

Protocole de chiffrement de données de transit TLS 1.3 : étude de cas sur un LMS avec Wireshark

Elhadji Mamadou MBAYE (elhadji.m.mbaye@aims-senegal.org)

African Institute for Mathematical Sciences (AIMS), Senegal

Supervised by: Professor Ado Adamou ABBA ARI

July 20, 2024

Submitted in partial fulfilment of the requirements for the award of Master of Science in Mathematical Sciences at AIMS Senegal

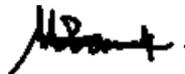


DECLARATION

This work was carried out at AIMS Senegal in partial fulfilment of the requirements for a Master of Science Degree.

I hereby declare that except where acknowledgement is made, this work has never been presented wholly or in part for the award of a degree at AIMS Senegal or any other University.

Student: Elhadji Mamadou Mbaye



Supervisor: Pr. Ado Adamou ABBA ARI

REMERCIEMENTS

Je tiens tout d'abord à exprimer ma profonde gratitude envers mon directeur de mémoire et mon tuteur, Pr. Ado Adamou Abba ARI et M. Nganeron Togdé, pour leur guidance experte, leurs conseils précieux et leur soutien constant tout au long de ce projet. Leurs connaissances approfondies et leur engagement ont été d'une aide inestimable pour mener à bien cette recherche. Mes remerciements vont également à tous les enseignants et membres du corps professoral d'AIMS Senegal, dont les enseignements ont enrichi mes connaissances et ont contribué à façonner ma compréhension des aspects techniques et théoriques du domaine. Enfin, je suis reconnaissant envers ma famille et mes amis pour leur soutien indéfectible et leur encouragement tout au long de cette période d'études.

DÉDICACES

À mes parents, pour leur soutien indéfectible, leurs encouragements constants et leur amour inconditionnel tout au long de ce parcours académique. Vous avez été ma source de motivation et de persévérance.

À mes professeurs et mentors, pour leur guidance, leurs conseils avisés et leur inspiration. Votre expertise et votre dévouement ont grandement contribué à la réussite de ce travail.

À mes amis et collègues, pour leur compréhension, leur camaraderie et leurs encouragements. Merci d'avoir été là durant les moments de doute et de célébration.

Et enfin, à tous ceux qui croient en la puissance de la connaissance et de la recherche pour un monde plus sûr et plus connecté.

Abstract

In a world where digital data security is paramount, encrypting data during its transit is an essential necessity. The TLS (Transport Layer Security) protocol has become a standard for ensuring the confidentiality and integrity of communications over the Internet. Specifically, version 1.3 of this protocol represents a significant advancement in terms of security and efficiency.

Despite the advancements brought by TLS 1.3, several questions remain about the real effectiveness of this protocol in various environments and its ability to counter increasingly sophisticated threats. This thesis explores the capability of TLS 1.3 to secure sensitive data in a practical scenario, through network traffic analysis with Wireshark, and compares the results to the security recommendations of ANSSI. Additionally, a comparison with TLS 1.2, using OpenSSL, was conducted to highlight the improvements brought by TLS 1.3.

The results obtained demonstrated that TLS 1.3 is effective in securing the exchange of sensitive data of the Moodle E-AIMS platform, meeting ANSSI's security standards. The comparison with TLS 1.2 revealed significant improvements in performance, notably due to a simplified handshake and more efficient encryption algorithms, which reduce latency and enhance the user experience.

The practical implications of this study highlight the importance of maintaining rigorous security practices and regularly updating server configurations. Theoretically, these results validate the advancements of TLS 1.3 in terms of security and performance compared to TLS 1.2 and enrich the existing literature on the protocol's benefits. By identifying and analyzing potential vulnerabilities, this research provides a valuable reference framework for future studies and proposes practical solutions to enhance data security in online educational environments.

Finally, this study makes an original contribution by empirically evaluating TLS 1.3 in a real educational context, proposing an accessible methodology for other researchers and professionals. By providing concrete guidelines to reinforce the security of sensitive data, this research opens new perspectives in the field of online data security.

Keywords: TLS 1.3, TLS 1.2, OpenSSL, data security, encryption, online educational platform, Wireshark, ANSSI, Moodle, PFS, handshake.

Résumé

Dans un monde où la sécurité des données numériques est primordiale, le chiffrement des données durant leur transit est une nécessité incontournable. La norme TLS (Transport Layer Security) s'est imposée comme un standard pour assurer la confidentialité et l'intégrité des communications sur Internet. Plus précisément, la version 1.3 de ce protocole représente une avancée significative en matière de sécurité et d'efficacité.

Malgré les avancées apportées par TLS 1.3, plusieurs questions subsistent quant à l'efficacité réelle de ce protocole dans des environnements variés et ses capacités à contrer des menaces de plus en plus sophistiquées. Ce mémoire se penche sur l'évaluation de la capacité de TLS 1.3 à sécuriser les données sensibles dans un scénario pratique, à travers l'analyse du trafic réseau avec Wireshark et en comparant les résultats aux recommandations de sécurité de l'ANSSI. De plus, une comparaison avec TLS 1.2, utilisant OpenSSL, a été réalisée pour mettre en évidence les améliorations apportées par TLS 1.3.

Les résultats obtenus ont démontré que TLS 1.3 est efficace pour sécuriser les échanges de données sensibles de la plateforme Moodle E-AIMS, respectant les normes de sécurité de l'ANSSI. La comparaison avec TLS 1.2 a révélé des améliorations significatives en termes de performances, notamment grâce à un handshake simplifié et des algorithmes de chiffrement plus efficaces, ce qui réduit la latence et améliore l'expérience utilisateur.

Les implications pratiques de cette étude soulignent l'importance de maintenir des pratiques de sécurité rigoureuses et de procéder à des mises à jour régulières des configurations de serveur. Théoriquement, ces résultats valident les avancées de TLS 1.3 en matière de sécurité et de performance par rapport à TLS 1.2, et enrichissent la littérature existante sur les bénéfices du protocole. En identifiant et en analysant les vulnérabilités potentielles, cette recherche offre un cadre de référence précieux pour les études futures et propose des solutions pratiques pour renforcer la sécurité des données dans les environnements éducatifs en ligne.

Enfin, cette étude apporte une contribution originale en évaluant empiriquement TLS 1.3 dans un contexte éducatif réel, proposant une méthodologie accessible pour d'autres chercheurs et professionnels. En fournissant des directives concrètes pour renforcer la sécurité des données sensibles, cette recherche ouvre de nouvelles perspectives dans le domaine de la sécurité des données en ligne.

Mots-clés : TLS 1.3, TLS 1.2, OpenSSL, Handshake, Moodle, sécurité des données, chiffrement, plateforme éducative en ligne, Wireshark, ANSSI.

Contents

Declaration	i
Remerciements	ii
Dedication	iii
Abstract	iv
Résumé	v
1 Introduction	1
2 Préliminaires	2
2.1 Introduction	2
2.1.1 Chiffrement des Données de Transit	3
2.1.2 Chiffrement symétrique	4
2.1.3 La cryptographie asymétrique	7
2.1.4 Fonctionnement de TLS	12
2.1.5 Pré-requis pour l'Utilisation de TLS 1.3	17
3 Revue de littérature	27
3.1 Introduction	27
3.2 Revue de Littérature sur TLS 1.3	27
3.2.1 Étude sur les Performances et la Sécurité	28
3.2.2 Étude sur les Vulnérabilités	28

3.2.3	Méthodologie pour les Tests contrôlés	28
3.2.4	Méthode de capture et de décryptage	34
3.2.5	Analyse des échanges entre client et serveur avec Wireshark	35
3.2.6	Étude des résultats de l'analyse de TLS 1.3 par rapport aux recommandations de l'ANSSI	40
3.2.7	Comparaison pratique entre TLS 1.2 et TLS 1.3 en termes de complexité cryptographique:	41
4	Résultats obtenus	46
4.1	Introduction	46
4.2	Résultats Théoriques	46
4.2.1	La performance et la sécurité	46
4.2.2	La Vulnérabilité	47
4.3	Résultats pratiques	48
4.4	Limite de la Recherche	48
5	Conclusion	50
	References	53

Chapter 1

Introduction

Dans un monde où les données numériques croissent de façon exponentielle, la sécurité de celles-ci durant leur transit dans le réseau internet est devenue une nécessité. Selon Statistica[6], deux tiers de la population mondiale utilise internet. Le protocole TLS (Transport Layer Security)[16] s'est imposé comme une norme essentielle pour assurer la confidentialité et l'intégrité des données. La version 1.3 de ce protocole représente une avancée majeure en termes de sécurité et d'efficacité. TLS 1.3[15] a été conçu pour remédier aux faiblesses de ses prédécesseurs en offrant un niveau de sécurité renforcé.

Malgré les améliorations apportées par TLS 1.3, des questions persistent quant à son efficacité réelle dans divers environnements et sa capacité à contrer les menaces de sécurité avancées[21]. Notre recherche explore si TLS 1.3 peut garantir la sécurité des données sensibles dans un cadre contrôlé, utilisant un certificat auto-signé , et examine les défis techniques associés à son implémentation. Nous évaluons également sa conformité aux recommandations de sécurité émises par des organismes comme l'ANSSI[1].

Ce mémoire vise à fournir une analyse complète du protocole TLS 1.3, de ses applications pratiques et des mesures de sécurité qui lui sont associées. Nous soulignons les défis actuels et futurs de la sécurisation des données en transit, tout en suggérant des pistes de recherche pour améliorer ce protocole crucial pour la confidentialité des communications numériques.

Dans cette introduction générale, nous posons d'abord les bases nécessaires à la compréhension de notre étude sur le protocole TLS 1.3, en définissant les concepts essentiels du chiffrement en transit et de l'analyse du trafic TLS 1.3. Le deuxième chapitre présente une revue de la littérature explorant les avancées significatives de TLS 1.3 par rapport à ses prédécesseurs, notamment en termes de sécurité renforcée et d'efficacité opérationnelle. Au troisième chapitre, nous discutons des résultats obtenus dans notre étude, basés sur l'analyse du trafic réseau capturé via Wireshark sur la plateforme E-AIMS. Cette analyse met en lumière les performances de TLS 1.3 et sa conformité aux normes de sécurité de l'ANSSI. Enfin, le chapitre final conclut en résumant les implications des résultats obtenus, en identifiant les défis techniques rencontrés et en proposant des orientations pour de futures recherches en sécurité informatique.

Chapter 2

Préliminaires

2.1 Introduction

Les fondements du chiffrement et des protocoles de sécurité revêtent une importance cruciale dans le domaine de la sécurité des données numériques. Cette section explorera plusieurs concepts clés nécessaires à la compréhension approfondie du protocole TLS 1.3 et de son application dans un environnement éducatif en ligne tel que E-AIMS.

Nous commencerons par examiner le chiffrement des données de transit, en mettant en lumière son rôle essentiel dans la protection des communications en ligne. Ensuite, nous aborderons le chiffrement symétrique, notamment l'algorithme AES, qui constitue un pilier de la sécurisation des données grâce à sa robustesse et à son efficacité.

La cryptographie asymétrique, représentée par des technologies telles que RSA[2] et Diffie-Hellman[24], sera également discutée pour sa capacité à gérer l'échange sécurisé des clés sans nécessiter de pré-échange sécurisé. Nous explorerons ensuite le fonctionnement détaillé du protocole TLS (2.1.4), en mettant en lumière les différences entre une connexion TLS classique et TLS 1.3, en soulignant les avancées significatives de cette dernière version en matière de sécurité et de performance.

Enfin, cette section présentera les pré-requis technologiques essentiels à notre étude, incluant Apache pour le serveur web, Wireshark[4] pour l'analyse du trafic réseau, OpenSSL[16] pour la gestion des certificats et des connexions sécurisées, ainsi que la plateforme E-AIMS comme cadre d'application pour nos analyses.

2.1.1 Chiffrement des Données de Transit

2.1.1.1 Définition et importance

Le chiffrement des données de transit est un processus important pour la sécurité des informations qui sont envoyées d'un point à un autre sur un réseau. Ce processus implique la transformation des données lisibles (texte en clair) en données codées (texte chiffré) à l'aide d'algorithmes de chiffrement. Les données chiffrées sont incompréhensibles pour quiconque n'ayant pas la clé de déchiffrement appropriée, garantissant ainsi la confidentialité des informations pendant leur transfert.

L'importance du chiffrement des données de transit réside dans la protection contre les interceptions et les accès non autorisés. Que ce soit pour des communications personnelles, des transactions financières ou des échanges de données sensibles entre entreprises, le chiffrement des données de transit assure que les informations restent sécurisées même lorsqu'elles traversent des réseaux potentiellement vulnérables comme l'internet public. Il est également essentiel de se conformer aux réglementations sur la protection des données, comme le RGPD (Règlement Général sur la Protection des Données)[11] en Europe, qui imposent des mesures strictes pour la sécurité des données personnelles.

2.1.1.2 Risques associés à l'absence de chiffrement des données en transit

Les données non chiffrées lors de leur transit sont exposées à diverses menaces, posant ainsi des risques significatifs pour la sécurité des informations et la confidentialité des utilisateurs. Voici quelques-uns des principaux dangers associés à ce scénario :

- **Attaques de type "man-in-the-middle" (MITM)** : Ce type d'attaque implique qu'un tiers intercepte la communication entre deux parties et peut potentiellement altérer les données transmises sans que les parties concernées en soient conscientes. En l'absence de chiffrement, les données sont vulnérables à la lecture et à la modification par des attaquants.
- **Vol de données sensibles** : Les données sensibles telles que les informations d'identification, les numéros de carte de crédit et les données personnelles peuvent être compromises si elles sont transmises sans chiffrement. Cela peut conduire à des fraudes, à des vols d'identité et à d'autres formes d'abus.
- **Non-conformité réglementaire** : De nombreuses réglementations exigent que les données personnelles et sensibles soient protégées par des mesures de sécurité appropriées, y compris le chiffrement lors de leur transit. Ne pas se conformer à ces exigences peut entraîner des sanctions légales et des amendes significatives.
- **Interception par des attaquants** : Les attaquants peuvent intercepter les données transitant sur le réseau en utilisant des techniques comme le "sniffing" de réseau, leur permettant ainsi de capturer et d'analyser les paquets de données échangés entre les systèmes.

2.1.2 Chiffrement symétrique

Historiquement, le premier concept de chiffrement à avoir émergé est celui du chiffrement symétrique[2], jouant un rôle crucial dans la sécurisation des communications entre deux parties. En cryptographie symétrique, une même clé est utilisée à la fois pour chiffrer et déchiffrer les données (cf. Figure 2.1). Cette clé doit demeurer confidentielle afin d'assurer la sécurité des échanges, d'où l'appellation de cryptographie à clé secrète pour ce domaine. Le chiffre de César est l'un des premiers algorithmes de cette catégorie et demeure le plus célèbre. Il opère en appliquant une rotation fixe à chaque symbole du texte clair dans l'alphabet utilisé, où cette rotation détermine la clé secrète dans ce contexte[2].

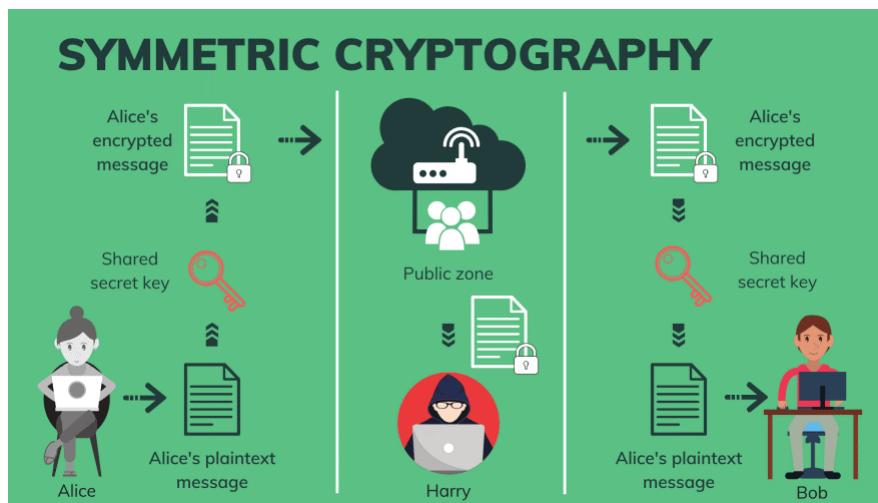


Figure 2.1: Chiffrement symétrique[18]

2.1.2.1 AES (Advanced Encryption Standard)

AES[17] est un algorithme de chiffrement symétrique basé sur le système Rijndael, conçu par Joan Daemen et Vincent Rijmen. Il opère en blocs de 128 bits (16 octets binaires) et utilise des clés de chiffrement de 128, 192 ou 256 bits. L'algorithme AES exécute plusieurs tours composés de transformations spécifiques.

Algorithme Rijndael

Rijndael est l'algorithme de base sur lequel AES est fondé. Développé par Joan Daemen et Vincent Rijmen, il a été choisi par le NIST comme standard pour remplacer le Data Encryption Standard (DES) en raison de sa robustesse, de ses performances et de sa flexibilité.

Durant le processus de sélection de l'AES, Rijndael a été sélectionné pour ses qualités de sécurité et de performance. Il utilise des clés de chiffrement de 128, 192 ou 256 bits, déterminant le nombre de tours d'exécution (10, 12 ou 14 tours respectivement). Avant chaque tour, une clé de tour, appelée RoundKey, est dérivée de la clé de chiffrement initiale.

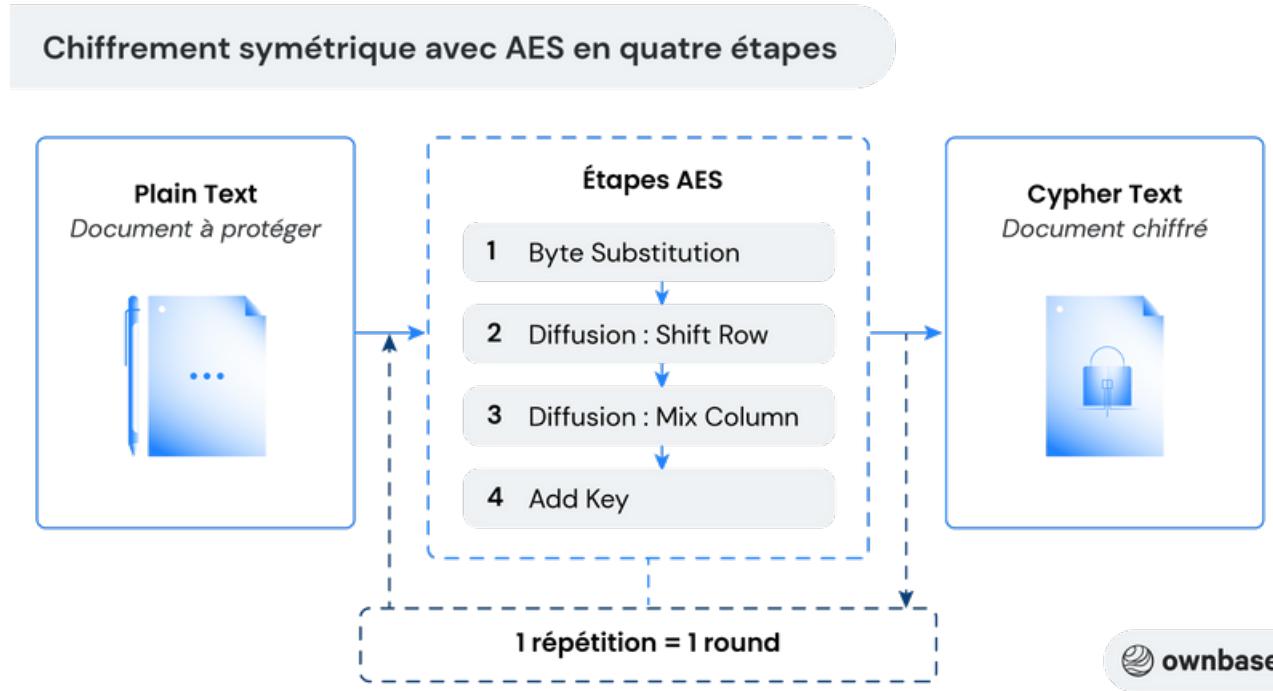


Figure 2.2: AES en quatre étapes

- **ByteSub:** Cette étape consiste à substituer chaque octet (byte) de l'état de données par un autre octet à l'aide d'une table de substitution appelée S-Box (Substitution Box). La S-Box est une table fixe dérivée de la clé de chiffrement. Elle est conçue pour introduire de la non-linéarité dans le processus de chiffrement, renforçant ainsi la sécurité de l'algorithme. Chaque octet de l'état est remplacé par le contenu correspondant de la S-Box.

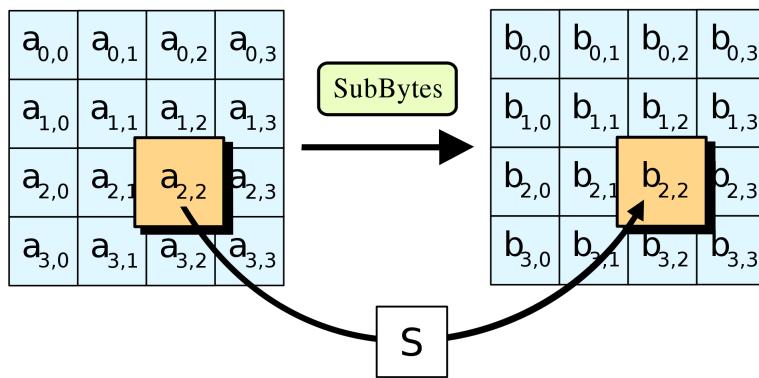


Figure 2.3: Étape Bytesub

- **ShiftRow :** Dans Rijndael, l'étape ShiftRow consiste à déplacer les octets dans chaque ligne du bloc de données. Cette opération, combinée avec MixColumn, fait partie du processus de

diffusion. Son objectif est de perturber l'organisation des bits dans les données, introduisant ainsi de la confusion pour renforcer la sécurité du chiffrement

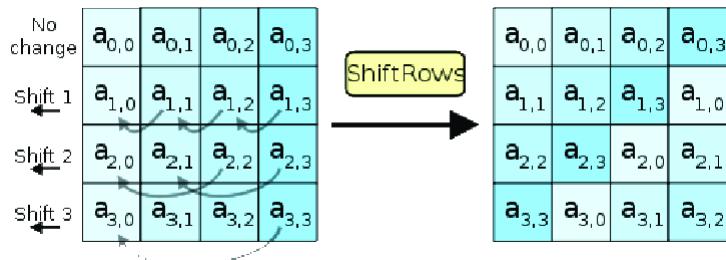


Figure 2.4: Étape shiftrow

- **MixColumn** : En AES, MixColumn mélange les octets du bloc de données à l'aide d'une transformation linéaire mathématique. Cette étape vise à impliquer plusieurs bits dans le chiffrement d'un seul, facilitant ainsi la liaison rapide de tous les éléments du bloc à travers des transformations matricielles. Le processus de déchiffrement est rendu plus complexe sans la connaissance de la clé correspondante.

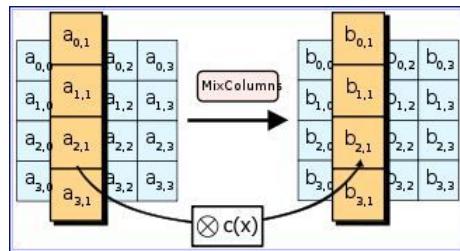


Figure 2.5: Étape mixcolumn

- **AddKey** : Dans AES, AddKey combine la clé de tour actuelle avec les valeurs du bloc de données. Cette étape est cruciale car elle effectue le chiffrement réel des données. Pour déchiffrer les données, il est impératif de disposer de la clé correspondante pour inverser cette opération.

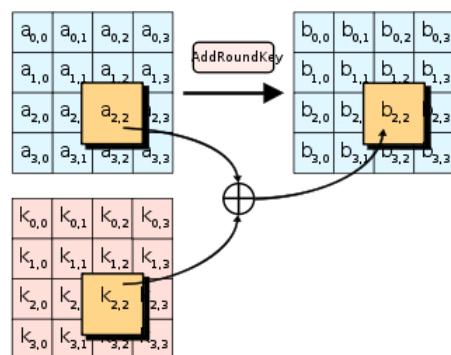


Figure 2.6: Étape d'ajout de la clé

Le processus de chiffrement avec l'Advanced Encryption Standard (AES) implique l'exécution répétée de quatre étapes pour chaque tour en fonction de la longueur de la clé utilisée. Le résultat final, appelé chiffrement, rend le contenu du message ou de l'information totalement indiscernable à l'œil nu. Lors du déchiffrement, Rijndael inverse séquentiellement toutes les étapes des tours effectués, permettant ainsi la reconstruction du texte clair à partir du texte chiffré.

La cryptographie symétrique pose deux défis majeurs dans son application. Tout d'abord, il est crucial d'échanger préalablement la clé secrète entre les parties impliquées pour garantir la sécurité des communications futures. Cette clé doit demeurer privée et connue uniquement des parties concernées, nécessitant un échange sécurisé initial.

Le deuxième défi réside dans la gestion des clés. Étant donné qu'une clé symétrique sécurise uniquement la communication entre deux individus, dans un réseau de n individus, il serait nécessaire de gérer jusqu'à $\frac{n(n-1)}{2}$ clés distinctes. Ainsi, le nombre de clés à gérer augmente quadratiquement avec le nombre d'individus sur le réseau.

Ces deux défis ont été des moteurs importants pour le développement de la cryptographie asymétrique à la fin des années 1970.

2.1.3 La cryptographie asymétrique

La cryptographie asymétrique^[2], aussi appelée cryptographie à clé publique, a été théorisée en 1976 par Whitfield Diffie et Martin Hellman. À l'époque de leur conceptualisation, bien qu'ils aient introduit le concept, aucune implémentation concrète n'était disponible. Ce n'est qu'avec l'invention du RSA et du protocole d'échange de clés Diffie-Hellman que les premières applications pratiques de la cryptographie asymétrique ont été développées.

Contrairement à la cryptographie symétrique qui utilise une seule clé pour le chiffrement et le déchiffrement, la cryptographie asymétrique repose sur l'utilisation de deux clés distinctes. Le destinataire choisit une clé privée pour le déchiffrement, la maintenant secrète. À l'aide d'une fonction à sens unique (ou fonction à trappe), il génère une clé publique utilisée par les expéditeurs pour chiffrer les messages.

L'utilisation de fonctions à trappe est essentielle en cryptographie asymétrique car elles permettent des opérations simples mais sont difficiles à inverser sans la trappe, qui est la clé privée dans ce contexte. La clé publique est distribuée largement, permettant à quiconque de chiffrer des messages que seul le destinataire légitime, possédant la clé privée correspondante, peut déchiffrer.

En résumé, la cryptographie asymétrique élimine le besoin de pré-partager un secret pour sécuriser les communications, offrant ainsi une méthode plus flexible et sécurisée pour échanger des informations confidentielles

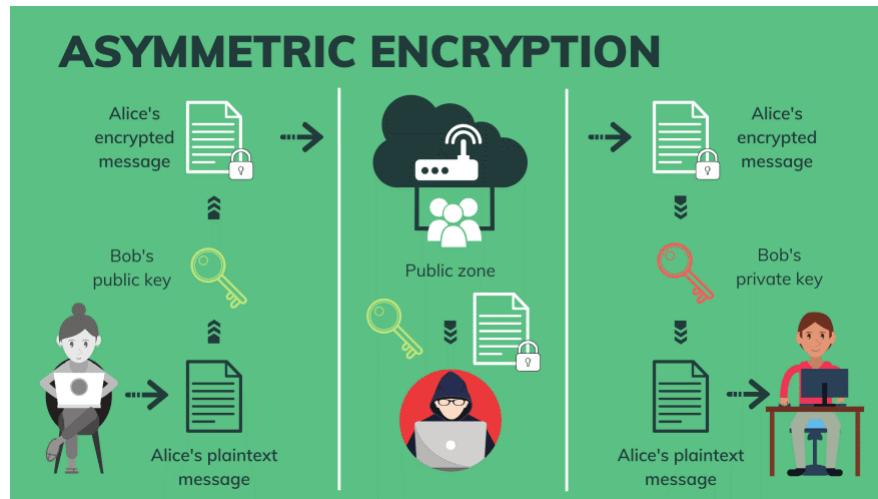


Figure 2.7: Chiffrement asymétrique[18]

Cependant, l'utilisation de clés publiques présente des défis. Il est difficile de vérifier avec certitude l'authenticité de la personne qui diffuse sa clé publique, ouvrant ainsi la possibilité à un attaquant de se faire passer pour un utilisateur légitime. Dans ce scénario, un attaquant pourrait intercepter et déchiffrer les messages destinés à la victime. Pour contrer ce risque, il est essentiel de mettre en place des infrastructures de gestion des clés publiques sécurisées.

En pratique, les méthodes de chiffrement asymétrique sont significativement plus lentes que celles du chiffrement symétrique, principalement en raison de la complexité des opérations mathématiques nécessaires et de la longueur des clés recommandées pour assurer un niveau de sécurité adéquat (par exemple, l'ANSSI recommande une taille minimale de clé RSA de 2048 bits).

De ce fait, les méthodes de chiffrement asymétrique ne sont pas adaptées pour sécuriser des canaux de communication à haut débit. En revanche, elles sont particulièrement efficaces pour sécuriser l'échange initial d'une clé de chiffrement symétrique, permettant ensuite l'utilisation d'algorithmes de chiffrement par blocs plus rapides pour le transfert de données.

L'émergence de la cryptographie asymétrique a introduit le concept de signature électronique. Tout comme une signature manuscrite apposée sur un document, une signature électronique peut garantir deux propriétés cryptographiques essentielles : l'authentification (identification de l'auteur du document signé) et l'intégrité (vérification que le document n'a pas été altéré après signature). La Figure 2.8 présente un schéma général de signature numérique.

En utilisant sa clé privée pour générer la signature, l'auteur permet à quiconque connaissant la clé publique associée et le document original de vérifier facilement la validité de la signature. La validité de la signature confirme ainsi que le détenteur de la clé privée a bien signé le document et que ce dernier n'a pas été altéré après la signature.

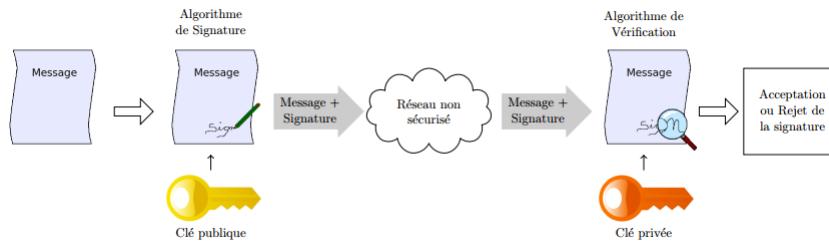


Figure 2.8: Signature numérique[2]

Ainsi, la signature électronique démontre de manière appropriée les propriétés cryptographiques nécessaires. Dans la section suivante, nous examinerons deux des algorithmes les plus couramment utilisés en cryptographie asymétrique et dans TLS : RSA et Diffie-Hellman.

2.1.3.1 RSA (Rivest-Shamir-Adleman)

L'algorithme RSA[2] a été développé en 1977 par Ronald Rivest, Adi Shamir et Leonard Adleman, représentant ainsi la première méthode de cryptographie à clé publique, concept introduit précédemment par Whitfield Diffie et Martin Hellman. Aujourd'hui, RSA est largement reconnu comme l'un des systèmes de cryptographie asymétrique les plus utilisés à travers le monde, bénéficiant d'une étude intensive tant sur le plan théorique que pratique.

La sécurité de RSA repose sur la difficulté supposée du problème de factorisation des grands nombres premiers. À ce jour, aucune méthode de résolution n'a démontré une complexité meilleure que sous-exponentielle, telles que le crible algébrique ou le Number Field Sieve.

Le plus grand module RSA factorisé à ce jour, utilisant des calculs distribués, possède une longueur de 768 bits. Pour prévenir la factorisation des modules, les tailles de module utilisées en pratique sont généralement d'au moins 1024 bits, voire 2048 bits.

2.1.3.1.1 Génération des clés

Le module public N (ou module RSA) est défini comme le produit de deux grands nombres premiers aléatoires p et q . La taille de N , exprimée en bits, est notée n . L'indicatrice d'Euler de N est alors $\phi(N) = (p - 1) \cdot (q - 1)$ [2].

L'exposant public e est choisi de telle sorte que $\gcd(e, \phi(N)) = 1$. Le couple $K_p = (N, e)$ forme la partie publique de la clé. La partie privée K_s consiste en l'exposant privé d , calculé comme l'inverse de e dans le groupe multiplicatif $(\mathbb{Z}/N\mathbb{Z})^*$, c'est-à-dire $d \cdot e \equiv 1 \pmod{\phi(N)}$ [2].

2.1.3.1.2 Chiffrement RSA

Le chiffrement RSA d'un message m consiste en une simple exponentiation modulaire avec l'exposant public :

$$C = m^e \pmod{N}$$

Le déchiffrement est réalisé de manière similaire, mais en utilisant l'exposant privé d :

$$m = C^d \mod N[2]$$

En l'absence d'erreurs pendant le chiffrement, la transmission ou le déchiffrement, \tilde{m} sera égal à m .

Une méthode alternative basée sur le théorème des restes chinois a été proposée par J.-J. Quisquater et C. Couvreur pour effectuer efficacement l'opération de déchiffrement, et elle est spécifiée dans la norme PKCS (Public-Key Cryptography Standards).

Exemple :

Considérons les nombres premiers $p = 11$ et $q = 17$. Ainsi, $n = 187$ et $\phi(n) = (11 - 1)(17 - 1) = 160$. Avec $e = 7$ et $d = 23$, Alice rend public le couple $(187, 7)$.

Bernard souhaite envoyer à Alice un message chiffré inférieur à $n = 187$, par exemple la date de l'anniversaire de Cédric, 10. Il calcule donc $10^7 \equiv 175 \mod 187$ et envoie le résultat 175 à Alice.

Alice calcule $175^{23} \mod 187$ comme suit :

- $175^2 \equiv 163 \mod 187$, donc $175^4 \equiv 144 \mod 187$.
- $175^8 \equiv 166 \mod 187$, $175^{16} \equiv 67 \mod 187$.
- $175^{23} \equiv 10 \mod 187$.

Alice retrouve donc correctement le message initial 10[5].

2.1.3.1.3 Signature RSA

La signature RSA S d'un message m est calculée par une exponentiation modulaire avec l'exposant privé : $S = m^d \mod N[2]$.

Pour vérifier la validité de cette signature, on utilise la clé publique du signataire :

$$m \stackrel{?}{=} S^e \mod N[2]$$

En pratique, on signe généralement une empreinte du message m plutôt que le message lui-même. On utilise une fonction de hachage H pour calculer cette empreinte :

$$m' = H(m)$$

La signature devient alors :

$$S = m'^d \mod N[2]$$

Le calcul de m' est également utilisé pour vérifier la signature. L'utilisation d'une fonction de hachage est combinée avec des schémas de remplissage comme RSA-OAEP pour le chiffrement et RSA-PSS pour la signature.

- **Sécurité de RSA**

La sécurité de RSA repose sur la difficulté de factoriser le module de chiffrement N pour retrouver les nombres premiers p et q . Les attaques contre RSA sont principalement basées sur cette factorisation, un problème difficile pour de grands nombres premiers. La taille des clés est cruciale pour renforcer la sécurité de RSA.

2.1.3.1.4 Diffie-Hellman (DH)

DH[19] est un protocole de chiffrement à clé publique utilisé pour l'échange sécurisé de clés secrètes sur un canal non sécurisé. Il permet à deux parties de convenir d'une clé de session secrète commune utilisée pour chiffrer les communications ultérieures.

L'algorithme d'échange de clés Diffie-Hellman résout le problème de l'échange sécurisé de clés dans un environnement où une communication directe sécurisée n'est pas disponible. Alice et Bob peuvent générer une clé partagée sans divulguer cette clé à un adversaire interceptant leurs communications.

Les étapes de l'algorithme sont les suivantes :

$$A \equiv g^a \pmod{p} \text{ pour Alice, et } B \equiv g^b \pmod{p} \text{ pour Bob}$$

Après avoir échangé ces valeurs calculées (publiquement mais de manière sécurisée), ils calculent leur clé partagée :

$$A' \equiv B^a \pmod{p} \text{ pour Alice, et } B' \equiv A^b \pmod{p} \text{ pour Bob}$$

Les valeurs calculées A' et B' sont identiques, ce qui constitue leur clé de session commune K_{AB} .

Création de paramètres publics	
Une partie de confiance choisit et publie un (grand) premier p et un entier g ayant un grand ordre premier dans \mathbb{F}_p^* .	
Première étape de Calculs privés	
Alice	Bob
Choisis un entier secret a et calcule $A \equiv g^a \pmod{p}$	Choisis un entier secret b et calcule $B \equiv g^b \pmod{p}$
Echange public de A et B	
Alice envoie A à Bob	Bob envoie B à Alice
Deuxième étape de calculs privés	
Alice	Bob
Calcule de $B^a \pmod{p}$	Calcule de $A^b \pmod{p}$
La valeur du secret partagé est $B^a \equiv (g^b)^a \equiv g^{ab} \equiv (g^a)^b \equiv A^b \pmod{p}$	

Table 2.1: Processus de création de paramètres publics et d'échange de clés.

L'algorithme de Diffie-Hellman est résumé ci-dessous.

Exemple : Alice et Bob choisissent $p = 941$ et $g = 627$. Alice choisit $a = 347$ et calcule $A \equiv 627^{347} \pmod{941} \equiv 390$. Bob choisit $b = 781$ et calcule $B \equiv 627^{781} \pmod{941} \equiv 691$. Alice envoie $A = 390$ à Bob et Bob envoie $B = 691$ à Alice via un canal non sécurisé. Bien que A et B soient publics, les valeurs secrètes $a = 347$ et $b = 781$ restent confidentielles. Alice et Bob calculent ensuite :

$$470 \equiv 627^{347 \cdot 781} \equiv A^b \cdot B^a \pmod{941}$$

Ainsi, 470 est leur secret partagé.

Supposons qu'Ève observe tout l'échange. Pour reconstituer le secret partagé d'Alice et Bob, elle doit résoudre l'une des congruences suivantes :

$$627^a \equiv 390 \pmod{941} \quad \text{ou} \quad 627^b \equiv 691 \pmod{941}$$

Si Ève peut résoudre le problème du logarithme discret, elle peut trouver a ou b , puis calculer facilement la clé partagée 627^{ab} . La sécurité de Diffie-Hellman repose sur la difficulté de ce problème.

Le protocole Diffie-Hellman (DH) est fondamental pour établir des communications sécurisées sur des canaux non sécurisés. Cependant, pour renforcer la sécurité, le Diffie-Hellman Éphémère (DHE) est souvent utilisé dans les applications modernes. Le DHE utilise des clés temporaires pour chaque session, offrant ainsi une confidentialité persistante même si une clé temporaire est compromise.

- **Diffie-Hellman Éphémère (DHE)[24]** : Le DHE fonctionne de manière similaire au protocole DH classique, mais utilise des clés temporaires générées pour chaque session. Cela inclut la génération de clés privées temporaires a et b , et le calcul de clés partagées temporaires $K = B^a \pmod{p}$ et $K = A^b \pmod{p}$.
- **L'Importance du DHE dans TLS 1.3[1]** : TLS 1.3 adopte systématiquement le DHE pour garantir la confidentialité persistante, assurant que les sessions passées restent sécurisées même en cas de compromission des clés à long terme.

2.1.4 Fonctionnement de TLS

Dans cette partie, nous allons présenter TLS[16] et ses spécificités par rapport à TLS 1.2. Nous fournirons également une explication des vulnérabilités de sécurité connues, souvent référencées sous la forme de Common Vulnerabilities and Exposures (CVE).

2.1.4.1 Détails d'une connexion TLS classique par rapport à TLS 1.3

TLS (Transport Layer Security), successeur de SSL (Secure Sockets Layer), est un protocole conçu pour sécuriser un canal de communication en fournissant plusieurs services : authentification unilatérale ou mutuelle, confidentialité, intégrité et non-rejet des données échangées de bout en bout.

Cette couche de sécurité peut être appliquée à divers canaux de communication, garantissant la transmission ordonnée des données. Pratiquement, SSL/TLS est principalement utilisé avec le protocole de transport TCP pour sécuriser des versions existantes (par exemple HTTPS = HTTP + TLS).

Le fonctionnement de TLS repose sur un processus étape par étape appelé handshake TLS. Durant ce processus, le client et le serveur s'authentifient mutuellement, échangent des clés de session et établissent un canal de communication sécurisé.

Le handshake commence par le client envoyant une liste d'algorithmes de chiffrement supportés, que le serveur utilise pour choisir un algorithme commun et envoyer son certificat (Certificate). Une fois l'échange de clés effectué à l'aide d'un algorithme asymétrique, une clé de session symétrique est générée pour chiffrer la session de communication.

TLS 1.3, la version la plus récente du protocole, apporte plusieurs améliorations, notamment une réduction du nombre d'étapes du handshake et une sécurité renforcée grâce à l'utilisation du Perfect Forward Secrecy (PFS). Cela garantit que même si la clé privée du serveur est compromise, les clés de session ne peuvent pas être compromises.

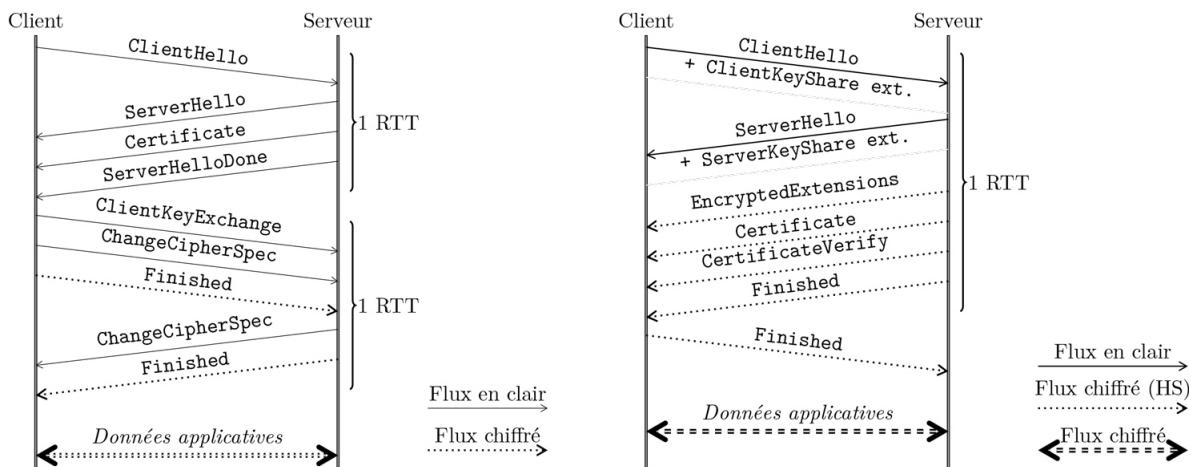


Figure 2.9: Comparaison de l'établissement de session TLS avec TLS 1.2 (à gauche) et TLS 1.3 (à droite)

2.1.4.1.1 Connexion classique

- **ClientHello:** Le client initie la connexion en envoyant un message ClientHello au serveur. Ce message comprend :

- Les versions de TLS supportées.
 - Une liste de suites cryptographiques supportées.
 - Les extensions supportées.
 - Un nonce aléatoire (un nombre utilisé une seule fois).
- **ServerHello:** Le serveur répond avec un message ServerHello qui contient :
 - La version de TLS choisie.
 - La suite cryptographique choisie.
 - Les extensions choisies.
 - Un nonce aléatoire.
 - **Certificate:** Le serveur envoie ensuite un message Certificate, qui contient le certificat numérique du serveur. Ce certificat est utilisé pour authentifier l'identité du serveur auprès du client.
 - **ServerHelloDone:** Après avoir envoyé le certificat, le serveur envoie un message ServerHelloDone pour indiquer qu'il a terminé la phase d'envoi des messages nécessaires pour l'établissement de la connexion.
 - **ClientKeyExchange:** Le client répond avec un message ClientKeyExchange, qui contient les informations nécessaires pour que le serveur puisse calculer la clé partagée. La méthode exacte dépend de l'algorithme d'échange de clés choisi, mais elle peut inclure, par exemple, une clé publique ou un secret chiffré avec la clé publique du serveur.
 - **ChangeCipherSpec (Client):** Le client envoie un message ChangeCipherSpec pour indiquer qu'il va commencer à utiliser les clés symétriques dérivées à partir de la clé partagée pour chiffrer les messages suivants.
 - **Finished (Client):** Le client envoie ensuite un message Finished, qui est chiffré avec les clés symétriques dérivées. Ce message inclut un hash de toutes les communications précédentes pour permettre au serveur de vérifier l'intégrité de l'échange.
 - **ChangeCipherSpec (Server):** Après avoir reçu et vérifié le message Finished du client, le serveur envoie son propre message ChangeCipherSpec pour indiquer qu'il commencera également à utiliser les clés symétriques dérivées pour chiffrer les messages suivants.
 - **Finished (Server):** Le serveur envoie un message Finished, également chiffré avec les clés symétriques dérivées, incluant un hash de toutes les communications précédentes pour que le client puisse vérifier l'intégrité de l'échange.

2.1.4.1.2 Différences et améliorations

Dans les versions précédentes, le client initiait la connexion avec le message ClientHello, où il spécifiait les algorithmes et les fonctionnalités supportés. Le serveur choisissait alors parmi ces paramètres pour la session et présentait son certificat. Ensuite, un second échange était

nécessaire pour l'établissement des clés avant que les données ne puissent être échangées. En revanche, dans TLS 1.3 par défaut, l'échange de clés se fait avec les extensions ClientKeyShare et ServerKeyShare (incluses respectivement dans les messages ClientHello et ServerHello).

TLS 1.3 négocie les algorithmes à trois endroits différents : l'extension `supported_groups` pour l'échange de clés, l'extension `signature_algorithms` pour l'authentification du serveur et les suites cryptographiques pour la protection des messages à l'aide d'un algorithme de chiffrement authentifié. Cela permet de réaliser la négociation des paramètres, l'échange de clés et l'authentification du serveur en un seul aller-retour, réduisant ainsi le temps nécessaire pour établir la session à 1 RTT (Round-Trip Time).

Les principales modifications par rapport à TLS 1.2 incluent :

- Le support exclusif des échanges de clés basés sur DHE (Diffie-Hellman éphémère) ou ECDHE (Elliptic Curve Diffie-Hellman éphémère), excluant en pratique le chiffrement RSA.
- Les groupes utilisés pour l'échange de clés (corps finis pour DHE ou courbes elliptiques pour ECDHE) ne peuvent plus être définis de manière arbitraire par le serveur, mais doivent être choisis parmi une liste prédéfinie de groupes.
- Si le client propose des parts secrètes pour des groupes non supportés par le serveur, ce dernier envoie un message `HelloRetryRequest`, imposant ainsi le groupe à utiliser, ce qui peut nécessiter un deuxième aller-retour, similaire à TLS 1.2 (c'est-à-dire 2 RTT). Dans ce cas, l'authentification du serveur se fait avec un message `CertificateVerify`, similaire à l'authentification client de TLS 1.2.

2.1.4.2 Définitions de CVE

Les CVE[10] (Common Vulnerabilities and Exposures) sont des identificateurs standardisés pour les failles de sécurité dans les logiciels et les protocoles. Un CVE est un identifiant unique attribué à une vulnérabilité de sécurité spécifique. Il se compose de quatre champs :

- Année : L'année de découverte de la vulnérabilité.
- Numéro : Un numéro séquentiel attribué à la vulnérabilité dans l'année de découverte.
- Tiret : Un séparateur.
- Numéro de version : Un numéro optionnel qui indique les mises à jour apportées à la description de la vulnérabilité.

Par exemple, CVE-2021-27856 fait référence à une vulnérabilité découverte en 2021 et identifiée par le numéro 27856.

1. CVE (Common Vulnerabilities and Exposures) spécifiques à TLS 1.3

- **CVE-2020-19722[13] : Vulnérabilité par canal latéral dans les bibliothèques TLS**

Cette vulnérabilité, découverte en 2020, affectait plusieurs bibliothèques TLS populaires, y compris OpenSSL, BoringSSL et GnuTLS. Elle permettait à un attaquant de récupérer des informations sensibles telles que des clés de chiffrement en observant les variations de temps d'exécution du processeur pendant le traitement des messages TLS 1.3.

Explication du lien entre CVE-2020-19722 et RSA :

La vulnérabilité CVE-2020-19722 exploite une faille dans l'implémentation de l'algorithme de Montgomery, qui est souvent utilisé pour accélérer les opérations de multiplication modulaire dans les bibliothèques cryptographiques, y compris celles qui implémentent l'algorithme RSA. L'exploitation réussie de cette vulnérabilité pourrait permettre à un attaquant de récupérer des informations sur les clés RSA utilisées par le serveur, telles que les exposants ou les modules. Ces informations pourraient ensuite être utilisées pour déchiffrer le trafic TLS chiffré ou pour forger des signatures numériques.

Impact sur les clés RSA :

Bien que la vulnérabilité CVE-2020-19722 ne compromette pas directement les clés RSA elles-mêmes, elle peut les rendre plus vulnérables aux attaques par canal latéral. Cela signifie qu'un attaquant pourrait potentiellement récupérer des informations sur les clés RSA en observant le comportement du serveur pendant les opérations cryptographiques.

Mesures d'atténuation pour protéger les clés RSA :

En plus des mesures d'atténuation générales recommandées pour CVE-2020-19722, il est important de prendre des mesures spécifiques pour protéger les clés RSA :

- Utiliser des clés RSA de longueur adéquate : Générer et utiliser des clés RSA d'une longueur de bit suffisante (au moins 2048 bits) pour résister aux attaques par force brute actuelles et futures.
- Stocker les clés RSA en toute sécurité : Protéger les clés RSA privées en les stockant dans des modules de sécurité matériels (HSM) ou en utilisant des techniques de chiffrement et d'authentification robustes.
- Mettre à jour régulièrement les bibliothèques cryptographiques : Appliquer régulièrement les mises à jour de sécurité des bibliothèques cryptographiques pour corriger les vulnérabilités connues, y compris celles liées aux algorithmes de multiplication modulaire comme l'algorithme de Montgomery.

- **CVE-2022-22898[9] : Vulnérabilité de déni de service (DoS) dans les bibliothèques TLS**

La vulnérabilité CVE-2022-22898 est une vulnérabilité de type déni de service (DoS) qui affecte certaines bibliothèques TLS. Cette vulnérabilité permet à un attaquant d'envoyer des paquets TLS malformés à un serveur vulnérable, ce qui peut entraîner le blocage du serveur ou la consommation excessive de ressources système. La vulnérabilité est classée comme un risque modéré et nécessite une attention particulière pour la corriger.

2. Explication technique de la vulnérabilité

La vulnérabilité réside dans la façon dont certaines bibliothèques TLS traitent les paquets TLS contenant des options malformées. Plus précisément, la vulnérabilité découle d'une erreur dans la gestion des options TLS "recherche de nom de serveur" (SNI) et "extension de point de terminaison" (Endpoint Extension). Un attaquant peut exploiter cette vulnérabilité en envoyant des paquets TLS malformés contenant des options SNI ou Endpoint Extension de grande taille ou invalides. Ces paquets peuvent submerger le serveur vulnérable, entraînant son blocage ou une consommation excessive de ressources système, ce qui peut affecter la disponibilité des services et compromettre les opérations normales.

3. Résolution de la vulnérabilité

La résolution de la vulnérabilité CVE-2022-22898 nécessite la mise à jour des bibliothèques TLS affectées vers des versions corrigées. Les fournisseurs de bibliothèques TLS ont publié des correctifs qui corrige l'erreur de traitement des options TLS malformées, éliminant ainsi la possibilité d'exploitation par déni de service.

4. Mesures d'atténuation supplémentaires

En plus de la mise à jour des bibliothèques TLS, il est recommandé de mettre en œuvre les mesures d'atténuation suivantes pour réduire davantage le risque d'exploitation :

- Mettre en place des pares-feux et des systèmes de détection d'intrusion (IDS) : Déployer des pares-feux et des IDS pour filtrer le trafic malveillant et détecter les tentatives d'exploitation de la vulnérabilité.
- Limiter le nombre de connexions TLS simultanées : Configurez les serveurs pour limiter le nombre de connexions TLS simultanées qu'un client peut établir, ce qui peut aider à atténuer l'impact d'une attaque par déni de service.
- Surveiller les performances du serveur : Surveiller en permanence les performances du serveur pour détecter toute activité anormale qui pourrait indiquer une attaque par déni de service.

2.1.5 Pré-requis pour l'Utilisation de TLS 1.3

2.1.5.1 Introduction

Pour la mise en œuvre et l'analyse de TLS 1.3, une compréhension basique des environnements Apache sur Linux, de la plateforme E-AIMS sur Linux, d'OpenSSL et de l'outil d'analyse de trafic Wireshark est nécessaire. Ces outils jouent un rôle crucial dans la configuration, le déploiement et la vérification des mécanismes de sécurité liés à TLS 1.3.

2.1.5.2 Présentation d'Apache sur Linux

Apache HTTP Server, souvent simplement appelé Apache, est un serveur web open source largement utilisé dans le monde entier. Initialement développé en 1995, Apache est utilisé pour

héberger des sites web et des applications en ligne.

Caractéristiques principales :

- **Gratuit et Open Source** : Apache est distribué sous une licence open source, ce qui permet une utilisation, une modification et une distribution libre sans coût.
- **Multi-plateforme** : Compatible avec diverses plates-formes telles que Linux, Unix, Windows, et d'autres systèmes d'exploitation.
- **Modularité** : Apache est modulaire, permettant de charger uniquement les modules nécessaires, optimisant ainsi les performances et la sécurité.
- **Support des protocoles** : Apache supporte les protocoles HTTP/1.1, HTTP/2, ainsi que des extensions comme SSL/TLS pour la sécurisation des communications web.
- **Extensibilité et Personnalisation** : Offre une large gamme de modules tiers et une API robuste pour étendre les fonctionnalités et personnaliser le comportement du serveur.
- **Support de la communauté** : Bénéficie d'une grande communauté d'utilisateurs et de développeurs contribuant à son évolution, avec des mises à jour de sécurité et de nouvelles fonctionnalités.

2.1.5.3 Présentation de E-AIMS Moodle

E-AIMS Moodle est une plateforme éducative en ligne conçue pour répondre aux exigences variées et évolutives des institutions d'enseignement et des apprenants, en offrant une infrastructure flexible et sécurisée pour la gestion de cours, la collaboration pédagogique, et l'évaluation des apprentissages.

Nous débutons par une présentation de son architecture et de ses fonctionnalités principales.

2.1.5.3.1 Page d'authentification

Sur la page d'accueil de la plateforme Moodle, saisissez votre nom d'utilisateur (*login*) et votre mot de passe (*password*). Une fois les deux champs saisis, cliquez sur "Connexion".

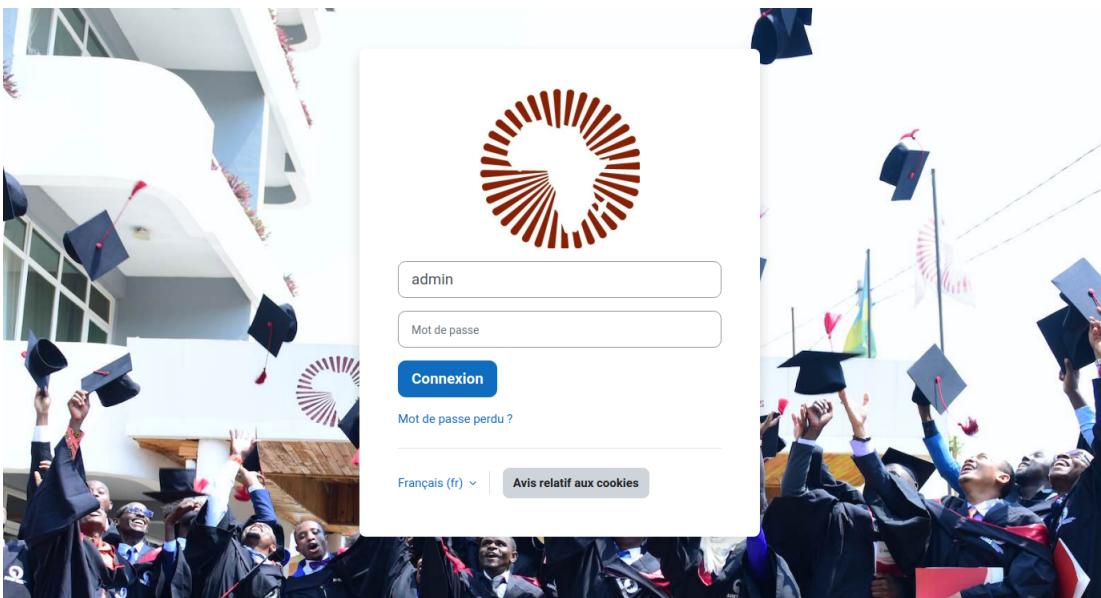


Figure 2.10: Page d'authentification

2.1.5.3.2 Tableau de bord

lorsque vous vous connectez à Moodle, vous accédez à votre tableau de bord.

Lors de votre première connexion, la partie centrale est vide puisque vous n'êtes inscrit à aucun cours. Vous devez rechercher vos cours via le bloc de recherche à droite.

Le tableau de bord dispose de ces éléments par défaut :

- Un bloc **CALENDRIER** où figurent tous les cours et événements concernant l'académie AIMS Sénégal. Vous pouvez déplacer le curseur de la souris sur un des jours en surbrillance pour voir la liste des activités et même cliquer sur une activité pour plus d'informations.

The screenshot shows the E-AIMS dashboard interface. At the top, there is a navigation bar with links for Accueil, Tableau de bord, Mes cours, Administration du site, and Kopere Dashboard. Below this is the 'Tableau de bord' section.

- Éléments consultés récemment:**
 - Annonces du site - E-AIMS SENEGAL
 - Tutorial - Linear algebra and applications
 - Announcements - Bio-Mathématiques
- Calendrier:** A calendar for June 2024. It shows several events listed on the right side of the month, including 'Journée du Donut ...', 'Journée du Cidre', and 'Journée mondiale ...'. A 'Nouvel événement' button is visible at the top right of the calendar area.
- Dernières annonces:** A section listing recent announcements:
 - 29 juin 2024, 01:28 by Admin User: Welcome to E-AIMS
 - Sujets antérieurs ...

Figure 2.11: Tableau de bord

- Une section élément consulté récemment
- Une section annonce du site pour partager des informations concernant tous les utilisateurs
- Une section qui regroupe tous les cours dont l'utilisateur a été inscrit

The screenshot shows the 'Vue d'ensemble des cours' section of the E-AIMS dashboard. At the top, there are four filter buttons: 'Tout', 'Rechercher', 'Trier par nom de cours', and 'Carte'. Below these are two course blocks:

- Classical Mechanics** (BLOCK 3): Includes a diagram comparing classical and quantum physics, showing a person climbing a hill versus tunneling through it. It also includes a quantum mechanics diagram showing an electron wave and potential energy.
- Computer security 1: Networking Security** (BLOCK 4): Shows a complex network visualization with multiple nodes and connections.

Figure 2.12: Bloc Annonces et Cours

2.1.5.3.3 Page d'accueil

la page d'accueil concerne principalement des informations académiques, des publications de travaux de recherche et des cours.

Elle est divisée en six parties :

- **Entête**

L'entête est composée de la barre principale puis d'un rappel sur les valeurs de AIMS.

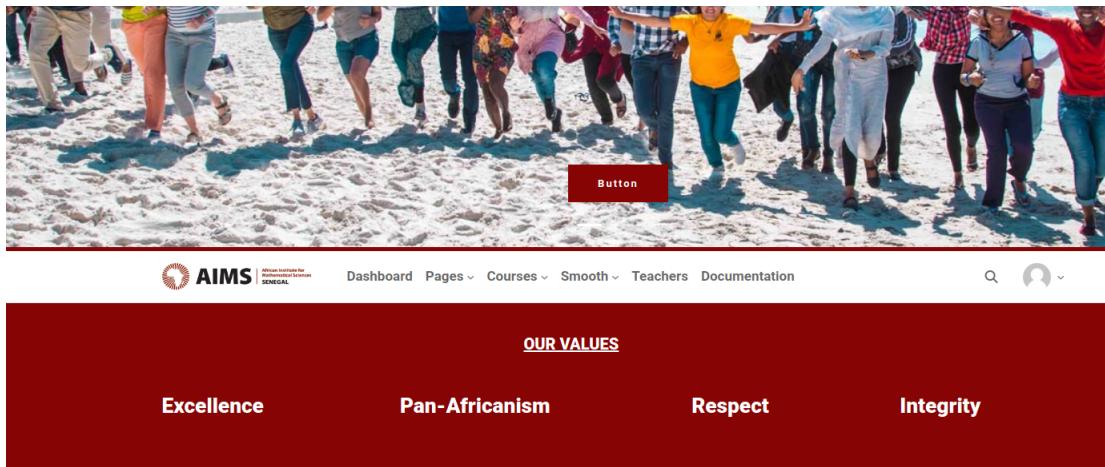
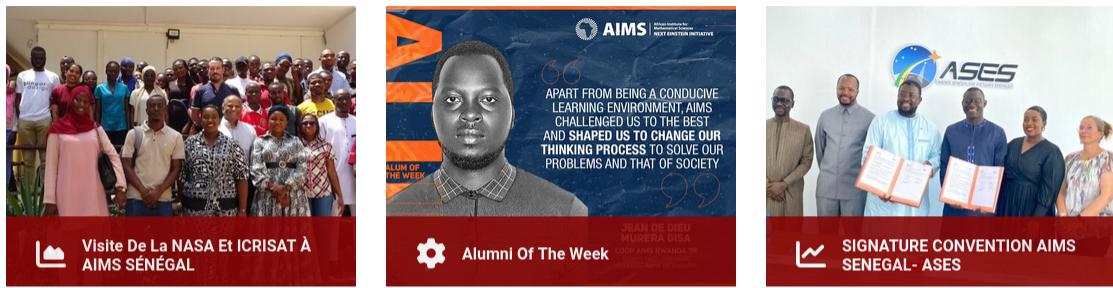


Figure 2.13: Entête

- **Information**

Elle constitue l'ensemble des informations concernant l'académie de AIMS, une publication de l'alumni de la semaine et les catégories de cours.



AVAILABLE CATEGORIES

List of registered categories

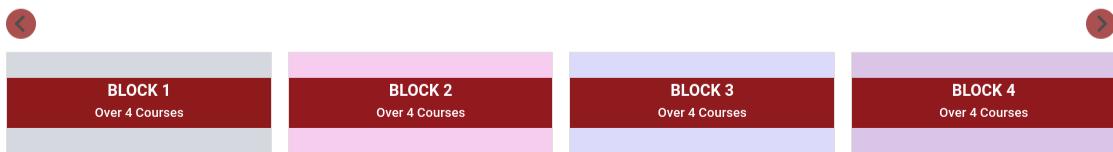


Figure 2.14: Information

- **Section projet**

Cette rubrique concerne les projets de mémoire, de recherche d'étudiants ou d'alumni de AIMS qui seront postés afin de rendre accessible leurs travaux au public de AIMS.

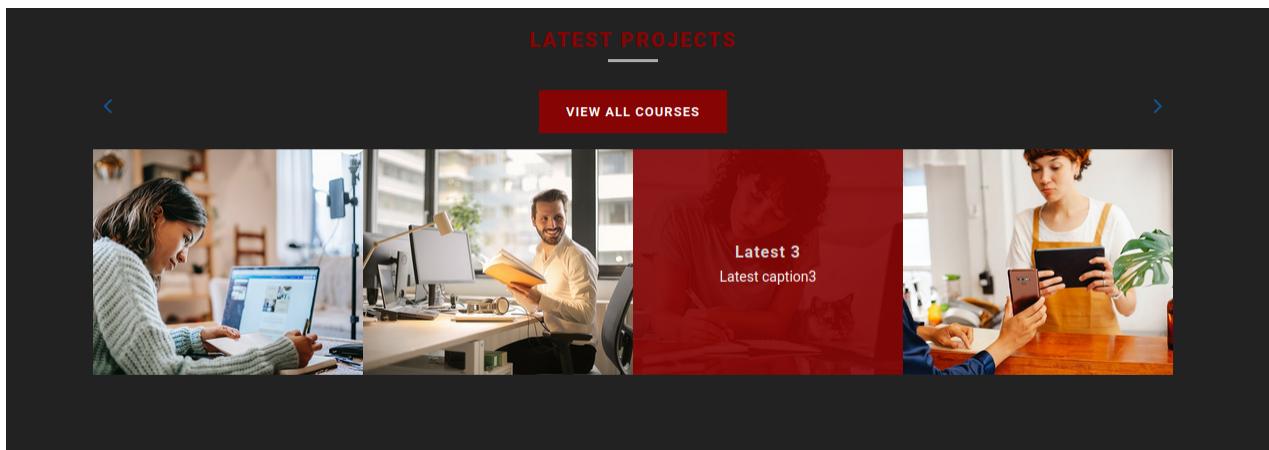


Figure 2.15: Section projet

- **Section recherche**

Cette section concerne le centre de recherche de AIMS et ses publications ,la chaire de recherche, les projets, les workshops et conférences etc.

RESEARCH CENTER

The Research Centre at AIMS Senegal came into existence in 2013 with the appointment of Professor **Mouhamed Moustapha Fall** as the German Research Chair in Mathematics and its Applications...

RESEARCH PUBLICATION

The Research Centre, led by Prof Fall, has published more than 50 articles with over 500 citations in internationally recognized peer-reviewed journals.

RESEARCH CHAIR

Mouhamadou Sy is the endowed Chair in Mathematics and its Applications at AIMS Senegal, funded by the Alexander von Humboldt Foundation. His research interests are: Nonlinear partial differential Equations, with a particular focus on stochastic approaches...

CONFERENCES AND WORKSHOPS

The AIMS Senegal Research Centre regularly hosts conferences, research schools and workshops to foster collaborations with local and international...

OVERVIEW

AIMS is in the top 20 academic institutions in Africa for weighted research outputs on the Nature Index list. With over 100 researchers in the network...

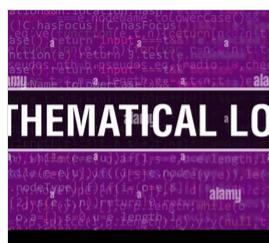
RESEARCH FELLOWS

AIMS Senegal is currently hosting four doctorate and one post-doctoral researcher.

Figure 2.16: Section recherche

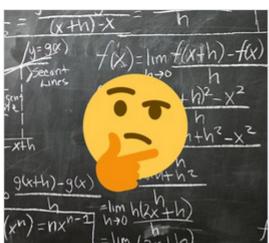
– Cours disponibles

Cette partie concerne les cours disponibles de la plateforme.



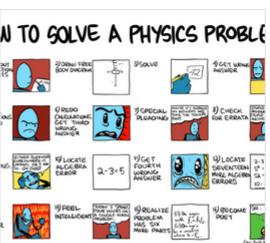
THEMATICAL LOGIC

Updated May 29, 2024



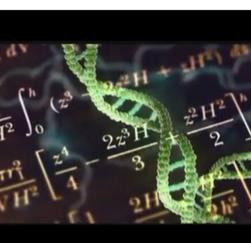
MATHEMATICAL PROBLEM SOLVING

Updated May 25, 2024



PHYSICS PROBLEM SOLVING

Updated May 25, 2024



BIO-MATHEMATICS

Updated May 25, 2024

Figure 2.17: Cours disponibles

– Vision, mission, témoignages et professeurs du mois

La dernière partie publie les témoignages de personnalités publiques, politiques, d'alumni, de professeurs, etc. sur AIMS et présente les professeurs du mois.

TESTIMONIALS

< >

- 66 Climate change will affect society and the generations unborn. Global issues of the changing climate and its devastating effects on humans and livelihood call for urgent action to address it



ADANNA HENRY-UKOHA
Lecturer, Department of Agricultural Economics & Extension, University of Port Harcourt, Aims Small Research Grant Recipient

OUR TEACHERS

We continue to serve with our distinguished teacher staff. Meet our teachers.



Figure 2.18: Témoignage

2.1.5.3.4 Fonctionnalités Principales

- **Gestion des cours**

Les enseignants peuvent créer, organiser et gérer des cours en ligne, incluant du contenu multimédia, des devoirs, des quiz, etc.

- **Interaction**

E-AIMS offre des fonctionnalités d'interaction telles que les forums de discussion, les chats en direct et les wikis pour encourager la collaboration entre les élèves.

- **Évaluation**

Les enseignants peuvent évaluer les progrès des élèves grâce à des activités d'évaluation variées, telles que les quiz, les devoirs en ligne et les sondages.

- **Suivi des progrès**

Les élèves et les enseignants peuvent suivre les progrès des cours et des activités grâce à des outils de suivi intégrés.

- **Personnalisation**

E-AIMS permet une personnalisation poussée de l'interface utilisateur et des fonctionnalités en fonction des besoins spécifiques de l'établissement ou de l'enseignant.

2.1.5.4 Présentation de Wireshark

Wireshark est un analyseur de paquets libre et gratuit largement utilisé pour le dépannage, l'analyse de réseaux informatiques, le développement de protocoles, l'éducation et la rétro-ingénierie. Comparable à tcpdump, Wireshark se distingue par son interface graphique conviviale

et ses nombreuses options de filtrage et de tri des paquets. Il permet de capturer et de décoder les paquets réseau, en comprenant les différentes structures d'encapsulation des protocoles de communication. Wireshark utilise le format pcap pour la capture des paquets, limitant ainsi son support aux types de réseaux pris en charge par pcap.

2.1.5.4.1 Interface d'étude de Wireshark

Nous allons utiliser l'interface analyseur de Wireshark constituée de deux parties.

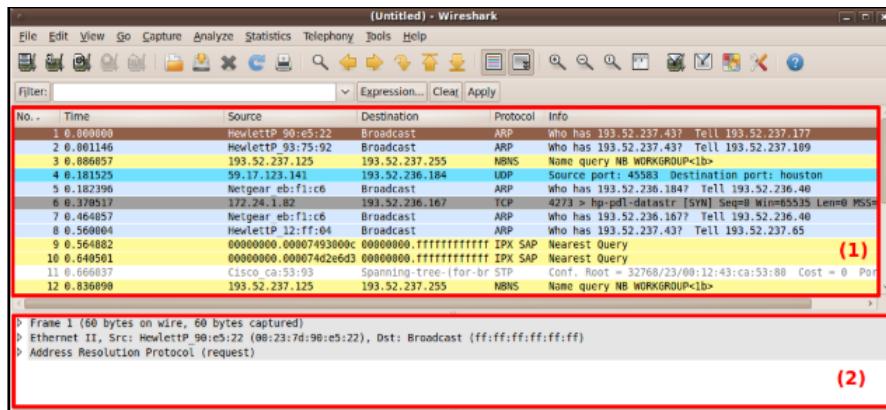


Figure 2.19: Interface de l'analyseur

- **Zone supérieure**, numérotée (1) sur la Figure 2.19 : liste l'ensemble des paquets capturés.
- **Zone centrale**, numérotée (2) sur la Figure 2.19 : affiche le détail d'un paquet sélectionné.

2.1.5.5 Présentation d'OpenSSL sur Linux

OpenSSL est une bibliothèque logicielle robuste et pleine de fonctionnalités permettant de sécuriser les communications sur les réseaux informatiques. Elle implémente les protocoles SSL (Secure Sockets Layer) et TLS (Transport Layer Security) pour assurer la confidentialité et l'intégrité des données en transit. OpenSSL est largement utilisé dans l'industrie pour sécuriser les transactions web, les courriels, les transferts de fichiers, et bien plus encore.

2.1.5.5.1 Caractéristiques principales

- **Cryptographie** : OpenSSL fournit une large gamme d'algorithmes cryptographiques, incluant le chiffrement symétrique (AES, DES, etc.), le chiffrement asymétrique (RSA, DSA, Diffie-Hellman, etc.), le hachage (SHA-256, MD5, etc.), ainsi que la génération de clés et de certificats.
- **Protocoles** : OpenSSL supporte les protocoles SSLv2, SSLv3, TLSv1, TLSv1.1, TLSv1.2, et TLSv1.3, permettant ainsi une sécurisation des communications pour une multitude d'applications réseau.

- **Outils en ligne de commande** : OpenSSL inclut une suite d'outils en ligne de commande pour gérer les tâches courantes liées à la cryptographie, telles que la génération de clés, la création de certificats, le chiffrement et le déchiffrement de fichiers, et le test des connexions SSL/TLS.

2.1.5.5.2 Installation d'OpenSSL sur Linux

La plupart des distributions Linux incluent OpenSSL dans leurs dépôts de logiciels, ce qui rend son installation simple et directe. Voici comment installer OpenSSL sur quelques distributions populaires :

- Ubuntu / Debian :
sudo apt update
sudo apt install openssl

2.1.5.5.3 Utilisation de base d'OpenSSL

Une fois installé, OpenSSL peut être utilisé pour une variété de tâches cryptographiques. Voici quelques commandes de base pour illustrer son utilisation :

```
1 # Génération d'une clé privée RSA
2 openssl genpkey -algorithm RSA -out private_key.pem
3
4 # Création d'une requête de signature de certificat (CSR)
5 openssl req -new -key private_key.pem -out csr.pem
6
7 # Génération d'un certificat auto-signé
8 openssl req -x509 -new -nodes -key private_key.pem -sha256 -days 365 -out
   certificate.pem
9
10 # Chiffrement d'un fichier avec AES-256
11 openssl enc -aes-256-cbc -salt -in fichier.txt -out fichier.enc
12
13 # Déchiffrement d'un fichier
14 openssl enc -aes-256-cbc -d -in fichier.enc -out fichier_dechiffre.txt
```

Chapter 3

Revue de littérature

3.1 Introduction

Cette section présente une vue d'ensemble des principales composantes de notre étude sur le protocole TLS 1.3, en se concentrant sur une revue de la littérature existante, la méthodologie utilisée, les analyses des résultats obtenus, et les recommandations de l'ANSSI. Nous proposons également une comparaison pratique entre TLS 1.2 et TLS 1.3, en mettant en évidence les différences en termes de complexité cryptographique. Enfin, nous récapitulons les résultats obtenus et discutons les limites de notre recherche.

3.2 Revue de Littérature sur TLS 1.3

TLS 1.3 a été largement étudié pour ses avancées en matière de sécurité et d'efficacité par rapport à ses prédecesseurs. Des recherches ont souligné son processus de handshake simplifié, ses mécanismes de chiffrement améliorés et ses performances accrues selon Rescorla[8]. Malgré ces améliorations, il est crucial d'évaluer ses limites et vulnérabilités potentielles dans un environnement contrôlé ou de développement.

Pour évaluer ces aspects, une approche pratique en environnement local peut être mise en place en utilisant Apache, une plateforme Moodle, et un certificat auto-signé. Cela permet de simuler un environnement sécurisé et de mener des tests rigoureux. Plusieurs études ont utilisé des certificats auto-signés dans des environnements contrôlés pour examiner l'implémentation et la sécurité de TLS 1.3. Bien que ceux-ci ne soient pas adaptés aux environnements de production, ils sont précieux pour des tests locaux à des fins éducatives.

3.2.1 Étude sur les Performances et la Sécurité

Abhay Pratap Singh et Mahendra Singh[20] ont mené en 2022 une étude comparative des protocoles de handshake TLS version 1.2 et TLS version 1.3. À l'aide de l'outil de surveillance réseau Wireshark, ils ont réalisé une surveillance passive du trafic SSL/TLS et étudié le processus de handshake.

3.2.2 Étude sur les Vulnérabilités

Bhargavan et Leurent[12] ont mené en 2016 des tests en environnement contrôlé pour simuler deux scénarios d'attaques par collision sur TLS et OpenVPN. Leur étude montre comment une attaque sur l'utilisation de 3DES en HTTPS peut être utilisée pour récupérer une session secrète de cookie, et comment une attaque similaire sur Blowfish peut compromettre les informations d'identification HTTP BasicAuth envoyées via OpenVPN.

Bien que leur recherche se concentre sur des versions antérieures de TLS, les principes sous-jacents restent pertinents pour TLS 1.3, qui introduit des améliorations significatives en matière de sécurité et d'efficacité.

Face à ces multiples paramètres, la performance, la sécurité et la vulnérabilité de TLS 1.3 vont être étudier dans un cadre spécifique de développement.

L'analyse de TLS 1.3 dans un environnement local est essentielle pour évaluer la sécurité et la robustesse de ce protocole dans des conditions réalistes. Cette approche s'inscrit dans une méthodologie où la simulation d'environnements contrôlés permet de détecter et de corriger les vulnérabilités potentielles avant leur déploiement à grande échelle.

Le certificat auto-signé permet d'explorer comment le protocole se comporte face à des configurations de certificats non validés, comme cela pourrait être rencontré dans des environnements de développement ou de test logiciel.

3.2.3 Méthodologie pour les Tests contrôlés

Dans le cadre de la recherche, l'environnement local a été privilégié pour sa flexibilité permettant de configurer et tester divers scénarios réseau sans les contraintes d'un environnement de production. Cependant, cela nécessite des solutions adaptées pour l'analyse et le déchiffrement du trafic réseau, notamment l'utilisation de certificats.

3.2.3.1 Configuration d'Apache avec TLS 1.3 et Moodle

Pour tester TLS 1.3 dans un environnement local il faut les étapes suivantes :

3.2.3.1.1 Installation et configuration d'Apache sur Linux

S'assurer d'avoir Apache installé et configuré pour prendre en charge TLS 1.3. Cela peut impliquer la mise à jour d'OpenSSL et la configuration des fichiers httpd.conf ou ssl.conf.

Nous allons couvrir l'installation et la configuration de base d'Apache sur une distribution Linux, ainsi que la sécurisation des communications avec TLS 1.3.

1. Installation :

Mettre à jour les paquets :

```
1 sudo apt update # Pour Debian/Ubuntu  
2
```

Cette commande met à jour la liste des paquets disponibles et leurs versions.

Installer Apache :

```
1 sudo apt install apache2 # Pour Debian/Ubuntu  
2
```

Cette commande installe le serveur Apache sur votre système.

2. Configuration de Base :

Accéder aux fichiers de configuration : Les fichiers de configuration d'Apache se trouvent généralement dans /etc/apache2/ pour Debian/Ubuntu.

Démarrage et Activation du Service :

Démarrer Apache :

```
1 sudo systemctl start apache2 # Pour Debian/Ubuntu  
2
```

Cette commande démarre le serveur Apache.

Activer Apache pour qu'il démarre au démarrage du système :

```
1 sudo systemctl enable apache2 # Pour Debian/Ubuntu  
2
```

Cette commande configure Apache pour qu'il démarre automatiquement au démarrage du système.

3. Sécurisation des Communications avec TLS 1.3 :

Installation des Modules TLS :

Installer OpenSSL :

```
1 sudo apt install openssl # Pour Debian/Ubuntu  
2
```

Cette commande installe OpenSSL, nécessaire pour le support de TLS 1.3.

Activer le module SSL d'Apache :

```
1 sudo a2enmod ssl # Pour Debian/Ubuntu  
2
```

Cette commande active le module SSL pour Apache sur Debian/Ubuntu. mod_ssl.

4. Configuration du SSL/TLS :

Créer un fichier de configuration SSL :

Pour Debian/Ubuntu :

```
1 sudo nano /etc/apache2/sites-available/default-ssl.conf  
2
```

Ajouter les configurations SSL :

Dans le fichier default-ssl.conf, ajoutez ou modifiez les lignes suivantes pour configurer SSL/TLS :

```
1 <VirtualHost *:443>  
2     ServerAdmin webmaster@localhost  
3     ServerName example.com  
4     DocumentRoot /var/www/html  
5     SSLEngine on  
6     SSLCertificateFile /etc/ssl/certs/your_domain_name.crt  
7     SSLCertificateKeyFile /etc/ssl/private/your_domain_name.key  
8     SSLCertificateChainFile /etc/ssl/certs/CA.crt  
9     SSLProtocol -all +TLSv1.3  
10    SSLCipherSuite TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:  
11        TLS_AES_128_GCM_SHA256  
12        <Directory /var/www/html>  
13            Options Indexes FollowSymLinks  
14            AllowOverride None  
15            Require all granted  
16        </Directory>  
17    </VirtualHost>
```

Activation du Site SSL :

Activer le site SSL :

```
1 sudo a2ensite default-ssl.conf # Pour Debian/Ubuntu  
2
```

Cette commande active la configuration du site SSL sur Debian/Ubuntu.

Redémarrer Apache pour appliquer les modifications :

```
1 sudo systemctl restart apache2 # Pour Debian/Ubuntu
```

```
2
```

Cette commande redémarre Apache pour appliquer les nouvelles configurations.

3.2.3.2 Déploiement de Moodle 4.3 sur Linux avec Apache et MariaDB:

Installation de Moodle sur le serveur Apache pour servir d'application web afin de faire des tests et configurations par un canal sécurisé HTTPS.

3.2.3.2.1 Prérequis

Avant de commencer l'installation, les éléments suivants doivent être disponibles :

- Un serveur exécutant une distribution Linux.
- Accès root ou sudo.
- Apache 2.4 ou supérieur.
- MariaDB 10.3 ou supérieur.
- PHP 7.4 ou supérieur, avec les extensions requises par Moodle.

3.2.3.2.2 Étape 1 : Mettre à jour le système

Mettre à jour la liste des paquets disponibles et mettre à jour les paquets installés:

```
1 sudo apt update
```

```
2 sudo apt upgrade -y
```

3.2.3.2.3 Étape 2 : Installer Apache, MariaDB et PHP

Installer Apache, MariaDB et PHP avec les extensions nécessaires pour Moodle

```
1 sudo apt install apache2 mariadb-server mariadb-client -y
```

```
2 sudo apt install php libapache2-mod-php php-mysql php-xml php-gd php-curl  
    php-zip php-mbstring php-intl php-soap php-xmlrpc php-ldap php-bz2 php-  
    redis -y
```

3.2.3.2.4 Étape 3 : Configurer MariaDB

Exécuter le script de sécurisation de MariaDB

```
1
```

```
2 sudo mysql_secure_installation
```

Suivre les instructions pour sécuriser MariaDB, puis se connectez à MariaDB pour créer la base de données et l'utilisateur pour Moodle : Se connecter à MariaDB en tant que root

```
1 sudo mysql -u root -p
```

Dans le shell MariaDB, exécuter les commandes suivantes qui signifient respectivement : Créer la base de données Moodle Crée l'utilisateur Moodle avec un mot de passe sécurisé Accorder tous les priviléges sur la base de données Moodle à l'utilisateur créé Appliquer les modifications Quitter le shell MariaDB

```
1 CREATE DATABASE moodle DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
2 CREATE USER 'moodleuser'@'localhost' IDENTIFIED BY 'your_password';
3 GRANT ALL PRIVILEGES ON moodle.* TO 'moodleuser'@'localhost';
4 FLUSH PRIVILEGES;
5 EXIT;
```

3.2.3.2.5 Étape 4 : Télécharger et installer Moodle

Naviguer vers le répertoire web par défaut Télécharger Moodle 4.3 Décompresser l'archive Moodle Déplacer les fichiers Moodle vers le répertoire approprié Crée le répertoire moodledata pour les données de Moodle Changer le propriétaire des répertoires Moodle et moodledata à www-data (utilisateur Apache) Définir les permissions appropriées pour les répertoires Moodle et moodledata

```
1 cd /var/www/html
2 sudo wget https://download.moodle.org/download.php/direct/stable43/moodle-latest-43.tgz
3 sudo tar -zxvf moodle-latest-43.tgz
4 sudo mv moodle /var/www/html/moodle
5 sudo mkdir /var/www/html/moodledata
6 sudo chown -R www-data:www-data /var/www/html/moodle /var/www/html/moodledata
7 sudo chmod -R 755 /var/www/html/moodle /var/www/html/moodledata
```

3.2.3.2.6 Étape 5 : Configurer Apache pour Moodle

Créer un fichier de configuration pour Moodle :

```
1 sudo nano /etc/apache2/sites-available/moodle.conf
```

Ajouter les lignes suivantes :

```
1 <VirtualHost *:80>
2   ServerAdmin admin@example.com
3   DocumentRoot /var/www/html/moodle
4   ServerName example.com
5   ServerAlias www.example.com
```

```

6 <Directory /var/www/html/moodle>
7     Options FollowSymLinks
8     AllowOverride All
9     Require all granted
10    </Directory>
11
12
13    ErrorLog ${APACHE_LOG_DIR}/moodle_error.log
14    CustomLog ${APACHE_LOG_DIR}/moodle_access.log combined
15 </VirtualHost>
```

Activer le site Moodle et le module rewrite en faisant :

Activer le nouveau site Moodle

Activer le module rewrite d'Apache

Redémarrer Apache pour appliquer les modifications

```

1 sudo a2ensite moodle.conf
2 sudo a2enmod rewrite
3 sudo systemctl restart apache2
```

3.2.3.2.7 Étape 6 : Installation de Moodle via le navigateur

Ouvrir le navigateur web et accéder à 'http://example.com' pour commencer l'installation de Moodle. Suivre les instructions à l'écran et fournissez les informations nécessaires, y compris les détails de la base de données créée précédemment.

3.2.3.3 Génération de Certificats Auto-signés :

Utilisation de OpenSSL pour créer un certificat auto-signé. Ce certificat sera utilisé pour chiffrer les communications entre le serveur Apache et les clients. Pour établir des connexions TLS en local, un certificat auto-signé est nécessaire, en remplacement des certificats émis par une autorité de certification (CA). Un certificat auto-signé est créé et signé par la même entité, souvent l'utilisateur lui-même, à l'aide d'outils comme OpenSSL.

3.2.3.3.1 Génération du Certificat avec OpenSSL

Pour générer un certificat auto-signé avec OpenSSL, vous pouvez suivre les étapes suivantes :

1. Création de la Clé Privée RSA :

Utiliser la commande suivante pour générer une clé privée RSA de 2048 bits :

```

1 openssl genpkey -algorithm RSA -out localhost.key -pkeyopt
rsa_keygen_bits:2048
```

Cette commande crée un fichier localhost.key contenant la clé privée RSA.

2. Génération du Certificat Auto-Signé :

Utiliser la commande suivante pour générer un certificat auto-signé valide pour 365 jours :

```
1 openssl req -new -x509 -key localhost.key -out localhost.crt -days 365  
      -subj "/CN=localhost"
```

Cette commande génère un fichier localhost.crt contenant le certificat auto-signé. L'option -subj spécifie le nom commun (CN) du certificat comme localhost.

Ces étapes vous permettent de créer et utiliser un certificat auto-signé pour sécuriser les communications HTTPS dans votre environnement local, essentiel pour l'analyse du trafic chiffré avec Wireshark.

3.2.4 Méthode de capture et de décryptage

La capture des paquets réseaux entre le client et le serveur Apache se fait en utilisant Wireshark.

Pour décrypter le trafic TLS 1.3, on configure un certificat auto-signé avec OpenSSL et on définit une variable d'environnement SSLKEYLOGFILE qui enregistre la clé de session à travers un fichier log:

Configuration de la variable d'environnement SSLKEYLOGFILE : SSLKEYLOGFILE est une variable d'environnement qui permet de capturer les clés de session utilisées par les bibliothèques SSL/TLS (comme OpenSSL) pour chiffrer et déchiffrer les communications. En enregistrant ces clés à travers un fichier, il devient possible d'utiliser des outils comme Wireshark pour décrypter le trafic SSL/TLS en temps réel. Cela est particulièrement utile pour le débogage et l'analyse du trafic réseau.

Création de la variable d'environnement sous Linux : Définissons la variable d'environnement SSLKEYLOGFILE pour indiquer à TLS où enregistrer les clés de session.

Ajouter la ligne suivante au fichier de configuration de Shell :

```
1 export SSLKEYLOGFILE=/path/to/sslkeys.log
```

Capture et filtrage des paquets réseau : Avant de procéder aux captures, nous allons configurer SSLKEYLOGFILE dans Wireshark :

Wireshark est configuré pour utiliser SSLKEYLOGFILE : Allez dans Edit > Preferences > Protocols > TLS.

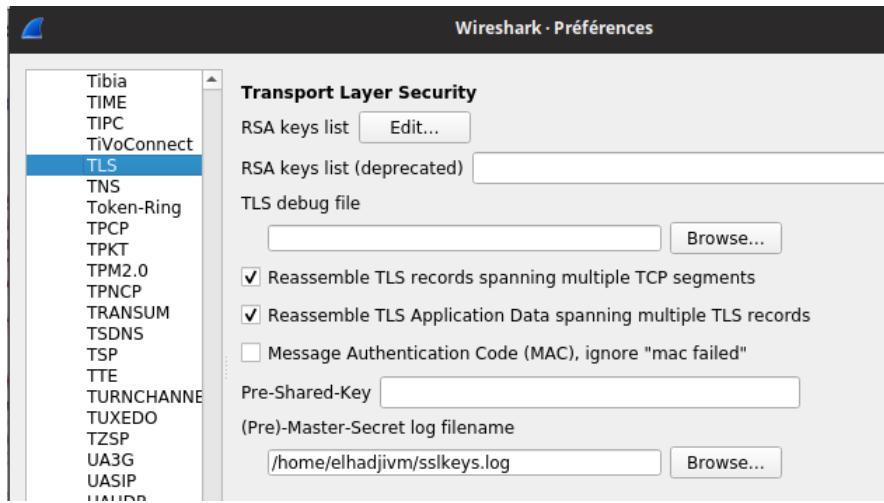


Figure 3.1: Configuration SSLKEYLOGFILE

3.2.5 Analyse des échanges entre client et serveur avec Wireshark

Après avoir démarré la capture dans l'interface loopback, Wireshark peut alors visualiser les paquets TLS capturés, permettant l'analyse détaillée des échanges réseau en utilisant un filtre pour afficher uniquement le port 443 et le protocole TLS.

3.2.5.1 Analyse de la poignée de main TLS

Une prise de contact TLS a lieu chaque fois qu'un utilisateur accède à un site utilisant HTTPS et également chaque fois que d'autres communications utilisent HTTPS. La poignée de main est une série de messages échangés par un client et un serveur. Cela implique plusieurs étapes et peut varier en fonction du type d'échange de clés et de la suite de chiffrement utilisée par les deux parties communicantes.

Le diagramme ci-dessous est un instantané du Handshake TLS entre un client et un serveur capturé à l'aide de Wireshark :

No.	Time	Source	Destination	Protocol	Length	Info
60	26.71671...	::1	::1	TLSv1.3	19...	Client Hello
62	26.71842...	::1	::1	TLSv1.3	18...	Client Hello
64	26.93235...	::1	::1	TLSv1.3	15...	Server Hello, Change Cipher Spec
65	26.93236...	::1	::1	TLSv1.3	15...	Server Hello, Change Cipher Spec
68	26.93621...	::1	::1	TLSv1.3	116	Change Cipher Spec
70	26.93692...	::1	::1	TLSv1.3	116	Change Cipher Spec
77	26.96950...	::1	::1	TLSv1.3	18...	Client Hello
79	26.97724...	::1	::1	TLSv1.3	15...	Server Hello, Change Cipher Spec
81	26.97857...	::1	::1	TLSv1.3	166	Change Cipher Spec
82	26.97906...	::1	::1	TLSv1.3	941	Application Data
83	26.97931...	::1	::1	TLSv1.3	389	Application Data
84	26.97960...	::1	::1	TLSv1.3	389	Application Data
92	28.53848...	::1	::1	TLSv1.3	88	Application Data

Figure 3.2: Trafic de TLS 1.3 au port 443

Le protocole TLS 1.3 utilise une même adresse comme source et destination, cela montre que la connexion se fait par localhost. Analysons chaque étape :

1. Étape 1 : Communication initiale Client Hello

No.	Time	Source	Destination	Protocol	Length	Info
20	137.0222...	::1	::1	TLSv1.3	19...	Client Hello
22	137.0244...	::1	::1	TLSv1.3	15...	Server Hello, Change Cipher Spec
24	137.0266...	::1	::1	TLSv1.3	116	Change Cipher Spec, Application Data
29	137.0285...	::1	::1	TLSv1.3	18...	Client Hello
35	137.1067...	::1	::1	TLSv1.3	15...	Server Hello, Change Cipher Spec
37	137.1080...	::1	::1	TLSv1.3	150	Change Cipher Spec, Application Data
39	137.1085...	::1	::1	TLSv1.3	10...	Application Data
41	137.1088...	::1	::1	TLSv1.3	373	Application Data
42	137.1091...	::1	::1	TLSv1.3	373	Application Data
44	137.3284...	::1	::1	TLSv1.3	488	Application Data
46	142.4002...	::1	::1	TLSv1.3	110	Application Data

Figure 3.3: ClientHello

Typiquement, le premier message dans le Handshake TLS est le message de bonjour client qui est envoyé par le client pour lancer une session avec le serveur.

Le client envoie bien sûr la liste des suites de chiffrement prises en charge, la version TLS prise en charge, l'heure UTC, le nombre aléatoire de 28 octets, l'ID de session, l'URL du serveur et devine quel protocole d'accord de clé le serveur est susceptible de sélectionner.

Le Client envoie également son partage de clé pour ce protocole d'accord de clé particulière.

```

    ▾ Transport Layer Security
      ▾ TLSv1.3 Record Layer: Handshake Protocol: Client Hello
        Content Type: Handshake (22)
        Version: TLS 1.0 (0x0301)
        Length: 1810
      ▾ Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 1806
        Version: TLS 1.2 (0x0303)
        Random: e34f75a94ddaa0cda46731aa63d38aa9671b9caf3d3e2e17...
        Session ID Length: 32
        Session ID: 54ce75267bf93d3d9ea85f37f6be35aaab70fb4ede8d1d18...
        Cipher Suites Length: 32
      ▾ Cipher Suites (16 suites)
        Cipher Suite: Reserved (GREASE) (0x2a2a)
        Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
        Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
        Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xccaa9)
        Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xccaa8)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
        Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
      Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
  
```

Figure 3.4: ClientHello zone centrale message

a. Explication de la figure 3.4

– Version : TLS 1.2 (0x0303)

Jusqu'à TLS 1.2, le client devait annoncer dans le champ version la plus haute du protocole qu'il supportait. Cependant, ce mécanisme n'a pas toujours été bien compris par certains éditeurs de logiciels ou fabricants d'équipements réseau, ce qui conduisait parfois à rejeter des messages avec une version trop élevée au lieu de répondre avec la version maximale supportée, comme spécifié. Pour cette raison, le groupe de travail TLS de l'IETF a choisi de figer le champ version à la version 1.2 et d'utiliser une nouvelle extension (`supported_versions`) pour annoncer les versions supportées.

– **Random:** Un nombre pseudo-aléatoire de 32 octets est utilisé pour calculer le secret de maître, nécessaire à la création de la clé de chiffrement.

– **Session ID:** Un identifiant unique utilisé par le client pour identifier une session.

– **Cipher Suite:** Le client supporte 16 combinaisons de chiffrement différentes pour le cryptage des données, par exemple `TLS_RSA_WITH_AES_128_GCM_SHA256` (0x009c).

– Extensions côté client

Extension: key_share (len=1263)

- Type: `key_share` (51)

- Length: 1263

- Key Share extension

- Client Key Share Length: 1261

- Key Share Entry: Group: Reserved (GREASE), Key Exchange length: 1

- Key Share Entry: Group: Unknown (25497), Key Exchange length: 1216

- Key Share Entry: Group: x25519, Key Exchange length: 32

Le client propose trois groupes de clés au serveur, dont le dernier, respectant la norme PFS, est le groupe x25519, recommandé pour l'échange de clés en TLS 1.3.

Extension: supported_versions (len=7)

- Type: supported_versions (43)
- Length: 7
- Supported Versions length: 6
- Supported Version: Unknown (0x0a0a)
- Supported Version: TLS 1.3 (0x0304)
- Supported Version: TLS 1.2 (0x0303)

L'extension *supported_versions* indique les versions TLS supportées par le client, incluant TLS 1.3 et TLS 1.2.

2. Étape 2 : Bonjour du serveur, modification des spécifications de chiffrement, serveur terminé et données d'application cryptées

tcp.port == 443 and tls						
No.	Time	Source	Destination	Protocol	Length	Info
22	137.0244...	::1	::1	TLSv1.3	15...	Server Hello, Change Cipher Spec, Application Data, Application Data
24	137.0266...	::1	::1	TLSv1.3	116	Change Cipher Spec, Application Data
29	137.0285...	::1	::1	TLSv1.3	18...	Client Hello
35	137.1067...	::1	::1	TLSv1.3	15...	Server Hello, Change Cipher Spec, Application Data, Application Data
37	137.1080...	::1	::1	TLSv1.3	150	Change Cipher Spec, Application Data
39	137.1085...	::1	::1	TLSv1.3	10...	Application Data
41	137.1088...	::1	::1	TLSv1.3	373	Application Data
42	137.1091...	::1	::1	TLSv1.3	373	Application Data
44	137.3284...	::1	::1	TLSv1.3	488	Application Data
46	142.4002...	::1	::1	TLSv1.3	110	Application Data

Figure 3.5: Zone supérieure du ServerHello

b. Explication de la figure 3.5

L'apparition de "Server Hello, Change Cipher Spec, Application Data" dans un seul paquet indique que le serveur a répondu avec plusieurs messages encapsulés ensemble.

En réponse au message « Client Hello », le serveur répond avec le « Server Hello » et le protocole d'accord de clé choisi s'il prend en charge TLS 1.3.

Le message « Server Hello » contient non seulement l'ID de session, l'heure UTC, un nombre aléatoire de 28 octets, la suite de chiffrement sélectionnée, mais également le partage de clé du serveur, son certificat et le message « Server Finished », et commence le cryptage des données. À partir de ce message, le serveur enverra les données au format crypté.

```

    -> TLSv1.3 Record Layer: Handshake Protocol: Server Hello
        Content Type: Handshake (22)
        Version: TLS 1.2 (0x0303)
        Length: 128
    -> Handshake Protocol: Server Hello (2)
        Handshake Type: Server Hello (2)
        Length: 124
        Version: TLS 1.2 (0x0303)
        Random: 3fa8342d4e4cd631b08dfef2da03d2623b130fc018b6fa0...
        Session ID Length: 32
        Session ID: da63d349a009bd3f5eaebd634643d6bd8383a8991d37a296...
        Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
        Compression Method: null (0)
        Extensions Length: 52
        > Extension: supported_versions (len=2)
        > Extension: key_share (len=36)
        > Extension: pre_shared_key (len=2)
    -> TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    -> TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
        Opaque Type: Application Data (23)
        Version: TLS 1.2 (0x0303)
        Length: 42
        Encrypted Application Data: 0bbadc7d2b986feb7dcaaee5fcc254595f8a8399ac7058e...
    -> TLSv1.3 Record Layer: Application Data Protocol: http-over-tls

```

Figure 3.6: Zone centrale du ServerHello

- **TLSv1.3 Record Layer:** Indique que ce message fait partie du protocole TLS 1.3.
- **Version:** Paramètre figé par le groupe de travail TLS de l'IETF à la version 1.2.
- **Random:** Valeur aléatoire générée par le serveur, utilisée pour établir des clés de session sécurisées.
- **Cipher Suite:** *TLS_AES_256_GCM_SHA384 (0x1302)* est la suite de chiffrement choisie par le serveur, utilisant AES avec une clé de 256 bits en mode GCM pour le chiffrement et SHA384 pour l'intégrité des données.

Côté serveur, les extensions suivantes sont observées :

- **Extension supported_versions (len=2)**
 - Type: supported_versions (43)
 - Length: 2
 - Supported Version: TLS 1.3 (0x0304) : Version TLS 1.3 supportée.
- **Extension key_share (len=36)**
 - Type: key_share (51)
 - Length: 36
 - Key Share Entry: Group: x25519, Key Exchange length: 32
 - Group: x25519 (29)
 - Key Exchange Length: 32
 - Key Exchange: 951d5fb4c7bd1f71990f6ebb99eb1a2b0dcd141b7e8612a0...

Le serveur a répondu en choisissant le groupe d'échange de clé x25519.

Le "Change Cipher Spec Message" est un archaïsme optionnel hérité des versions précédentes de TLS, utilisé pour rendre TLS 1.3 compatible avec TLS 1.2. Ainsi, les équipements réseau intermédiaires ont moins de risque de couper la connexion.

Le certificat et les étapes de partage de clé sont toujours chiffrés pour assurer la sécurité de la session. Cela révèle que la clé de session n'a pas été enregistrée dans le fichier *sslkeys.log* associé à la variable d'environnement *SSLKEYLOGFILE*.

Si c'était le cas, la succession de code de Key Exchange serait en clair et les informations sur le certificat du serveur seraient disponibles.

3. Étape 3 : Modifier les spécifications de chiffrement, les données du client terminé et l'application cryptée

tcp.port == 443 and tls						
No.	Time	Source	Destination	Protocol	Length	Info
24	137.0266...	::1	::1	TLSv1.3	116	Change Cipher Spec, Application Data
29	137.0285...	::1	::1	TLSv1.3	18...	Client Hello
35	137.1067...	::1	::1	TLSv1.3	15...	Server Hello, Change Cipher Spec, Application Data, Application Da
37	137.1080...	::1	::1	TLSv1.3	150	Change Cipher Spec, Application Data
39	137.1085...	::1	::1	TLSv1.3	10...	Application Data
41	137.1088...	::1	::1	TLSv1.3	373	Application Data
42	137.1091...	::1	::1	TLSv1.3	373	Application Data
44	137.3284...	::1	::1	TLSv1.3	488	Application Data
46	142.4002...	::1	::1	TLSv1.3	110	Application Data

Figure 3.7: Étape 3

c. Explication de la figure 3.7

- **TLSv1.3 Record Layer:** Application Data Protocol: http-over-tls
- Opaque Type: Application Data (23)
- Version: TLS 1.2 (0x0303)
- Length: 69
- Encrypted Application Data: 385da0ff2a780f91a6ebbbee2790843fe2a8b55bee3564d4...

Désormais, le client vérifie le certificat partagé par le serveur, génère des clés symétriques (car il possède le partage de clé du serveur) et envoie le message "Client Finished". À partir de ce moment, le client et le serveur communiquent en chiffrant les messages sous la forme d'Encrypted Application Data. Cela marque la fin du processus d'établissement de liaison du protocole TLS 1.3.

Dans TLS v1.3 par rapport aux versions précédentes, le processus global est réduit de six à trois étapes, ce qui permet d'économiser environ 25 à 50 % du temps nécessaire pour terminer le processus TLS.

3.2.6 Étude des résultats de l'analyse de TLS 1.3 par rapport aux recommandations de l'ANSSI

L'Agence nationale de la sécurité des systèmes d'information (ANSSI)[1] est un service français créé pour défendre les systèmes d'information de l'État et fournir des conseils en matière de sécurité.

Dans cette section, nous allons vérifier la conformité des résultats du chiffrement TLS 1.3 par rapport aux recommandations de l'ANSSI, basées sur l'analyse des paquets réseau.

- Suites de chiffrement négociées:

Code TLS	Suite cryptographique
0x1302	TLS_AES_256_GCM_SHA384
0x1301	TLS_AES_128_GCM_SHA256
0x1304	TLS_AES_128_CCM_SHA256
0x1303	TLS_CHACHA20_POLY1305_SHA256

Table 3.1: Suites TLS 1.3 recommandées

Côté serveur: *TLS_AES_256_GCM_SHA384*

Utilisation de AES avec une clé de 256 bits en mode GCM pour le chiffrement et SHA384 assurant l'intégrité.

- Examen du processus d'échange de clés:

Recommandation R6 - Échanger les clés en assurant toujours la PFS (Perfect Forward Secrecy).

Le serveur utilise Key Share Entry avec le groupe x25519, assurant un échange de clés sécurisé et éphémère basé sur les courbes elliptiques.

- Authentification du serveur par certificat:

Recommandation R8 - Authentifier le serveur par certificat.

Le serveur est authentifié via l'échange de certificats et de signatures, conformément aux méthodes asymétriques telles que ECDSA (Elliptic Curve Digital Signature Algorithm) ou EdDSA (Edwards-curve Digital Signature Algorithm).

- Chiffrement symétrique:

Recommandation R9 - Privilégier AES ou ChaCha20.

Utilisation de AES avec une clé de 256 bits en mode GCM pour le chiffrement par bloc conformément à la recommandation.

- Fonction de hachage:

Recommandation R11 - Utiliser SHA-2 comme fonction de hachage.

SHA384, appartenant à la famille SHA-2, est utilisé pour l'intégrité des données conformément à la recommandation.

3.2.7 Comparaison pratique entre TLS 1.2 et TLS 1.3 en termes de complexité cryptographique:

Dans cette partie, nous nous penchons sur une comparaison pratique entre TLS 1.2 et TLS 1.3 en explorant leur complexité cryptographique à travers l'utilisation d'OpenSSL pour forcer une connexion à TLS 1.2. En focalisant spécifiquement sur TLS 1.2, nous examinons les suites

cryptographiques supportées et les protocoles de chiffrement utilisés dans le cadre d'une connexion sécurisée établie à l'aide d'OpenSSL.

Par cette comparaison pratique, nous visons à évaluer les améliorations apportées par TLS 1.3 en termes de simplification du handshake, et d'efficacité opérationnelle par rapport à la version précédente.

3.2.7.1 Méthodologie:

Configurer l'environnement pour enregistrer les clés de session.

Capturer le trafic réseau avec tcpdump.

Établir une connexion SSL/TLS avec openssl s_client.

Déchiffrer le trafic capturé avec tshark en utilisant le fichier de clés de session.

Analyser le trafic déchiffré avec tshark ou Wireshark.

1. Exporter la variable d'environnement SSLKEYLOGFILE

```
1 export SSLKEYLOGFILE=/home/elhadjivm/Bureau/sslkeys.log
```

export : C'est une commande utilisée pour définir une variable d'environnement dans le shell.
SSLKEYLOGFILE : C'est le nom de la variable d'environnement spécifique pour enregistrer les clés de session SSL/TLS. **/home/elhadjivm/Bureau/sslkeys.log** : C'est le chemin complet du fichier où les clés de session seront enregistrées. Remplacez ce chemin par celui où vous souhaitez enregistrer le fichier.

2. Capture du trafic avec tcpdump

```
1 sudo tcpdump -i lo -w tls_capture.pcap
```

sudo : Cette commande exécute tcpdump avec des privilèges élevés, nécessaires pour capturer le trafic réseau.

tcpdump : C'est l'outil utilisé pour capturer les paquets réseau.

-i lo : Cette option spécifie l'interface réseau loopback à partir de laquelle capturer le trafic.

-w tls_capture.pcap : Cette option indique à tcpdump d'enregistrer les paquets capturés dans un fichier nommé **tls_capture.pcap**.

3. Connexion SSL avec openssl s_client

```
1 openssl s_client -connect localhost:443 -tls1_2 -debug -msg -keylogfile /  
home/elhadjivm/Bureau/sslkeys.log
```

openssl s_client : C'est une commande qui lance un client SSL/TLS générique pour tester les connexions SSL/TLS.

-connect localhost:443 : Cette option spécifie l'adresse et le port du serveur auquel se connecter.

-tls1_2 : Cette option force l'utilisation de TLS 1.2 pour la connexion.

-debug : Cette option affiche des informations de débogage détaillées sur la connexion SSL/TLS.

-msg : Cette option affiche les messages de protocole SSL/TLS.

-keylogfile /home/elhadjivm/Bureau/sslkeys.log : Cette option spécifie le fichier de log des clés de session, nécessaire pour déchiffrer le trafic capturé.

4. Déchiffrer le trafic avec tshark

```
1 tshark -r tls_capture.pcap -o tls.keylog\_file:/home/elhadjivm/Bureau/  
    sslkeys.log -w decrypted\_capture.pcap
```

tshark : C'est la version en ligne de commande de Wireshark, utilisée pour analyser les paquets réseau.

-r tls_capture.pcap : Cette option indique à tshark de lire le fichier tls_capture.pcap contenant les paquets capturés.

-o tls.keylog_file:/home/elhadjivm/Bureau/sslkeys.log : Cette option spécifie le fichier de log des clés de session à utiliser pour déchiffrer le trafic SSL/TLS.

-w decrypted_capture.pcap : Cette option indique à tshark d'écrire le trafic déchiffré dans un nouveau fichier nommé decrypted_capture.pcap.

5. Analyser le trafic déchiffré avec tshark

```
1 tshark -r decrypted\_capture.pcap
```

tshark : Outil utilisé pour analyser les paquets réseau.

-r decrypted_capture.pcap : Cette option indique à tshark de lire le fichier decrypted_capture.pcap contenant le trafic déchiffré.

Sortie du Débogage

Connexion Établie

```
1 SCSS  
2 CONNECTED(00000004)
```

La connexion au serveur local sur le port 443 a été établie.

ClientHello (TLS 1.2)

```

1 >>> ??? [length 0005]
2   16 03 01 00 b7
3 >>> TLS 1.2, Handshake [length 00b7], ClientHello
4   01 00 00 b3 03 03 a3 f2 76 8b 5a bb 3e a3 d3 b3
5   20 b3 f0 01 7c b2 8e 44 9b 34 c3 22 70 fb 48 45
6   9e d2 c8 82 01 05 00 00 38 c0 2c c0 30 00 9f cc
7   a9 cc a8 cc aa c0 2b c0 2f 00 9e c0 24 c0 28 00
8   6b c0 23 c0 27 00 67 c0 0a c0 14 00 39 c0 09 c0
9   13 00 33 00 9d 00 9c 00 3d 00 3c 00 35 00 2f 00
10  ff 01 00 00 52 00 0b 00 04 03 00 01 02 00 0a 00
11  0c 00 0a 00 1d 00 17 00 1e 00 19 00 18 00 23 00
12  00 00 16 00 00 00 17 00 00 00 0d 00 2a 00 28 04
13  03 05 03 06 03 08 07 08 08 08 09 08 0a 08 0b 08
14  04 08 05 08 06 04 01 05 01 06 01 03 03 03 01 03
15  02 04 02 05 02 06 02

```

ServerHello (TLS 1.2)

```

1 yaml
2 read from 0x5559f69f0180 [0x5559f69f6de3] (5 bytes => 5 (0x5))
3 0000 - 16 03 03 00 41
4 <<< ??? [length 0005]
5   16 03 03 00 41
6 read from 0x5559f69f0180 [0x5559f69f6de8] (65 bytes => 65 (0x41))
7 0000 - 02 00 00 3d 03 03 a0 de-5e fa 82 55 ab c7 08 79
8 0010 - de f0 07 1f ec 93 31 19-fe 18 e5 c9 e0 ab 44 4f
9 0020 - 57 4e 47 52 44 01 00 c0-30 00 00 15 ff 01 00 01
10 0030 - 00 00 0b 00 04 03 00 01-02 00 23 00 00 00 17 00
11 0040 - 00

```

Certificate (TLS 1.2)

```

1 read from 0x5559f69f0180 [0x5559f69f6de3] (5 bytes => 5 (0x5))
2 0000 - 16 03 03 03 97
3 <<< ??? [length 0005]
4   16 03 03 03 97
5 read from 0x5559f69f0180 [0x5559f69f6de8] (919 bytes => 919 (0x397))
6 0000 - 0b 00 03 93 00 03 90 00-03 8d 30 82 03 89 30 82
7 0010 - 02 71 02 14 36 77 7b 38-c5 3f f9 2f 58 e0 a0 b9
8 ...
9 0390 - bd f5 35 e6 f8 dd 77

```

Interprétation des Lignes Clé

ClientHello:

- 03 03 indique que le client utilise TLS 1.2.
- La suite de valeurs représente les différentes options proposées par le client, y compris les suites cryptographiques et les extensions.

ServerHello:

- 03 03 indique que le serveur a accepté d'utiliser TLS 1.2.
- c0 30 est l'identifiant de la suite cryptographique choisie (TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384).

Certificate:

- Les données contiennent le certificat X.509 du serveur. Ce certificat est utilisé pour vérifier l'identité du serveur et établir une connexion sécurisée.

Analyse Comparative

TLS 1.3 présente plusieurs améliorations significatives par rapport à TLS 1.2 :

- **Sécurité Améliorée** : TLS 1.3 utilise des mécanismes de chiffrement plus modernes comme AES-256-GCM et SHA384, offrant une sécurité renforcée par rapport à TLS 1.2.
- **Efficacité du Handshake** : L'utilisation de x25519 pour l'échange de clés simplifie et accélère le processus de connexion par rapport aux méthodes plus complexes utilisées dans TLS 1.2.
- **Confidentialité** : Malgré l'indisponibilité de la clé de session pour le déchiffrement, la confidentialité des données est maintenue grâce à TLS 1.3, illustrant son efficacité dans la protection des communications.

Cette comparaison met en lumière les progrès significatifs en termes de sécurité, d'efficacité et de conformité aux normes de sécurité avec l'adoption de TLS 1.3 par rapport à TLS 1.2. La transition vers TLS 1.3 est non seulement recommandée pour ses avantages en matière de sécurité, mais elle démontre également une meilleure performance globale dans la protection des données sensibles lors des échanges sur le réseau.

Chapter 4

Résultats obtenus

4.1 Introduction

Dans cette section, nous présenterons les résultats obtenus lors de l'analyse du protocole TLS 1.3, en nous appuyant à la fois sur des bases théoriques et des expérimentations pratiques. Les résultats théoriques incluront une comparaison des résultats des études sur des paramètres de TLS 1.3 par rapport aux résultats obtenus dans ce mémoire. Les résultats pratiques, quant à eux, seront tirés de l'analyse de captures de trafic réseau effectuées à l'aide de Wireshark sur la plateforme E-AIMS, mettant en évidence les performances et la robustesse de TLS 1.3 en situation réelle.

4.2 Résultats Théoriques

4.2.1 La performance et la sécurité

Avec TLS 1.2, le processus de handshake nécessitait plusieurs échanges pour établir une connexion sécurisée, augmentant ainsi la latence. En revanche, TLS 1.3 est plus performant en utilisant la PFS et en simplifiant le handshake.

- **Algorithmes de Chiffrement Modernes** : TLS 1.3 adopte des mécanismes de chiffrement plus robustes, tels que AES-256-GCM et SHA384, qui offrent une meilleure sécurité comparativement à ceux utilisés dans TLS 1.2.
- Déclaration implicite des informations relatives au protocole d'échange de clé et de l'algorithme d'authentification du serveur par mesures de sécurité.
- **Résistance aux Attaques** : La conception de TLS 1.3 intègre des protections supplémentaires contre plusieurs types d'attaques, y compris celles qui visaient les versions antérieures du protocole.

Efficacité du Handshake :

- **Réduction de la Latence** : TLS 1.3 simplifie le processus de handshake en utilisant des algorithmes d'échange de clés plus rapides, tels que x25519, réduisant ainsi le temps nécessaire pour établir une connexion sécurisée.
- **Optimisation des Échanges** : Le nombre de messages échangés lors du handshake est réduit, ce qui améliore l'efficacité et la rapidité des connexions.

Confidentialité Renforcée :

- **Protection des Données** : Même en l'absence de la clé de session pour déchiffrer le trafic capturé, TLS 1.3 assure une confidentialité accrue des données grâce à ses algorithmes de chiffrement avancés et ses améliorations de protocole.
- **En-tête Sécurisé** : Les informations sensibles des certificats et des échanges sont mieux protégées dans TLS 1.3, réduisant ainsi les risques de compromission des données.

Nos résultats sur la performance de TLS 1.3 par rapport à TLS 1.2 sont conformes avec ceux de Abhay Pratap Singh et Mahendra Singh

4.2.2 La Vulnérabilité

Version de TLS et Conformité aux Recommandations de l'ANSSI

Il a été confirmé que la communication entre le serveur et le client s'est effectuée en utilisant la version TLS 1.3. Les algorithmes de la suite de chiffrement utilisée, à savoir AES-256-GCM pour le chiffrement et SHA384 pour le hachage, sont conformes aux recommandations de l'Agence nationale de la sécurité des systèmes d'information (ANSSI). Cela assure un niveau de sécurité élevé pour les communications.

Échange de Clé et Handshake

L'algorithme x25519 a été utilisé pour l'échange de clés, assurant un mécanisme sécurisé et efficace pour établir la session cryptée. Le processus de handshake TLS a été complété avec succès, permettant de vérifier l'authenticité des parties et d'établir une connexion sécurisée.

Transmission des Données

Les données ont transité normalement entre le serveur et le client en mode chiffré, confirmant ainsi que la sécurité et l'intégrité des données ont été maintenues tout au long de la session.

Les recherches de Bhargavan et Leurent ont identifié des scénarios spécifiques où TLS est vulnérable en menant des attaques contre des algorithmes utilisant une taille de clé de chiffrement de 64 bits. D'après les résultats obtenus, TLS 1.3 a été utilisant une taille de clé de 256 bits au niveau de la suite de chiffrement. La sécurité a été donc endurcie.

4.3 Résultats pratiques

L'analyse approfondie du trafic réseau a révélé plusieurs avantages significatifs de TLS 1.3 par rapport à TLS 1.2 dans l'environnement de Moodle d'après le tableau suivant.

Étapes de la connexion	TLS 1.3	TLS 1.2
ClientHello	Random, supported_versions, key_share, Cipher suite, signature_algorithm	Random, Cipher suite, supported_versions
ServerHello	TLS 1.3; TLS AES 256 GCM SHA384, x25519	TLS 1.2; TLS ECDHE RSA WITH AES 256 GCM SHA384
Encrypted data/Server Certificate	Encrypted data	Server Certificate
Encrypted data/Server Key Exchange	Encrypted data	Server Key Exchange
Encrypted data/ServerHelloDone	Encrypted data	ServerHelloDone
Encrypted data/Client Key Exchange	Encrypted data	Client Key Exchange
Change Cipher Spec, Finished	Encrypted data	Finished

Table 4.1: Comparaison des étapes de connexion de TLS 1.3 et TLS 1.2

4.4 Limite de la Recherche

Malgré les résultats positifs obtenus lors de l'analyse du trafic réseau et de l'évaluation de la sécurité du protocole TLS 1.3, certaines limitations ont été rencontrées. Ces limitations doivent être prises en compte pour une compréhension complète de l'étude et pour orienter les recherches futures.

4.4.0.1 Non-collecte de la Clé de Session

La limite rencontrée au cours de cette étude a été le non déchiffrement des données échangées durant la session de TLS 1.3. Cela est due à l'absence de la clé de session dans le fichier log de la variable d'environnement `SSLKEYLOGFILE`. Les raisons possibles pour lesquelles la clé de session n'a pas été collectée incluent :

- **Certificat Auto-signé** : L'utilisation de certificats auto-signés peut entraîner des restrictions supplémentaires dans la collecte des clés de session.

- **Mesures de Sécurité de TLS 1.3 :** TLS 1.3 a été conçu avec des mesures de sécurité renforcées, ce qui peut parfois limiter l'accès aux clés de session pour des raisons de sécurité.

4.4.0.2 Implications de la Limite

L'incapacité de décrypter le trafic réseau signifie que certaines analyses détaillées des échanges de données chiffrées n'ont pas pu être réalisées. En particulier :

- **Analyse du Contenu des Messages :** Il n'a pas été possible d'examiner le contenu exact des messages échangés, ce qui limite la capacité à vérifier l'absence de données sensibles ou malveillantes dans les communications.
- **Évaluation Complète des Algorithmes :** Bien que les algorithmes utilisés aient été confirmés par les informations des certificats et les spécifications de la suite de chiffrement, une évaluation plus détaillée de leur mise en œuvre réelle n'a pas été possible sans décryptage.

En conclusion, bien que l'incapacité de déchiffrer le trafic réseau ait limité certaines analyses, les résultats obtenus confirment la robustesse et l'efficacité de TLS 1.3 dans l'environnement étudié.

Les solutions alternatives et les recherches futures qui seront proposées en perspective vont permettre de surmonter ces limitations et d'approfondir l'évaluation de la sécurité des communications chiffrées.

Chapter 5

Conclusion

Le protocole TLS 1.3 représente une avancée majeure dans le domaine de la sécurisation des données en transit. À travers ce mémoire, nous avons exploré de manière approfondie les mécanismes de ce protocole, ses algorithmes de chiffrement, et son application pratique dans un environnement local à des fins de développement en utilisant un serveur Apache et la plateforme E-AIMS Moodle.

L'analyse approfondie du trafic réseau et la comparaison entre TLS 1.3 et TLS 1.2 ont permis de mettre en évidence les performances et la sécurité accrues du protocole TLS 1.3. Les résultats confirment que TLS 1.3 offre une sécurité renforcée grâce à l'utilisation d'algorithmes de chiffrement modernes tels que AES-256-GCM et SHA384. Le processus de handshake optimisé avec l'algorithme x25519 a non seulement simplifié mais aussi accéléré l'établissement des connexions sécurisées, réduisant ainsi la latence.

Ces améliorations substantielles par rapport à TLS 1.2 positionnent TLS 1.3 comme une solution robuste pour protéger les échanges de données sensibles sur les plateformes éducatives en ligne.

Les conclusions tirées de cette étude soutiennent fortement l'adoption continue de TLS 1.3 dans les environnements éducatifs en ligne comme Moodle. Ainsi, les institutions éducatives peuvent non seulement renforcer la sécurité des données des utilisateurs mais aussi inspirer une confiance accrue dans la protection de leurs informations personnelles.

Cependant, cette étude a rencontré des limites significatives, notamment l'incapacité de procéder à une analyse approfondie du contenu des messages échangés en raison de l'indisponibilité de la clé de session. Cette contrainte a restreint l'évaluation détaillée des performances réelles des algorithmes de chiffrement utilisés. Pour surmonter ces défis, il est crucial d'explorer des stratégies telles que l'utilisation de certificats provenant d'autorités de confiance et l'utilisation des proxies SSL/TLS tels que Squid avec SSL-Bump, capables d'intercepter et de déchiffrer le trafic TLS 1.3 en temps réel.

Pour des recherches ultérieures sur TLS 1.3, plusieurs domaines méritent une exploration approfondie. Tout d'abord, avec l'émergence des ordinateurs quantiques, la résistance aux attaques quantiques devient cruciale. TLS 1.3 pourrait être étudié pour intégrer des algorithmes de chiffrement post-quantiques afin de maintenir sa robustesse face à cette nouvelle menace. En parallèle,

l'amélioration de la confidentialité des métadonnées représente un autre défi important. Bien que TLS 1.3 protège les données en transit, les métadonnées telles que les adresses IP peuvent encore être exposées, nécessitant des recherches sur des méthodes pour renforcer cette confidentialité.

De plus, malgré ses avantages en termes de performance, il reste essentiel d'évaluer comment TLS 1.3 fonctionne dans des environnements variés, comme les réseaux à faible bande passante ou les systèmes embarqués. Cette étude permettrait de mieux comprendre son efficacité et d'identifier d'éventuels ajustements nécessaires.

Un autre domaine clé concerne l'interopérabilité et la compatibilité de TLS 1.3 avec d'autres protocoles et applications existants. Il est crucial de déterminer comment assurer une transition en douceur vers TLS 1.3 sans compromettre la sécurité ni perturber les systèmes déjà en place.

Enfin, une analyse continue des failles de sécurité est nécessaire pour identifier et corriger les vulnérabilités potentielles de TLS 1.3. Bien que conçu pour être sécurisé, aucun protocole n'est immunisé contre les nouvelles menaces. Des recherches approfondies dans ce domaine peuvent contribuer à renforcer la résilience de TLS 1.3 contre les attaques sophistiquées.

En concentrant les efforts de recherche sur ces aspects, il est possible de promouvoir une meilleure compréhension et une mise en œuvre plus sécurisée de TLS 1.3 dans divers contextes technologiques et industriels.

References

- [1] ANSSI. Recommandations de sécurité relatives à tls. *none*, 2020.
- [2] Alexandre Berzati. *Analyse cryptographique des altérations d'algorithmes*. *none*, 2011.
- [3] CLOUDFARE. Why use tls 1.3. 2023.
- [4] Benoit Darties. *Tutoriel d'utilisation de Wireshark*. *none*, 2018.
- [5] Site de Stéphane Flon. *Sur l'algorithme RSA*. *none*, 2020. Available from www.booleanopera.fr/TIPE/ADS/ADS_RSA.pdf.
- [6] Statista Research Department. Nombre d'utilisateurs d'internet dans le monde 2014-2023. 2023.
- [7] Osterweil E. and al. *Using Wireshark to analyze TLS 1.3 traffic in a local environment*. *none*, 2018.
- [8] Rescorla E. Tls 1.3. *none*, 2015. Available from <https://rwc.iacr.org/2015/Slides/RWC-2015-Rescorla-TLS.pdf>.
- [9] HackerOne. Cve-2021-22898 detail. *none*, 2021. Available from <https://nvd.nist.gov/vuln/detail/cve-2021-22898>.
- [10] Red Hat. Une cve, qu'est-ce que c'est ? *none*, 2021.
- [11] Bercy Info. Le règlement général sur la protection des données (rgpd), mode d'emploi. 2023.
- [12] Bhargavan K. and Leurent G. *On the Practical (In-)Security of 64-bit Block Ciphers*. ACM Conference on Computer and Communications Security (CCS), 2016.
- [13] Mitre. Cve-2020-19722 detail. *none*, 2021. Available from <https://nvd.nist.gov/vuln/detail/CVE-2020-19722>.
- [14] Levillain Olivier. Ssl/tls, 3 ans plus tard. *none*, 2015.
- [15] Levillain Olivier. Une étude de l'écosystème tls. *none*, 2016.
- [16] Levillain Olivier. Prise en main de tls 1.3 avec openssl 1.1.1. *GNU/Linux Magazine*, 2019.

- [17] SafeHouse. Aes : comment fonctionne réellement le cryptage le plus avancé. 2024.
- [18] TECH SCHOOL. A complete overview of ssl/tls and its cryptographic system. *none*, 2020.
Available from <https://dev.to/techschoolguru>.
- [19] Joseph H. Silverman. *Diffie–Hellman key exchange*. *none*, 2008.
- [20] Abhay Pratap Singh and Mahendra Singh. *Handshake Comparison Between TLS V1.2 and TLS V1.3 Protocol*. *none*, 2022.
- [21] Nick Sullivan. Un examen détaillé de la rfc 8446 (alias tls 1.3). 2018.
- [22] Nicolas Thauvin. *TP N°2 Réseaux 2A - Mise en œuvre du protocole HTTPS Module TR-C8*. *none*, 2009.
- [23] Nir Y. and A. Langley. *ChaCha20 and Poly1305 for IETF Protocols*. *none*, 0000. Available from <https://www.rfc-editor.org/info/rfc8439>.
- [24] P. Saint-Andre Y. Sheffer, R. Holz. Recommendations for secure use of transport layer security (tls) and datagram transport layer security (dtls). *none*, 2015.