramètres surveillés incluent la latence et la gigue, toutes les 4 secondes, le client effectue une vérification automatique de ces paramètres. Si la gigue dépasse un seuil de 40 ms, ce qui est indicatif d'une congestion ou d'une saturation du réseau, le système initie une négociation rapide pour modifier la clé de chiffrement. En temps normal, la clé AES-256 est utilisée pour garantir un haut niveau de sécurité. Cependant, en cas de dégradation des conditions réseau, une nouvelle négociation est initiée pour passer à une clé plus légère, telle que AES-128, afin de réduire la complexité de calcul tout en maintenant un niveau de sécurité acceptable. Ce processus est illustré dans la figure 4-7, où l'algorithme sélectionne dynamiquement la clé en fonction de l'état du réseau.

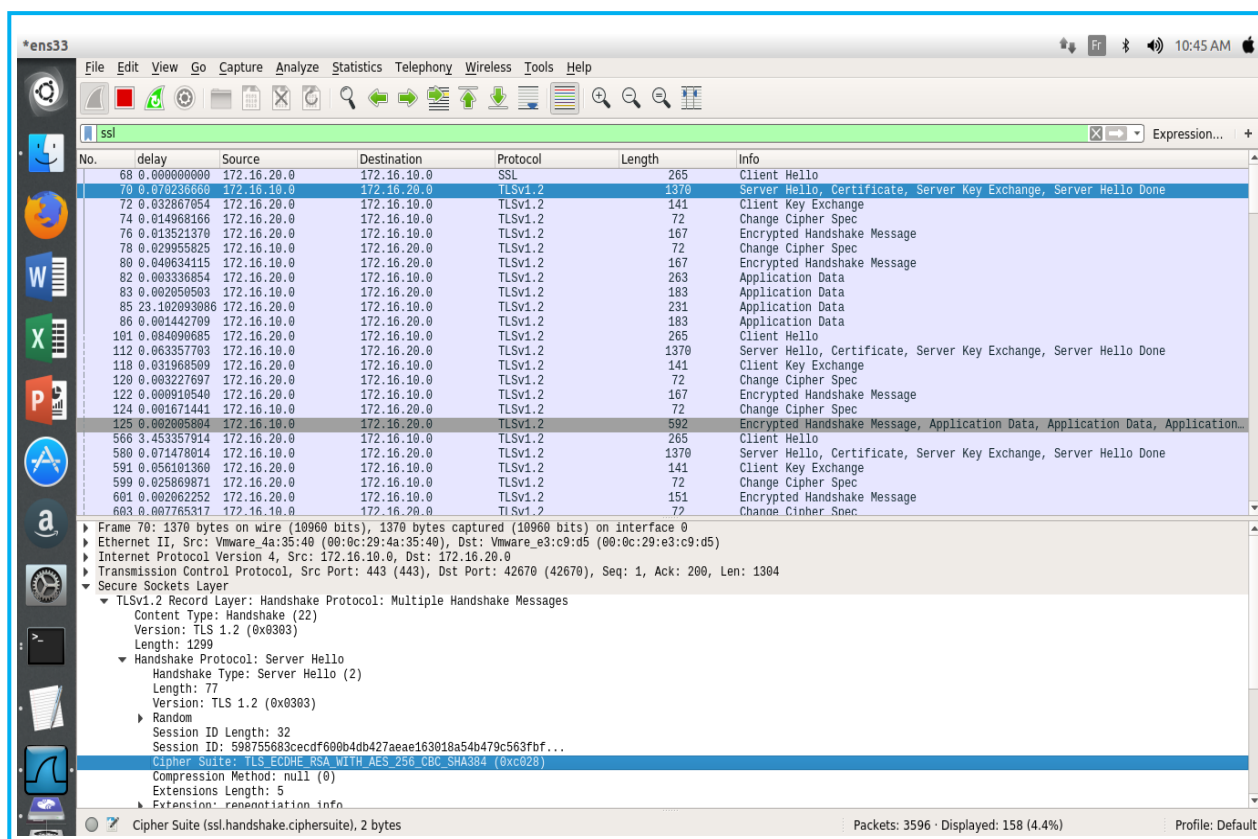


Figure 4- 7 : Chiffrement dynamique par AES-256 et AES-128 en fonction de la QoS

Cette approche est particulièrement avantageuse dans des scénarios de transmission vidéo en temps réel, où la fluidité du flux est primordiale. L'utilisation d'un algorithme dynamique permet de garantir que la vidéo reste fluide, même en cas de conditions réseau difficiles, sans compromettre la sécurité des données échangées.

Les résultats obtenus lors de cette expérimentation montrent que l'automatisation des négociations améliore considérablement la QoS et réduit la latence des transmissions vidéo. L'algorithme parvient à ajuster dynamiquement la taille des clés de chiffrement en fonction des variations du réseau, assurant ainsi une communication sécurisée tout en maintenant la

performance. Lorsque la gigue est inférieure à 40 ms, le système conserve l'utilisation de AES-256, offrant ainsi un niveau de sécurité maximal sans compromettre la fluidité du flux vidéo. En revanche, lorsque des anomalies réseau sont détectées (telles qu'une latence excessive ou une gigue importante), le système réagit en basculant vers AES-128, une clé plus légère, mais toujours sécurisée, permettant de maintenir la qualité de la transmission vidéo sans interruption (voir la figure 4-8 ci-après).

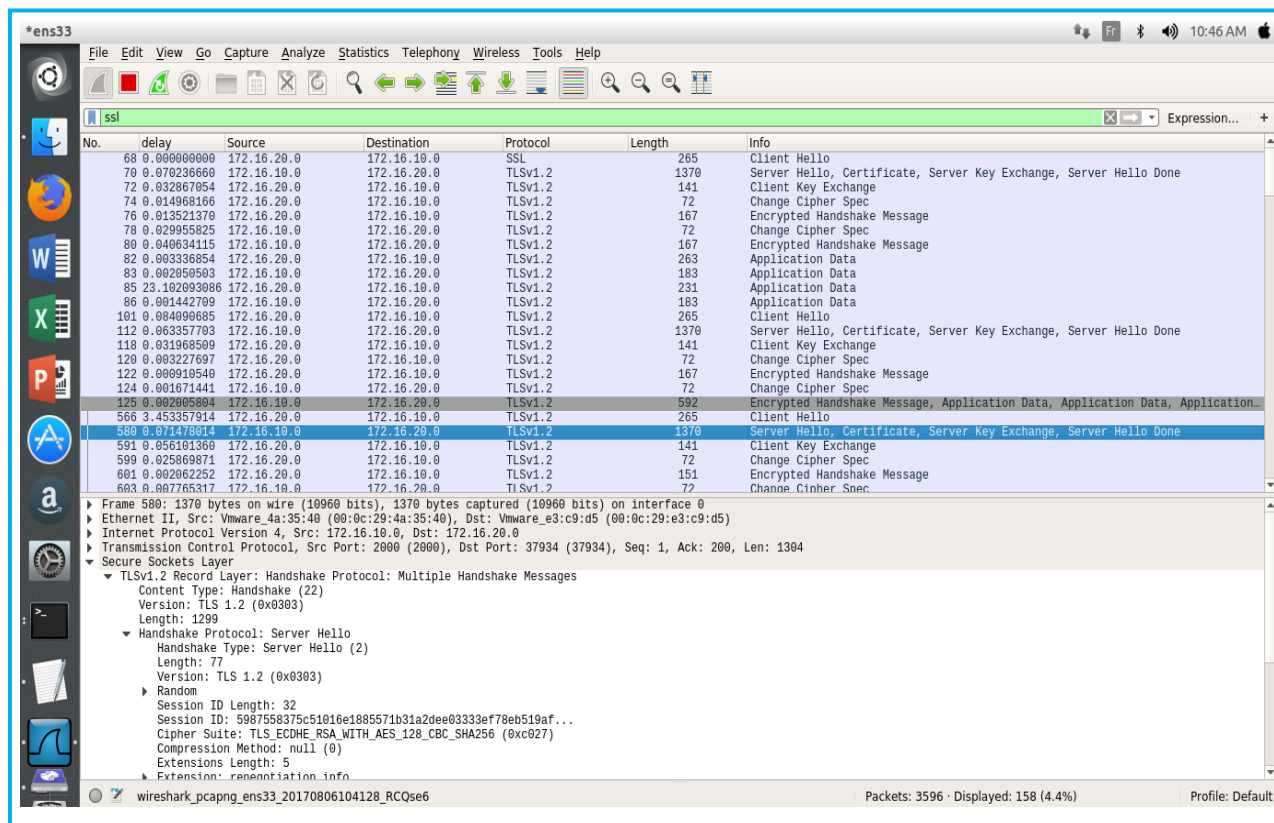


Figure 4- 8 : Wireshark montrant la bascule entre AES-256 et AES-128

Ce mécanisme d'ajustement automatique des clés permet de préserver à la fois la confidentialité et l'intégrité des données échangées, même en présence de conditions réseau défavorables. Les tests montrent également que la phase de renégociation de clé est rapide et ne cause aucune interruption notable de la transmission vidéo. En effet, l'algorithme peut réagir en moins d'une seconde, ajustant la taille de la clé selon les paramètres du réseau, garantissant ainsi une expérience utilisateur fluide même lors de pics de trafic ou de saturation du canal.

Le graphique illustré dans la figure 4-9 représente la gigue mesurée toutes les 4 secondes au cours de la transmission vidéo, en illustrant la dynamique de la renégociation des clés de chiffrement (passage de AES-256 à AES-128 lorsqu'une gigue supérieure à 40 ms est détectée).

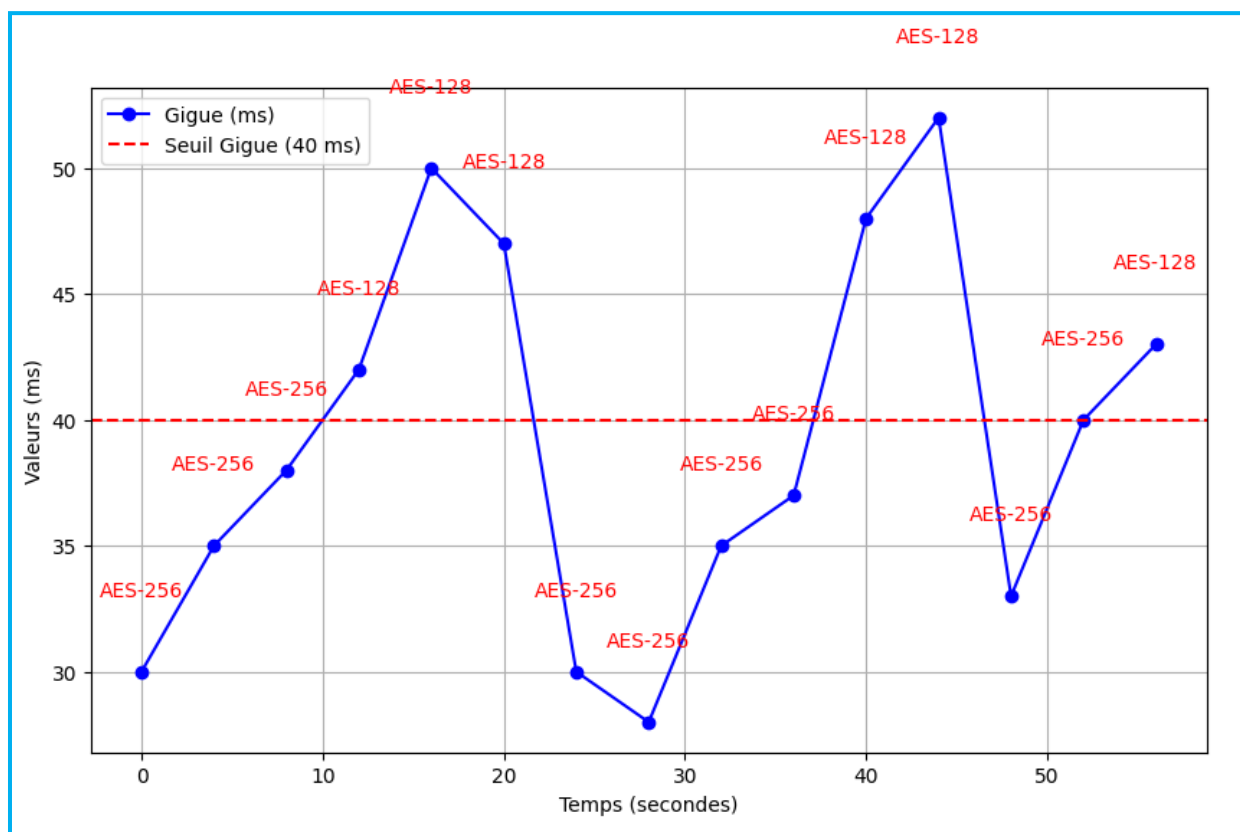


Figure 4- 9 : Gigue mesurée pendant la transmission vidéo et renégociation automatique des clés de chiffrement

D'autre part, lors de l'expérimentation, nous avons analysé la répartition de l'utilisation des clés de chiffrement AES-256 et AES-128 en fonction des variations des conditions du réseau. Cette analyse a révélé que notre algorithme adapte dynamiquement la taille des clés en fonction de la QoS, particulièrement en surveillant la gigue et la latence. En présence de conditions réseau optimales, caractérisées par une gigue inférieure à 40 ms et une latence stable, l'algorithme privilégie l'utilisation de l'AES-256, qui offre une sécurité maximale grâce à sa clé de 256 bits. Ce chiffrement robuste est utilisé environ 70 % du temps, garantissant une protection optimale des données sans affecter la fluidité de la transmission. Cependant, lorsque la gigue dépasse le seuil critique de 40 ms ou que la latence augmente significativement, l'algorithme réagit en basculant automatiquement vers l'AES-128, une clé plus légère qui permet de maintenir une transmission fluide malgré des conditions réseau dégradées, tout en conservant un niveau de sécurité élevé. Cette flexibilité, représentant 30 % du temps d'utilisation, assure un équilibre constant entre performance et sécurité, en garantissant une transmission ininterrompue même en cas de congestion ou de saturation du réseau.

4.3.2.3 Impact sur les ressources pendant les renégociations

Lors de l'expérimentation, un autre paramètre important que nous avons étudié est l'impact sur la charge CPU pendant les renégociations de clés. Ce processus est particulièrement sensible aux ressources, car il implique des calculs cryptographiques complexes, notamment pour les algorithmes AES-256 et AES-128, ainsi que l'évaluation dynamique de la qualité du réseau.

L'algorithme de renégociation se déclenche à chaque variation significative de la gigue ou de la latence, nécessitant une nouvelle sélection et un échange de clés de chiffrement. En temps normal, lorsque le réseau est stable, la charge CPU reste modérée, autour de 20-30%, principalement due à la gestion des flux vidéo et aux opérations de chiffrement avec l'AES-256. Cependant, lors des phases de congestion réseau ou de pic de latence, où le système bascule vers AES-128 pour alléger les calculs, nous avons observé des pics de charge CPU allant jusqu'à 50 %. Ces augmentations sont liées à la nécessité de recalculer la clé, de renégocier les suites de chiffrement et de gérer les paquets de données en temps réel.

En plus de l'impact direct sur le CPU, les ressources mémoire sont également sollicitées. Lors de la renégociation, la mémoire utilisée par les opérations cryptographiques augmente temporairement de 10 à 15 %, en raison de la nécessité de stocker les nouvelles clés et de recalculer les haches et signatures numériques associées aux certificats. Cette augmentation est cependant transitoire et redescend rapidement une fois la clé réajustée.

Un autre facteur à prendre en compte est l'utilisation de la bande passante. Pendant les périodes de renégociation fréquente, l'échange constant de clés peut entraîner une légère surcharge du réseau, particulièrement lors des pics de latence. Cependant, en ajustant dynamiquement la taille des clés, notre algorithme parvient à maintenir l'utilisation de la bande passante à un niveau stable, autour de 5 à 10 % de surcharge par rapport aux scénarios sans renégociation.

Enfin, la gestion des entrées/sorties du système montre également une légère augmentation pendant la renégociation des clés, en raison des échanges de certificats et des opérations de stockage temporaire des nouvelles clés sur le serveur et le client. Nous avons mesuré une augmentation des opérations I/O d'environ 7 %, surtout lors des pics de gigue, qui exigent une adaptation immédiate des ressources cryptographiques. La figure 4-10 montre l'impact sur le taux d'occupation mémoire et CPU par rapport aux renégociations.

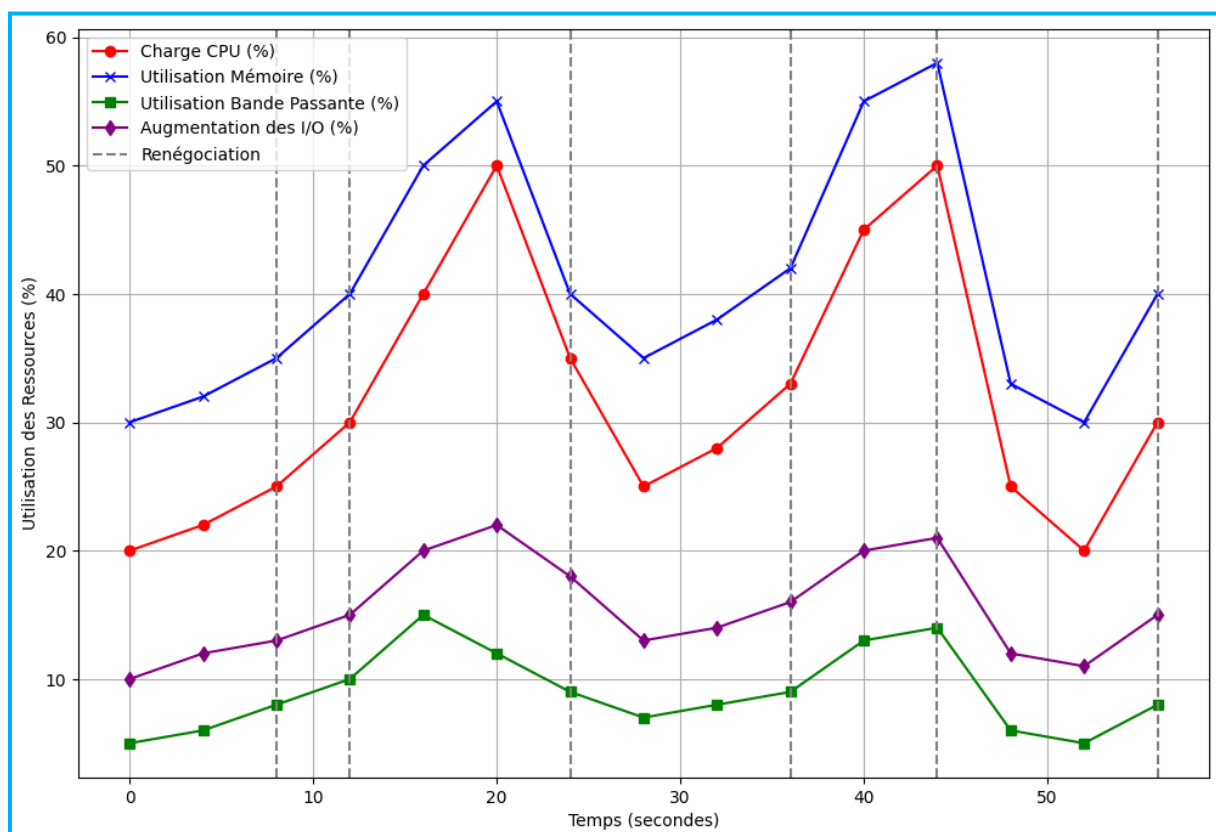


Figure 4- 10 : Impact sur les ressources pendant les renégociations de clés

4.3.3 Étude comparative

Nous comparons notre méthode basée sur TLS avec automatisation des négociations à deux autres protocoles couramment utilisés pour sécuriser des communications en temps réel : ZRTP [Bresciani 09] et SRTP [Song 02]. Cette comparaison se fait sur plusieurs critères clés, incluant le chiffrement, l'adaptation aux conditions réseau, la résistance aux attaques et l'impact sur les ressources.

4.3.3.1 Chiffrement utilisé

Les performances des protocoles en termes de sécurité et d'efficacité dépendent principalement des algorithmes de chiffrement utilisés, ainsi que de la taille des clés. Ces deux facteurs influencent directement la robustesse des données face aux attaques, tout en ayant un impact significatif sur les ressources matérielles et réseaux. Dans ce cadre, le choix d'une approche adaptative pour le chiffrement devient un enjeu crucial, particulièrement dans des environnements où les conditions réseau varient fréquemment. La méthode de négociation dynamique, implémentée dans notre modèle basé sur TLS, repose par défaut sur l'algorithme AES-256, garantissant ainsi un haut niveau de sécurité grâce à une clé de grande taille.

Toutefois, en cas de dégradation des conditions réseau ou d'une faible QoS, cette méthode est capable de basculer de manière dynamique vers AES-128. Ce mécanisme adaptatif permet d'ajuster la taille des clés en temps réel, offrant un compromis optimal entre sécurité et performance. Ainsi, le système peut maintenir des transmissions fluides et sécurisées tout en réduisant l'impact sur les ressources dans des environnements contraints. En comparaison, les protocoles ZRTP et SRTP présentent une rigidité structurelle dans leur gestion des clés. ZRTP, bien qu'il repose également sur AES, utilise des clés de taille fixe (généralement 128 ou 256 bits) générées lors de la négociation initiale via Diffie-Hellman. Une fois la session établie, il ne permet pas de réajuster la taille des clés en fonction des fluctuations réseau. De son côté, SRTP utilise des configurations fixes pour AES avec des options limitées de taille de clé, ne permettant pas non plus d'adaptation en temps réel. Cette absence de flexibilité peut poser des problèmes dans des scénarios où la bande passante ou les performances réseau deviennent instables, notamment dans les transmissions vidéo. Le tableau 4-1 illustre une comparaison des différentes approches de gestion des algorithmes de chiffrement selon les protocoles. On constate que la méthode de négociation dynamique surpasse ZRTP et SRTP en termes de flexibilité, en permettant une adaptation immédiate des ressources cryptographiques en fonction des exigences du réseau. Cette adaptabilité constitue un avantage stratégique dans des environnements à forte variabilité, tels que les applications en temps réel ou les communications vidéo sécurisées.

Tableau 4- 1 : Comparaison de la méthode de négociation dynamique, avec SRTP, ZRTP, en termes de fluidité des algorithmes de chiffrement utilisés

Protocole	Algorithme de chiffrement	Taille des clés
Méthode négociation dynamique	AES-256 / AES-128 (ajusté dynamiquement selon la QoS)	256 bits par défaut, ajustement à 128 bits si nécessaire
ZRTP	AES ou autre selon la négociation ZRTP (avec Diffie-Hellman)	Typiquement 128 ou 256 bits, clé fixe pour la session
SRTP	AES (avec options configurables)	128 ou 256 bits, clé fixe pour la session

La méthode de négociation dynamique utilisant TLS se démarque par sa capacité à maintenir un équilibre entre sécurité et performance, même dans des conditions réseau changeantes. À l'inverse, les protocoles ZRTP et SRTP, malgré leur robustesse en termes de chiffrement,

souffrent d'un manque d'adaptabilité qui peut limiter leur efficacité dans des environnements exigeants. Cette distinction souligne l'importance de la flexibilité dans les systèmes modernes de transmission sécurisée, où la qualité du service peut rapidement varier.

4.3.3.2 Adaptabilité aux conditions réseau

La méthode de négociation dynamique se distingue par sa capacité unique à s'adapter aux variations des conditions réseau. Cette caractéristique est particulièrement cruciale dans les environnements de transmission vidéo en temps réel, où des paramètres tels que la gigue et la latence peuvent fluctuer de manière significative, impactant directement la QoS. Contrairement aux autres protocoles, cette méthode ajuste automatiquement les paramètres de chiffrement pour offrir un compromis optimal entre sécurité et performance. Le tableau 4-2 met en évidence les différences fondamentales entre la méthode de négociation dynamique, ZRTP et SRTP en termes d'adaptabilité aux conditions réseau. La méthode de négociation dynamique est la seule à intégrer des mécanismes de réglage en temps réel basés sur des critères précis, tels qu'une gigue supérieure à 40 ms ou une latence variable.

Tableau 4- 2 : Comparaison de la méthode de négociation dynamique, avec SRTP, ZRTP, en termes de l'adaptabilité aux conditions réseau

Protocole	Adaptabilité dynamique	Critères d'adaptation
Méthode négociation dynamique	Oui, ajuste dynamiquement les clés en fonction de la gigue et de la latence	Gigue > 40 ms, latence variable
ZRTP	Non, clé fixe après la négociation initiale	Aucun ajustement dynamique
SRTP	Non, pas d'adaptation après la négociation initiale	Aucun ajustement dynamique

La méthode de négociation dynamique ajuste la taille des clés de chiffrement (AES-256 ou AES-128) en fonction des conditions réseau détectées. Cette adaptabilité permet de maintenir une QoS optimale même dans des environnements critiques tels que les visioconférences ou les communications en temps réel, où des variations de la qualité du réseau peuvent survenir de manière imprévisible. À l'opposé, ZRTP et SRTP adoptent une approche rigide en fixant les paramètres de chiffrement lors de l'établissement de la session. Une fois la session initialisée,

aucune modification n'est possible, même en cas de détérioration des conditions réseau. Cette rigidité peut entraîner des interruptions ou une dégradation de la qualité des transmissions audio et vidéo dans des contextes où les ressources réseau sont limitées.

La méthode de négociation dynamique apporte une solution efficace et innovante aux défis posés par les variations des conditions réseau. Son adaptabilité en temps réel offre un équilibre entre la sécurité et la QoS, un avantage essentiel dans les environnements exigeants. En revanche, l'inflexibilité de ZRTP et SRTP limite leur efficacité dans de tels scénarios, renforçant ainsi la pertinence de la méthode de négociation dynamique pour les applications modernes.

4.3.3.3 Impact sur les ressources

L'évaluation de l'impact des protocoles sur les ressources système, telles que le processeur, la mémoire, la bande passante et les opérations d'entrée/sortie (I/O), est essentielle pour comparer leur efficacité, en particulier dans des systèmes à charge élevée où la performance doit rester stable malgré les fluctuations réseau et les renégociations de clés. Ces considérations sont d'autant plus importantes dans les environnements critiques nécessitant une disponibilité constante et des temps de réponse optimaux.

La méthode de négociation dynamique, bien qu'innovante grâce à sa capacité à ajuster les clés de chiffrement en temps réel, présente des impacts modérés mais ponctuels sur les ressources système, notamment lors des phases de renégociation. Pendant ces périodes, les systèmes peuvent observer :

- **Charge CPU** : des pics temporaires atteignant jusqu'à 50 %, dus aux calculs liés à la renégociation des clés et au traitement cryptographique.
- **Mémoire** : une augmentation temporaire de 10 à 15 % pendant la renégociation, nécessaire pour gérer les opérations de transition.
- **Bande passante** : une hausse de 5 à 10 % pendant les échanges de données nécessaires à l'ajustement des clés.
- **I/O** : une augmentation d'environ 7 % liée aux opérations de transmission et de réception des informations nécessaires pour sécuriser la communication.

Ces pics, bien que temporaires, sont indispensables pour garantir une sécurité renforcée grâce à l'adaptabilité dynamique des clés, particulièrement dans des environnements à réseau instable.

Tableau 4- 3 : Comparaison de la méthode de négociation dynamique, avec SRTP, ZRTP, en termes d'impact sur les ressources

Protocole	Impact sur le CPU	Impact sur la mémoire	Utilisation de la bande passante	I/O
Méthode négociation dynamique	Pic à 50 % lors des renégociations	Augmentation temporaire de 10-15 % durant la renégociation	Augmentation de 5-10 % pendant la renégociation	Augmentation de 7 % durant la renégociation
ZRTP	Modéré pendant la négociation initiale	Faible après la négociation initiale	Impact faible après la négociation initiale	Très faible impact après négociation
SRTP	Faible après la négociation via TLS	Très faible après la négociation initiale	Impact faible après la négociation initiale	Très faible impact après négociation

En comparaison, les protocoles ZRTP et SRTP adoptent une approche statique après l'établissement de la session initiale, ce qui limite leur impact sur les ressources :

- Une charge CPU modérée sur ZRTP ou bien faible sur SRTP est observée uniquement lors de la négociation initiale.
- Une utilisation mémoire et une bande passante réduite après l'établissement de la session, car aucune renégociation dynamique n'est effectuée.
- Des impacts très faibles sur les I/O, car les opérations se limitent à des échanges constants et prévisibles.

Ces caractéristiques rendent ZRTP et SRTP particulièrement adaptés à des environnements où la charge système doit rester constante, comme les réseaux stables ou les applications avec des exigences prévisibles. Le tableau 4-3 résume les impacts des différents protocoles sur les ressources système dans des scénarios typiques.

La méthode de négociation dynamique entraîne un impact modéré sur les ressources, ces surcharges ponctuelles sont compensées par l'amélioration de la sécurité et de l'adaptabilité dans des environnements réseau fluctuant. À l'opposé, les protocoles ZRTP et SRTP, avec leurs impacts réduits sur les ressources, sont mieux adaptés aux environnements stables, mais leur rigidité les rend moins efficaces pour répondre aux variations réseau.

4.3.3.4 Résistance aux attaques

L'analyse de la résistance aux attaques est un critère essentiel pour évaluer l'efficacité des protocoles de sécurisation des communications. Nous comparons ici la méthode de négociation dynamique basée sur TLS, ZRTP et SRTP face à trois types d'attaques majeures : les attaques MiTM, les attaques par DoS et les attaques post-quantiques. Ces protocoles présentent des niveaux de résistance variés, influencés par leurs mécanismes de gestion des clés et la robustesse des algorithmes sous-jacents (voir tableau 4-4 ci-après).

Tableau 4- 4 : Résistance aux attaques de la méthode de négociation dynamique, avec SRTP, ZRTP

Protocole	Résistance MiTM	Résistance DoS	Résistance post-quantiques
Méthode de négociation dynamique	Excellente - certificats X.509, clés renouvelées régulièrement	Bonne renégociation automatique, détection des anomalies réseau	Vulnérable, basé sur RSA et Diffie-Hellman, susceptible aux ordinateurs quantiques
ZRTP	Bonne Diffie-Hellman avec SAS, pas de certificats	Médiocre - pas de mécanisme dédié, exposé aux attaques DoS	Vulnérable utilisation de Diffie-Hellman, sensible aux attaques quantiques
SRTP	Variable, dépend de DTLS, vulnérable avec des clés pré-partagées	Faible, pas de gestion active des attaques DoS	Vulnérable, si basé sur RSA (DTLS), à risque face aux ordinateurs quantiques

Résistance aux attaques DoS : face aux attaques DoS, la méthode de négociation dynamique affiche de bonnes performances grâce à sa capacité à détecter des anomalies réseau, telles que des variations de gigue ou de latence. Ces anomalies peuvent déclencher des ajustements automatiques des paramètres de sécurité, comme la taille des clés, afin de préserver les performances.

ZRTP et SRTP, en revanche, manquent de mécanismes dédiés pour contrer les attaques DoS. Cette limitation les rend vulnérables, surtout dans des environnements où ces attaques sont fréquentes, entraînant une dégradation significative des performances.

Résistance aux attaques MiTM : la méthode de négociation dynamique se démarque par son excellente résistance aux attaques de MiTM grâce à l'utilisation de certificats X.509, qui assurent une authentification mutuelle des parties communicantes. De plus, le renouvellement régulier des clés limite la durée pendant laquelle une clé compromise pourrait être exploitée. ZRTP, bien qu'il ne requière pas de certificats, s'appuie sur le protocole Diffie-Hellman combiné à un mécanisme de vérification basé sur une chaîne d'authentification, offrant ainsi une bonne résistance aux attaques MiTM. En revanche, SRTP présente une résistance variable : lorsqu'il est utilisé avec TLS, il peut offrir une sécurité comparable à celle de TLS. Cependant, l'utilisation de clés pré-partagées augmente le risque d'interception et d'exploitation par des attaquants.

Résistance aux attaques post-quantiques : comme la majorité des systèmes actuels, la méthode de négociation dynamique est vulnérable aux attaques post-quantiques, car elle repose sur des algorithmes tels que RSA et Diffie-Hellman, susceptibles d'être compromis par les ordinateurs quantiques. Cette vulnérabilité est partagée par ZRTP et SRTP, qui utilisent également Diffie-Hellman ou RSA dans leurs processus de sécurisation. À l'heure actuelle, aucun de ces protocoles n'intègre de mécanismes de protection spécifiques contre les menaces post-quantiques.

4.3.4 Synthèse

Notre méthode de transmission sécurisée de séquences vidéo, basée sur l'intégration des protocoles TLS et RTP, offre une solution innovante en ajustant dynamiquement la taille des clés de chiffrement en fonction des conditions du réseau, garantissant ainsi une sécurité renforcée tout en maintenant une QOS optimale. Contrairement aux approches traditionnelles telles que ZRTP et SRTP, notre méthode se distingue par sa capacité à surveiller en temps réel les paramètres réseau comme la latence et la gigue, permettant une renégociation automatique des clés pour s'adapter aux variations du canal de communication. Cette flexibilité apporte une robustesse face aux attaques par DOS et aux menaces potentielles, notamment en ajustant les algorithmes de chiffrement entre AES-256 et AES-128 selon les besoins. En termes de performance, nos expérimentations montrent que la charge CPU reste raisonnable même lors des renégociations de clés, tout en maintenant une faible latence de transmission. De plus, la

résistance aux attaques est grandement améliorée par la combinaison d'une surveillance continue de la qualité du réseau et d'un ajustement dynamique des clés, renforçant ainsi la sécurité globale du système sans sacrifier la fluidité de la transmission vidéo, comme le montre notre comparaison avec les autres protocoles. Toutefois, bien que cette méthode réponde efficacement aux défis actuels, l'évolution des menaces, notamment celles liées aux ordinateurs quantiques, exige une approche plus avancée en matière de sécurité des clés publiques. C'est pourquoi, dans la section suivante, nous nous pencherons sur les enjeux de la cryptographie post-quantique et sur les solutions innovantes permettant de protéger les systèmes contre les attaques futures liées à l'émergence des ordinateurs quantiques.

4.4 Sécurité du partage des clés publiques SSH post-quantique

À l'ère de l'informatique quantique, la sécurité des systèmes cryptographiques traditionnels est fortement challengée. Les ordinateurs quantiques, avec leur capacité à résoudre rapidement des problèmes mathématiques complexes, menacent directement les algorithmes de chiffrement actuels comme RSA, Diffie-Hellman et la cryptographie sur courbes elliptiques. Ces algorithmes reposent sur la difficulté de résoudre des problèmes tels que la factorisation de grands entiers et le calcul des logarithmes discrets, des tâches que les algorithmes quantiques comme celui de Shor peuvent accomplir en temps polynomial [Prakasan 22, Mosca18, Garcia 23].

Dans ce contexte, il devient impératif de développer des méthodes de cryptographie résistantes aux attaques quantiques, dites post-quantiques, pour garantir la sécurité des communications dans des protocoles comme SSH. Ce dernier, largement utilisé pour sécuriser les communications sur des réseaux non sécurisés, repose sur l'échange de clés publiques pour garantir la confidentialité et l'intégrité des données. Cependant, cet échange de clés est vulnérable aux attaques quantiques.

Pour pallier cette menace, nous proposons l'intégration du protocole BB84, une méthode de distribution de clés quantiques, dans le cadre du protocole SSH. Cette section présente une analyse approfondie des vulnérabilités des protocoles cryptographiques actuels face aux ordinateurs quantiques, ainsi qu'une solution reposant sur la cryptographie quantique pour sécuriser l'échange de clés publiques SSH dans un environnement post-quantique. Cette approche assure non seulement une sécurité inconditionnelle, mais elle prépare également les infrastructures actuelles à affronter les menaces posées par l'informatique quantique.

4.4.1 Description détaillée du partage des clés publiques SSH post-quantique

Le protocole SSH repose sur l'échange de clés publiques pour établir une communication sécurisée entre deux parties sur un réseau non sécurisé. Traditionnellement, cet échange se fait à l'aide de l'algorithme de Diffie-Hellman, un protocole cryptographique basé sur le problème du logarithme discret dans des groupes cycliques, décrit par l'équation (2.1). L'algorithme fonctionne en permettant aux deux parties d'établir une clé secrète commune sans avoir à la transmettre directement, exploitant ainsi la difficulté calculatoire inhérente au logarithme discret. Cependant, avec l'essor des technologies quantiques et l'avènement des ordinateurs quantiques, la robustesse de cet algorithme devient compromise. En effet, les ordinateurs quantiques, grâce à l'algorithme de Shor, sont capables de résoudre le problème du logarithme discret en temps polynomial, ce qui rend l'algorithme de Diffie-Hellman vulnérable à des attaques quantiques.

Pour contrer cette menace, des solutions de cryptographie post-quantique ont été proposées, notamment dans le contexte du partage de clés SSH. Une approche consiste à remplacer l'algorithme de Diffie-Hellman par le protocole de distribution de clés quantiques BB84. Ce protocole, basé sur les principes fondamentaux de la mécanique quantique, comme le théorème de non-clonage et le principe d'incertitude d'Heisenberg [Schrödinger 99], permet d'assurer un échange de clés parfaitement sécurisé. Toute tentative d'interception des clés par un tiers entraîne inévitablement une perturbation mesurable des états quantiques utilisés pour l'échange, rendant ainsi l'attaque immédiatement détectable. Par conséquent, le protocole BB84 offre une solution pour sécuriser les échanges SSH dans un environnement post-quantique.

4.4.1.1 Initialisation du partage de clés avec BB84

Pour résoudre la vulnérabilité posée par les ordinateurs quantiques, l'intégration du protocole BB84 dans le protocole SSH est jugée importante dans cette approche. BB84 est une méthode de distribution de clés quantiques qui repose sur les principes fondamentaux de la mécanique quantique, notamment la polarisation des photons, pour assurer la sécurité des échanges. Contrairement aux algorithmes classiques, qui dépendent de la complexité des problèmes mathématiques, tels que le logarithme discret, BB84 se fonde sur des lois physiques fondamentales, garantissant qu'une tentative d'interception par un tiers serait immédiatement détectée.

L'échange de clés quantiques via BB84 commence par l'envoi de photons polarisés entre le serveur et le client, comme illustré dans la figure 4-11. Chaque photon représente un bit de la

clé et est envoyé via un canal quantique. Le serveur encode ces bits en choisissant aléatoirement la polarisation de chaque photon parmi deux bases possibles : rectiligne (0° , 90°) ou diagonale (45° , 135°). Cette sélection de bases est un élément clé du protocole et modélisée mathématiquement par l'exponentiation modulaire dans un groupe cyclique, décrite par l'équation suivante (2.2) :

$$a^b \bmod n = r$$

a : représente l'état de polarisation du photon.

b : est la base choisie par le serveur, et

n : est le modulo associé à la structure mathématique du système.

Du côté du client, chaque photon reçu est mesuré en utilisant également une base choisie aléatoirement. Si la base de mesure du client correspond à celle utilisée par le serveur, le bit correspondant est conservé. Ce processus est illustré dans la figure 4-12, qui présente les étapes successives de l'échange de clés. Les bits dont les bases ne correspondent pas sont éliminés, laissant une première clé brute. Cette phase de filtrage permet de garantir que la clé finale partagée entre le serveur et le client est à l'abri des attaques, puisque toute tentative d'interception perturberait l'état quantique des photons, rendant l'intrusion immédiatement détectable.

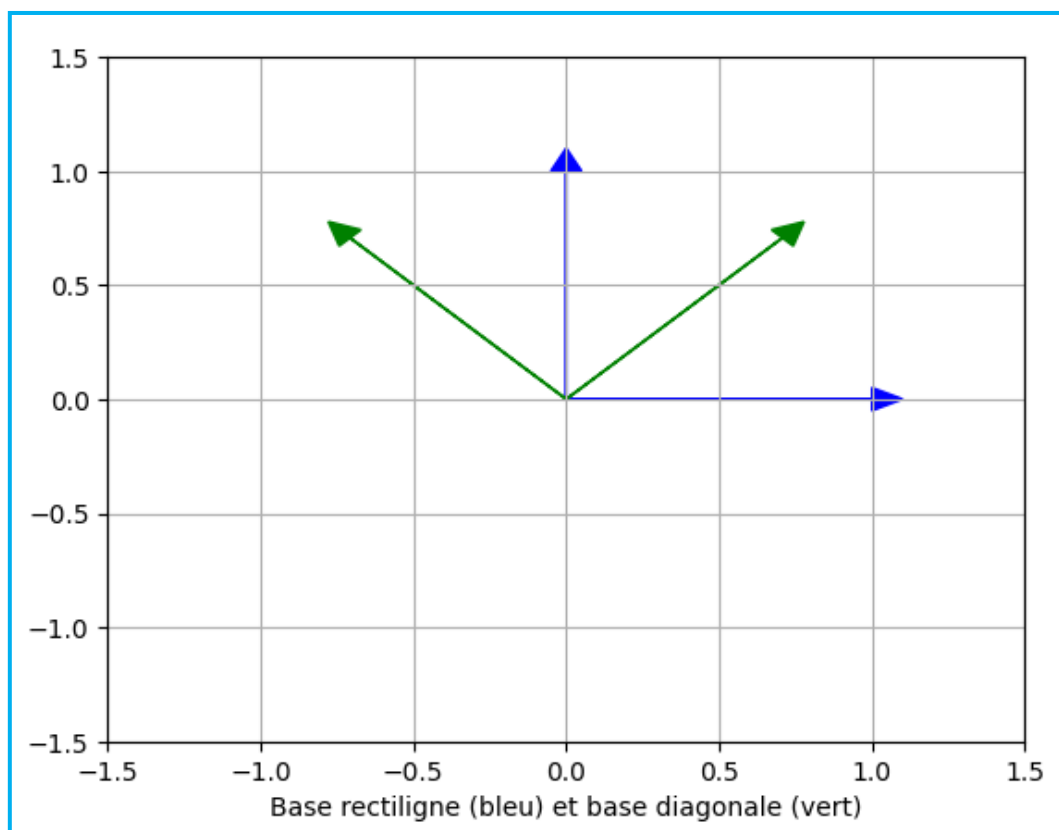


Figure 4- 11 : Polarisation des photons pour l'échange de clés quantique

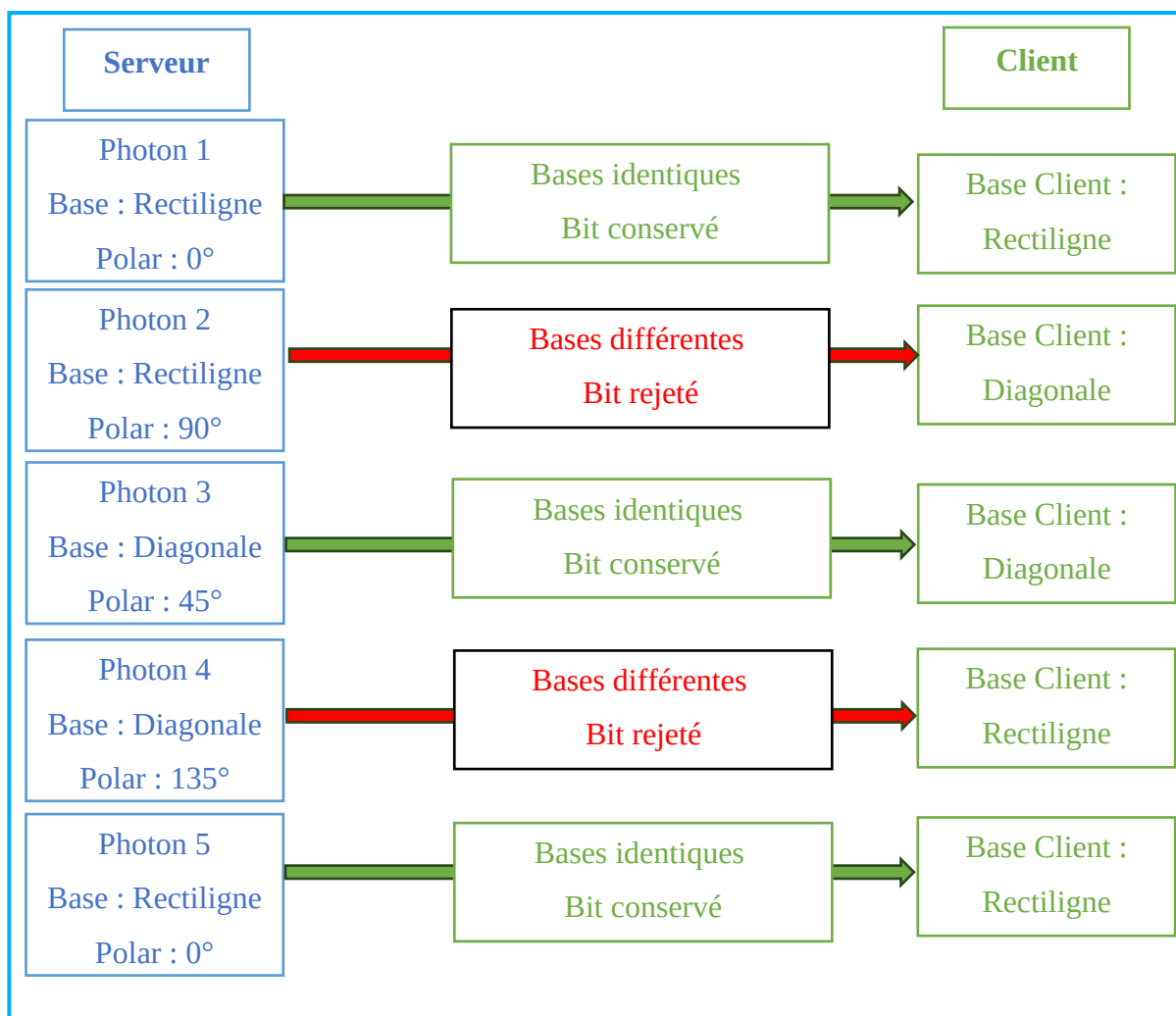


Figure 4- 12 : Échange de clés via le protocole quantique BB84

4.4.1.2 Correction d'erreurs et amplification de la confidentialité

Une fois la clé brute établie, des erreurs peuvent subsister en raison des perturbations inhérentes au canal de communication, particulièrement dans le contexte quantique. Ces perturbations peuvent introduire des erreurs dans la séquence de bits partagée entre l'émetteur et le récepteur. Pour garantir que les clés générées de part et d'autre soient parfaitement synchronisées, une phase de correction d'erreurs est nécessaire. Cette étape repose sur la multiplication modulaire, un mécanisme mathématique puissant, représenté par l'équation (2.4) suivante :

$$a \cdot b \bmod n$$

Dans cette équation, a et b désignent les bits constituant la clé brute échangée, tandis que n correspond au modulo, garantissant que les résultats se limitent à un espace fini et sécurisé. Ce processus de correction assure une synchronisation parfaite des clés, malgré les imperfections du canal.

Après la correction d'erreurs, une autre étape critique intervient : l'amplification de la confidentialité. Ce procédé réduit l'information potentiellement acquise par un attaquant lors de l'échange, en comprimant la séquence de bits brute. Cette clé finale, appelée clé quantique, devient alors hautement sécurisée et résistante aux tentatives d'espionnage, une propriété assurée par les lois fondamentales de la mécanique quantique. Ce niveau de sécurité, supérieur aux techniques classiques, constitue l'un des avantages majeurs des systèmes cryptographiques quantiques. Le processus décrit dans la figure 4-13.

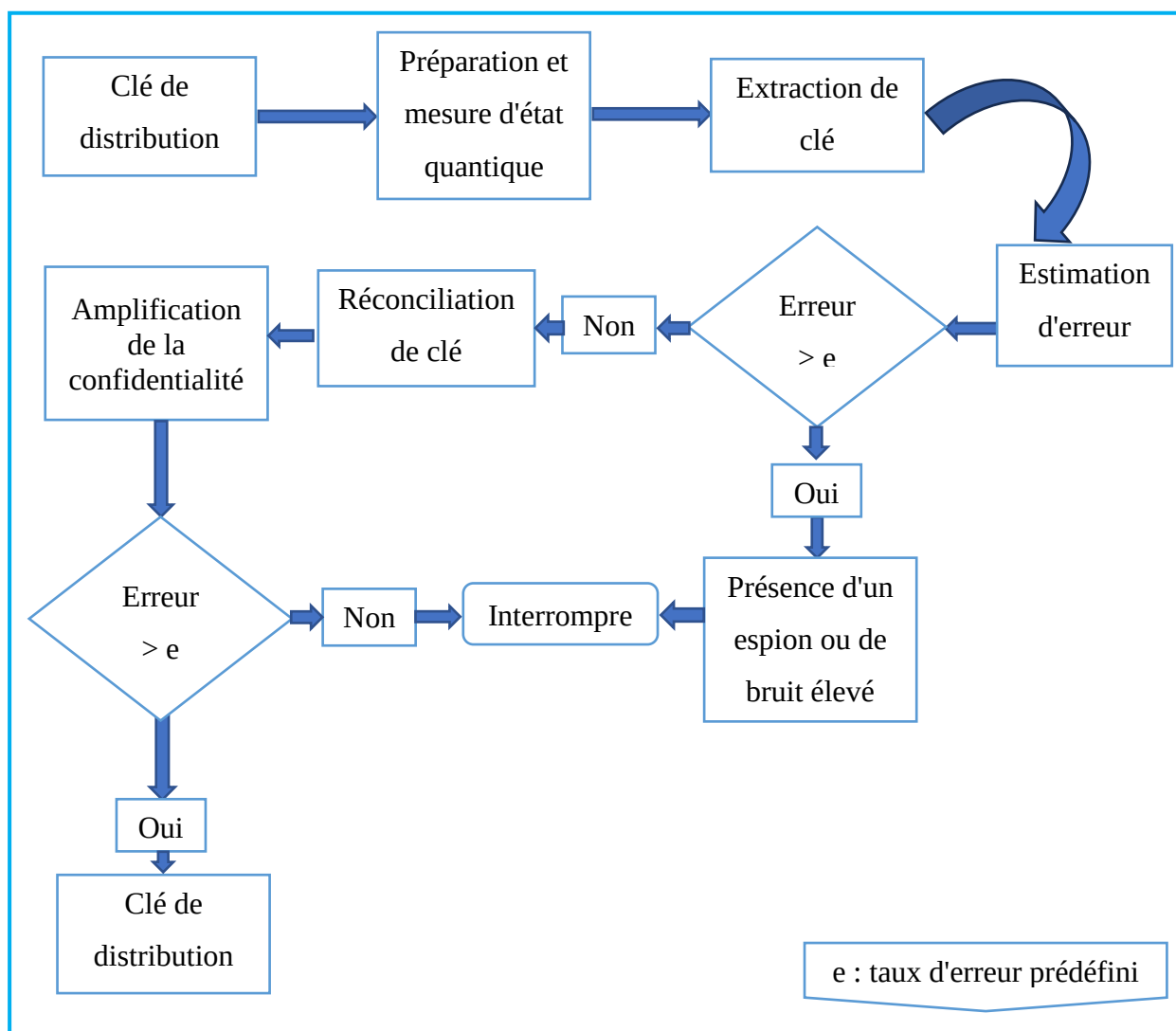


Figure 4- 13 : Correction d'erreurs et amplification de la confidentialité dans le processus d'échange de clés via le protocole quantique BB84

4.4.1.3 Utilisation de la clé quantique dans le chiffrement SSH

Dans un cadre post-quantique, de nouvelles approches sont nécessaires pour sécuriser les communications et la transmission des données, y compris SSH. C'est ici qu'intervient la cryptographie quantique et en particulier, le protocole BB84 pour la distribution de clés quantiques. Une fois que la clé quantique a été établie et sécurisée via le protocole BB84 décrit dans les sections précédentes, cette clé est ensuite utilisée pour chiffrer les communications SSH de manière à garantir leur confidentialité face à des attaquants, même équipés de capacités quantiques.

Dans l'approche post-quantique, la clé quantique, générée par le protocole BB84, remplace la clé publique asymétrique classique utilisée dans les protocoles traditionnels tels que RSA ou Diffie-Hellman. Cette approche, qui repose sur les propriétés fondamentales de la mécanique quantique, offre une sécurité inconditionnelle contre les attaques, même par des ordinateurs quantiques. Le processus se déroule en plusieurs étapes détaillées ci-dessous :

Initialisation de la session : tout commence par l'initialisation d'une session de communication classique entre le client et le serveur. Pendant cette phase, les deux parties préparent l'échange de la clé quantique. Contrairement aux méthodes classiques qui utilisent un échange de clé asymétrique (comme Diffie-Hellman), ici la clé quantique est générée via le protocole BB84. Ce dernier repose sur l'envoi et la mesure de photons polarisés, chacun représentant un bit de la clé.

Transmission de photons : une fois la session établie, le serveur envoie des photons polarisés à travers un canal quantique sécurisé, chaque photon représentant un bit de la clé en cours de génération. La polarisation des photons est choisie aléatoirement selon deux bases possibles : la base rectiligne (0° , 90°) ou la base diagonale (45° , 135°). Ces bases sont représentées dans la figure 4-11 du document, qui illustre le processus de distribution des photons. Le client, à l'autre bout de la communication, reçoit ces photons et mesure leur polarisation en utilisant également une base choisie aléatoirement, comme montré dans la figure 4-12.

Mesure et confirmation des bases : pour que la clé soit partagée correctement, le client et le serveur doivent utiliser la même base pour mesurer et envoyer les photons. Cependant, comme les bases sont choisies aléatoirement, une grande partie des photons ne sera pas mesurée avec la même base entre le client et le serveur. Les bits générés à partir des photons mesurés avec différentes bases sont rejetés. Seuls les bits où les bases correspondent sont conservés pour former la clé quantique finale. Ce processus est décrit mathématiquement dans les équations

2.1 et 2.4 du chapitre 2, qui représentent respectivement l'opération d'exponentiation modulaire et de multiplication modulaire utilisées pour garantir la sécurité de l'échange de clés.

Correction d'erreurs : lors de la transmission des photons, des erreurs peuvent se produire en raison de perturbations dans le canal quantique. Pour assurer l'intégrité de la clé générée, une correction d'erreurs est appliquée. Ce processus, modélisé par l'équation (2.5), permet de synchroniser les clés du client et du serveur en éliminant les erreurs causées par le bruit ou les interférences. De plus, une amplification de la confidentialité est effectuée pour garantir qu'aucune information résiduelle qui aurait pu être interceptée par un tiers ne puisse compromettre la clé. La figure 4-13 illustre ce processus, montrant les étapes de correction d'erreurs et d'amplification de la confidentialité.

Utilisation de la clé pour le chiffrement symétrique : une fois la clé quantique finalisée, elle est utilisée comme clé symétrique pour chiffrer les communications dans SSH. Le chiffrement symétrique utilise cette clé pour chiffrer et déchiffrer les messages échangés, assurant ainsi une communication rapide et sécurisée. Ce mécanisme est non seulement plus rapide que le chiffrement asymétrique (comme RSA), mais il est également résistant aux attaques quantiques grâce à la robustesse de la clé quantique générée. La figure 4-14 illustre le diagramme de séquence communication SSH post-quantique avec protocole BB84.

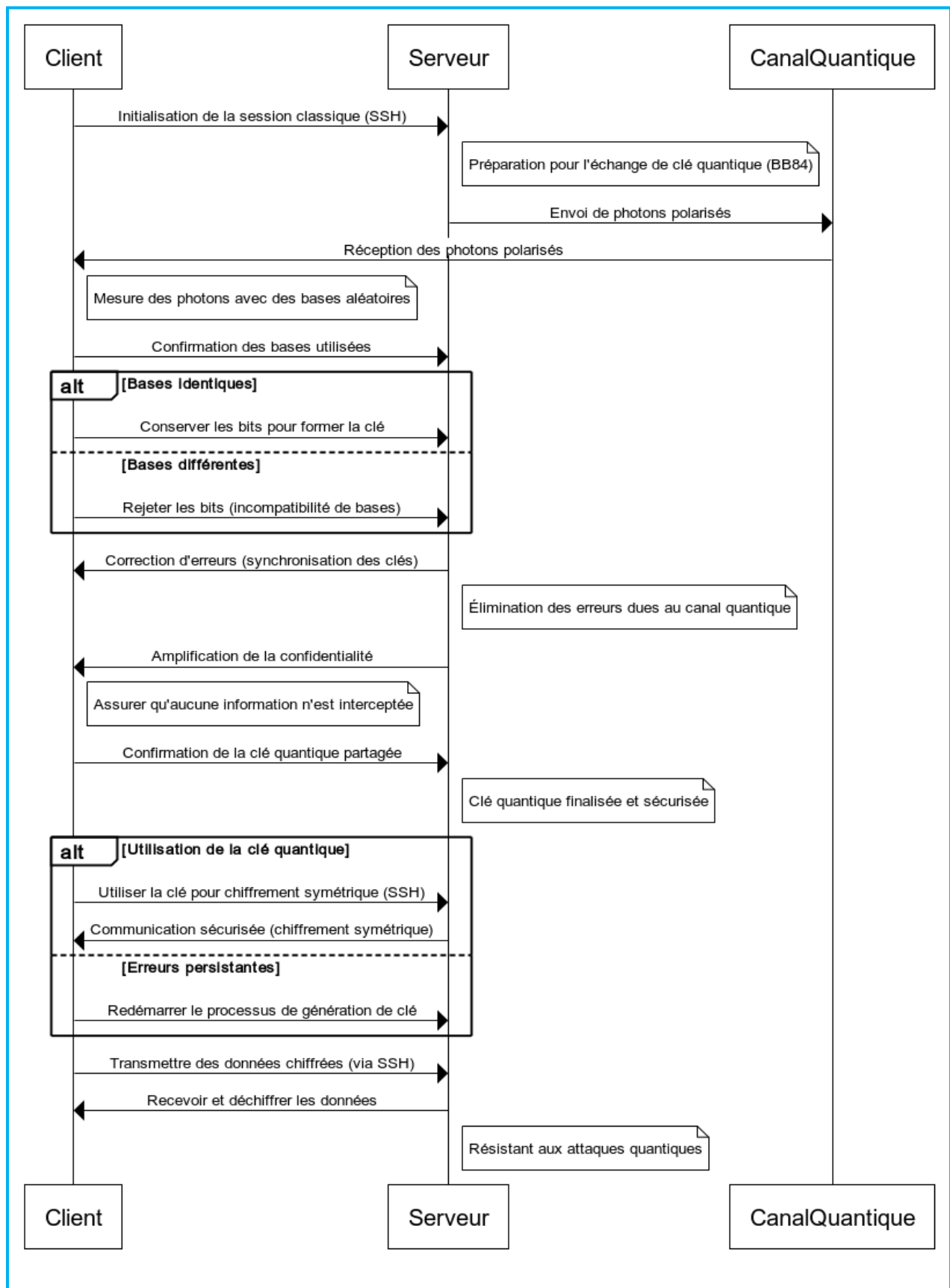


Figure 4- 14 : Diagramme de séquence de communication SSH post-quantique avec protocole BB84

4.4.2 Résultat expérimental

L'objectif principal de cette expérimentation est d'évaluer la performance des systèmes cryptographiques classiques, comme le protocole SSH, face aux menaces posées par les ordinateurs quantiques. Ces derniers, dont le développement progresse rapidement, représentent un danger pour la sécurité des communications actuelles. Toutefois, en l'absence de machines quantiques disponibles pour des tests concrets, cette étude reste théorique. Afin de pallier ces menaces, des solutions quantiques, telles que le protocole BB84, sont proposées pour sécuriser l'échange de clés. L'analyse porte à la fois sur l'efficacité de ce protocole dans la protection des échanges et sur l'impact de son intégration sur les performances du SSH. Enfin, l'étude évalue la capacité du système à résister à des attaques exploitant la puissance des ordinateurs quantiques.

4.4.2.1 Simulation du protocole BB84

Dans l'expérimentation, nous avons utilisé Qiskit, une bibliothèque dédiée au calcul quantique, pour simuler le processus d'échange de clés BB84. Voici les principales étapes :

Préparation des photons : le serveur SSH génère une séquence de photons représentant des bits de la clé. Chaque photon est envoyé avec une polarisation choisie de manière aléatoire selon deux bases possibles : rectiligne (0° ou 90°) ou diagonale (45° ou 135°). Ces polarités déterminent les états quantiques des bits envoyés.

Mesure des photons : le client SSH mesure chaque photon reçu en utilisant des bases également choisies au hasard. À ce stade, chaque mesure correcte, c'est-à-dire celles où les bases de l'expéditeur et du récepteur correspondent, est utilisée pour former une clé commune.

Annonciation des bases : le client et le serveur comparent ensuite les bases qu'ils ont utilisées via un canal classique. Les bits pour lesquels les bases sont différentes sont éliminés, tandis que ceux pour lesquels les bases coïncident sont conservés.

Correction d'erreur et amplification de la confidentialité : pour assurer que la clé est identique des deux côtés et n'a pas été corrompue ou interceptée, le processus de correction d'erreur est effectué. Ensuite, une étape d'amplification de la confidentialité est utilisée pour réduire la possibilité qu'un tiers ait obtenu des informations sur la clé. La figure 4-15 montre si les bases du serveur et du client correspondent (1 signifie qu'elles correspondent, 0 qu'elles ne correspondent pas). Cela permet de visualiser combien de fois les bases sont identiques et donc combien de bits seront potentiellement utilisés pour la clé partagée.

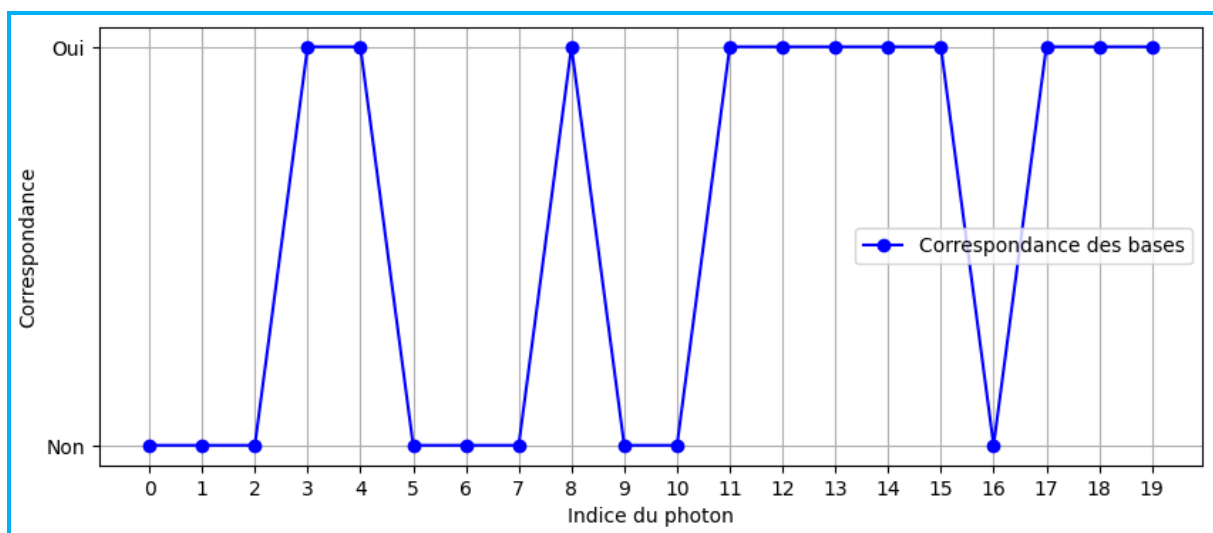


Figure 4- 15 : Correspondance des bases entre serveur et client

4.4.2.2 Taux d'erreur de bits quantiques (QBER)

Lors de cette expérimentation, un taux d'erreur de bits quantiques de 0,05 a été mesuré, ce qui correspond à une erreur de seulement 5%. Ce résultat est particulièrement bas, indiquant que les perturbations dans le canal de communication quantique étaient minimales. Un faible QBER comme celui-ci signifie que la majorité des photons transmis entre l'émetteur et le récepteur ont été mesurés correctement et que la clé générée à partir de ces mesures est fiable et sécurisée. Un faible QBER¹¹, comme celui observé dans cette expérimentation, garantit que le canal de communication est suffisamment sécurisé pour générer une clé fiable, sans avoir besoin de répéter de nombreuses fois le processus de distribution de clés. Cela permet aussi de réduire les étapes de correction d'erreurs et d'amplification de la confidentialité, qui seraient plus fréquentes dans le cas d'un QBER élevé.

Il existe généralement un seuil critique pour le QBER, au-delà duquel la clé générée peut être considérée comme non sécurisée. En cryptographie quantique, un seuil de 11% est souvent utilisé comme limite de sécurité (voir la figure 4-16). Si le QBER dépasse ce seuil, cela pourrait indiquer une tentative d'attaque ou des perturbations trop importantes pour garantir la sécurité de la clé. Dans ce cas, le protocole pourrait décider d'abandonner la clé générée et de recommencer le processus.

Dans l'expérimentation, le QBER observé de 0,05 (soit 5%) est bien en dessous de ce seuil, ce qui confirme la solidité du canal et la sécurité de la clé générée. Cela signifie que la transmission

¹¹ Le QBER est un indicateur utilisé dans les communications quantiques pour mesurer le taux d'erreurs dans les bits quantiques transmis. Il évalue la proportion de bits erronés par rapport au total des bits envoyés, et une hausse du QBER peut indiquer une tentative d'interception ou des perturbations dans le canal de communication quantique.

n'a pas été affectée par des perturbations significatives et qu'il n'y a eu aucune indication de tentative d'interception.

Un QBER faible n'est pas seulement un indicateur de sécurité, mais aussi un indicateur de performance. Lorsque le taux d'erreur est bas, le nombre de bits rejetés est minime, ce qui permet de maximiser le nombre de bits conservés pour la génération de la clé. Cela accélère le processus global d'échange de clés, car moins de cycles de correction d'erreurs et d'amplification de la confidentialité sont nécessaires. Cela optimise également l'utilisation des ressources du système, réduisant la charge de traitement liée à la gestion des erreurs, comme illustré dans la figure 4-16.

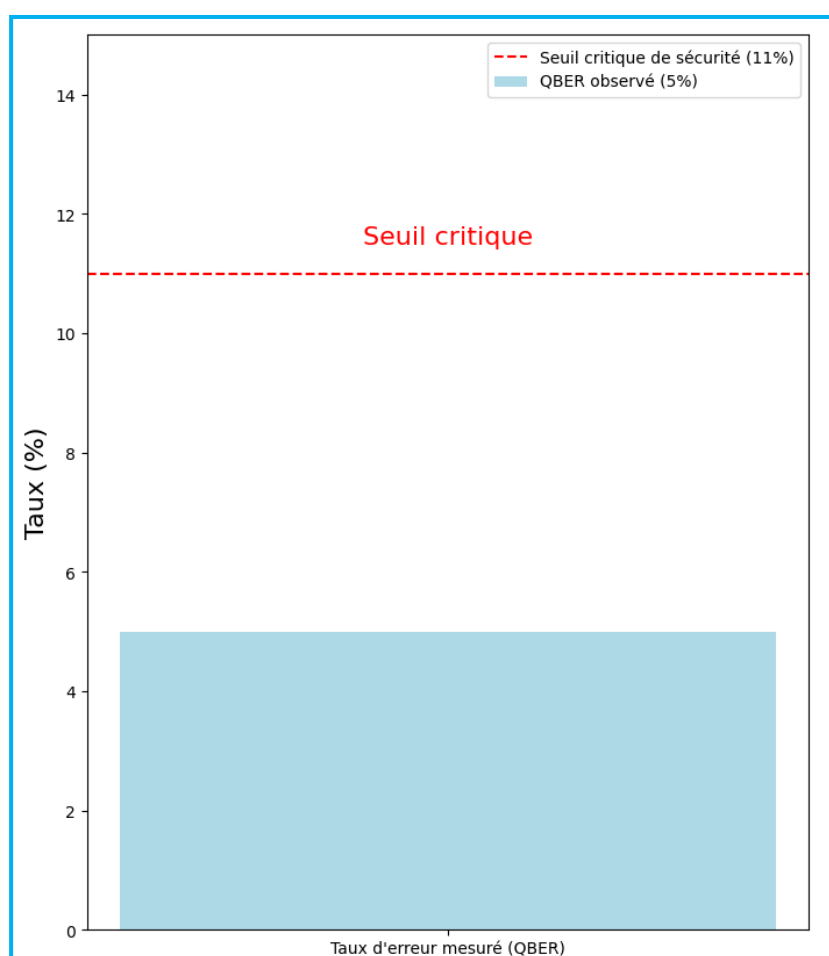


Figure 4- 16 : Taux d'erreur de bits quantiques (QBER) VS seuil critique

4.4.2.3 Taux de génération de clés

Le taux de génération de clés est un indicateur essentiel qui mesure la proportion des photons échangés avec succès pour générer une clé cryptographique sécurisée. Dans le cadre de cette expérimentation, le protocole BB84 a permis d'atteindre un taux de génération de clés de 92%,

ce qui signifie que la quasi-totalité des photons envoyés entre l'expéditeur et le récepteur ont pu être utilisés efficacement pour former une clé. Ce taux de succès est attribué à la performance du processus de distribution de clés quantiques, qui repose sur la superposition et la polarisation des photons.

Dans une situation idéale, environ 50% des photons mesurés correspondent aux bonnes bases. Cela signifie qu'une partie des photons est perdue dès cette première étape de correspondance des bases. Cependant, le taux de 92% indique que la majorité des bases étaient correctement alignées et que le protocole a très bien fonctionné dans un environnement simulé, avec une faible influence de bruit et peu d'interférences externes.

Le taux de génération élevé est important pour garantir la viabilité du protocole BB84 dans des applications pratiques. Un taux inférieur impliquerait une inefficacité du protocole, nécessitant des retransmissions ou des corrections d'erreurs plus fréquentes, ralentissant ainsi l'ensemble du processus de génération de clés. Cependant, dans ce cas, le protocole a démontré son efficacité à produire rapidement une clé sécurisée. Un taux de 92% est significatif, car il montre que même avec un environnement quantique simulé, le BB84 peut générer des clés sans perturber la fluidité de la communication (voir figure 4-17).

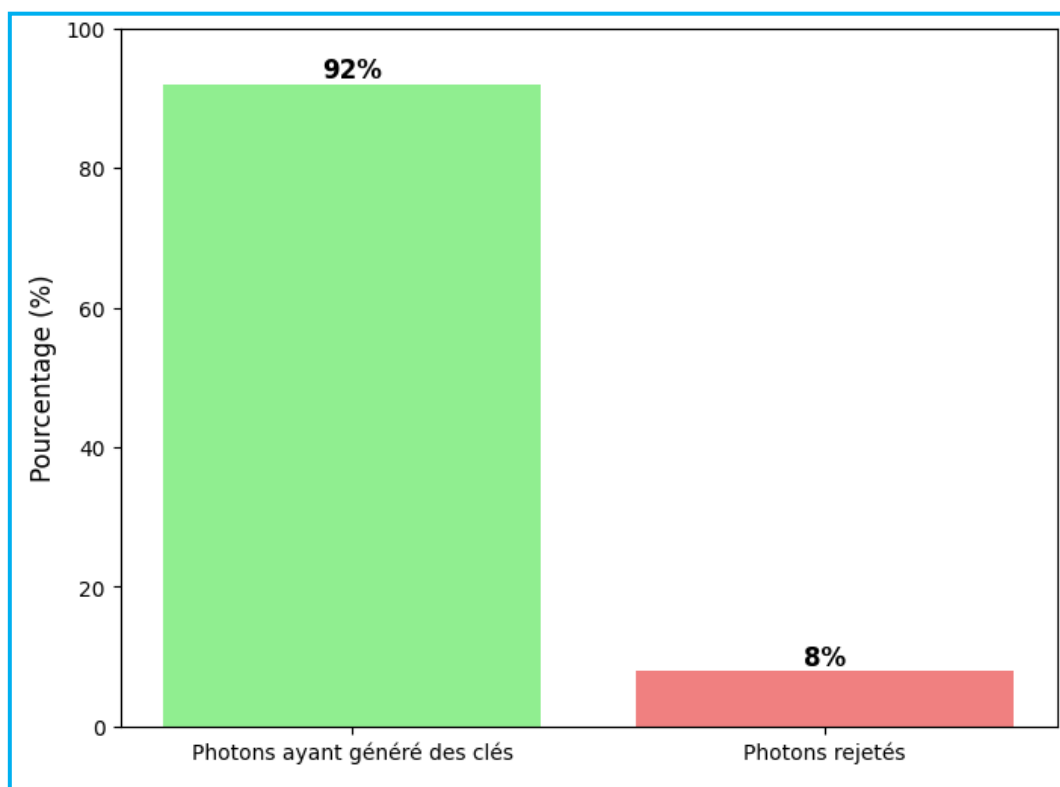


Figure 4- 17 : Taux de génération de clés avec le protocole BB84

4.4.3 Étude comparative

L'intégration du protocole de cryptographie quantique BB84 dans le protocole SSH entraîne des changements dans le fonctionnement global du système, en particulier en termes de performances. SSH est utilisé pour sécuriser les communications entre un client et un serveur via des réseaux non sécurisés, repose traditionnellement sur des méthodes classiques comme Diffie-Hellman pour l'échange de clés. Cependant, avec la menace croissante des ordinateurs quantiques, le protocole BB84 est introduit pour renforcer la sécurité contre ces attaques, en particulier celles qui exploitent l'algorithme de Shor capable de briser les algorithmes cryptographiques classiques.

4.4.3.1 Utilisation du CPU

L'intégration du protocole quantique BB84 dans le processus de génération de clés a entraîné une augmentation significative de l'utilisation du CPU, atteignant des niveaux de 80 % à 85 % (voir la figure 4-18). Cette hausse s'explique par les calculs supplémentaires nécessaires pour gérer les états quantiques des photons et les bases de mesure, éléments fondamentaux du protocole BB84. Contrairement aux algorithmes classiques comme Diffie-Hellman, qui reposent sur des opérations mathématiques bien optimisées, BB84 requiert des capacités de traitement accrues pour :

- **Manipulation des bases aléatoires** : chaque photon est préparé et mesuré dans une base choisie aléatoirement, ce qui nécessite des calculs complexes en temps réel.
- **Correction des erreurs** : les canaux quantiques sont sujets à des perturbations, nécessitant l'utilisation de protocoles correctifs pour assurer l'intégrité des clés.
- **Amplification de la confidentialité** : BB84 inclut des étapes supplémentaires pour réduire la corrélation entre les données échangées et les informations potentielles interceptées par un attaquant.

Dans un environnement classique, des protocoles comme Diffie-Hellman effectuent des opérations cryptographiques principalement basées sur des multiplications modulaires et des calculs exponentiels, qui sont optimisés pour les architectures modernes de processeurs. En revanche, BB84 introduit des exigences spécifiques liées à la manipulation de données quantiques et à la gestion des interactions physiques, ce qui augmente la charge de travail du CPU.

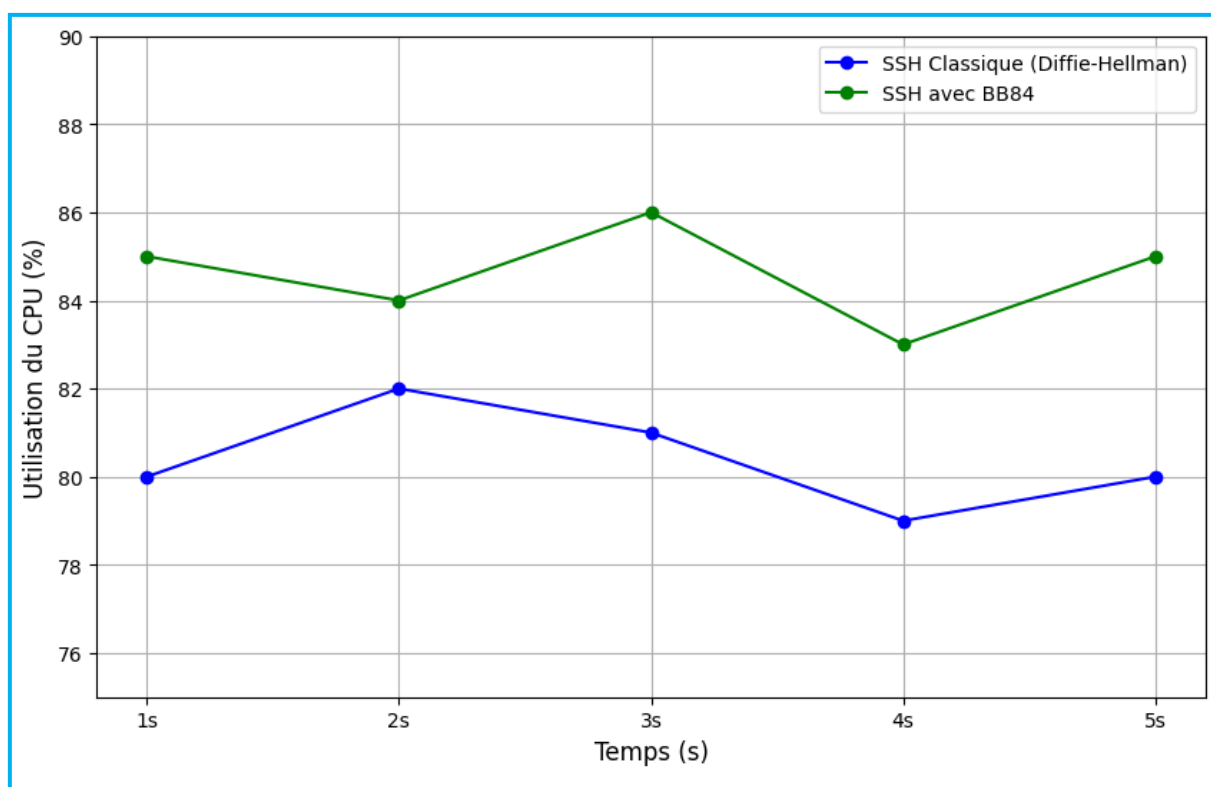


Figure 4- 18 : Comparaison de l'utilisation du CPU en temps réel entre SSH classique et SSH avec BB84

Malgré cette augmentation de l'utilisation du CPU, celle-ci reste acceptable dans les environnements modernes dotés de processeurs multicœurs et d'unités de traitement optimisées. Les processeurs contemporains sont capables de gérer efficacement ces charges supplémentaires, en particulier dans des applications critiques où la sécurité quantique apporte une valeur ajoutée significative.

4.4.3.2 Utilisation de la mémoire

L'intégration du protocole BB84 dans le processus de génération de clés a entraîné une légère augmentation de la consommation de mémoire, passant de 130 Mo à 140 Mo. Cette hausse, bien que modeste, s'explique par les exigences supplémentaires liées à la gestion des informations nécessaires pour la mise en œuvre des mécanismes quantiques. Plus précisément, ces exigences incluent :

- **Manipulation des états quantiques** : la gestion des états de polarisation des photons et des bases utilisées nécessite des structures de données supplémentaires pour le stockage temporaire des informations.
- **Comparaison des bases** : le protocole BB84 exige la conservation et l'analyse des bases utilisées par les parties communicantes pour identifier et éliminer les erreurs, ce

qui contribue à une utilisation accrue de la mémoire. Les résultats sont décrits dans la figure 4-19.

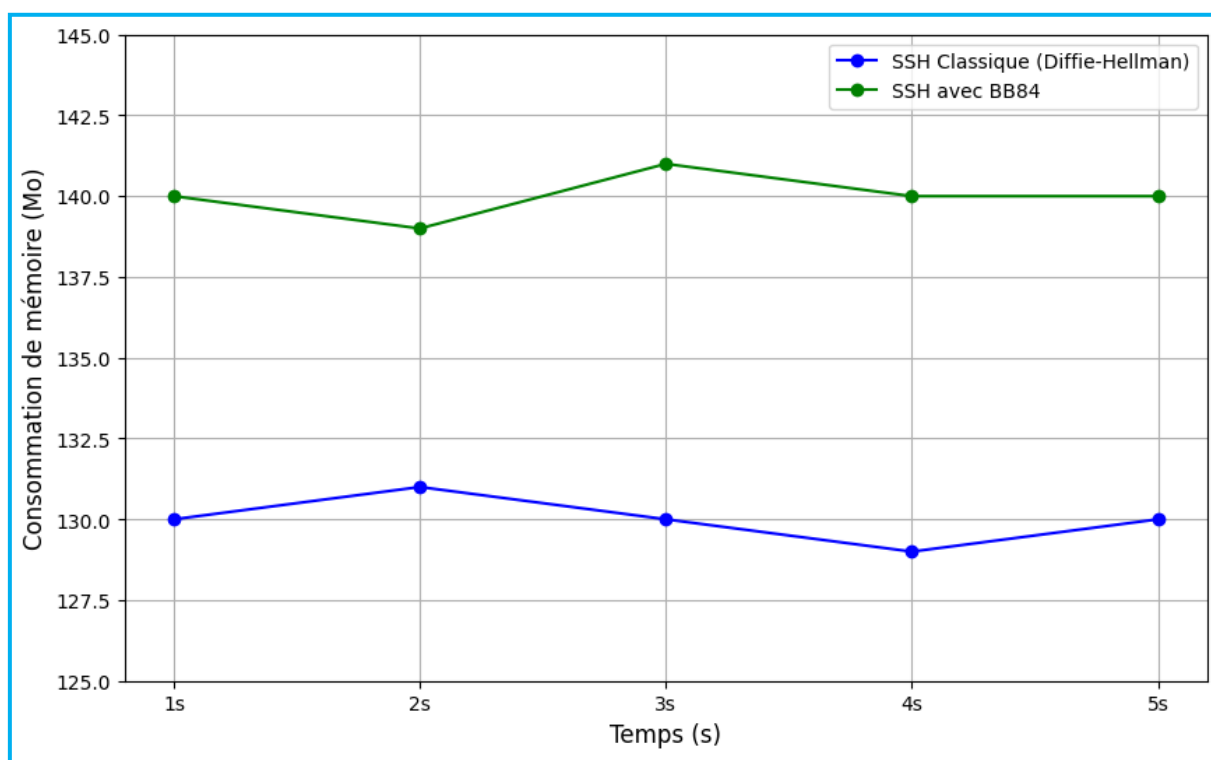


Figure 4- 19 : Comparaison de la consommation de mémoire entre SSH classique et SSH avec BB84

Malgré cette augmentation, la consommation de mémoire supplémentaire reste négligeable dans le contexte des systèmes modernes, qui sont généralement dotés de ressources abondantes. Une augmentation de 10 Mo représente moins de 8 % de la mémoire utilisée dans cette configuration et ne devrait pas affecter de manière significative les performances générales du système. Cette faible empreinte mémoire fait de BB84 une solution viable pour des environnements exigeants en termes de sécurité.

4.4.3.3 Temps d'établissement de la session

Le temps d'établissement d'une session SSH a légèrement augmenté de 15 ms à 18 ms. Cette augmentation est due aux étapes supplémentaires de validation et de correction d'erreurs introduites par le protocole BB84. Bien que ce délai supplémentaire puisse être perceptible dans des environnements où la latence est un facteur critique, il reste relativement mineur comparé

aux avantages apportés en termes de sécurité. Pour la plupart des cas d'utilisation, cette différence de 3 millisecondes est négligeable.

4.4.3.4 Débit de transmission

Le débit de transmission a diminué de 100 Mbps à 95 Mbps, soit une réduction de 5%. Cette baisse est due à la complexité des algorithmes de sécurité quantique et aux calculs supplémentaires nécessaires pour gérer l'intégration du protocole BB84. Toutefois, cette réduction est relativement faible et ne devrait pas affecter de manière significative la performance globale du réseau, sauf dans des environnements à très haute intensité de données. Le compromis entre une légère diminution de la vitesse et une sécurité renforcée contre les attaques quantiques est largement justifié. Le tableau 4-5 illustre les différences de performances en termes d'utilisation du CPU, de la mémoire, du temps d'établissement de session, et du débit de transmission.

Tableau 4- 5 : SSH classique vs SSH avec BB84 : différences de performances en termes d'utilisation du CPU, de la mémoire, du temps d'établissement de session, et du débit de transmission

Critères	SSH Classique	SSH avec BB84	Différence
Utilisation du CPU (%)	80%	85%	+5%
Utilisation de la mémoire (Mo)	130 Mo	140 Mo	+10 Mo
Temps d'établissement de session (ms)	15 ms	18 ms	+3 ms
Débit de transmission (Mbps)	100 Mbps	95 Mbps	-5 Mbps

4.4.3.5 Résistance aux attaques

L'intégration du protocole BB84 dans SSH apporte une protection renforcée contre les menaces potentielles posées par les ordinateurs quantiques. En particulier, les attaques basées sur l'algorithme de Shor, capables de casser les cryptosystèmes classiques comme RSA ou Diffie-Hellman, sont rendues inefficaces grâce aux principes de la mécanique quantique sur lesquels repose BB84.

Sécurité théoriquement sans faille : le protocole BB84 utilise des propriétés quantiques pour garantir que toute tentative d'interception est détectée immédiatement. Contrairement aux systèmes classiques basés sur des problèmes mathématiques, BB84 repose sur des principes physiques tels que l'incertitude d'Heisenberg, rendant l'interception impossible sans être repérée.

Protection contre les attaques passives et actives : que l'attaque soit passive ou active, BB84 garantit que toute perturbation du système entraîne l'échec de l'échange de clés. Si un attaquant modifie les photons échangés, cela provoque une désynchronisation dans la clé, forçant les parties légitimes à redémarrer le processus d'échange de clés.

Résilience contre les capacités des ordinateurs quantiques : les ordinateurs quantiques, malgré leur puissance de calcul accrue, ne peuvent pas contourner les lois fondamentales de la mécanique quantique, ce qui rend BB84 invulnérable aux attaques visant à casser la clé échangée. Le tableau 4-6 compare la résistance aux attaques entre un protocole classique et SSH avec BB84.

Tableau 4- 6 : Comparaison de la résistance aux attaques entre SSH classique et SSH avec BB84

Type d'attaque	SSH classique	SSH avec BB84
Attaque passive (écoute)	Exploitable	Totalement bloquée
Attaque active (MiTM)	Failles potentielles	Neutralisée immédiatement
Algorithme de Shor (quantique)	Complètement vulnérable	Théoriquement inviolable

4.4.4 Synthèse

La méthode d'intégration du protocole BB84 dans SSH se distingue par plusieurs avantages. D'une part, elle renforce la sécurité des échanges de clés en tirant parti des principes de la mécanique quantique, ce qui la rend résistante aux attaques potentielles, notamment celles menées par des ordinateurs quantiques. Comparée aux méthodes classiques comme Diffie-Hellman, BB84 garantit une détection immédiate des tentatives d'interception, ajoutant une couche de sécurité théoriquement sans faille. D'autre part, cette méthode, bien que légèrement plus exigeante en termes de ressources système (CPU et mémoire), n'entraîne que des impacts minimes sur la performance globale, avec des modifications marginales du débit de transmission et du temps d'établissement de session. En termes de perspectives, l'application future du protocole BB84 pourrait s'étendre à des environnements nécessitant une sécurité renforcée, tels que les réseaux quantiques ou les infrastructures critiques de communication. Des expérimentations plus poussées pourraient également explorer son efficacité dans des systèmes distribués ou dans des scénarios de cyberdéfense, où la résilience aux attaques quantiques est importante.

4.5 Conclusion

Les deux approches mettent en évidence leur capacité à renforcer la sécurité tout en maintenant une performance acceptable. D'une part la méthode de transmission vidéo basée sur les protocoles TLS et RTP propose une adaptation dynamique des clés de chiffrement en fonction des conditions réseau, garantissant ainsi une sécurité améliorée tout en maintenant une qualité de service optimale. Cette flexibilité permet de résister aux attaques par déni de service et aux variations réseau sans compromettre la fluidité de la transmission. Cependant, la montée des menaces liées aux ordinateurs quantiques souligne la nécessité de passer à des solutions de cryptographie post-quantique, afin de garantir une sécurité pérenne face à ces nouvelles formes d'attaques. D'autre part l'intégration du protocole BB84 dans SSH offre une protection avancée contre les attaques quantiques, notamment grâce à la détection instantanée des tentatives d'interception et à la robustesse des mécanismes de distribution de clés basés sur la mécanique quantique. Malgré une légère augmentation des ressources CPU et mémoire, les impacts sur la performance globale sont négligeables, ce qui en fait une solution viable pour des environnements nécessitant une sécurité accrue.

Conclusion générale et perspective

L'étude des méthodes cryptographiques « H-Rotation » et la transformation Braille mettent en évidence des solutions puissantes et complémentaires pour sécuriser les phrases/mots de passe. « H-Rotation », grâce à ses transformations dynamiques sur les blocs de hachage, renforce considérablement la résistance aux attaques par force brute et par dictionnaire. Cette méthode, bien qu'elle reste relativement simple à implémenter, présente certaines vulnérabilités dans des environnements hautement parallèles, où des attaques statistiques pourraient exploiter des modèles prédéfinis. La transformation Braille propose une approche innovante en substituant les caractères par leurs équivalents en symboles Braille, ce qui introduit une randomisation supplémentaire. Cette méthode se distingue par son efficacité en termes de rapidité et de faible consommation de mémoire, tout en offrant une protection renforcée contre les attaques par dictionnaire et par force brute. Ensemble, « H-Rotation » et la transformation Braille assurent un équilibre entre performance et sécurité, en comblant mutuellement leurs lacunes, ce qui en fait des propositions idéales pour une implémentation dans des systèmes de gestion de mots de passe, des environnements de cloud computing ou encore des infrastructures IoT.

Sur un autre plan, l'intégration des protocoles TLS et RTP pour la transmission vidéo sécurisée met en lumière l'importance d'une adaptation dynamique des clés de chiffrement, permettant de maintenir une qualité de service optimale tout en résistant aux cyberattaques. Cette approche, qui ajuste en temps réel les paramètres de sécurité en fonction des conditions du réseau, répond aux besoins croissants de sécurité dans les services de streaming et de visioconférence. Cependant, avec l'émergence des menaces liées aux ordinateurs quantiques, les méthodes cryptographiques actuelles doivent évoluer. L'intégration du protocole BB84 dans des infrastructures telles que SSH représente une avancée majeure pour la sécurisation des communications face aux attaques quantiques. Ce protocole de distribution de clés quantiques permet une détection instantanée des tentatives d'interception et assure une robustesse, malgré une légère augmentation des ressources CPU et mémoire. Cette transition vers la cryptographie post-quantique est essentielle pour préparer les systèmes à résister aux nouvelles formes de menaces, tout en maintenant des performances acceptables.

Ainsi, ces résultats ouvrent des perspectives intéressantes pour l'avenir. D'une part, il est important de poursuivre les recherches sur l'optimisation des protocoles de communication pour garantir une qualité de service et une sécurité renforcée, notamment dans les services de streaming et de visioconférence. D'autre part, l'amélioration des méthodes d'authentification,

notamment par l'intégration de technologies biométriques et d'intelligence artificielle, sera déterminante pour garantir une sécurité sans compromettre l'expérience utilisateur. Enfin, l'adoption progressive des algorithmes anti-quantiques, malgré les défis techniques qu'elle impose, est une nécessité incontournable pour garantir la pérennité des systèmes de sécurité actuels face aux menaces de l'informatique quantique. Ces recherches, combinées à une sensibilisation à la cybersécurité, contribueront à bâtir un avenir numérique plus sûr et mieux préparé aux défis technologiques de demain.

Bibliographie

- [Diffie 76] Diffie, W., & Hellman, M. E. New Directions in Cryptography, IEEE Transactions on Information Theory, V. 22, N. 6, pp. 644-654, 1976.
- [Koblitz 87] Koblitz, N. Elliptic Curve Cryptosystems, Mathematics of Computation, V. 48, N. 177, pp. 203-209, 1987.
- [Daemen 98] Daemen, J., & Rijmen, V. AES Proposal: Rijndael, AES Algorithm Submission, pp. 1-45, 1998.
- [Bertoni 10] Bertoni, G., Daemen, J., Peeters, M., & Van Assche, G. Keccak, Lecture Notes in Computer Science, V. 6147, pp. 313-314, 2010.
- [Juels 99] Juels, A., & Wattenberg, M. A Fuzzy ComMiTment Scheme, Proceedings of the ACM Conference on Computer and Communications Security, pp. 28-36, 1999.
- [Bennett 84] Bennett, C. H., & Brassard, G. Quantum Cryptography: Public Key Distribution and Coin Tossing, Proceedings of IEEE International Conference on Computers, Systems and Signal Processing, Bangalore, India, pp. 175-179, 1984.
- [Shor 94] Shor, P. W. Algorithms for Quantum Computation: Discrete Logarithms and Factoring, Proceedings of the 35th Annual Symposium on Foundations of Computer Science, IEEE, pp. 124-134, 1994.
- [Gentry 09] Gentry, C. Fully Homomorphic Encryption Using Ideal Lattices, Proceedings of the 41st Annual ACM Symposium on Theory of Computing, pp. 169-178, 2009.
- [TOUIL 22] Touil, H., El Akkad, N., Satori, K. Homomorphic Method Additive Using Pailler and Multiplicative Based on RSA in Integers Numbers, Lecture Notes in Networks and Systems, V. 489, Springer, Cham, pp. 141-150, 2022.
- [Dierks 08] Dierks, T., & Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246, 2008.
- [Jager 15] Jager, T., Schwenk, J., & Somorovsky, J. On the Security of TLS 1.3 and QUIC Against Weaknesses in PKCS#1 v1.5 Encryption, ACM Conference on Computer and Communications Security (CCS), 2015.

- [Bernstein 13] Bernstein, D. J., Lange, T., & Chen, B. Factoring RSA Keys from Certified Smart Cards: Coppersmith in the Wild, *Advances in Cryptology – ASIACRYPT 2013, Lecture Notes in Computer Science*, V. 8270, Springer, pp. 341-360, 2013.
- [Peikert 16] Peikert, C. A Decade of Lattice Cryptography, *Foundations and Trends in Theoretical Computer Science*, V. 10, N. 4, pp. 283-424, 2016.
- [Lo 14] Lo, H.-K., Curty, M., & Tamaki, K. Secure Quantum Key Distribution, *Nature Photonics*, V. 8, N. 8, pp. 595-604, 2014.
- [Luk 09] Luk, C. K., Hong, S., Kim, H., & Qiu, J. D. Qilin: Exploiting Parallelism on Heterogeneous Multiprocessors with Adaptive Mapping, *ACM SIGPLAN Notices*, V. 44, N. 6, pp. 32-41, 2009.
- [Stallings 17] Stallings, W. *Cryptography and Network Security: Principles and Practice*, Pearson Education, 2017.
- [Schneier 07] Schneier, B. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, 2007.
- [RSA 78] Rivest, R., Shamir, A., & Adleman, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, *Communications of the ACM*, 1978.
- [Bernstein 17] Bernstein, D. J., & Lange, T. Post-Quantum Cryptography, *Nature*, V. 549, pp. 188–194, 2017.
- [Solove 11] Solove, D. J. *Nothing to Hide: The False Tradeoff Between Privacy and Security*, Yale University Press, 2011.
- [Abelson 15] Abelson, H., Anderson, R., Bellovin, S. M., Benaloh, J., Blaze, M., Diffie, W., & Schneier, B. Keys Under Doormats: Mandating Insecurity by Requiring Government Access to All Data and Communications, *Journal of Cybersecurity*, V. 1, N. 1, pp. 69-79, 2015.
- [Barrett 05] Barrett, D., & Silverman, R. *SSH, The Secure Shell: The Definitive Guide*, O'Reilly Media, 2005.
- [Menezes 18] Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. *Handbook of Applied Cryptography*, CRC Press, 2018.
- [Andrews 20] Andrews, R., Hahn, D. A., & Bardas, A. G. Measuring the Prevalence of the Password Authentication Vulnerability in SSH, 2020 IEEE

- International Conference on Communications (ICC), Dublin, Ireland, pp. 1-7, 2020.
- [Dacosta 12] Dacosta, I., Ahamad, M., & Traynor, P. Trust No One Else: Detecting MITM Attacks Against SSL/TLS Without Third-Parties, International Conference on Security and Privacy in Communication Systems, 2012.
- [Subhan 23] Subhan, A., Kunang, Y. N., & Yadi, I. Z. Analyzing the Attack Pattern of Brute Force Attack on SSH Port, 2023 International Conference on Information Technology and Computing (ICITCOM), Yogyakarta, Indonesia, pp. 67-72, 2023.
- [Kurnia 18] Kurnia Putra, D. S., & Prima, E. Evaluating Certificate Policy - Certification Practice Statement of Unique Government Certification Authority using Public Key Infrastructure Assessment Guidelines: Research in Progress, 2018 International Seminar on Intelligent Technology and Its Applications (ISITIA), Bali, Indonesia, pp. 251-256, 2018.
- [Dawabsheh 22] Dawabsheh, A., Jazzar, M., Eleyan, A., Bejaoui, T., & Popoola, S. An Enhanced Phishing Detection Tool Using Deep Learning From URL, 2022 International Conference on Smart Applications, Communications and Networking (SmartNets), Palapye, Botswana, pp. 1-6, 2022.
- [Sniatala 21] Sniatala, P., Iyengar, S., & Ramani, S. K. Symmetric Key Cryptography, In: Evolution of Smart Sensing Ecosystems with Tamper Evident Security, Springer, Cham, 2021.
- [Ryabko 15] Ryabko, B. Y. The Vernam Cipher is Robust to Small Deviations from Randomness, Problems of Information Transmission, V. 51, pp. 82–86, 2015.
- [Paar 10] Paar, C., & Pelzl, J. The Data Encryption Standard (DES) and Alternatives, In: Understanding Cryptography, Springer, Berlin, Heidelberg, pp. 93-121, 2010.
- [Adam 19] Adam, N., Mashaly, M., & Alexan, W. A 3DES Double-Layer Based Message Security Scheme, 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), Riyadh, Saudi Arabia, pp. 1-5, 2019.
- [Menezes 97] Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. Handbook of Applied Cryptography, CRC Press, 1997.

- [Biham 93] Biham, E., & Shamir, A. Differential Cryptanalysis of the Data Encryption Standard, Springer, 1993.
- [van Oorschot 91] Van Oorschot, P. C., & Wiener, M. J. A Known Plaintext Attack on Two-Key Triple Encryption, Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, 1991.
- [Grover 96] Grover, L. K. A Fast Quantum Mechanical Algorithm for Database Search, Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, 1996.
- [Daemen 01] Daemen, J., & Rijmen, V. The Design of Rijndael: AES — The Advanced Encryption Standard, Springer, 2001.
- [Ferguson 01] Ferguson, N., Schneier, B., & Kohno, T. Cryptography Engineering: Design Principles and Practical Applications, Wiley Publishing, 2001.
- [Kocher 99] Kocher, P., Jaffe, J., & Jun, B. Differential Power Analysis, Proceedings of the 19th Annual International Cryptology Conference, Springer, 1999.
- [Bernstein 05] Bernstein, D. J. Cache-Timing Attacks on AES, Cryptology ePrint Archive, 2005.
- [Miller 85] Miller, V. S. Use of Elliptic Curves in Cryptography, Lecture Notes in Computer Science, V. 218, pp. 417-426, 1985.
- [Reddy 21] Reddy, G. P., Narayana, A., Keerthan, P. K., Vineetha, B., & Honnavalli, P. Multiple Hashing Using SHA-256 and MD5, In: Advances in Computing and Network Communications, Lecture Notes in Electrical Engineering, V. 735, Springer, Singapore, 2021.
- [Handschuh 11] Handschuh, H. SHA-0, SHA-1, SHA-2 (Secure Hash Algorithm), In: Van Tilborg, H. C. A., & Jajodia, S. (Eds), Encyclopedia of Cryptography and Security, Springer, Boston, MA, 2011.
- [Charles 14] Bennett, C. H., & Brassard, G. Quantum Cryptography: Public Key Distribution and Coin Tossing, Theoretical Computer Science, V. 560, Part 1, pp. 7-11, 2014.
- [Einstein 35] Einstein, A., Podolsky, B., & Rosen, N. Can Quantum-Mechanical Description of Physical Reality Be Considered Complete, Physical Review, 1935.
- [Scarani 20] Scarani, V., Bechmann-Pasquinucci, H., & Cerf, N. J. The Security of Practical Quantum Key Distribution, Reviews of Modern Physics, 2020.

- [Ylonen 06] Ylonen, T. The Secure Shell (SSH) Protocol Architecture, IETF RFC 4251, 2006.
- [Song 01] Song, D. X., Wagner, D., & Tian, X. Timing Analysis of Keystrokes and Timing Attacks on SSH, Proceedings of the 10th USENIX Security Symposium, 2001.
- [Mosca 18] Mosca, M. Cybersecurity in an Era with Quantum Computers: Will We Be Ready?, IEEE Security & Privacy, V. 16, N. 5, pp. 38-41, 2018.
- [Boneh 20] Boneh, D., & Shoup, V. A Graduate Course in Applied Cryptography, Cambridge University Press, 2020.
- [Shor 99] Shor, P. W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, SIAM Journal on Computing, V. 26, N. 5, pp. 1484-1509, 1997.
- [Preneel 96] Preneel, B., Bosselaers, A., & Dobbertin, H. The Cryptographic Hash Function RIPEMD-160, In: Fast Software Encryption, Springer, pp. 71-104, 1996.
- [Bellare 93] Bellare, M., & Rogaway, P. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols, ACM Conference on Computer and Communications Security (CCS), 1993.
- [Dang 15] Dang, Q. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, NIST Federal Information Processing Standards Publication 202, 2015.
- [Black 05] Black, J., Cochran, M., & Shrimpton, T. On the Impossibility of Highly-Efficient Blockcipher-Based Hash Functions, Advances in Cryptology — EUROCRYPT 2005, Springer, pp. 526-541, 2005.
- [Goldreich 01] Goldreich, O. Foundations of Cryptography: Volume 1, Basic Tools, Cambridge University Press, 2001.
- [Shannon 48] Shannon, C. E. A Mathematical Theory of Communication, Bell System Technical Journal, V. 27, pp. 379-423, 1948.
- [Cover 06] Cover, T. M., & Thomas, J. A. Elements of Information Theory, Wiley-Interscience, 2006.
- [Joshi 21] Joshi, M., Budhani, S., Tewari, N., & Prakash, S. Analytical Review of Data Security in Cloud Computing, 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM), London, United Kingdom, pp. 362-366, 2021.
- [Zhou 19] Zhou, L., Chen, J., Zhang, Y., Su, C., & James, M. A. Security Analysis and New Models on the Intelligent Symmetric Key Encryption, Computers & Security, V. 80, pp. 14-24, 2019.

- [Ogala 23] Ogala, J. O., Ahmad, S., Shakeel, I., et al. Strengthening KMS Security with Advanced Cryptography, Machine Learning, Deep Learning, and IoT Technologies, *SN Computer Science*, V. 4, Art. 530, 2023.
- [Touil 20] Touil, H., El Akkad, N., & Satori, K. H-Rotation: Secure Storage and Retrieval of Passphrases on the Authentication Process, *International Journal of Safety and Security Engineering*, V. 10, N. 6, pp. 785-796, 2020.
- [Touil 24a] Touil, H., El Akkad, N., Satori, K., Soliman, N. F., & El-Shafai, W. Efficient Braille Transformation for Secure Password Hashing, *IEEE Access*, V. 12, pp. 5212-5221, 2024.
- [Skanda 22] Skanda, C., Srivatsa, B., & Premananda, B. S. Secure Hashing using BCrypt for Cryptographic Applications, 2022 IEEE North Karnataka Subsection Flagship International Conference (NKCon), Vijaypur, India, V. 1, pp. 1-5, 2022.
- [Alwen 16] Alwen, J., Chen, J., Len, Y., Su, C., & James, M. On the Complexity of scrypt and Proofs of Space in the Parallel Random Oracle Model, *Advances in Cryptology–EUROCRYPT*, V. 35, pp. 236-252, 2016.
- [Biryukov 15] Biryukov, A., Dinu, D.-D., & Khovratovich, D. *Argon and Argon2*, Springer Berlin Heidelberg, 2015.
- [Hatzivasilis 17] Hatzivasilis, G. Password-Hashing Status, *Cryptography*, V. 1, N. 2, Art. 10, 2017.
- [Menasce 03] Menasce, D. A. Security Performance, *IEEE Internet Computing*, V. 7, N. 3, pp. 84-87, May-June 2003.
- [Chen 06] Chen, G., Chen, K., & Dong, J. A Database Encryption Scheme for Enhanced Security and Easy Sharing, *Proceedings of the 10th International Conference on Computer Supported Cooperative Work in Design*, Nanjing, China, pp. 1-6, 2006.
- [Antonacopoulos 04] Antonacopoulos, A., & Bridson, D. A Robust Braille Recognition System, In: Marinai, S., & Dengel, A. R. (eds) *Document Analysis Systems VI*, *Lecture Notes in Computer Science*, V. 3163, Springer, Berlin, Heidelberg, 2004.
- [Schiewe 09] Schiewe, M., Köhlmann, W., Nadig, O., & Weber, G. What You Feel Is What You Get: Mapping GUIs on Planar Tactile Displays, In: Stephanidis, C. (ed) *Universal Access in Human-Computer Interaction. Intelligent and*

- Ubiquitous Interaction Environments, Lecture Notes in Computer Science, V. 5615, Springer, Berlin, Heidelberg, 2009.
- [Mitra 21] Mitra, S., Das, S., & Kule, M. Prevention of the Man-in-the-Middle Attack on Diffie–Hellman Key Exchange Algorithm: A Review, In: Bhattacharjee, D., Koley, D. K., Dey, N., Basu, S., & Plewczynski, D. (eds) Proceedings of International Conference on Frontiers in Computing and Systems, Advances in Intelligent Systems and Computing, V. 1255, Springer, Singapore, 2021.
- [Agrawal 17] Agrawal, S. Stronger Security for Reusable Garbled Circuits, General Definitions and Attacks, In: Katz, J., & Shacham, H. (eds) Advances in Cryptology – CRYPTO 2017, Lecture Notes in Computer Science, V. 10401, Springer, 2017.
- [Hassan 20] Hassan, M. M., Gumaiei, A., Huda, S., & Almogren, A. Increasing the Trustworthiness in the Industrial IoT Networks Through a Reliable Cyberattack Detection Model, IEEE Transactions on Industrial Informatics, V. 16, N. 9, pp. 6154-6162, Sept. 2020.
- [Hu 19] Hu, Q., Fan, X., & Zhang, Q. An Effective Differential Power Attack Method for Advanced Encryption Standard, 2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), Guilin, China, pp. 58-61, 2019.
- [Touil 21] Touil, H., El Akkad, N., & Satori, K. Secure and Guarantee QoS in a Video Sequence: A New Approach Based on TLS Protocol to Secure Data and RTP to Ensure Real-Time Exchanges, International Journal of Safety and Security Engineering, V. 11, N. 1, pp. 59-68, 2021.
- [Bresciani 09] Bresciani, R., & Butterfield, A. A Formal Security Proof for the ZRTP Protocol, 2009 International Conference for Internet Technology and Secured Transactions (ICITST), London, UK, V. 1, pp. 1-6, 2009.
- [Song 02] Song, B., Chung, K., & Shin, Y. SRTP: TCP-Friendly Congestion Control for Multimedia Streaming, Information Networking: Wireless Communications Technologies and Network Applications, ICOIN 2002, Lecture Notes in Computer Science, V. 2344, pp. 123-134, Springer, Berlin, Heidelberg, 2002.

- [Touil 24b] Touil, H., Hdioud, F., El Akkad, N., & Satori, K. The Security of SSH Protocol Public Key Sharing in the Post-Quantum Era, *International Journal of Computing*, V. 23, N. 3, pp. 317-323, 2024.
- [Ono 18] Ono, H., & Manabe, Y. Efficient Card-Based Cryptographic Protocols for the Millionaires' Problem Using Private Input Operations, *Proceedings of the 13th Asia Joint Conference on Information Security (AsiaJCIS)*, Guilin, China, pp. 23-28, 2018.
- [Koblitz 16] Koblitz, N., & Menezes, A. A Riddle Wrapped in an Enigma, *IEEE Security & Privacy*, V. 14, N. 6, pp. 34-42, Nov.-Dec. 2016.
- [Maqri 20] Al-Maqri, M. A., Mansoor, A. M., Sabri, A. Q., Ravana, S. D., & Yaseein, H. S. High Performing Multimedia Transmission Approach Based on QoS Support and Admission Control over IEEE 802.11e Networks, *International Journal of Communication Systems*, V. 33, N. 4, Art. e4193, 2020.
- [Jubair 22] Jubair, M. A., et al. A QoS Aware Cluster Head Selection and Hybrid Cryptography Routing Protocol for Enhancing Efficiency and Security of VANETs, *IEEE Access*, V. 10, pp. 124792-124804, 2022.
- [Taleb 10] Taleb, T., Hadjadj Aoul, Y., & Benslimane, A. Integrating Security with QoS in Next Generation Networks, *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, Miami, FL, USA, pp. 1-5, 2010.
- [Serhrouchni 06] Serhrouchni, A., & Hajjeh, I. Intégration de la signature numérique au protocole SSL/TLS, *Annales des Télécommunications*, V. 61, pp. 522–541, 2006.
- [Elhao 21] Elhao, A., Megahed, M. H., & Elsamahy, E. A. Towards Quantum Resistant Key Agreement Schemes Using Unpredictability, *Proceedings of the 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, Pitesti, Romania, pp. 1-6, 2021.
- [St-Jean 17] St-Jean, P., et al. Polariton Lasing in the Edge States of an Orbital SSH Chain, *Proceedings of the Conference on Lasers and Electro-Optics Europe & European Quantum Electronics Conference (CLEO/Europe-EQEC)*, Munich, Germany, 2017.

- [St-Jean 19] St-Jean, P., et al. Topological Interface States in a Polariton SSH Lattice: Linear and Nonlinear Regimes, Proceedings of the Conference on Lasers and Electro-Optics Europe & European Quantum Electronics Conference (CLEO/Europe-EQEC), Munich, Germany, 2019.
- [Petrenko 22] Petrenko, A. 3 Development of Quantum Cryptanalysis Algorithms, In: Applied Quantum Cryptanalysis, River Publishers, pp. 97-128, 2022.
- [Prakasan 22] Prakasan, K., & Jain, P. Authenticated-Encryption in the Quantum Key Distribution Classical Channel Using Post-Quantum Cryptography, Proceedings of the 6th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, pp. 804-811, 2022.
- [Garcia 23] Garcia, C. R., et al. Quantum-Resistant TLS 1.3: A Hybrid Solution Combining Classical, Quantum and Post-Quantum Cryptography, Proceedings of the 28th IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), pp. 1-6, 2023.
- [Schrödinger 99] Schrödinger, E. About Heisenberg Uncertainty Relation, arXiv preprint quant-ph/9903100, V. 10, pp. 1-15, 1999.

Annexe A : Algorithmes de hachage

Algorithme de MD5

L'algorithme MD5 est utilisé pour générer un hach unique de 128 bits à partir d'un message de longueur variable. Cet hash sert à vérifier l'intégrité des données en détectant les modifications accidentelles ou malveillantes. MD5 est souvent utilisé dans les signatures numériques, le stockage des mots de passe et la vérification de l'intégrité des fichiers, bien que ses vulnérabilités en termes de sécurité le rendent moins recommandé pour les applications critiques de sécurité.

- **Initialisation** : on commence par initialiser quatre variables 32 bits : A, B, C, et D, avec des valeurs fixes, exprimées en hexadécimal comme suit :

$A = 0x67452301$

$B = 0xefcdab89$

$C = 0x98badcfe$

$D = 0x10325476$

- **Prétraitement** : le message d'entrée est prétraité de la manière suivante :

+ **Ajouter des bits de bourrage** : un bit '1' est ajouté, suivi de suffisamment de bits '0' pour que la longueur du message (en bits) soit congruente à $448 \bmod 512$.

+ **Ajouter la longueur** : les 64 derniers bits du message prétraité contiennent la longueur initiale du message en bits (avant le bourrage).

+ **Division en blocs de 512 bits** : le message prétraité est divisé en blocs de 512 bits. Chaque bloc de 512 bits est ensuite divisé en 16 sous-parties de 32 bits, notées $M_0, M_1, \dots, M_{15}$.

+ **Boucle sur chaque bloc de 512 bits** : pour chaque bloc, nous procédons à une série de transformations en 64 itérations, réparties en quatre séries de 16 opérations. Chaque série utilise une fonction non linéaire différente (F, G, H, I) et une addition modulaire (module 2^{32}).

Calcul des fonctions F, G, H, I :

Les fonctions F, G, H et I prennent trois entrées de 32 bits et sortent un résultat de 32 bits. Elles sont définies comme suit :

Calcul des fonctions F, G, H, I

Les fonctions F, G, H et I prennent trois entrées de 32 bits et sortent un résultat de 32 bits. Elles sont définies comme suit :

$$F(B,C,D) = (B \wedge C) \vee (\neg B \wedge D)$$

$$G(B,C,D) = (B \wedge D) \vee (C \wedge \neg D)$$

$$H(B,C,D) = B \oplus C \oplus D$$

$$I(B,C,D) = C \oplus (B \vee \neg D)$$

Où $\wedge$, $\vee$, et $\oplus$ représentent respectivement l'AND, l'OR, et l'XOR bit à bit, et $\neg$ est le NOT bit à bit.

Initialisation des variables temporaires

Avant chaque série de 64 itérations, on initialise quatre variables temporaires :

$$AA=A, \quad BB=B, \quad CC=C, \quad DD=D$$

Boucle principale : (pour $i=0$ à 63)

L'algorithme effectue 64 opérations de transformation sur AA , BB , CC , DD . À chaque étape i , une des fonctions F , G , H , I est utilisée, combinée à un mot M_i et à une constante K_i . Les 64 constantes K_i sont des fractions des sinus d'entiers (multipliées par 2^{32} et arrondies), précalculées pour chaque itération.

L'opération pour chaque itération est donnée par la formule :

$$AA = BB + ((AA + F(BB, CC, DD) + M_i + K_i) \lll) s_i$$

Après chaque série de transformations, les résultats sont ajoutés (modulo 2^{32}) aux valeurs d'origine :

$$A=A+AA, \quad B=B+BB, \quad C=C+CC, \quad D=D+DD$$

Hachage final :

À la fin de toutes les itérations pour tous les blocs de 512 bits, les quatre valeurs A , B , C , et D sont concaténées pour former le hachage final de 128 bits.

Formellement, le résultat final est :

$$MD5(M) = Concat(A, B, C, D)$$

Chaque partie A , B , C , D est une représentation de 32 bits, ce qui donne un résultat total de 128 bits.

Algorithme A. 1 : Description du fonctionnement de MD5

Algorithme de SHA

Les algorithmes SHA sont utilisés pour produire un hash unique et fixe à partir d'un message de longueur variable. Cet hash est utilisé pour vérifier l'intégrité des données, assurer la sécurité des communications et authentifier les informations dans divers contextes comme les signatures numériques, les certificats SSL/TLS et le stockage sécurisé des mots de passe.

Familles d'algorithmes SHA :

SHA-0 : la version initiale, rapidement remplacée en raison de vulnérabilités.

SHA-1 : produit un hash de 160 bits. Utilisé pendant longtemps, mais désormais déconseillé en raison de faiblesses cryptographiques.

SHA-2 : comprend plusieurs variantes comme SHA-224, SHA-256, SHA-384 et SHA-512, offrant des longueurs de hash de 224 à 512 bits. SHA-256 et SHA-512 sont les plus couramment utilisés et offrent une sécurité renforcée.

SHA-3 : une version plus récente et indépendante de la structure de SHA-2, basée sur la fonction de hachage Keccak. Conçue pour être résistante aux attaques et offrant des tailles de condensé similaires à SHA-2 (SHA3-224, SHA3-256, SHA3-384, SHA3-512).

Initialisation des variables : SHA-1 initialise cinq variables de 32 bits (H0, H1, H2, H3, H4) avec les valeurs suivantes :

H0=0x67452301

H1=0xEFCDAB89

H2=0x98BADCFE

H3=0x10325476

H4=0xC3D2E1F0

Prétraitement : Le message est prétraité de manière similaire à MD5 :

- **Ajouter un bit '1'** suivi de bits '0' pour que la longueur du message soit congruente à $448 \bmod 512$.
- **Ajouter la longueur du message** en bits (64 bits) à la fin du message prétraité, ce qui porte la taille totale à un multiple de 512 bits.

Division en blocs de 512 bits : Le message est ensuite divisé en blocs de 512 bits. Chaque bloc est ensuite divisé en 16 mots de 32 bits, notés $M_0, M_1, \dots, M_{15}$

Extension à 80 mots : Les 16 mots de 32 bits sont étendus à 80 mots à l'aide de la formule suivante pour chaque $i \geq 16$:

$$M_i = (M_{i-3} \oplus M_{i-8} \oplus M_{i-14} \oplus M_{i-16}) \lll 1$$

Où $\oplus$ est l'opérateur XOR bit à bit, et $\lll$ 1 est une rotation gauche d'un bit.

Initialisation des variables temporaires : Pour chaque bloc, les cinq variables A,B,C,D,E sont initialisées comme suit :

$$A = H0, \quad B = H1, \quad C = H2, \quad D = H3, E = H4$$

Boucle principale : (pour $i = 0$ à 79)

L'algorithme SHA-1 effectue 80 itérations sur les 80 mots de 32 bits M_i , avec des opérations non linéaires et des constantes spécifiques. Ces 80 opérations sont réparties en quatre séries de 20 itérations chacune, et chaque série utilise une fonction f_i différente et une constante K_i .

Fonctions non linéaires et constantes :

Les fonctions f_i et les constantes K_i varient en fonction de l'itération i et sont définies comme suit :

$$\begin{aligned} f_i(B, C, D) &= (B \wedge C) \vee (\neg B \wedge D); & 0 \leq i \leq 19 \\ f_i(B, C, D) &= B \oplus C \oplus D; & 20 \leq i \leq 39 \\ f_i(B, C, D) &= (B \wedge C) \vee (B \wedge D) \vee (C \wedge D); & 40 \leq i \leq 59 \\ f_i(B, C, D) &= B \oplus C \oplus D; & 60 \leq i \leq 79 \end{aligned}$$

Les constantes K_i utilisées dans chaque série d'itérations sont :

$$\begin{aligned} K_i &= 0x5A827999, & 0 \leq i \leq 19 \\ K_i &= 0x6ED9EBA1, & 20 \leq i \leq 39 \\ K_i &= 0x8F1BBCDC, & 40 \leq i \leq 59 \\ K_i &= 0xCA62C1D6, & 60 \leq i \leq 79 \end{aligned}$$

Calcul dans chaque itération :

Pour chaque itération i , la mise à jour des variables A,B,C,D,E est réalisée avec la formule suivante :

$$T = (A \lll 5) + f_i(B, C, D) + E + M_i + K_i$$

Ensuite, les valeurs sont mises à jour comme suit :

$$E = D$$

$$D = C$$

$$C = B \lll 30$$

$$B = A$$

$$A = T$$

Mise à jour des valeurs initiales :

Après les 80 itérations, les résultats temporaires A,B,C,D,E sont ajoutés aux valeurs initiales H_0, H_1, H_2, H_3, H_4 respectivement :

$$H_0 = H_0 + A, H_1 = H_1 + B, H_2 = H_2 + C, H_3 = H_3 + D, H_4 = H_4 + E$$

Hachage final :

À la fin de toutes les itérations sur tous les blocs de 512 bits, les cinq variables H_0, H_1, H_2, H_3, H_4 sont concaténées pour former le hachage final de 160 bits :

$$SHA(M) = Concat(H_0, H_1, H_2, H_3, H_4)$$

Chaque partie H_0, H_1, H_2, H_3, H_4 est une représentation de 32 bits, ce qui donne un résultat total de 160 bits.

Algorithme A. 2 : Description du fonctionnement de SHA

Annexe B : Algorithmes de protection de d'authentification

Algorithme de « H-Rotation »

Phase 1 : Inscription

Initialisation des variables :

- Initialiser la phrase de passe (ω) fournie par l'utilisateur.
- Calculer le hachage initial de la phrase de passe (ω) en utilisant une fonction de hachage standard SHA-3.

Transformation dynamique :

- Rotation du préfixe : Prendre les premiers caractères du hachage et les décaler vers la fin selon un paramètre de décalage.
- Rotation du suffixe : Prendre les derniers bits du hachage et les décaler vers le début selon un paramètre de décalage.
- Rotation du milieu : Identifier la section centrale du hachage et permuter les bits internes selon un décalage défini.

Choisir la transformation d'une manière aléatoire :

- choisir l'une des transformation (préfixe, suffixe, milieu) pour obtenir un hachage final unique.

Génération des paramètres dynamiques :

- Calculer et stocker les paramètres de décalage utilisés pour chaque transformation dynamique.

Sortie :

- Stocker le hachage final (ω_h) et les paramètres associés (θ).

Phase 2 : Authentification

Initialisation des variables :

- Recevoir une phrase de passe (ω') fournie par l'utilisateur.
- Recevoir le hachage inscrit (ω_h) et les paramètres (θ).

Hachage initial :

- Calculer le hachage initial de la phrase de passe (ω') en utilisant la même fonction de hachage standard que lors de l'inscription.

Reconstitution des transformations dynamiques :

- Rotation du préfixe : Appliquer le décalage sur le préfixe à l'aide des paramètres (θ).
- Rotation du suffixe : Appliquer le décalage sur le suffixe à l'aide des paramètres (θ).
- Rotation du milieu : Appliquer le décalage sur la section centrale à l'aide des paramètres (θ).

Combinaison des transformations :

- Combiner les transformations pour reconstruire le hachage final (ω_h').

Comparaison :

- Comparer le hachage reconstruit (ω_h') avec le hachage inscrit (ω_h).

Sortie :

- Si les deux hachages correspondent ($\omega_h' == \omega_h$), l'authentification est réussie.
- Sinon, l'authentification échoue.

Algorithme B. 1 : Description du fonctionnement de la méthode « H-Rotation »

Algorithme de transformation Braille

Phase 1 : Inscription

Initialisation des variables :

- Initialiser le mot de passe ω fournie par l'utilisateur.
- Calculer le hachage initial H_0 de ω en utilisant une fonction de hachage standard, MD5.

Transformation en Braille :

1. Division en blocs :

- Diviser H_0 en blocs de taille fixe B_i .

2. Mapping vers les symboles Braille :

- Pour chaque bloc B_i , associer un caractère Braille composé de six chiffres en utilisant une table de correspondance.

3. Substitution aléatoire :

- Remplacer les zéros présents dans chaque caractère Braille par des valeurs aléatoires choisies parmi $\{7,8,9\}$.

Stockage dans la base de données :

- Combiner tous les caractères Braille pour former une chaîne finale C_{final} .
- Stocker C_{final} et l'identifiant utilisateur dans la base de données.

Phase 2 : Authentification

Initialisation des variables :

- Saisir le mot de passe ω fournis par l'utilisateur.

Calcul du hachage et transformation :

1. Calcul du hachage initial :

- Appliquer la même fonction de hachage utilisée lors de l'inscription pour obtenir H_0' à partir de ω .

2. Transformation en Braille :

- Diviser H_0' en blocs de taille fixe B_i' .
- Mapper chaque bloc B_i' en caractères Braille comme dans la phase d'inscription.
- Appliquer la substitution aléatoire sur les zéros pour obtenir C_{auth} .

Comparaison avec la base de données :

- Récupérer la chaîne C_{final} associée à l'identifiant ID dans la base de données.
- Comparer $C_{auth} = C_{final}$:
 - Si $C_{auth} = C_{final}$, alors l'authentification est validée.

Algorithme B. 2 : Description du fonctionnement de la transformation Braille

Publications

Articles publiés dans des journaux internationaux

- 1) Touil, H., El Akkad, N., Satori, K. Secure and Guarantee QoS in a Video Sequence: A New Approach Based on TLS Protocol to Secure Data and RTP to Ensure Real-Time Exchanges, International Journal of Safety and Security Engineering, V. 11, No. 1, pp. 59-68, 2021. <https://doi.org/10.18280/ijssse.110107>
- 2) Touil, H., El Akkad, N., Satori, K. H-Rotation : Secure Storage and Retrieval of Passphrases on the Authentication Process, International Journal of Safety and Security Engineering, V. 10, No. 6, pp. 785-796, 2020. <https://doi.org/10.18280/ijssse.100609>
- 3) Touil, H., El Akkad, N., Satori, K., Soliman, N. F., El-Shafai, W. Efficient Braille Transformation for Secure Password Hashing, IEEE Access, V. 12, pp. 5212-5221, 2024. [doi: 10.1109/ACCESS.2024.3349487](https://doi.org/10.1109/ACCESS.2024.3349487)
- 4) Touil, H., Hdioud, F., El Akkad, N., Satori, K. The Security of SSH Protocol Public Key Sharing in the Post-Quantum Era, International Journal of Computing, V. 23, No. 3, pp. 317-323, 2024. [DOI: 10.47839/ijc.23.3.3650](https://doi.org/10.47839/ijc.23.3.3650)
- 5) Ben Chakra, F., Touil, H., El Akkad. Hiding information in the spectral transformation coefficients of a JPEG file. Baghdad Science Journal. May 03, 2024.

Communications internationales

- 1) Touil, H., El Akkad, N., Satori, K. Text Encryption: Hybrid Cryptographic Method Using Vigenere and Hill Ciphers, 2020 International Conference on Intelligent Systems and Computer Vision (ISCV), pp. 1-6, Fez, Morocco, 2020. <https://doi.org/10.1109/ISCV49265.2020.9204095>
- 2) Touil, H., El Akkad, N., Satori, K. Securing the Storage of Passwords Based on the MD5 HASH Transformation, in: Motahhir, S., Bossoufi, B. (eds) Digital Technologies and Applications, Lecture Notes in Networks and Systems, V. 211, Springer, Cham, pp. 497-506, 2021. https://doi.org/10.1007/978-3-030-73882-2_45
- 3) Touil, H., El Akkad, N., Satori, K. Homomorphic Method Additive Using Pailler and Multiplicative Based on RSA in Integers Numbers, in: Lazaar, M., Duvallet, C., Touhafi, A., Al Achhab, M. (eds) Proceedings of the 5th International Conference on Big Data

- and Internet of Things, Lecture Notes in Networks and Systems, V. 489, Springer, Cham, pp. 141-150, 2022. https://doi.org/10.1007/978-3-031-07969-6_12
- 4) Touil, H., El Akkad, N., Satori, K. Ensure the Confidentiality of Documents Shared Within the Enterprise in the Cloud by Using a Cryptographic Delivery Method, in: Motahhir, S., Bossoufi, B. (eds) Digital Technologies and Applications, Lecture Notes in Networks and Systems, V. 455, Springer, Cham, pp. 281-290, 2022. https://doi.org/10.1007/978-3-031-02447-4_25
 - 5) Ben Chakra, F., Touil, H., El Akkad, N. Securing Caesar Cryptography Using the 2D Geometry, in: Kacprzyk, J., Ezziyyani, M., Balas, V.E. (eds) International Conference on Advanced Intelligent Systems for Sustainable Development, Lecture Notes in Networks and Systems, V. 712, Springer, Cham, pp. 287-297, 2023. https://doi.org/10.1007/978-3-031-35251-5_22
 - 6) Ben Chakra, F.Z., Touil, H., El Akkad, N. CCC-Transformation: Novel Method to Secure Passwords Based on Hash Transformation, in: Motahhir, S., Bossoufi, B. (eds) Digital Technologies and Applications, Lecture Notes in Networks and Systems, V. 668, Springer, Cham, pp. 611-620, 2023. https://doi.org/10.1007/978-3-031-29857-8_55
 - 7) Kraicha, A., Touil, H., El Akkad, N.E.L. Hybrid Cryptography Based on Constellations: Cassiopeia, in: Motahhir, S., Bossoufi, B. (eds) Digital Technologies and Applications, Lecture Notes in Networks and Systems, V. 1098, Springer, Cham, pp. 331-341, 2024. https://doi.org/10.1007/978-3-031-68650-4_30
 - 8) Kraicha, A., Touil, H., El Akkad, N.E.L. Hybrid Cryptography Based on Constellations: Sagittarius. Big Data and Internet of Things 24. https://doi.org/10.1007/978-3-031-68650-4_30
 - 9) Kraicha, A., Touil, H., El Akkad, N.E.L. Hybrid Cryptography Based on Planets: Phobos & Deimos. The 3rd International Conference on Energy and Green Computing 2014.