Prévention des attaques de confiance en temps réel dans l'IoT social

Mariam Masmoudi

THÈSE

En vue de l'obtention du
DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Cotutelle internationale: Université de Sfax

Présentée et soutenue par

Mariam MASMOUDI

Le 18 décembre 2023

Prévention des attaques de confiance en temps réel dans l'IoT Social

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
IRIT : Institut de Recherche en Informatique de Toulouse

Thèse dirigée par
Florence SèDES et Ikram AMOUS

Jury

M. Thierry DELOT, Rapporteur
M. Mahdi KHEMAKHEM, Rapporteur
Mme Corinne Amel ZAYANI, Examinatrice
M. Stéphane FRéNOT, Examinateur
Mme Raoudha BEN JEMAA, Examinatrice
Mme Florence SèDES, Directrice de thèse
Mme Ikram AMOUS, Co-directrice de thèse
Mme Frédérique LAFOREST, Présidente

Avant-propos

Cette thèse a été réalisée en cotutelle entre l'Université de Sfax-Tunisie (Ecole doctorale en « Economie, Gestion et Informatique» - Laboratoire Multimedia, Information Systems and Advanced Computing Laboratory : MIRACL) et l'Université Toulouse III Paul Sabatier (Ecole Doctorale 475 des Mathématiques, Informatique, Télécommunications de Toulouse : EDMITT et au sein de l'Institut de Recherche en Informatique de Toulouse : IRIT). Cette thèse a été financée par le PHC Utique du ministère français des Affaires étrangères et du ministère de l'Enseignement supérieur et de la Recherche, ainsi que par le ministère tunisien de l'Enseignement supérieur et de la Recherche scientifique dans le cadre du projet CMCU numéro 18G1431 intitulé GECO - Gestion de la confiance dans un environnement mobile et social.

L'ensemble des résultats obtenus au cours de cette thèse a donné lieu aux publications suivantes :

Liste des articles scientifiques :

Dans des conférences internationales :

— Masmoudi Mariam, Amous Ikram, Zayani Corinne Amel, Sèdes Florence (2023). Real-Time Mitigation of Trust-Related Attacks in Social IoT. In 12th International Conference on Model and Data Engineering. MEDI (2023)

— Jmal Raouf, Masmoudi Mariam, Amous Ikram, Zayani Corinne Amel, Sèdes Florence (2023). Apache Spark based Deep Learning for Social Transaction Analysis. In the 19th International Conference on Web Information Systems and Technologies. WEBIST (2023)

— Masmoudi Mariam, Zayani Corinne Amel, Amous Ikram, Sèdes Florence (2021). A New Blockchain-Based trust management model. In 25th International Conference on Knowledge-Based and Intelligent Information Engineering Systems. KES 2021. Procedia Computer Science, 192, 1081-1091. (2021)

— Masmoudi Mariam, Abdelghani Wafa, Amous Ikram, Sèdes Florence (2020). Deep learning for trust-related attacks detection in social internet of things. In Advances in E-Business Engineering for Ubiquitous Computing : Proceedings of the 16th International Conference on e-Business Engineering (ICEBE) (pp. 389-404). Springer International Publishing. (2020)

Dans des revues internationales :

— Masmoudi Mariam, Amous Ikram, Zayani Corinne Amel, Sèdes Florence (2023). Trust Attack Prevention based on Spark-Blockchain in Social IoT : A Survey. In the International Journal of Information Security 2023. (2023) Submitted

— Masmoudi Mariam, Amous Ikram, Zayani Corinne Amel, Sèdes Florence (2023). Real-time Prevention of Trust-related Attacks in Social IoT using Blockchain and Apache Spark. In Computer Communication Journal 2023. (2023) Submitted

*Je dédie ce travail à : Mes très chers parents **Mohamed** et **Najoua**, mon cher mari **Heni**, ma sœur **Abir**, mes frères **Firas** et **Houssem**, mes neveux **Assil** et **Majd**, ma belle famille, tous mes proches et amis pour leur soutien moral, leur aide aussi précieuse qu'inestimable et grâce à vous je n'ai manqué de rien « MERCI ».*

Remerciement

Je vais exprimer par ces quelques lignes de remerciement ma gratitude envers tous ceux qui par leur présence, leur soutien, leur disponibilité et leurs conseils, ont permis d'accomplir ce travail. Le résultat d'un tel projet résulte en effet d'un travail d'équipe et non d'une seule personne.

En premier lieu, vifs et chaleureux sont les remerciements que j'adresse à **Madame Ikram AMOUS** et **Madame FLORENCE SèDES** qui m'ont fait l'honneur d'être mes directrices de thèse et pour m'avoir si chaleureusement accueilli dans leurs équipes. Je vous remercie pour la confiance que vous avez investie en moi en acceptant d'encadrer mon travail. Ainsi que pour vos conseils intéressants, vos encouragements continus, ainsi que le temps que vous m'avez consacré malgré vos grandes occupations.

Madame CORINNE AMEL ZAYANI, ma co-encadrante pour la qualité de son encadrement. Je vous remercie profondément pour vos encouragements continus et aussi d'être toujours là pour m'écouter, m'aider et me guider à retrouver le bon chemin par vos précieux conseils. Cette thèse est le fruit d'une collaboration de plus de quatre années avec elle. C'est à ses côtés que j'ai compris ce que rigueur et précision voulaient dire.

Je remerci **Monsieur YOUNES BOUJELBENE**, le doyen de la faculté des Sciences Economiques et de Gestion de Sfax et **Monsieur ABDELWAHED MOKNI**, le président de l'Université de Sfax, pour m'avoir accepté et me donné l'opportunité de faire une thèse en cotutelle.

Je tiens aussi à remercier **Monsieur JEAN MARC PIERSON**, directeur de l'Institut de Recherche en Informatique de Toulouse (IRIT) qui m'a accueilli pendant mes séjours depuis 2019 au sein de son laboratoire. C'est grâce à lui que j'ai pu concilier avec bonheur recherche théorique et appliquée pendant cette thèse. Un grand merci aussi à tous les membres de IRIT et en particulier à **Madame CLEMENTINE ROGER**, qui ont répondu avec calme et patience aux demandes pour être capable de faire mes séjours.

Je remercie également **Madame MARIE HELENE VIGNAL**, directrice de l'école doctorale EDMITT pour m'avoir accepté et me donné l'opportunité de faire une thèse en cotutelle.

Je tiens d'autre part à remercier les respectables membres de jurys **Thiery DELOT**, **Mahdi KHEMAKHEM**, **Fréderique LAFOREST**, **Stéphane FRéNOT** et **Raoudha BEN JMEAA** pour avoir accepté participer à ce jury de thèse et d'évaluer ce travail.

Enfin, je ne peux achever ce mémoire sans exprimer ma gratitude à toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de me rencontrer et de répondre à mes questions durant mes recherches.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Contexte et problématique

Grâce aux progrès rapides de l'internet, l'Internet des Objets, appelée également Internet of Things (IoT) en anglais, est apparu comme une révolution fondamentale et innovante dans le monde de la technologie, de l'information et de la communication. Ainsi, l'IoT est un environnement où les objets physiques quotidiens deviennent connectés à l'Internet. Ces objets produisent un trafic énorme de données suite à des interactions et des communications intensives entre des milliards d'utilisateurs, hôtes et objets connectés (Zheng et al., 2022). Incontestablement, la vision de l'IoT a un impact considérable sur la vie quotidienne des utilisateurs en fournissant des informations et des services, ce qui constitue sa principale force (Ramanathan, 2015). De ce fait, les utilisateurs peuvent ensuite tirer parti des services qui résultent de ces interactions. Néanmoins, plusieurs défis freinent la concrétisation de la vision de l'IoT, notamment la gestion de la confiance, la confidentialité et la sécurité (Abdelghani et al., 2016).

Afin de répondre aux défis mentionnés, une nouvelle approche, appelée Internet des Objets sociaux (IoT Social) ou Social Internet of Things en anglais, a été développée (Abdelghani et al., 2016). Cette approche intègre la dimension sociale dans l'Internet des Objets pour proposer une variété d'applications et de services innovants aux utilisateurs finaux. L'adoption de cette nouvelle approche attire de plus en plus l'attention de différents utilisateurs. En effet, grâce à l'IoT social, les utilisateurs et les objets ont la possibilité d'établir des relations sociales avec d'autres utilisateurs et objets, de rejoindre des communautés et de créer leurs propres réseaux sociaux. Ils peuvent également découvrir et utiliser les services proposés par d'autres utilisateurs, ce qui enrichit leur expérience de l'IoT social (Ramanathan, 2015). De plus, la tendance émergente d'adopter cette vision présente d'autres avantages. Tout d'abord, elle permet d'améliorer la navigabilité et la découverte des services. Elle garantit l'évolutivité comme dans les réseaux sociaux humains. Enfin, elle permet d'augmenter la quantité et la variété des données, ce qui améliore l'intelligence des services et leur capacité à s'adapter aux besoins situationnels et contextuels des utilisateurs (Roopa et al., 2019).

Bien que l'IoT social présente de nombreux avantages, il est confronté à plusieurs défis qui peuvent réduire la qualité de ses performances et aggravent la vulnérabilité de la confidentialité, de la sécurité des utilisateurs et la confiance entre les utilisateurs et les objets connectés. L'un des principaux défis est la composante de découverte de services car elle peut être influencée par des compor-

tements malveillants tels que les "attaques de confiance", qui peuvent fausser les résultats de recommandation de services (Abdelghani et al., 2018). Grâce à ces attaques, les utilisateurs peuvent promouvoir leurs valeurs de confiance et mettre en avant des services malveillants comme étant dignes de confiance, ce qui compromet la sécurité et la fiabilité des demandeurs de services (Masmoudi et al., 2021). De ce fait, ils peuvent ne pas être sûrs de la fiabilité des services et des objets connectés qu'ils utilisent et des informations qu'ils partagent avec eux (Masmoudi et al., 2020). Ils restent méfiants et vigilants à l'égard de ce nouveau réseau. Pour que les utilisateurs puissent utiliser les produits et services de l'IoT social en toute confiance, il est impératif de mettre en place un mécanisme de gestion de la confiance. Ce mécanisme doit garantir des communications sécurisées, une analyse fiable des données, des services de qualité et des interactions fiables. La prévention en temps réel de tous les types d'attaques de confiance est donc un défi crucial à relever pour assurer un environnement fiable et digne de confiance, càd lors de génération des transactions entre les utilisateurs.

Notre problématique générale pourrait se résumer avec les interrogations suivantes : (i) Comment pouvons-nous instaurer un environnement de confiance pour l'IoT social et renforcer la confiance et la sécurité des utilisateurs malgré les risques potentiels d'attaques de confiance ? (ii) Comment pouvons-nous mettre en place un mécanisme de gestion de la confiance efficace pour l'IoT social qui assure la prévention de tous les types d'attaques de confiance en temps réel ?

Objectifs de la thèse

L'objectif de cette thèse est de concevoir et développer un mécanisme de gestion de la confiance efficace pour la prévention en temps réel de tous les types d'attaques de confiance dans l'IoT social, en utilisant des technologies avancées telles que la blockchain et Apache Spark. Ce mécanisme doit aussi tenir compte des propriétés de sécurité. Pour atteindre cet objectif, cette thèse vise à proposer une architecture en couches qui permet de combiner ces technologies avec l'IoT social. Ainsi, nous avons conçu une architecture pour collecter, classifier et analyser en temps réel les flux de transactions des utilisateurs de l'IoT social en utilisant notre protocole de consensus basé sur Apache Spark. Cette architecture vise à prévenir toute manipulation malveillante et à identifier les attaques de confiance. Si une attaque est détectée, notre protocole de consensus annule automatiquement la transaction en question. Pour cela, il utilise des techniques d'apprentissage d'Apache Spark pour identifier les transactions malveillantes et les attaques de confiance en temps réel, garantissant ainsi la prévention des attaques en temps réel. De plus, notre protocole permet de stocker de manière décentralisée et immuable les transactions fiables entre les utilisateurs de l'IoT social dans la blockchain. Les résultats de cette thèse devront montrer l'efficacité de la solution proposée pour la prévention en temps réel des attaques de confiance dans l'IoT social, ainsi que ses avantages par rapport aux autres travaux existants.

Contributions

Les principales contributions de cette thèse, qui offrent une vision globale des apports de notre travail de recherche, sont les suivantes :

1. Conception d'un mécanisme de gestion de la confiance efficace : La thèse propose un mécanisme novateur pour la prévention en temps réel de tous les types d'attaques de confiance dans l'IoT social. Ce mécanisme intègre des technologies avancées telles que la blockchain et Apache Spark, tout en prenant en considération les propriétés de sécurité essentielles, qui seront détaillées dans les chapitres ultérieurs. Cela vise à instaurer la confiance dans l'IoT social.

2. Introduction de nouveaux facteurs, qui seront examinés plus en détail dans les prochains chapitres et sont également connus sous le nom de caractéristiques, qui sont déduits de chaque modèle d'attaque de confiance. Ces facteurs permettent de détecter rapidement si un utilisateur a initié une transaction malveillante, qui est en réalité une attaque de confiance.

3. Proposition d'une architecture en couches qui permet d'intégrer la blockchain, Apache Spark avec l'IoT social. Cette architecture offre une solution complète pour la gestion de la confiance et la prévention des attaques de confiance en temps réel.

4. Développement d'un protocole de consensus basé sur Apache Spark : La thèse propose un protocole de consensus innovant basé sur Apache Spark. Ce protocole garantit l'intégrité des transactions et la prévention des manipulations malveillantes. Il utilise des techniques d'apprentissage d'Apache Spark pour identifier en temps réel les transactions malveillantes qui se révèlent être des attaques de confiance, assurant ainsi une prévention rapide et efficace de ces attaques.

5. Validation expérimentale de l'efficacité de la solution proposée : La thèse propose une évaluation expérimentale pour démontrer l'efficacité de la solution dans la prévention en temps réel des attaques de confiance dans l'IoT social. Les résultats obtenus mettront en évidence les avantages de la solution par rapport aux travaux existants, mettant en lumière sa capacité à assurer la confiance et la sécurité dans l'environnement de l'IoT social.

Organisation de la thèse

Notre rapport de thèse se compose de cinq chapitres que nous avons élaborés pour répondre à notre problématique générale et à nos objectifs spécifiques :

Le *premier chapitre* de ce rapport se concentre sur les concepts fondamentaux de notre domaine d'étude, notamment l'IoT, l'IoT social, la confiance et les mécanismes de gestion de la confiance et ses différentes phases. Nous avons expliqué en détail chacune de ces phases, à savoir la phase de composition, la phase d'agrégation, la phase de propagation et la phase de mise à jour de la confiance, qui constituent un processus complexe et crucial dans les environnements dynamiques et complexes tels que l'IoT social. Ce chapitre constitue une base solide pour la compréhension de notre travail, car il nous permet de mieux définir les termes spécifiques liés à notre domaine et de situer notre étude dans son contexte approprié. Nous nous penchons également sur la littérature existante concernant les mécanismes de gestion de la confiance pour l'IoT social. Il est important de noter que la confiance est un aspect crucial pour la réussite de l'IoT social, et les mécanismes de gestion de la confiance sont donc essentiels pour assurer un environnement sûr et fiable. Nous menons également une analyse des différentes catégories d'approches pour la gestion de la confiance proposées dans la littérature, qu'il s'agisse de la détection ou de la prévention des attaques de confiance. Nous examinons en détail les avantages et les limites de chaque catégorie d'approches. Après cette analyse, nous constatons que la prévention en temps réel de tous les types d'attaques de confiance dans l'IoT social est la stratégie la plus efficace.

Le *deuxième chapitre* de ce rapport est dédié à l'étude des techniques qui ont été exploitées dans des domaines connexes à l'IoT social pour prévenir efficacement les attaques de confiance en temps réel, tout en prenant en compte les propriétés de sécurité. Nous effectuons une comparaison approfondie de ces techniques dans ce chapitre, en mettant en évidence les avantages et les limites de chacune d'elles. Parmi les différentes techniques étudiées, nous avons identifié la blockchain comme la plus appropriée pour la prévention des attaques de confiance. En comparaison avec les autres techniques, telles que le multi-agent, la blockchain présente de nombreux avantages, notamment en termes de sécurité, de transparence et de résilience. Afin de prendre en compte les différents scénarios d'attaques de confiance possibles, nous avons proposé un nouveau protocole de consensus plus sophistiqué pour la blockchain. Ce protocole utilise des techniques d'apprentissage pour détecter les transactions malveillantes et identifier les attaques de confiance en temps réel. En détectant et en annulant ces transactions malveillantes, le protocole garantit la prévention des attaques de confiance en temps réel. Pour cela, afin de choisir le moteur de traitement de flux le plus approprié pour détecter les attaques de confiance en temps réel en utilisant des techniques d'apprentissage, nous avons effectué une évaluation approfondie des différents moteurs de traitement de flux disponibles. Cette analyse nous a permis de conclure qu'Apache Spark est le moteur de traitement de flux le plus adapté à notre objectif, grâce à sa vitesse de traitement, sa facilité de mise en œuvre et sa compatibilité avec d'autres technologies comme blockchain. Nous avons donc proposé d'intégrer Apache Spark dans notre protocole de consensus de la blockchain.

Le *troisième chapitre* de notre rapport se concentrera sur la présentation dé-

taillée de la solution que nous avons proposée pour la prévention des attaques de confiance dans l'IoT social. Nous commençons par présenter la modélisation du système IoT social utilisée pour notre étude, suivie d'une description de l'architecture en couche qui combine plusieurs technologies avancées telles que la blockchain et Apache Spark avec l'IoT social. Nous expliquons ensuite en détail l'architecture de notre solution appelée SparkChain, qui permet de prévenir en temps réel tous les types d'attaques de confiance tout en tenant compte des propriétés de sécurité. Ce mécanisme repose sur la blockchain et Apache Spark. Nous allons également présenter en détails la phase de composition de la confiance et les différents facteurs proposés. Ces facteurs sont déduits de la description des différentes attaques de confiance.

Dans le *quatrième chapitre* de ce rapport, nous allons poursuivre notre présentation de notre approche SparkChain, en décrivant les méthodes proposées pour agréger les facteurs proposés dans le chapitre 3 et comment ils seront transformés en scores de confiance. Ce score aide notre protocole de consensus à prendre la décision de valider ou annuler les transactions effectuées par les utilisateurs dans l'IoT social. Nous avons détaillé notre nouveau protocole de consensus qui utilise Apache Spark pour l'analyse en temps réel les transactions. Ce protocole permet une détection rapide des attaques de confiance et une réponse immédiate en les interrompant et les annulant. Nous aborderons les méthodes de propagation et de mise à jour de ces scores de confiance. Nous expliquerons comment la blockchain peut être utilisée pour créer un registre décentralisé et immuable de transactions fiables entre les utilisateurs de l'IoT social. Nous montrerons comment Apache Spark peut être utilisé dans notre protocole de la blockchain pour l'analyse en temps réel des transactions générées par les utilisateurs. Nous détaillerons les différentes étapes de notre approche, telles que la collecte des données, la classification des données, la détection des attaques de confiance en temps réel et la réponse aux attaques détectées par les interrompre et les annuler.

Le *cinquième chapitre* de notre rapport est dédié à la présentation des outils et des technologies que nous avons utilisés tout au long de notre développement. Nous y exposons également l'évaluation de notre système, qui consiste à mesurer l'efficacité de notre approche dans la prévention des attaques de confiance. Nous présentons les différents scénarios de test que nous avons mis en place pour évaluer notre approche et notre protocole de consensus, ainsi que les métriques que nous avons utilisées pour mesurer leurs performances. Enfin, nous discutons des résultats de nos tests et analysons les avantages et les limites de notre travail. Ce chapitre permet ainsi de donner une vision concrète de notre proposition et de son potentiel en tant que solution pour la gestion de la confiance dans l'IoT social.

Enfin, le dernier chapitre de ce rapport sera dédié à la conclusion, où nous ferons une synthèse sur nos résultats, discuterons des limites de notre approche et proposerons des pistes pour des recherches futures.

1 Concepts fondamentaux de l'IoT social et gestion de la confiance

Sommaire

1.1 Introduction

Les avantages et les capacités multiples de l'Internet ont donné naissance à l'Internet des Objets, appelée également Internet of Things (IoT), qui peut être considéré comme étant une véritable révolution dans le monde de la technologie, de l'information et de la communication. De plus, l'intégration de la composante sociale dans l'IoT a donné naissance à un nouveau réseau nommé Social Internet of Things (Social IoT) ou l'IoT Social. Ce dernier permet aux personnes et aux objets d'interagir afin d'offrir une variété de services et d'applications attrayants.

Néanmoins, ce paradigme est confronté à un certain nombre de défis qui réduisent la qualité de ses performances. L'un des défis les plus importants est la confiance, plus précisément les attaques de confiance. Ces attaques peuvent influencer les résultats de la découverte de services. En conséquence, la gestion de la confiance devient une exigence majeure dans l'IoT social afin de prévenir ces attaques et de garantir des expériences dignes de confiance aux utilisateurs finaux en fournissant des services qualifiés et une sécurité garantie. C'est dans ce contexte que s'inscrit notre travail. Nous nous intéressons au problème de la gestion de la confiance dans le réseau Social IoT.

Dans ce chapitre, nous définissons l'Internet des Objets (IoT) et son architecture. Ensuite, nous présentons l'environnement de l'IoT Social, son architecture et ses défis. Puis, nous exposons le concept de la confiance dans l'IoT Social, le mécanisme de gestion de la confiance et nous détaillons les différentes attaques de confiance dans l'IoT Social. Finalement, nous présentons une revue sur les mécanismes de gestion de la confiance proposé dans la littérature pour l'IoT Social.

1.2 L'Internet des Objets (IoT)

L'Internet des Objets est au centre de l'attention des consommateurs. Il s'agit de la tendance dans le monde de la technologie. Pour cette raison, la promesse d'un monde peuplé d'objets connectés offre d'innombrables opportunités aux utilisateurs.

La notion même d'Internet des Objets, qui est sujet à interprétation, mérite d'être définie. Tout au long de cette partie, nous commençons par définir un objet connecté pour ensuite mettre l'accent sur l'internet des objets et enfin, terminer par son architecture.

1.2.1 La définition d'un objet connecté

Un objet connecté est nommé aussi «objet communicant», «objet interactif» ou «objet intelligent». D'après le dictionnaire Larousse le terme objet est défini comme «chose solide considérée comme un tout, fabriqué par l'homme et destinée à un certain usage» et le terme connecter se traduit par faire «unir, lier des

choses entre elles», et au sens technique «établir une liaison électronique entre divers organes ou machine» ou «établir une liaison avec un réseau informatique».

L'objet connecté est donc un artéfact fabriqué par l'être humain afin de pouvoir collecter des données, de produire et recevoir de l'information à partir des autres objets connectés. Certains objets sont mobiles et caractérisés par une position qui évolue au cours du temps. Ces objets peuvent être des téléphones intelligents (smartphones), des montres intelligentes ou encore des voitures. D'autres objets sont fixes, à savoir des téléviseurs intelligents, ou encore des capteurs intelligent, etc. Tous ces objets possèdent au minimum un identifiant unique attaché à une identité exprimant d'une part ses propriétés (type, couleur, poids, etc.) et son état c'est-à-dire l'ensemble de ses caractéristiques pouvant évoluer au cours du temps (position, niveau de batterie, etc.).

1.2.2 La définition de l'IoT

L'Internet des Objets correspond à un ensemble d'objets connectés pour communiquer entre eux via le réseau internet (Asemani et al., 2019). C'est également grâce aux solutions de l'Internet des Objets que certains individus pourront contrôler leurs données et objets à distance. À ces perspectives, l'IoT correspond à un monde où il y a eu plus «d'objets» connectés à Internet que de personnes (Ramanathan, 2015).

La valeur de l'Internet des Objets s'articule généralement autour de deux grands volets : objets et connectivité. La connectivité représente les réseaux et leurs équipements et semble être la couche la plus avancée de l'internet des objets (Zheng et al., 2022). Cette couche est présente dans le quotidien des utilisateurs qui disposent d'une interface pour interagir avec l'Internet des Objets, via leurs smartphones et tablettes. Quant à la couche des objets, elle correspond aux capteurs intégrés dans des objets physiques.

1.2.3 L'architecture de l'IoT

L'internet des objets s'organise en trois couches (Layers) comme proposé par (Zheng et al., 2011, 2022) : couche de perception (Perception Layer), couche de communication (Communication Layer) et couche d'application (Application Layer) comme montre la figure 1.1.

La **couche de perception** comprend tout le matériel physique, comme les capteurs intelligents et les actionneurs (RFID, NFC, caméra, GPS, etc.). La **couche de communication** a pour rôle de connecter les objets physiques entre eux. Finalement, la **couche d'application** se charge de la présentation de données aux utilisateurs finaux.

1.3 L'Internet des Objets Social (IoT Social)

Récemment, un certain nombre d'activités de recherche indépendantes ont examiné les possibilités d'intégration des concepts de réseaux sociaux dans les

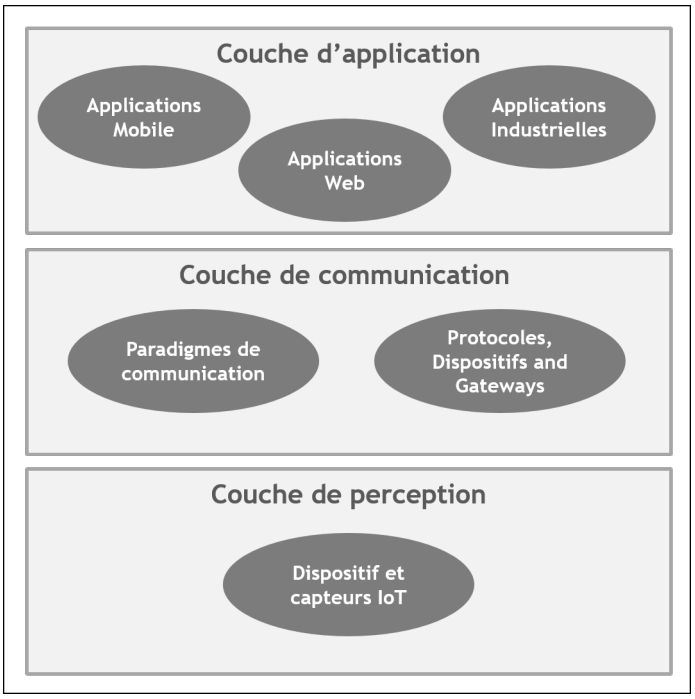

FIGURE 1.1 – *Architecture de l'IoT (Zheng et al., 2011, 2022)*

solutions de l'IoT. Le paradigme qui en résulte est appelé Internet des Objets Social (IoT Social).

Dans cette section, nous commençons par une définition de ce nouveau paradigme. Ensuite, nous présentons son architecture. Après, nous présentons les différentes relations qui peuvent être établies entre ces objets. Finalement, nous discuterons ses principaux défis.

1.3.1 De l'IoT à l'IoT Social

Comme nous l'avons mentionné précédemment, l'IoT (Chen et al., 2016) fait référence à la connexion via Internet de différents dispositifs, tels que des capteurs, des actionneurs, des caméras, des véhicules, etc. pour collecter et échanger des données. L'objectif principal de l'IoT est de fournir des services intelligents et personnalisés.

L'IoT social, quant à lui, se concentre sur l'utilisation de l'IoT pour améliorer les relations sociales entre les êtres humains. En effet, l'IoT social est un nouveau paradigme dans lequel l'IoT et les SN (Social Network) sont combinés (Abdelghani et al., 2016). L'objectif principal de l'IoT social est d'encourager les interactions sociales entre les individus, de renforcer les liens sociaux et de faciliter la communication. En outre, il vise à résoudre les problèmes d'évolutivité, de confiance et de découverte de services en modélisant l'IoT comme un réseau social.

Grâce à ses fonctionnalités sociales, les utilisateurs peuvent établir des relations sociales pour interagir avec d'autres personnes, échanger et partager des

informations, bénéficier des services fournis par les appareils d'autres utilisateurs et évaluer les services (Abdelghani et al., 2018).

Les dispositifs de l'IoT social peuvent inclure des plateformes de médias sociaux, des jouets connectés, des dispositifs de surveillance de la santé, etc. Alors que ses services peuvent être un objet capable de prédire les conditions routières, les services de Google Maps, Moovit et Waze peuvent être liés non seulement au domaine de la gestion du trafic routier mais aussi à d'autres domaines tels que l'apprentissage en ligne, dont les services sont edx, Udemy, Mooc et les chaînes YouTube qui offrent des cours et des sessions de formation en ligne.

Ainsi, alors que l'IoT se concentre sur la collecte et l'échange de données pour améliorer les processus, l'IoT social se concentre sur l'interaction sociale et le bien-être des utilisateurs.

1.3.2 L'architecture de l'IoT Social

Plusieurs architectures sont suggérées dans la littérature. Nous illustrerons dans cette partie l'architecture la plus simplifiée et récente de l'IoT social.

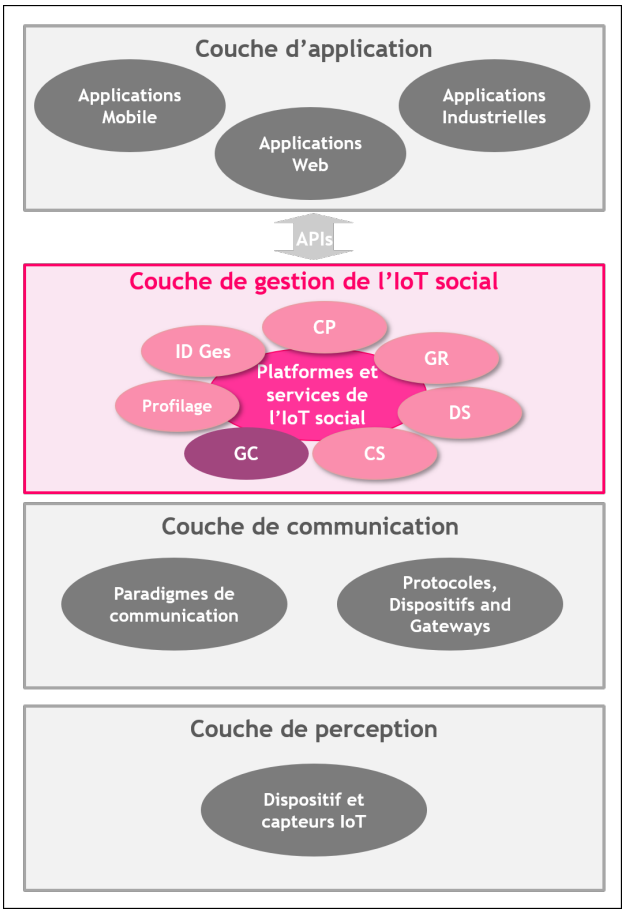

FIGURE 1.2 – *Architecture de l'IoT Social (Gulati et Kaur, 2019)*

TABLE 1.1 – *Composantes essentielles de la couche de gestion de l'IoT Social*

Composants	Objectifs
Gestion des ID (ID Ges)	identifier les objets du réseau
Profilage	configurer des informations sur les objets
Contrôle par le Propriétaire (CP)	définir les activités pouvant être réalisées par les objets définir l'information partagée
Gestion Relationnelle (GR)	établir et entretenir des relations entre les objets
Découverte des Services (DS)	trouver l'objet qui peut fournir le service requis dans le réseau social.
Composition de Services (CS)	permettre des interactions entre objets, soit en récupérant des informations authentiques, soit en trouvant un service spécifique fourni par un autre objet
Gestion de la Confiance (GC)	évaluer la confiance des différents objets pour les interactions et les découvertes de service

Cette architecture à quatre niveaux généralisés a été proposée par (Gulati et Kaur, 2019) en fonction des services et des composants définis dans la littérature pour l'IoT Social. Elle comprend : (i) une couche objet ou de perception où des objets IoT et des dispositifs équipés de capteurs sont présents ; (ii) une couche de communication qui définit les technologies et les protocoles de communication pour communiquer entre les dispositifs et les réseaux de l'IoT Social ; (iii) une couche de gestion de l'IoT social qui comprend les éléments essentiels à la gestion des services de l'IoT social décrits au tableau 1.1 ; iv) une couche d'application qui fournit des interfaces à diverses applications de l'IoT social. La figure 1.2 illustre l'architecture proposée par (Gulati et Kaur, 2019).

1.3.3 Les types de relation

La gestion des relations sociales entre les nœuds de l'IoT Social a été explorée dans diverses études, telles que (Lin et Dong, 2017; Chen et al., 2015; Abdelghani et al., 2022). Ces recherches visent à identifier les multiples types de relations qui peuvent émerger entre les divers acteurs de l'IoT Social, incluant :

(i) la relation d'"amitié" qui représente l'intimité, (ii) la relation de "contact social" qui représente la proximité et l'affinité et (iii) la relation "communauté d'intérêt" qui renvoie à des connaissances ou des expériences communes. Ces trois types de relations sociales ont été introduites par (Chen et al., 2016) et sont inspirées des interactions présentes dans les réseaux sociaux traditionnels, reliant les propriétaires d'objets.

Dans la même veine, (Atzori et al., 2012, 2011) proposent diverses formes de socialisation entre les objets. La relation "parentale" est définie entre les objets similaires, construits à la même période par le même propriétaire. Les objets peuvent aussi établir une relation de "Proximité" ou une relation de "Co-travail", comme dans le cas des humains lorsqu'ils partagent des expériences personnelles (par exemple, cohabitation) ou publiques (par exemple, le travail). Un autre type de relation est défini pour les objets appartenant au même utilisateur et est

nommé relation de "propriété". La dernière relation est établie lorsque les objets qui entrent en contact, de façon continue, pour des raisons purement liées aux relations entre leurs propriétaires et est nommée relation "Sociale".

Nous remarquons que dans le réseau IoT Social, (Chen et al., 2016; Atzori et al., 2012, 2011) sont partagés entre ceux qui s'appuient sur les réseaux pair-à-pair et optent uniquement pour les relations entre objets et d'autres qui se basent sur les réseaux sociaux et ne considèrent que les relations entre les propriétaires d'objets. Cependant, les réseaux IoT Social sont une combinaison des deux et les auteurs devraient considérer à la fois les objets et les acteurs humains. Ainsi, différents types de relations peuvent opérer, tels que : les relations utilisateurs-utilisateurs, dispositifs-dispositifs et utilisateurs-dispositifs pour donner suite à un réseau bipartite et multidimensionnel.

1.3.4 Les défis du réseau IoT Social

L'IoT social est confronté à un certain nombre de défis liés liés à la découverte de services, ce qui peut affecter la qualité de ses performances. Pour instaurer un système robuste et accroître sa convivialité ainsi que son applicabilité, il est nécessaire de mettre en avant plusieurs conditions préalables.

Par exemple, le composant de découverte de services doit renvoyer un résultat qui correspond aux préférences des utilisateurs. Si ce système ne parvient pas à fournir le service pertinent, il devra fournir une liste de services recommandés (Roopa et al., 2019). En outre, l'une des conditions préalables les plus saillantes est la capacité à fournir aux utilisateurs des services dignes de confiance. Par conséquent, le composant de gestion de la confiance, conçu pour le système de découverte, doit résister aux comportements malveillants à l'encontre d'un demandeur de services. Comme certains utilisateurs sont en concurrence les uns avec les autres pour être choisis comme fournisseurs de services (Masmoudi et al., 2020), certains d'entre eux ont recours à des comportements malveillants et lancent des attaques dites de confiance. Ces dernières empêchent un demandeur de services d'estimer la fiabilité des services reçus et de choisir un service digne de confiance.

Parmi les diverses questions qui se posent, ce rapport se concentre sur les suivantes : Comment créer un environnement de confiance qui puisse résister aux attaques de confiance ? Comment isoler les nœuds qui se comportent mal ? Comment évaluer la confiance des transactions effectuées entre les utilisateurs ? Il est nécessaire de mettre en place un mécanisme de gestion de la confiance dans l'IoT social afin d'augmenter les performances de découverte de la qualité de service.

1.4 La gestion de la confiance dans l'IoT Social

À l'ère de l'IoT social, la confiance a été considérée comme un facteur important pour fournir des expériences sécurisées, fiables et transparentes en fournissant un système de découverte de services digne de confiance (Chahal et al.,

2020; Khan et al., 2020; Roopa et al., 2019; Abdelghani et al., 2016). Par conséquent, un mécanisme de gestion de la confiance est une condition préalable pour augmenter la fiabilité du système de découverte de services.

Tout au long de cette section, notre première étape consiste à définir le concept de confiance. L'étape suivante consiste à présenter les différents attaques de confiance citées dans la littérature et les phases du mécanisme de gestion de la confiance.

1.4.1 La confiance

Avec l'évolution rapide du nombre d'utilisateurs d'objets connectés dans les environnements IoT social, les utilisateurs sont confrontés au défi de la sélection des utilisateurs confiants et du service le plus approprié à leur situation. De ce fait, l'accent est mis sur l'impact de la confiance pour la prise de décision.

Avant d'entrer dans les détails du mécanisme de gestion de la confiance proposés dans la littérature, nous allons d'abord définir la confiance et les notions qui s'articulent autour de ce terme.

1.4.1.1 Définition de la confiance

Le concept de confiance a été largement étudié dans divers domaines, notamment les sciences sociales, l'économie, la philosophie et le cyberespace. En fait, la confiance est un terme complexe qui n'a pas de définition commune. Il est défini différemment dans la littérature, en fonction du domaine de recherche.

Selon (Abdelghani et al., 2016), la confiance est une relation qui comprend au moins deux entités principales : un "Trustor", ou la personne qui fait confiance, et un "Trustee", ou la personne à qui l'on fait confiance. En outre, comme l'indiquent (Jayasinghe et al., 2018), le Trustor représente une entité censée initier une transaction avec une autre entité, tandis que le Trustee est la seconde entité qui fournit les informations nécessaires au Trustor pour sa demande. Toutefois, dans les sciences sociales, une attention particulière est accordée à l'effet de la confiance sur la prise de décision (Beldjilali, 2016). De même, dans le contexte de l'informatique en nuage (Cloud computing en anglais), la confiance vise à garantir des environnements plus fiables pour aider les utilisateurs à prendre la bonne décision et à sélectionner les utilisateurs, les appareils et les services les plus dignes de confiance.

Par conséquent, il n'existe donc pas de définition unique de la confiance. Néanmoins, il est clair que la confiance est un élément clé de l'IoT social. Les dispositifs IoT sont conçus pour collecter des données provenant de différents contextes, et ces données peuvent être utilisées pour prendre des décisions importantes, telles que la gestion de la sécurité, la gestion de la santé, etc. Dans ce contexte, la confiance est un élément crucial pour garantir que les utilisateurs peuvent se fier aux données collectées et que les décisions prises sont basées sur des données fiables. Il est essentiel de mettre en place des mesures de sécurité

appropriées, de garantir la confidentialité et la transparence des données et de gérer les risques afin d'instaurer une confiance durable entre les utilisateurs en utilisant un mécanisme de gestion de la confiance. Son objectif est d'améliorer la sécurité et la confiance en soutenant le processus de prise de décision.

1.4.1.2 Les propriétés de la confiance

Dans la littérature, plusieurs travaux ont cherché à quantifier la notion de confiance et ont proposé différentes mesures selon les propriétés considérées (Lin et Dong, 2017; Abdelghani et al., 2022).

- La confiance peut être **directe** ou **indirecte**. Elle est considérée directe si elle est bâtie sur la base d'expériences passées établies directement entre le « trustor » et le « trustee ». Elle est considérée indirecte si elle est bâtie sur la base de l'expérience des autres entités avec le « trustee ».
- La confiance peut être **locale** ou **globale**. La confiance locale est une mesure qui dépend du couple truster/trustee considéré et qui varie d'un couple d'entités à un autre. Autrement dit, une entité i peut être considérée digne de confiance du point de vue d'une entité j, tandis qu'une autre entité k, peut considérer i comme non digne de confiance. La confiance est dite globale, si chaque nœud ou entité dans le réseau a une valeur de confiance unique qui est bâtie sur la base de l'expérience passée de l'ensemble des entités du réseau avec le nœud en question. La confiance globale est aussi appelée réputation ou valeur de confiance dans le réseau.
- La confiance peut être **symétrique** ou **asymétrique**. Néanmoins, la majorité des travaux la considèrent comme étant asymétrique puisque si une entité i a confiance en une entité j, ceci n'implique pas forcément que j a confiance en i. Certains travaux, abordent la notion de confiance symétrique ou réciproque et l'appellent relation d'intimité.
- La confiance peut être **subjective** ou **objective**. La confiance est intrinsèquement une opinion personnelle subjective qui est basée sur divers facteurs ou preuves auxquels certains peuvent attribuer plus ou moins d'importance. Néanmoins, dans certains cas, elle est considérée objective, lorsque la confiance est calculée par exemple en fonction des propriétés QoS d'un dispositif.
- La confiance est **contextuelle** si le degré de confiance d'une entité dans une autre dépend principalement du contexte en question. En effet, une entité i peut accorder de la confiance à une entité j dans un domaine donné mais pas dans un autre.
- La confiance est **dynamique** si elle varie et évolue d'une façon non-monotone avec le temps. Elle peut être périodiquement rafraîchie ou révoquée et doit pouvoir s'adapter aux conditions changeantes de l'environnement dans lequel la décision de confiance a été prise.
- La confiance est **composée** si elle consiste en une composition de nombreux différents facteurs tels que : la fiabilité, l'honnêteté, l'affinité, la véracité ou encore la compétence.

1.4.2 Les attaques de confiance

Une attaque est un comportement malveillant établi par un nœud malveillant effectuée pour briser les fonctionnalités de base d'un système donné et pour atteindre une variété de fins malveillantes. Un nœud malveillant, en général, peut effectuer des attaques de protocole de communication pour perturber les opérations du réseau. Nous estimons que ce type d'attaque est déjà traité par les techniques de détection d'intrusion et n'est donc pas abordé dans notre travail.

Dans le contexte de l'IoT Social, nous nous intéressons aux attaques de confiance. Dans ce type d'attaques, un nœud malicieux peut améliorer sa propre réputation ou valeur de confiance dans le réseau pour accéder à des fonctions plus élevées dans le système ou perturber le système généralement de manière à réduire son efficacité globale dans le découverte de service. De ce fait, une attaque de confiance est définie comme un comportement malveillant qui empêche un demandeur de service d'estimer la fiabilité du service reçu et de sélectionner un service social digne de confiance (Masmoudi et al., 2020, 2021).

Ainsi, un utilisateur malveillant peut effectuer les attaques suivantes (Masmoudi et al., 2020, 2021; Abdelghani et al., 2018; Guo et al., 2017; Ekbatanifard et Yousefi, 2019) :

— Attaque Auto-Promotive (AAP), en anglais **Self-Promoting Attack (SPA)** : où un utilisateur peut promouvoir sa valeur de confiance dans le réseau (s'approprier des votes élevées) afin d'être sélectionné par d'autres utilisateurs.

— Attaque Discriminatoire (AD), en anglais **Discriminatory Attack (DA)** : lorsqu'un utilisateur malveillant fournit de mauvais votes à la majorité des utilisateurs (soit malveillants soit bienveillants).

— Attaque de Mauvaise Attitude (AMA) en anglais **Bad-Mouthing Attack (BMA)** : où un utilisateur peut ruiner la valeur de confiance d'autres utilisateurs bienveillants en fournissant de mauvais votes pour diminuer leurs chances d'être sélectionnés par d'autres utilisateurs.

— Attaque de Bourrage de Vote (ABV), en anglais **Ballot-Stuffing Attack (BSA)** : où un utilisateur malveillant peut augmenter la valeur de confiance d'autres nœuds malveillants en attribuant des votes élevés pour augmenter leurs chances d'être choisi par d'autres.

— Attaque de Service Opportuniste (ASO) en anglais, **Opportunistic Service Attack (OSA)** : où un utilisateur malveillant peut fournir des services de bonne qualité pour obtenir une valeur de confiance élevée dans le réseau, puis commencer à fournir des attaques de types BSA ou BMA.

— Attaque On-Off (AOO), en anglais **On-Off Attack (OOA)** : lorsqu'un utilisateur malveillant effectue des attaques de type BMA, BSA ou SPA au hasard pour éviter d'être étiqueté comme utilisateur malveillant.

Ces attaques peuvent avoir des conséquences graves, telles que la perte de données, la violation de la vie privée, la perte d'une vie précieuse dans certains cas, etc. Pour prévenir ces attaques, il est important de mettre en œuvre des mesures de sécurité appropriées, telles que des mécanismes de surveillance des transactions entre les utilisateurs en temps réel, l'authentification, le cryptage des données, le contrôle d'accès, etc. Pour cela, nous avons besoin d'un mécanisme robuste de gestion de la confiance qui réponde à ces besoins. Avant cela, le mécanisme de gestion de la confiance mérite d'être défini.

1.4.3 Le mécanisme de gestion de la confiance

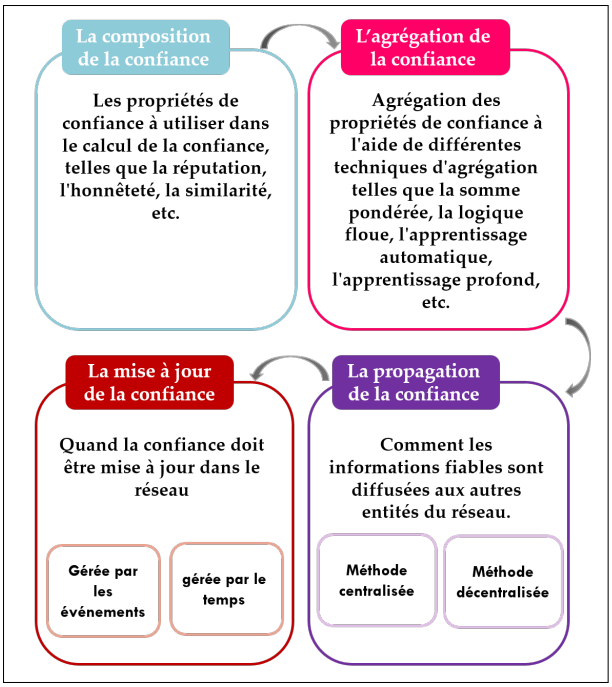

FIGURE 1.3 – *Le mécanisme de gestion de la confiance*

Le mécanisme de gestion de la confiance est un système ou un processus qui établit la confiance entre deux nœuds en diminuant l'opportunisme et la vulnérabilité entre les nœuds inconnus. Ainsi, les utilisateurs peuvent prendre des risques, établir la confiance et décider s'ils veulent interagir (Mendoza et Kleinschmidt, 2018). Ce mécanisme peut être considéré comme un facteur de construction de la confiance en fournissant une communication directe entre des nœuds inconnus, comme dans le cas de l'IoT social. Il représente un moyen d'encourager la collaboration de confiance entre les nœuds tout en réduisant l'importance de la participation ou du fonctionnement anormal des nœuds non dignes de confiance (Guo et al., 2017). En effet, en utilisant des mécanismes de gestion de la confiance, les systèmes IoT sociaux peuvent garantir que seules les entités dignes de confiance sont autorisées à accéder et à interagir avec des données et des services sensibles. Cette mesure renforce la sécurité et la fiabilité globales du système tout en améliorant l'expérience de l'utilisateur. Par conséquent, la gestion de la confiance est devenue un défi majeur pour fournir des interactions

fiables, des expériences agréables et dignes de confiance, et des services qualifiés. À ce stade, la question posée est la suivante : Comment la confiance entre les nœuds peut-elle être évaluée, établie, maintenue et révoquée à l'aide d'un mécanisme de gestion de la confiance au sein du réseau de l'IoT social ?

À cet égard, (Abdelghani et al., 2016; Guo et al., 2017) se sont concentrés sur les mécanismes de gestion de la confiance à l'aide d'une approche multiservice basée sur quatre dimensions différentes, à savoir **la composition** de la confiance, **l'agrégation** de la confiance, **la propagation** de la confiance et **la mise à jour** de la confiance, comme illustré dans la figure 1.3. Ces dimensions sont définies comme suit :

— **La composition de la confiance** comprend les éléments à prendre en compte dans l'évaluation de la confiance en répondant à la question suivante : Quelles propriétés de confiance devraient être utilisées dans le calcul de la confiance ? Les composantes de la confiance les plus courantes dans la littérature sont la réputation, l'honnêteté et la similarité (Abdelghani et al., 2018; Masmoudi et al., 2020; Chen et al., 2016; Truong et al., 2016; Xia et al., 2019).

— **L'agrégation de la confiance** est le meilleur moyen de collecter des preuves pour l'évaluation de la confiance en agrégeant les propriétés de la confiance à l'aide de différentes techniques d'agrégation telles que la somme pondérée, la logique floue, le modèle bayésien, la théorie de la croyance, l'analyse de régression, l'apprentissage automatique et l'apprentissage profond (Masmoudi et al., 2020; Abdelghani et al., 2018).

— **La propagation de la confiance** se concentre sur la diffusion d'informations de confiance aux autres entités d'un réseau. La propagation de la confiance peut être soit centralisée, lorsqu'une entité centrale est responsable de la diffusion de la confiance à travers toutes les entités, soit décentralisée, lorsque chaque entité calcule indépendamment les valeurs de confiance et propage les informations de confiance aux autres entités (Abdelghani et al., 2016).

— **La mise à jour de la confiance** concernent le moment où la confiance doit être mise à jour dans le réseau. Les mises à jour des informations de confiance peuvent être effectuées après un événement ou une transaction (méthode gérée par les événements)) ou périodiquement (méthode gérée par le temps) (Abdelghani et al., 2016).

En résumé, compte tenu de l'importance du mécanisme de gestion de la confiance dans l'évaluation, l'établissement et le maintien de la confiance entre les nœuds au sein du réseau de l'IoT social ainsi que dans le traitement des attaques de confiance, nous examinerons les travaux existants dans la section suivante afin d'identifier les principales lacunes et limitations et de suggérer des orientations futures potentielles en termes de prévention des attaques de confiance.

1.5 Les mécanismes de gestion de la confiance proposés dans l'IoT Social

La gestion de la confiance a été largement étudiée dans de nombreux environnements réseau tels que les réseaux peer-to-peer, ad hoc, grid, les réseaux de capteurs sans fil, etc. Cependant, il existe peu de travaux de recherche qui traitent de la gestion de la confiance dans l'IoT social, en particulier ceux qui traitent des attaques de confiance. Selon la littérature, pour faire face à ces attaques, nous distinguons d'abord deux catégories d'approches, celles qui détectent les attaques de confiance (Kumar et al., 2020; Kowshalya et Valarmathi, 2017a,b; Talbi et Bouabdallah, 2020; Truong et al., 2016; Ekbatanifard et Yousefi, 2019; Masmoudi et al., 2020; Abdelghani et al., 2018) et d'autres qui préviennent et empêchent ces attaques (Abderrahim et al., 2017; Masmoudi et al., 2021). Chaque approche a ses propres avantages et inconvénients, que nous allons aborder dans les sous sections suivantes.

Dans cette section, nous allons étudier les travaux dont l'objectif principal était la détection et la prévention des attaques liées à la confiance dans le réseau IoT social. Un tableau comparatif est présenté dans le tableau 1.2 afin de combler les principales lacunes des travaux existants. Pour cela, nous allons comparer et analyser ces différentes approches selon les critères suivantes : (i) catégorie d'approche : détection ou prévention des attaques, (ii) attaques traitées, (iii) propriétés de confiance, (iv) méthode d'agrégation (v) méthode de propagation et (vi) méthode de mise à jour. De plus, nous allons classer ces travaux en fonction des attaques traitées dans la figure 1.4.

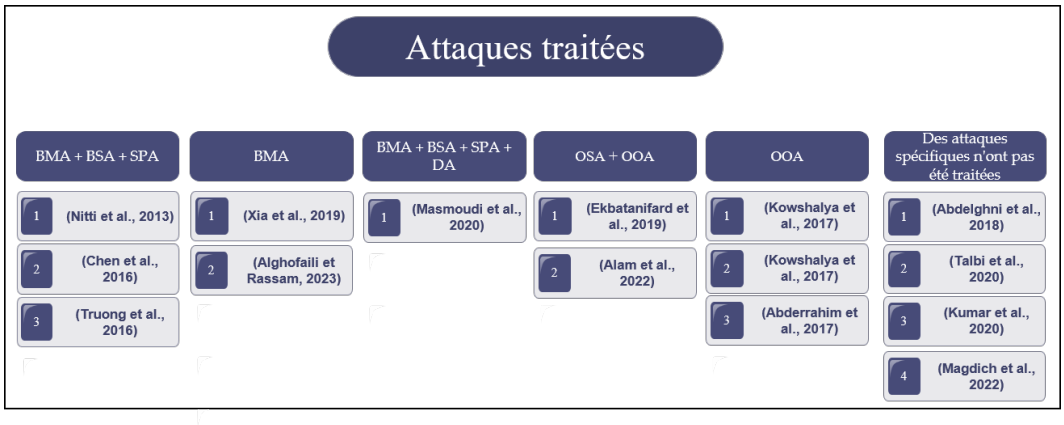

Figure 1.4 – *Typologie des attaques de confiance traitées dans l'IoT social : une classification des travaux de la littérature*

Les mécanismes de gestion de la confiance proposés sont basés sur différentes propriétés de confiance (ou facteurs) telles que la centralité, l'honnêteté, la recommandation, les expériences directes, la similarité, l'observation directe, le feedback, etc. Dans la phase d'agrégation, ils ont utilisé des méthodes comme la moyenne pondérée, l'apprentissage automatique, l'apprentissage profond, le

TABLE 1.2 – *Comparaison des mécanismes de gestion de la confiance proposés dans l'IoT social*

Mécanismes de gestion de la confiance	Catégorie d'approche		Attaques traitées	Propriétés de confiance	Méthode d'agrégation	Méthode de propagation	Méthode de mise à jour
	Détection	Prévention					
(Alghofaili et Rassam, 2023)	✓	✗	BMA	(i) perte de paquets, (ii)retard, (iii)débit	Apprentissage Profond	Centralisée	Gérée par le temps
(Magdich et al., 2022)	✓	✗	Des attaques spécifiques n'ont pas été traités	(i) réputation (ii) recommandation , (iii)similitude sociale , (iv)hostilité au travail , (v)communauté d'intérêts , (vi)coopération et (vii)I-SoR	Apprentissage automatique	Décentralisée	Gérée par le temps
(Alam et al., 2022)	✓	✗	OOA + DA + OSA	(i)Confiance social, (ii) qualité du service	Blockchain	Decentralisée	Gérée par le temps
(Kumar et al., 2020)	✓	✗	Des attaques spécifiques n'ont pas été traités	(i) contexte, (ii) transitivité	Apprentissage en profondeur + Blockchain	Décentralisée	-
(Masmoudi et al., 2020)	✓	✗	BMA + BSA + SPA + DA	(i) réputation, (ii)honnêteté, (iii) qualité du fournisseur, (iv) similarité, (v) expérience directe, (vi) fréquence des notations, (vii) tendance des notations	Apprentissage en profondeur	-	-
(Xia et al., 2019)	✓	✗	BMA	Confiance de familiarité : (i) confiance directe, (ii) recommandation. Confiance de similarité : (i) confiance de similarité externe (centralité), (ii) interne (communauté d'intérêts)	Méthode de logique floue	Centralisée	Gérée par le temps
(Talbi et Bouabdallah, 2020)	✓	✗	Des attaques spécifiques n'ont pas été traités	(i) expériences directes, (ii) expériences indirectes	Somme pondéré	Centralisée	Gérée par le temps
(Ekbatanifard et Yousefi, 2019)	✓	✗	OSA + OOA	(i) renvoyé sur le service, (ii) intimité, (iii) sociabilité, (iv) importance de la transaction	Méthode combinatoire	Décentralisée	Gérée par les évènements
(Abdelghani et al., 2018)	✓	✗	Des attaques spécifiques n'ont pas été traités	(i) réputation, (ii) honnêteté, (iii) qualité du fournisseur, (iv) similarité, (v) expérience directe, (vi) la fréquence des notations, (vii) tendance des notations	Apprentissage automatique	-	-
(Kowshalya et Valarmathi, 2017a)	✓	✗	OOA	(i) confiance directe, (ii) centralité, (iii) communauté d'intérêts, (iv) coopération, (v) score de service	Modèle bayésien	Centralisée	Gérée par le temps
(Kowshalya et Valarmathi, 2017b)	✓	✗	OOA	(i) observations dirigent, (ii) recommandations indirectes, (ii) centralité, (iv) énergie, (v) score de service	Méthode de logique floue	Centralisée	Gérée par le temps
(Abderrahim et al., 2017)	✗	✓	OOA	(i) observations dirigent, (ii) observations indirectes	Technique du filtre de Kalman	Centralisée	Gérée par le temps
(Truong et al., 2016)	✓	✗	BMA + BSA + SPA	(i) recommandation, (ii) réputation, (iii) connaissances, (iv) préférences de l'utilisateur, (v) contexte du service	Théorie de l'utilité multicritère	-	Gérée par les événements
(Chen et al., 2016)	✓	✗	BMA + BSA + SPA	(i) réputation, (ii) relations sociales, (iii) niveau d'énergie	Somme pondérée	Centralisée	-
(Nitti et al., 2013)	✓	✗	BMA + BSA + SPA	Approche subjective : (i) centralité, (ii) expérience directe, (iii) expérience indirecte Objectif d'approche : (i) centralité, (ii) observations à court terme, (iii) observations à long terme	Somme pondérée	Décentralisée	-

filtre de Kalman, la méthode de la logique floue, etc. Dans la phase de propagation, certains d'entre eux ont utilisé une méthode centralisée tandis que d'autres ont employé une méthode décentralisée. Tandis que, dans la phase de mise à jour, ils ont utilisé une méthode gérée par le temps ou par les événements.

1.5.1 L'approche de détection des attaques

L'approche de détection des attaques consiste à scanner et à identifier les actions qui tentent de compromettre la confidentialité ou l'intégrité d'un réseau (Dagorn, 2006). En fait, le processus de détection tente de rechercher des signes suspects qui peuvent indiquer des activités anormales ou suspectes sur la cible analysée (un réseau) afin d'alerter les utilisateurs de la présence d'un comportement anormal. En outre, il peut fournir des informations précieuses sur l'attaque, telles que le type d'attaque et les systèmes affectés, ce qui peut faciliter la réponse à l'incident. Cependant, le processus de détection ne peut pas arrêter ou bloquer les activités anormales ou les attaques. De plus, ces activités anormales ont été découvertes après qu'elles se soient produites. On dit que le processus de détection est généralement "a posteriori" sur la courbe temporelle, indiquant ainsi que les attaques peuvent avoir déjà causé des dommages avant d'être détectées. Cette situation peut entraîner des temps d'arrêt, des pertes de données et d'autres conséquences négatives.

A cet égard, la majorité de ces mécanismes se sont concentrés sur la détection des attaques hors ligne (Ekbatanifard et Yousefi, 2019; Kowshalya et Valarmathi, 2017a,b; Xia et al., 2019; Truong et al., 2016; Kumar et al., 2020). Dans le travail de (Talbi et Bouabdallah, 2020), les auteurs ont proposé un mécanisme de gestion de la confiance caractérisé par deux propriétés : (i) les expériences directes et (ii) les expériences indirectes. Ces propriétés ont été agrégées à l'aide de la méthode de la somme pondérée. Ce mécanisme peut détecter les attaques de confiance hors ligne sans spécifier leurs types. Cependant, les nœuds malveillants ne peuvent pas être isolés. En outre, une méthode dirigée par le temps a été utilisée pour mettre à jour les informations de confiance. Un problème crucial dans les méthodes basées sur le temps est le choix de la granularité temporelle la plus appropriée. À cet égard, l'identification des attaques en temps réel peut constituer une solution idéale.

Ce travail (Ekbatanifard et Yousefi, 2019) a présenté un nouveau mécanisme de gestion de la confiance qui consiste en quatre propriétés distinctes : (i) le retour d'information sur le service, (ii) l'intimité, (iii) la sociabilité et (iv) l'importance de la transaction. Une méthode combinatoire a été utilisée pour agréger ces propriétés. Une méthode décentralisée est également appliquée pour propager les informations de confiance. En outre, une méthode de mise à jour pilotée par les événements dans laquelle le nœud i interagit avec le nœud j est utilisée. Ce mécanisme ne permet pas d'isoler les nœuds malveillants. Toutefois, de manière hors ligne, ils ont réussi à détecter les attaques OSA et OOA.

L'étude menée par (Kumar et al., 2020), visait à développer un mécanisme de gestion de la confiance basé sur une approche d'intelligence artificielle pour

renforcer la confiance entre les dispositifs du réseau IoT social. Une méthode décentralisée est également appliquée pour propager les informations de confiance. Bien que ce mécanisme ne puisse pas empêcher les attaques, il pourrait détecter les attaques en mode hors ligne sans spécifier le type de l'attaque lancée par l'utilisateurs malicieux.

Dans (Xia et al., 2019), les auteurs ont introduit deux mécanismes de gestion de la confiance utilisant quatre propriétés différentes : (i) confiance directe, (ii) recommandation, (iii) confiance de similarité externe, et (iv) communauté d'intérêt interne. Une méthode centralisée a été utilisée dans les deux mécanismes pour propager les informations de confiance. Leur détection des attaques BMA s'est faite hors ligne. Cependant, ils dépendent fortement d'une entité centralisée pour collecter et calculer les valeurs de confiance. Or, cette entité centralisée représente un point de défaillance potentiel, en raison de ses exigences élevées. Il est donc recommandé d'utiliser une méthode décentralisée ou distribuée dans laquelle une telle entité centralisée n'est pas nécessaire.

(Kowshalya et Valarmathi, 2017a,b) ont proposé deux mécanismes de gestion de la confiance composés de différentes propriétés : (i) confiance directe, (ii) centralité, (iii) communauté d'intérêt, (iv) coopération, et (v) score de service. Ces mécanismes sont basés sur des méthodes bayésiennes et logiques floues. En outre, les informations de confiance ont été non seulement propagées à l'aide de la méthode centralisée, mais également mises à jour à l'aide de la méthode gérée par le temps. Les deux mécanismes ont permis de détecter les attaques OOA en mode hors ligne plutôt que d'isoler les nœuds malveillants.

(Chen et al., 2016; Nitti et al., 2013) proposent deux approches de gestion de la confiance avec des propriétés différentes combinées en utilisant la méthode de la somme pondérée. Pour propager les valeurs de confiance, la première approche met en œuvre une méthode centralisée, tandis que la seconde utilise une méthode décentralisée. Les deux approches permettent de détecter les attaques SPA, BMA et BSA en mode hors ligne.

Dans une étude réalisée par (Truong et al., 2016), un mécanisme de confiance incluant deux entités IoT Social a été proposé. Ce mécanisme intègre diverses propriétés de confiance agrégées sur la base de la théorie de l'utilité multicritère. Les valeurs de confiance ont été propagées de manière semi-centralisée et la mise à jour a été faite à l'aide d'une méthode événementielle. Le mécanisme proposé ne permet pas d'isoler les nœuds malveillants. Toutefois, de manière hors ligne, ils ont réussi à détecter les attaques SPA, BMA et BSA.

Même dans nos travaux précédents (Masmoudi et al., 2020; Abdelghani et al., 2018), nous nous sommes principalement concentrés sur la détection des attaques. Nous avons d'abord utilisé une méthode d'apprentissage automatique pour classer les interactions précédentes des utilisateurs en deux classes principales (attaques ou non-attaques). Nous avons ensuite (Masmoudi et al., 2020) amélioré le travail réalisé par (Abdelghani et al., 2018) en utilisant la technique d'apprentissage profond. De même, nous avons spécifié le type d'attaques en

les classant en cinq catégories différentes (BMA, BSA, SPA, DA et non-attaque). Nous nous sommes appuyés sur sept propriétés : (i) la réputation, (ii) l'honnêteté, (iii) la qualité du fournisseur, (iv) la similarité des utilisateurs, (v) l'expérience directe, (vi) la fréquence des évaluations et (vii) la tendance des évaluations. Comme nous n'avons l'intention de détecter que les attaques en mode hors ligne, ces méthodes ne seront pas en mesure d'interrompre les transactions malveillantes ou les attaques de confiance. En d'autres termes, sur la base de ces études, nous ne serons pas en mesure de prévenir les attaques en temps réel.

1.5.2 L'approche de prévention des attaques

En revanche, l'approche de prévention des attaques (Dagorn, 2006) est une approche proactive de la sécurité qui implique d'éviter les attaques avant qu'elles ne se produisent. Cette démarche vise à diminuer les risques de dommages, à prévenir les interruptions et à éviter les pertes de données. En outre, elle vise non seulement à anticiper et à empêcher les activités indésirables et suspectes de circuler dans le réseau, mais aussi à garantir qu'aucune nouvelle activité ou attaque malveillante ne puisse s'infiltrer dans le réseau. Pour ce faire, elle surveille le comportement des utilisateurs et bloque les attaques à la source, par exemple en bloquant le trafic malveillant au niveau du pare-feu. Cela empêche l'attaque d'atteindre le système et de causer des dommages (Kenkre et al., 2015). Une stratégie de sécurité globale peut combiner la détection et la prévention pour prévenir les attaques en temps réel, maximiser la protection et minimiser les risques.

À cet égard, dans le travail mené par (Abderrahim et al., 2017), les auteurs ont proposé un mécanisme de gestion de la confiance qui permet de prévenir les attaques OOA et l'isoler les nœuds malveillants. Ce mécanisme consiste en deux propriétés différentes : (i) les observations directes et (ii) les observations indirectes. Elles ont été agrégées à l'aide de la technique du filtre de Kalman. Les auteurs ont utilisé non seulement une méthode centralisée pour propager les informations de confiance, mais aussi une méthode temporelle pour les mettre à jour. Cependant, certaines attaques sont plus dangereuses que d'autres, en fonction du contexte dans lequel elles ont été appliquées (Masmoudi et al., 2020). De plus, une attaque peut causer des dommages irréversibles, perturber un système et réduire son efficacité (Abdelghani et al., 2016). Dans un tel scénario, si un utilisateur ou "demandeur de service", souhaite vérifier l'état des routes, il recevra plusieurs services sans pouvoir estimer leur fiabilité. Cela peut s'expliquer par le fait que certains utilisateurs ou "fournisseurs de services" peuvent améliorer leur valeur de confiance dans le réseau pour propager des comportements malveillants. Il est donc essentiel de prévenir tous les types d'attaques (BMA, BSA, SPA, DA, OSA et OOA) pour obtenir un environnement fiable en mettant en œuvre une mesure de sécurité appropriée, telle que des mécanismes de surveillance des transactions en temps réel entre les utilisateurs. Pour ce faire, nous avons besoin d'un mécanisme robuste de gestion de la confiance qui réponde à ces besoins.

TABLE 1.3 – *Une analyse comparative visant à résoudre les principales limitations des travaux existants concernant la prévention des attaques de confiance en temps réel*

Critère	Travaux existants	Notre travail
Approche utilisée	-La majorité des travaux ont été portés sur la détection d'attaques. -Un seul travail s'est concentré sur la prévention des attaques.	-Nous nous sommes concentrés sur la prévention des attaques.
Temps réel	-Détection et prévention en mode hors ligne	- Nous nous articulons sur la dimension temps réel.
Attaque adressée pour la détection	-BMA, BSA, SPA, DA, OSA, OOA (chaque attaque a été traitée séparément) -Sans spécificité le type d'attaque traité	-Toutes les attaques de confiance (BMA, BSA, SPA, DA, OSA et OOA)
Attaque adressée pour la prévention	-OOA	-Toutes les attaques de confiance (BMA, BSA, SPA, DA, OSA et OOA)
Propriétés de sécurité	-Ne tient pas compte les propriétés de sécurité	-Nous prenons en compte les propriétés de sécurité
Méthode de mise à jour	-Pas en temps réel	-En temps réel

D'autre part, nous remarquons que les mécanismes de gestion de la confiance proposés dans la littérature ne prennent pas en compte les propriétés de sécurité (Calvaresi et al., 2018), telles que la confidentialité, l'intégrité, la cryptographie, la transparence, l'authentification, l'autorisation et l'immutabilité. Pour souligner l'importance de ces propriétés de sécurité, nous comparons dans le chapitre suivant des techniques et technologies dont le point commun est la prévention des attaques. Cette comparaison nous aidera à sélectionner la technique ou la technologie la plus appropriée pour la prévention des attaques tout en maintenant les propriétés de sécurité.

En résumé, l'un des principaux défis dans les environnements IoT sociaux est de développer un mécanisme de gestion de la confiance basé sur une architecture décentralisée qui peut identifier et interrompre les transactions malveillantes en temps réel afin de prévenir tous les types d'attaques de confiance d'une part et d'autre part de maintenir les propriétés de sécurité. A cet égard, un tableau comparatif est présenté dans le tableau 1.3.

1.6 CONCLUSION

Ce chapitre a permis de présenter les différents environnements de l'IoT et l'IoT social, la notion de confiance ainsi que les concepts qui y sont associés, les différentes attaques de confiance et le mécanisme de gestion de la confiance.

Suite à cette étude, nous avons constaté que l'IoT social se présente comme un paradigme prometteur, mais soulève des interrogations quant à l'absence de confiance. Ainsi, les préoccupations concernant les attaques sur la confiance représentent les principaux obstacles à l'adoption de cette technologie. En conséquence, divers mécanismes de gestion de la confiance ont été suggérés dans la littérature pour aborder cette problématique.

La majorité des travaux sont conçus pour détecter les attaques hors ligne avec

ou sans spécifier le type d'attaque réalisée. En revanche, les travaux visant à prévenir ces attaques ne sont capables d'en prévenir qu'un seul type, en utilisant un moyen préventif inefficace qui entraîne une perte de temps. Pour obtenir un environnement de confiance, il est nécessaire de traiter en continu les transactions, c'est-à-dire de réaliser le traitement de flux de transactions pour prévenir ces attaques au niveau de la génération des transactions, en temps réel. De plus, ces travaux ne prennent pas en compte les propriétés de sécurité telles que la cryptographie, la transparence, l'immuabilité, etc.

Dans ce but, nous allons examiner dans le prochain chapitre les techniques et technologies visant à prévenir les attaques dans les domaines liés à l'IoT social tels que les réseaux sociaux et l'IoT, tout en prenant également en compte les propriétés de sécurité telles que la cryptographie, la transparence et l'immuabilité. Cette comparaison nous permettra de sélectionner la technique ou la technologie la plus appropriée pour prévenir les attaques tout en maintenant les propriétés de sécurité. Ensuite, nous passerons en revue certains travaux connexes qui ont utilisé un moteur de traitement de flux pour identifier et détecter les attaques en temps réel lors de la génération des transactions. En combinant la technique appropriée choisie avec le moteur de traitement de flux sélectionné, nous pourrons assurer une prévention plus efficace des attaques de confiance en temps réel.

En résumé, un mécanisme amélioré vise à prévenir tous types d'attaques de confiance (BMA, BSA, SPA, DA, OSA et OOA) en temps réel tout en tenant compte des propriétés de sécurité est un défi majeur pour les environnements IoT sociaux.

2 Comparaison des techniques pour une prévention efficace des attaques de confiance

Sommaire

2.1 Introduction

La littérature scientifique met en avant diverses techniques, technologies et moteurs de traitement de flux de données pour prévenir en temps réel des différentes formes d'attaques dans plusieurs domaines de recherche. Dans ce chapitre, nous présentons une étude des techniques qui ont été exploitées dans des domaines connexes à l'IoT social pour prévenir efficacement les attaques de confiance en temps réel, tout en prenant en compte les propriétés de sécurité. A cet égard, nous effectuons une comparaison détaillée de ces techniques, en mettant en évidence les avantages et les limites de chacune d'elles. Parmi ces techniques, nous avons identifié la blockchain comme la plus appropriée pour la prévention des attaques de confiance, grâce à ses propriétés de sécurité, de transparence et de résilience. En ce qui concerne l'identification des attaques de confiance en temps réel, nous avons déterminé qu'Apache Spark est le meilleur moteur de traitement de flux, grâce à sa vitesse de traitement élevée, sa facilité de mise en œuvre et sa compatibilité avec d'autres technologies telles que la blockchain.

Ce chapitre a pour objectif de présenter une étude approfondie des techniques et des technologies existantes dans la littérature pour la prévention des attaques et des intrusions, ainsi que d'introduire brièvement la technologie de la blockchain. En outre, nous présentons également un défilé des moteurs de traitement de flux existants pour l'identification des attaques et des intrusions en temps réel, tout en introduisant brièvement le Framework Apache Spark. Enfin, nous concluons ce chapitre en résumant les principales conclusions de cette étude comparative des différentes techniques et technologies, et en discutant de leurs avantages et limites respectives.

2.2 Exploration des techniques et technologies existantes pour la prévention des attaques

Afin de concevoir un mécanisme de gestion de la confiance efficace et robuste qui vise à prévenir tous les types d'attaques de confiance possibles tout en maintenant les propriétés de sécurité requises, nous allons mener une analyse approfondie et une comparaison minutieuse des différentes techniques et technologies existantes qui ont pour objectif commun la prévention des différentes formes d'attaques (Calvaresi et al., 2018). Nous examinerons attentivement les avantages et les limites de chaque technique, en mettant en lumière leurs spécificités et leurs performances respectives en matière de prévention des attaques, de sécurité, de transparence et de résilience. Sur la base de ces comparaisons détaillées, nous serons en mesure de sélectionner la technique ou la technologie la plus appropriée pour notre mécanisme de gestion de la confiance, qui sera capable de prévenir efficacement toutes les attaques de confiance potentielles, tout en maintenant les propriétés de sécurité requises pour l'IoT social.

2.2.1 Technique Multi-agents

Dans le travail de (Bakhsh et al., 2019), les auteurs ont présenté un système adaptatif de détection et de prévention des intrusions pour l'Internet des Objets (IoT) afin de renforcer la sécurité parallèlement à la croissance des appareils connectés à Internet. Une fois que le système reçoit le paquet, il examine son comportement. Si le paquet est suspect, il est soit bloqué, soit abandonné. Le système proposé utilise une technique multi-agents pour détecter et prévenir les attaques, telles que les virus, les attaques par Déni de Service (DdS) et les attaques par Déni de Service Distribué (DdSD).

En effet, les systèmes multi-agents sont des systèmes informatiques composés d'un ensemble d'entités autonomes appelées "agents". Chacun est capable de prendre des décisions indépendantes et d'interagir avec son environnement et avec d'autres agents pour atteindre des objectifs spécifiques. Cette technique est capable de valider ou annuler les transactions. Cependant, elle n'est pas capable de garantir la sécurité des transactions et des données, tels que l'intégrité des données, l'anonymisation et la gestion de l'identité.

2.2.2 Technique d'apprentissage automatique

(Feng et al., 2019) ont proposé un nouveau modèle d'analyse des utilisateurs pour identifier les victimes potentielles en analysant les informations personnelles et les comportements sociaux des utilisateurs. Ils attribuent à chaque utilisateur un score de sécurité afin d'alerter les utilisateurs sur les risques élevés. Ce score permet de guider les entreprises pour prévenir les cyberattaques du réseau social. Pour ce faire, ils ont mis en œuvre une technique d'apprentissage.

Dans (Farishta et al., 2022), les auteurs ont proposé une approche basée sur des techniques d'apprentissage automatique pour identifier les attaques de type Cross-Site Scripting (XSS) et injection SQL puisque les applications web sont vulnérables à cette forme d'attaques.

Toutefois, il est important de souligner que les travaux de (Feng et al., 2019; Farishta et al., 2022; Kebede et al., 2022; Kathirkamanathan et al., 2022; Ansari et al., 2022) présentent certaines limites. En effet, ils ne permettent ni de valider ni d'annuler une transaction entre différents utilisateurs. En outre, la sécurité des transactions et des données n'est pas garantie, ce qui peut engendrer des risques de sécurité importants.

2.2.3 Technologie de la blockchain

(Khan et al., 2019) ont proposé une nouvelle architecture décentralisée basée sur la confiance qui utilise la technologie blockchain pour prévenir les attaques de la couche de contrôle d'accès au support "MAC" dans les réseaux ad hoc véhiculaires "VANET". Les attaques sont : les attaques par Déni de Service (DdS), les attaques par modification de données, les attaques par vol d'identité, les attaques par Sybil et les attaques par rejeu.

Dans (Lu et al., 2018), les auteurs ont suggéré un mécanisme de confiance décentralisé basé sur la blockchain, nommé Blockchain-based Anonymous Reputation System "BARS" pour empêcher la distribution de faux messages (attaques) dans les Vehicular Ad-hoc NETworks "VANETs". Ce mécanisme comprend 3 blockchains pour enregistrer les messages diffusés par différents véhicules, les transactions et les clés révoquées.

(Nakayama et al., 2018) se sont appuyés sur un algorithme nommé SPAM Attack Guard Algorithm Using Blockchain "SAGA-BC" pour prévenir les attaques de spam en utilisant la technologie blockchain. Une personne qui envoie un courriel en utilisant cet algorithme doit payer le coût de traitement avec des cryptomonnaies. Cependant, divers courriels de spam ne sont pas reçus normalement, car les adresses des courriels de spam sont indistinctes. Si un e-mail envoyé à l'aide de cet algorithme est reçu normalement sur un compte e-mail de destination, ce coût est remboursé. Par conséquent, les serveurs de messagerie et/ou les expéditeurs de courrier pourront facilement juger les courriers entrants en utilisant cet algorithme.

L'utilisation de la blockchain dans (Khan et al., 2019; Lu et al., 2018; Nakayama et al., 2018; Ibrahim et al., 2022) a permis de répondre à deux défis majeurs dans la gestion de la confiance dans l'IoT social : la sécurité des transactions, et la validation ou l'annulation des transactions entre les différents nœuds. En effet, grâce à ses propriétés de sécurité avancées, la blockchain garantit l'intégrité des données, l'anonymisation et la gestion des identités des utilisateurs. De plus, la blockchain permet la validation ou l'annulation des transactions, sans avoir recours à un tiers de confiance. Cette caractéristique assure une transparence et une fiabilité accrues des échanges entre les différents nœuds, tout en offrant une grande flexibilité et une grande adaptabilité aux besoins spécifiques de chaque environnement IoT social.

2.2.4 Technique du filtre de Kalman

(Abderrahim et al., 2017) ont suggéré un mécanisme centralisé de gestion de la confiance qui utilise des techniques de filtre de Kalman pour estimer les comportements des nœuds et prévenir les attaques OOA dans un réseau IoT social. L'architecture du réseau est un groupe de nœuds appelé communauté d'intérêt (CoI). Chaque nœud de la communauté partage les mêmes intérêts. D'ailleurs, elle est gérée par un administrateur qui est responsable du calcul et de l'enregistrement des valeurs de confiance des membres de sa communauté. Il peut également exclure un nœud malveillant de sa communauté. Dans ce travail, les auteurs ont établi des transactions artificielles pendant une période de temps prédéfinie pour pouvoir acquérir une connaissance initiale des membres et calculer leurs valeurs de confiance afin d'estimer leur prochain comportement pour la prévention des attaques.

Cependant, la technique du filtre de Kalman, bien qu'utile pour la prédiction des comportements malveillants et bienveillants, ne permet ni d'annuler ni de

valider des interactions entre les utilisateurs. De plus, elle ne garantit pas la sécurité des transactions, tels que l'intégrité des données, l'anonymisation et la gestion de l'identité. En effet, son rôle se limite à l'analyse de données passées et présentes pour prédire des comportements futurs, sans offrir de mécanisme de protection contre des actes malveillants.

2.2.5 Synthèse

Nous avons pour objectif dans cette sous-section de classifier les différentes études que nous avons recensées, comme l'illustre la figure 2.1. Nous allons ensuite procéder à une analyse et une comparaison détaillées de ces travaux, en nous appuyant sur plusieurs critères dans le tableau 2.1 : (i) les types d'attaques visées, (ii) les techniques ou technologies employées, (iii) les domaines d'application étudiés, (iv) les types d'architecture utilisés (centralisée ou décentralisée), (v) la capacité de la technique ou de la technologie à annuler ou à valider une transaction, et (vi) la capacité de la technique ou de la technologie à garantir certaines propriétés de sécurité (Calvaresi et al., 2018), telles que l'intégrité des données, l'anonymisation et la gestion de l'identité. En examinant ces critères, nous espérons fournir une synthèse complète de l'état de l'art en matière de prévention des attaques, en mettant en évidence les forces et les limites des différentes approches proposées dans la littérature.

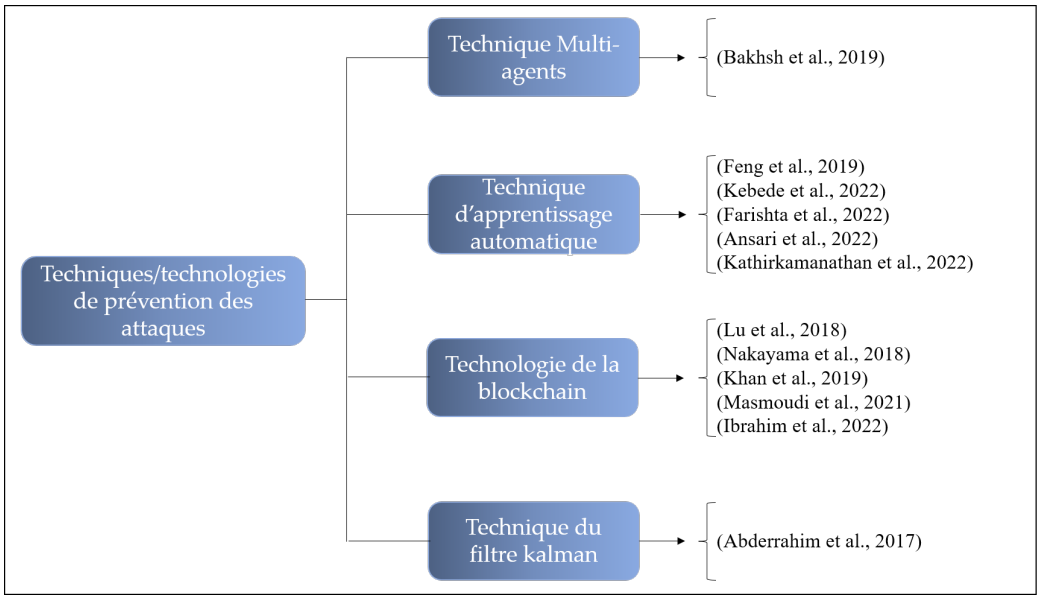

FIGURE 2.1 – *Classement des méthodes et des technologies employées pour la prévention des attaques dans les réseaux sociaux et l'IoT*

De nombreuses techniques et technologies ont été développées pour prévenir diverses formes d'attaques dans différents domaines de recherche, telles que les attaques DoS, DDoS, les attaques de spam et les injections SQL. Toutefois, dans le cadre de notre étude, nous avons choisi de nous concentrer sur deux domaines spécifiques, à savoir l'IoT et les réseaux sociaux, car notre domaine d'étude "IoT

TABLE 2.1 – *Analyse comparative des méthodes et technologies employées pour prévenir différentes formes d'attaques*

	Type d'attaque adressée	**Technique / Technologie**	**Contexte**	**Architecture**	**Valider ou annuler les transactions**	**Propriétés de sécurité**			
						Intégrité des données	Anonymisation	Gestion des identités	Auditabilité
(Bakhsh et al., 2019)	DoS, DDos, etc.	Multi-agents	IoT	Décentralisé	✓	✗	✗	✗	✗
(Feng et al., 2019)	Spear-, phishing, etc.	Apprentissage automatique	Réseaux Sociaux en ligne	-	✗	✗	✗	✗	✗
(Khan et al., 2019)	Attaques de la "MAC"	Blockchain	Réseaux (VANET)	Décentralisé	✓	✓	✓	✓	✓
(Abderrahim et al., 2017)	Attaques OOA	iltre de kalman	IoT social	Centralisé	✗	✗	✗	✗	✗
(Lu et al., 2018)	messages falsifiés	Blockchain	Réseaux (VANET)	Décentralisé	✓	✓	✓	✓	✓
(Nakayama et al., 2018)	Attaques de spam	Blockchain	Boîte de réception e-mail	Décentralisé	✓	✓	✓	✓	✓
(Farishta et al., 2022)	Attaque XSS + injection SQL	Apprentissage automatique	Des applications Web	Centralisé	✗	✗	✗	✗	✗

social" est une combinaison de ces deux domaines. Dans cette optique, nous avons examiné diverses techniques et technologies utilisées pour prévenir ces formes d'attaques. Parmi celles-ci, on peut citer les systèmes multi-agents, qui ont été employées dans des travaux tels que ceux de (Bakhsh et al., 2019) pour prévenir les attaques dans l'IoT. De plus, les techniques d'apprentissage ont également été utilisées dans diverses études pour prévenir les attaques, telles que celles menées par (Feng et al., 2019). De même, la technologie blockchain, qui a été mise en œuvre dans des travaux tels que ceux de (Nakayama et al., 2018; Khan et al., 2019; Lu et al., 2018). Enfin, les techniques de filtre de Kalman ont été exploitées pour prévenir les attaques de types OOA dans les travaux de (Abderrahim et al., 2017).

Contrairement à d'autres technologies telles que les systèmes multi-agents et la blockchain, les techniques d'apprentissage et de filtre de Kalman ont une capacité limitée à annuler ou à valider des transactions. En effet, les techniques d'apprentissage sont principalement utilisées pour classifier les événements ou les comportements comme étant bienveillants ou malicieux, tandis que les techniques de filtre de Kalman sont principalement exploitées pour prédire et estimer le comportement des utilisateurs dans un système. Ces techniques peuvent être utilisées pour détecter des événements malveillants qui pourraient révéler une attaque ou une violation de la sécurité. Toutefois, elles ne sont pas en mesure d'annuler ou de valider des transactions, car elles sont passives et se contentent de prédire et de classifier les événements. Il est donc important de considérer leurs limites lors de la conception de stratégies de sécurité pour notre domaine de recherche IoT social. Toutefois, ces techniques sont complémentaires aux techniques actives telles que la blockchain et les systèmes multi-agents et peuvent être utilisées ensemble pour une protection plus robuste contre les attaques.

D'un autre côté, les technologies blockchain et les systèmes multi-agents ont la capacité de détecter et de prévenir les attaques en utilisant des mécanismes d'intervention active. Les systèmes multi-agents peuvent être conçus pour détecter les menaces à la sécurité et coordonner les actions entre plusieurs agents afin de répondre efficacement aux attaques. Les technologies blockchain, quant à elles, peuvent être exploitées pour le traitement des transactions, avec la capacité d'empêcher les transactions malveillantes et de garantir la fiabilité du système. En outre, les deux technologies ont des architectures décentralisées, ce qui les rend plus résilientes aux attaques en question.

Cependant, malgré leurs avantages, les systèmes multi-agents peuvent être sujets à plusieurs problèmes de sécurité, tels que l'intégrité des données, l'anonymisation, la gestion des identités et la sécurité des données. Ces problèmes ont été identifiés dans la littérature, comme le montrent les travaux de recherche de (Afanasyev et al., 2019; Calvaresi et al., 2018). Pour surmonter ces problèmes, des chercheurs ont proposé des solutions basées sur la technologie blockchain.

Par ailleurs, la technologie blockchain a attiré une attention particulière en raison de ses avantages significatifs dans une multitude de domaines, tels que

la finance, les soins de santé, la banque (Khan et al., 2019) et l'apprentissage en ligne (Mikroyannidis et al., 2020), entre autres. En effet, elle est largement adoptée et utilisée à grande échelle (Iqbal et al., 2019) et a montré des résultats satisfaisants (Guidi, 2020) dans l'atténuation des risques de sécurité (Panarello et al., 2018) grâce à ses caractéristiques. Ces caractéristiques incluent notamment l'intégrité des données (immuabilité), qui garantit que toutes les transactions et données enregistrées ne peuvent pas être modifiées ou supprimées, l'anonymisation en utilisant des clés publiques pour préserver la vie privée des utilisateurs, la transparence (Smik, 2018) qui permet à toutes les transactions d'être visibles par tous les nœuds du réseau, l'auditabilité qui permet de vérifier l'exactitude de tout bloc à tout moment, la gestion de l'identité de l'utilisateur, et la sécurité des données.

Dans le cadre de la prévention des attaques de confiance dans l'IoT social, notre objectif principal est de détecter et d'interrompre les transactions malveillantes qui sont souvent à l'origine de ces attaques. Pour ce faire, nous recherchons une technologie capable de valider ou d'annuler ces transactions tout en maintenant les propriétés de sécurité nécessaires, comme l'a souligné (Calvaresi et al., 2018). De plus, nous cherchons une technologie dotée d'une architecture décentralisée, ce qui est essentiel dans ce contexte.

C'est pourquoi la technologie blockchain se présente comme une solution idéale. Elle a été largement étudiée et adoptée dans divers domaines, notamment la finance, les soins de santé et l'apprentissage en ligne, en raison de ses caractéristiques uniques. Parmi ces caractéristiques, on retrouve l'immuabilité, l'anonymisation, la transparence, l'auditabilité, la gestion de l'identité de l'utilisateur et la sécurité des données. Ces attributs font de la blockchain un outil efficace pour atténuer les risques de sécurité et prévenir les attaques de confiance auxquelles nous sommes confrontés.

Nous souhaitons souligner que notre choix de la technologie blockchain repose sur notre besoin de prévenir les attaques de confiance dans l'IoT social, non seulement en détectant les comportements malveillants, mais également en intervenant activement pour empêcher ces attaques. Cette technologie répond à cette exigence en offrant la validation et l'annulation de transactions de manière décentralisée tout en maintenant les propriétés de sécurité requises. Nous sommes à présent prêts à plonger dans l'univers de la blockchain qui a été développée par une personne ou un groupe de personnes sous le pseudonyme de Satoshi Nakamoto en 2008 (Nakamoto et Bitcoin, 2008).

La blockchain est constituée techniquement d'un nombre illimité de blocs chaînés composés d'un en-tête de bloc et d'un corps de bloc (Khan et al., 2019). L'en-tête de bloc contient des informations importantes telles que le hash du bloc précédent, la date et l'heure de création du bloc et un numéro de séquence unique, tandis que le corps du bloc contient les données de transaction, c'est-à-dire les informations sur les transferts de données entre les parties impliquées.

Chaque bloc de cette technologie est connecté au bloc précédent de manière cryptographique en incluant un identifiant unique dans son en-tête de bloc, tel

que le hash de bloc, l'horodatage, ou le hash de transaction du bloc précédent (Khan et al., 2019). Ce processus itératif permet de confirmer l'intégrité du bloc précédent, remontant ainsi jusqu'au bloc initial appelé "bloc genèse". Afin d'assurer l'intégrité d'un bloc et des transactions qu'il contient, le bloc est généralement signé numériquement. Cette signature numérique garantit que les blocs de la blockchain ne peuvent pas être modifiés. En effet, pour modifier un bloc, il est nécessaire de modifier tous les blocs qui le précèdent dans la chaîne, ce qui serait extrêmement difficile à réaliser, surtout avec la grande quantité de nœuds qui participent au réseau. C'est cette complexité qui rend la modification des blocs très difficile et protège ainsi l'intégrité de la blockchain. Ainsi, l'immutabilité des données représente l'une des caractéristiques fondamentales de la blockchain. Grâce à cette caractéristique, les données stockées dans la chaîne peuvent être considérées comme indélébiles, ce qui est essentiel pour garantir la sécurité et la confiance dans les transactions effectuées sur la blockchain.

La validation des transactions dans la blockchain repose sur des règles de validation établies dès le début par les développeurs. Ces règles sont également appelées "protocoles de consensus". Comme décrit dans (Bashir, 2017), un protocole de consensus est une série d'étapes pour approuver un état ou une valeur proposée. Plusieurs protocoles consensuels ont été utilisés dans la littérature pour assurer la validité et la sécurité de la blockchain. Parmi ces protocoles, on peut citer la Preuve de Travail (PoW) qui consiste à résoudre des problèmes mathématiques complexes pour valider les transactions, la Preuve de Participation (PoS) qui nécessite la possession de jetons pour valider les transactions, et la Tolérance à la Défaillance Byzantine Pratique (PBFT) qui nécessite un consensus entre les nœuds pour valider les transactions. D'autres protocoles de consensus existent également, chacun avec ses avantages et ses inconvénients en termes de sécurité. La figure 2.2 illustre les étapes d'ajout d'un nouveau bloc à la blockchain.

Ce processus de création d'un nouveau bloc dans la blockchain commence par l'interaction entre deux utilisateurs dans le réseau qui génère une nouvelle transaction. Ensuite, cette transaction est soumise à un protocole de consensus pour être examinée et validée. Après la validation, elle est ajoutée avec d'autres transactions validées pour former un nouveau bloc. Ce dernier est ensuite inclus dans la blockchain existante, ce qui rend l'ensemble de la chaîne de blocs immuable. Ainsi, tout changement dans un bloc nécessite la modification de tous les blocs, ce qui en fait une technologie hautement sécurisée et fiable pour l'enregistrement des transactions.

Il est important de souligner que les protocoles de consensus peuvent différer d'une implémentation à une autre. Toutefois, le principe fondamental de la blockchain demeure inchangé : il s'agit d'une base de données distribuée, décentralisée et sécurisée grâce à l'utilisation de la cryptographie et des protocoles de consensus.

Selon les travaux de (Chen et al., 2019), il apparaît que les protocoles de consensus classiques de la blockchain ne sont pas adaptés aux réseaux sociaux,

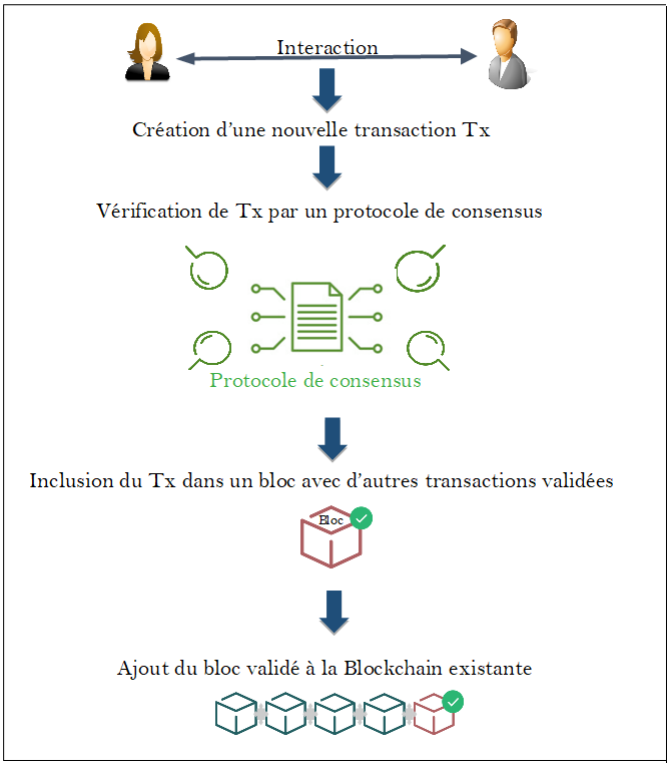

FIGURE 2.2 – *Processus d'ajout d'un bloc à la blockchain*

en particulier l'IoT social. De plus, une lacune majeure de la blockchain est son incapacité à intégrer la sécurité basée sur les attaques de confiance étudiées. Pour illustrer ce point, prenons le cas de l'apprentissage en ligne (E-learning). Lorsqu'un apprenant effectue des transactions avec des clés publiques et privées, le protocole de consensus les valide et les ajoute dans la blockchain en toute confiance. Cependant, en réalité, ces transactions peuvent constituer des attaques malveillantes déguisées. Cette situation met en évidence le besoin urgent d'un nouveau protocole de consensus plus sophistiqué, qui prend en compte les différents scénarios des attaque de confiance possibles.

Pour toutes ces raisons, les chercheurs ont proposé divers protocoles de consensus en fonction de leurs besoins et de leurs objectifs, notamment ceux décrits dans (Chen et al., 2019; Yahia, 2019; Shala et al., 2019; Zou et al., 2018). Cependant, en raison des lacunes existantes dans la sécurité basée sur les attaques de confiance, il est nécessaire de développer un nouveau protocole de consensus pour la technologie blockchain. L'objectif du nouveau protocole proposé est de renforcer la sécurité de la blockchain en détectant et annulant les transactions malveillantes pour garantir la sécurité de la blockchain.

Trouver un protocole de consensus approprié pour prévenir les attaques de confiance est un défi majeur. Ainsi, dans nos travaux précédents (Masmoudi et al., 2020; Abdelghani et al., 2018), nous avons démontré que la technique d'apprentissage, en particulier la classification, est capable d'analyser les comportements des utilisateurs et de les classer en plusieurs catégories avec des résultats

satisfaisants. Étant donné que la classification est largement utilisée pour identifier les attaques en temps réel, nous proposons d'utiliser cette technique dans notre protocole de consensus pour classifier les transactions en temps réel. Cette classification nous permettra de détecter les transactions malveillantes et d'identifier les attaques de confiance.

Nous sommes convaincus que cette approche est particulièrement pertinente dans le contexte des réseaux sociaux et de l'IoT social, où les interactions entre les utilisateurs peuvent être très complexes et où les attaques de confiance sont susceptibles de se produire plus fréquemment. En intégrant cette technique d'apprentissage à notre protocole de consensus, nous espérons apporter une solution innovante à cette problématique et contribuer ainsi à renforcer la sécurité de la blockchain. C'est pourquoi, dans notre travail actuel, nous proposons d'utiliser la technique d'apprentissage dans le cadre d'un nouveau protocole de consensus qui vise à classifier les transactions en temps réel. Grâce à cette classification, nous serons en mesure de classifier les transactions en temps réel comme étant malveillantes ou sécurisées. En détectant les transactions malveillantes, notre protocole de consensus pourra prévenir efficacement les attaques de confiance. Cette approche permettra donc d'ajouter une couche supplémentaire de sécurité à la blockchain, en offrant une surveillance continue et une détection rapide des transactions malveillantes, garantissant ainsi un environnement plus sûr pour les utilisateurs.

Pour cela, dans la prochaine section, nous nous concentrerons sur l'analyse des différentes options de moteurs de traitement de flux de données qui utilisent des techniques d'apprentissage. Cette analyse nous permettra de choisir le moteur de traitement de flux de données le plus approprié pour notre nouveau protocole de consensus. L'objectif est de parvenir à une détection automatique des transactions malveillantes en temps réel, permettant ainsi d'améliorer la rapidité et la précision de la détection des attaques.

2.3 Analyse des moteurs de traitement de flux existants pour l'identification des attaques et des intrusions en temps réel

Le traitement des transactions au niveau de leurs générations est crucial pour lutter contre les attaques à un niveau plus fin et aider à prévenir les attaques à temps. Ce niveau permet d'interrompre les transactions malveillantes en temps réel. Dans cette section, nous allons examiner plusieurs travaux connexes qui ont utilisé des moteurs de traitement de flux pour identifier les attaques et les intrusions en temps réel. Nous allons comparer et analyser ces différents travaux dans le tableau 2.2 et en nous basant sur plusieurs critères tels que la technique utilisée, les attaques traitées et les bibliothèques exploitées.

TABLE 2.2 – *Analyse comparative des techniques basées sur Apache Spark pour la détection en temps réel des attaques et des intrusions*

	Technique	**Attaques ciblées**	**Bibliothèques**	
			Spark Streaming	**Spark MLlib**
(Awan et al., 2021)	Uun système distribué pour prédire les attaques DDoS en temps réel basé sur différents algorithmes d'apprentissage automatique.	DDoS	✓	Random Forest et Perceptron multicouche
(Patil et al., 2020)	Un nouveau système de détection DDoS en temps réel basé sur Apache Spark, appelé S-DDoS.	DDoS	✓	K-Means
(Han et al., 2017)	Un nouveau système de détection d'attaques DDoS basé sur Spark en particulier le modèle K-means.	DDoS	✓	K-Means
(Marir et al., 2018)	Une nouvelle approche distribuée pour détecter les comportements anormaux dans les réseaux à grande échelle.	Un comportement anormal	✓	SVM
(Saravanan et al., 2020)	Un classificateur basé Apache Spark pour la détection des intrusions.	Intrusion	✓	Régression logistique, SVM et arbre de décision
(Zhang et al., 2018)	Un framework basé sur un algorithme Random Forest utilisant Apache Spark afin de détecter les intrusions.	Intrusion	✓	Random Forest
(Abid et Jemili, 2020)	Un système de détection d'intrusion en temps réel .	Intrusion	✓	K2 algorithm
(Zhou et al., 2018)	Un système de surveillance du trafic de l'Internet en ligne pour détecter les attaques DDoS en temps réel.	DDoS	✓	Naïve Bayes, régression logistique et arbre de décision
(Patil et al., 2022b) ; **(Patil et al., 2022a)**	Deux classificateurs basé sur Spark Streaming et Kafka. Ces systèmes visent à classer différents types d'attaques DDoS.	DDoS	✓	Arbre de décision, Bayes naïf, régression logistique et Random Forest

2.3.1 Analyse des moteurs de traitement de flux de données

Plusieurs travaux ont été effectués dans la littérature pour comparer les moteurs de traitement de flux de données open-source tels que Apache Spark, Flink, Storm et Hive (Gorasiya, 2019; Perera et al., 2016). Ces études ont montré qu'il n'y avait pas d'option meilleure que les autres, chacun ayant ses avantages et ses inconvénients. Le choix dépend des besoins et des exigences spécifiques de l'application.

Cependant, ces dernières années, la combinaison du moteur de traitement de flux distribué Apache Spark (Rubin, 2015) avec la technologie blockchain a fait l'objet de recherches croissantes. Cette combinaison permet de renforcer la sécurité et l'intégrité des données en utilisant la blockchain pour effectuer, stocker et partager des transactions de manière sûre, tandis que Spark peut être utilisé pour traiter et analyser ces transactions en temps réel afin de trouver des modèles et des patrons. Cette approche est particulièrement efficace pour les réseaux qui nécessitent à la fois la sécurité et le traitement en temps réel.

De plus, grâce à sa capacité de traitement distribué, Apache Spark est bien adapté à la détection d'attaques en temps réel. Il est capable de traiter de grands volumes de données, permettant ainsi de détecter rapidement les menaces potentielles et d'y répondre. En capitalisant sur les fonctionnalités avancées d'apprentissage et de traitement de flux offertes par Spark, il est possible d'analyser les flux de données entrants, permettant ainsi une détection précoce et une prévention efficace des éventuelles attaques.

À cet égard, plusieurs études récentes se sont intéressées à l'utilisation de Spark pour la détection en temps réel d'attaques et d'intrusions (Awan et al., 2021; Patil et al., 2020, 2022a,b; Saravanan et al., 2020; Marir et al., 2018; Zhang et al., 2018). Ces études ont utilisé deux bibliothèques de Spark : MLlib (Machine Learning library, bibliothèque d'apprentissage automatique), qui permet de former des algorithmes d'apprentissage automatique et de créer des modèles pour prédire les étiquettes de données futures ; et Spark Streaming (bibliothèque de traitement en temps réel), qui permet de traiter les flux de données et de prédire correctement un nouvel élément sur la base du modèle. Ces approches ont donné de bons résultats, confirmant l'efficacité de Spark pour la détection des différents formes d'attaques en temps réel.

De plus, les études examinées dans (Zhang et al., 2018) ont présenté un cadre novateur pour la détection d'intrusions dans les réseaux de données à haut débit en utilisant un algorithme Random Forest (RF) intégré à Apache Spark. Les résultats de leur modèle ont révélé une précision remarquable dans la détection d'intrusions en un temps record. En effet, leur modèle a été en mesure de détecter un nombre considérablement plus élevé d'intrusions en seulement 0,01 seconde, ce qui le place en tête par rapport à d'autres modèles existants en termes de rapidité et d'efficacité. Ces résultats suggèrent que l'utilisation d'Apache Spark est une solution prometteuse pour la détection précoce des intrusions dans les réseaux à haut débit.

(Saravanan et al., 2020) ont conçu un classificateur pour la détection d'intrusion en utilisant Apache Spark, une solution d'analyse de données massives. Comparé aux approches existantes, leur modèle a produit des résultats supérieurs avec un taux de faux positifs satisfaisant. Ils ont évalué leur modèle de classification en utilisant des données de sécurité réseau et ont obtenu des taux de précision pour différents algorithmes tels que la égression logistique, la machine à vecteur de support, la SVM avec descente stochastique de gradient et l'arbre de décision. Les résultats obtenus pour ces différents algorithmes étaient respectivement de 93,9%, 92,8%, 91,1% et 96,8%.

Une autre étude menée par (Marir et al., 2018) a exploré une nouvelle méthode de détection de comportements anormaux dans les réseaux à grande échelle en utilisant une approche distribuée, appelée "méthode profonde distribuée". Les auteurs ont utilisé l'algorithme (Support Vector Machine) avec une variété de techniques pour améliorer les performances de prédiction, notamment en réduisant le nombre de caractéristiques. Pour cela, Apache Spark a été utilisé pour former et tester le modèle, tandis qu'un réseau de croyance profonde a été utilisé pour réduire le nombre de caractéristiques. Les résultats principaux ont montré que le modèle proposé surpassait les approches existantes.

Les auteurs de (Han et al., 2017) ont proposé un nouveau système de détection d'attaques DDoS basé sur le cadre Spark. Ils ont utilisé l'algorithme K-means pour garantir la précision de la détection et ont constaté que l'utilisation de Spark a considérablement réduit le temps nécessaire pour détecter les attaques DDoS, tout en améliorant considérablement l'efficacité de la détection. Les résultats des expériences ont montré que ce système peut non seulement détecter efficacement les attaques DDoS, mais aussi détecter tous les types d'attaques DDoS en temps réel, avec un taux de fausses alertes faible.

Les auteurs de (Patil et al., 2020) ont développé un système de détection en temps réel des attaques DDoS, nommé S-DDoS, basé sur Apache Spark. Ce système utilise l'algorithme de clustering K-means pour détecter le trafic d'attaque DDoS en temps réel. Les résultats obtenus montrent que le système S-DDoS est capable de différencier efficacement les attaques DDoS des flux de trafic réseau avec une précision de détection élevée (98%).

Dans leur étude, (Awan et al., 2021) ont proposé une approche basée sur les big data pour prédire en temps réel les attaques DDoS en utilisant différents algorithmes d'apprentissage automatique. Pour évaluer la performance de l'algorithme, ils ont utilisé Apache Spark, un système distribué, ainsi qu'un modèle de classification. Deux algorithmes d'apprentissage automatique, Random Forest et le perceptron multi-couches, ont été utilisés, ainsi que les bibliothèques Scikit ML et Spark-MLlib du cadre de big data pour la détection des attaques DoS. Les résultats ont montré que leur approche a atteint une précision moyenne de 99,5% avec une détection en temps réel en quelques millisecondes. De plus, grâce à la distribution des calculs en mémoire par Spark, leur approche a été plus rapide que l'approche non big data en termes de temps de formation et de test.

(Abid et Jemili, 2020) ont développé un système de détection d'intrusion en temps réel basé sur l'algorithme de graphe k2 pour détecter différents types d'attaques en temps réel. Le système utilise Spark MLlib et Spark Streaming pour l'apprentissage et la classification. L'algorithme k2 est utilisé pour la classification des attaques et les performances sont améliorées grâce à l'utilisation du moteur de streaming Apache Spark. Les résultats ont montré que le système proposé est efficace pour détecter des attaques en temps réel avec une grande précision.

Selon (Zhou et al., 2018), un système de surveillance du trafic Internet en temps réel a été recommandé pour détecter les attaques DDoS à l'aide de Spark Streaming et de l'apprentissage automatique. Le système est composé de trois parties : le collecteur, le système de messagerie et le processeur de flux. Pour sélectionner les caractéristiques de réseau les plus pertinentes pour l'algorithme de détection DDoS basé sur l'apprentissage automatique, ils ont utilisé une méthode de sélection de caractéristiques basée sur la corrélation. Trois algorithmes d'apprentissage automatique (Naïve Bayes, Régression logistique et Arbre de décision) ont été utilisés pour comparer leur précision de détection. Les résultats ont montré que le système pouvait détecter les trois types d'attaques DDoS les plus courantes avec une précision de plus de 99,3%. De plus, le système a bien fonctionné pour les grands volumes de trafic internet.

Le système de détection d'intrusion proposé par (Wirz et al., 2022) se concentre sur la détection d'attaques courantes telles que le forçage automatisé des connexions Web, les attaques par inondation HTTP, les injections SQL et les scripts intersites (XSS) en utilisant Apache Kafka et Apache Spark. Les auteurs ont également développé une nouvelle méthode de comparaison pour améliorer le taux de faux positifs dans la détection des SQLi, un problème récurrent dans la plupart des IDS existants. Leur système a été capable d'identifier avec précision des modèles malveillants et de générer des alertes par SMS tout en enregistrant les événements dans un Google Cloud Storage Bucket. En somme, la mise en œuvre du système de détection d'intrusion basé sur le cloud a démontré des performances élevées et une efficacité notable dans la détection d'attaques.

Enfin, en utilisant Spark Streaming et Kafka, (Patil et al., 2022b) ont développé un classificateur distribué appelé SSK-DDoS qui vise à classer les flux réseau légitimes et différents types d'attaques DDoS. Pour sa conception, ils ont utilisé une Spark MLlib distribuée basée sur un cluster Hadoop, qui a ensuite été déployé sur la plateforme de streaming Spark. Les auteurs ont comparé la précision de détection de quatre algorithmes d'apprentissage automatique différents : l'arbre de décision, les Bayes naïfs, la régression logistique multinominale et le Random Forest. Les résultats ont montré que la méthode proposée était capable de classer efficacement les flux réseau en sept catégories distinctes.

2.3.2 Synthèse

Les recherches précédentes ont souligné l'importance d'Apache Spark pour détecter en temps réel les attaques et les intrusions. Ainsi, pour améliorer la

prévention en temps réel d'attaques de confiance étudiées, nous proposons de combiner Apache Spark avec la blockchain, en particuler de l'utiliser dans notre nouveau protocole de consensus. A cet égard, nous suggérons l'intégration du module Apache Spark MLlib, une bibliothèque de machine learning scalable, dans le protocole pour améliorer l'identification (Assefi et al., 2017) des différents types des attaques de confiance possibles. Cette bibliothèque propose des algorithmes de classification intégrés, tels que la régression linéaire, la table de décision, la forêt aléatoire, les Bayes naïfs, le SVM, etc., qui accélèrent la phase d'apprentissage et améliorent la création de modèles et de patrons. En outre, nous recommandons l'utilisation de la bibliothèque Spark Streaming pour traiter les transactions de flux en temps réel et prédire l'étiquette de chaque transaction (Awan et al., 2021). Cette étiquette permet de spécifier si une transaction est une attaque ou non, ainsi que le type d'attaque. Elle est également utilisée pour valider les transactions légitimes (sécurisées) et de rejeter les transactions malveillantes en temps réel.

En somme, la combinaison de la technologie blockchain et le moteur de traitement de flux Apache Spark est devenue un défi majeur pour les environnements IoT social. En effet, utiliser Apache Spark dans le protocole de consensus permet de mettre en place un mécanisme de gestion de la confiance plus performant et efficace, en mesure de prévenir les différentes attaques de confiance en temps réel tout en préservant les propriétés de sécurité indispensables.

Il est maintenant primordial de fournir une définition complète d'Apache Spark, en décrivant ses principales caractéristiques, fonctionnalités et avantages. En effet, Apache Spark est un moteur de traitement de données en temps réel open-source qui a été conçu pour offrir une solution de traitement de données rapide et unifiée pour les applications Big Data (Shaikh et al., 2019). Ce moteur a été développé au sein du AMPLab de l'UC Berkeley et prend en charge différentes API telles que Python, Java, Scala et R. Il offre également plusieurs bibliothèques, telles que Spark SQL, qui permet d'interroger des données structurées en utilisant SQL et Hive Query Language (HQL), Spark Streaming, qui facilite le traitement de flux de données en temps réel, MLlib, qui fournit des fonctionnalités d'apprentissage automatique, et GraphX, un moteur de traitement de graphes qui fournit des opérateurs pour la manipulation de graphes et des algorithmes de graphes courants, comme illustré dans la figure 2.3.

Apache Spark fonctionne en lisant les données d'un fichier et en les transformant en un ensemble de données distribuées résilientes (Resilient Distributed Dataset : RDD) (Jonnalagadda et al., 2016). Les RDD sont des éléments essentiels qui sont manipulés dans toutes les applications Spark. Ils permettent d'effectuer des processus de calcul en mémoire et de surmonter les défaillances. Les RDD sont comparables à une table dans une base de données, où les données sont stockées sur différentes partitions et sont immuables. Pour effectuer des modifications, il faut appliquer une transformation qui renverra un nouveau RDD tout en conservant le RDD d'origine inchangé. Les RDD prennent en charge deux types d'opérations : les transformations et les actions. Une transformation ne retourne

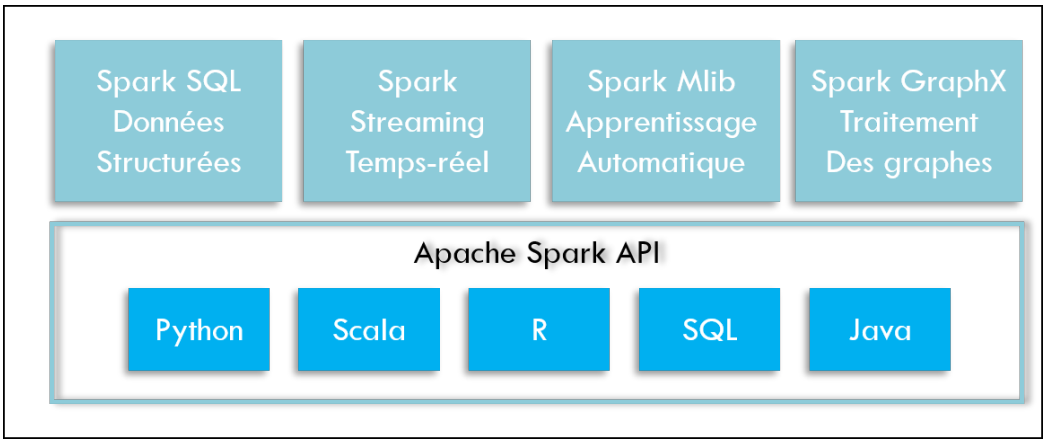

FIGURE 2.3 – *Ensemble d'outils d'Apache Spark*

pas une valeur immédiatement, mais renvoie un nouveau RDD qui est le résultat de la transformation appliquée à l'ensemble de données. Quelques exemples de fonctions de transformation sont Map, Filter, GroupByKey, etc. Lorsqu'une fonction d'action est appelée sur un objet RDD, toutes les opérations de traitement de données sont calculées et le résultat est renvoyé. Quelques exemples de fonctions d'action sont Collect, Count, CountByKey, etc.

La bibliothèque d'apprentissage automatique de Spark (MLlib) est un moyen d'appliquer des algorithmes d'apprentissage automatique sur les données distribuées résilientes (RDD) créées à partir de fichiers lus par Spark (Meng et al., 2016). Avec MLlib, il est possible de prétraiter, de mélanger, de former des modèles et de faire des prédictions sur les données en utilisant une variété d'algorithmes d'apprentissage automatique, tels que la classification, le regroupement, la régression, le filtrage collaboratif, les primitives d'optimisation sous-jacentes. En outre, MLlib s'intègre de manière transparente avec d'autres bibliothèques Spark telles que Spark Streaming, Spark SQL, ce qui permet une analyse de données distribuées complète.

Par contre, Spark Streaming traite des flux de données en micro-batchs (Shaikh et al., 2019). Il est basé sur le concept fondamental de RDD, ce qui permet une intégration transparente avec Spark. De ce fait, les données historiques et en temps réel peuvent être fusionnées en toute fluidité. Il fournit une abstraction de flux continu de données appelée DStream, qui est immuable et peut être utilisée comme un ensemble de données distribuées par Spark. DStream améliore l'efficacité du traitement en continu de données.

Pour mieux comprendre le fonctionnement de Spark Streaming, chaque RDD dans un DStream divise les données d'entrée en lots en fonction d'un intervalle de temps appelé fenêtre. Par exemple, si l'intervalle de temps est de 3 secondes, les données seront rassemblées toutes les 3 secondes et stockées dans un RDD. Cela permet de traiter les données en temps réel, en continu et en les partitionnant en lots.

Spark Streaming présente plusieurs avantages tels que sa facilité d'utilisation, sa haute vitesse de traitement, son équilibrage de charge, sa faible latence, sa grande évolutivité et sa tolérance aux pannes. Tous ces avantages font de Spark Streaming un choix idéal pour l'analyse en temps réel de grandes quantités de données. En somme, Spark Streaming offre une solution efficace pour la gestion de flux de données en temps réel à grande échelle.

En utilisant Spark Streaming avec Spark MLlib, il est possible d'appliquer des algorithmes d'apprentissage automatique aux données en temps réel pour effectuer des prédictions en continu. Les flux de données continus traités par Spark Streaming peuvent être prétraités, mélangés et utilisés pour former des modèles d'apprentissage automatique à l'aide de Spark MLlib. Les modèles peuvent ensuite être utilisés pour prédire les événements futurs à mesure que les données sont diffusées. Cette combinaison permet une analyse de haute qualité et une prise de décision en temps réel.

Dans cette optique, nous visons à combiner Spark MLlib et Spark Streaming dans notre travail afin de surveiller en temps réel les transactions effectuées par les utilisateurs, afin de détecter les transactions malveillantes et les interrompre pour prévenir les attaques de confiance. Pour atteindre cet objectif, nous comptons combiner les bibliothèques Spark MLlib et Spark Streaming. Nous allons exploiter Spark Streaming pour collecter et traiter rapidement les flux de données continus en micro-batchs, et utiliser Spark MLlib pour construire des modèles d'apprentissage afin de détecter les transactions malveillantes en temps réel.

En intégrant Spark Streaming et Spark MLlib, nous sommes en mesure de combiner les données historiques avec les données en temps réel pour une analyse complète et précise des transactions des utilisateurs. Ainsi, nous pouvons identifier les modèles de transaction suspects et les interrompre avant qu'ils ne causent des dommages. L'analyse de données en temps réel nous permet également d'adapter notre modèle de détection d'attaque de confiance en fonction des nouvelles données, afin d'améliorer constamment notre capacité à détecter les transactions frauduleuses (Masmoudi et al., 2023).

2.4 Conclusion

Au cours de ce chapitre, nous avons mené une exploration approfondie des différentes techniques et technologies qui sont utilisées dans la littérature pour prévenir les différents formes d'attaques dans les domaines connexes à notre domaine d'étude, tels que l'IoT et le réseau social. Nous avons commencé par présenter les différentes approches et techniques qui ont été développées pour lutter contre les attaques dans ces domaines, afin de dégager des idées clés et de pouvoir les appliquer à notre propre contexte. Parmi les technologies identifiées, la blockchain a émergé comme la plus pertinente pour notre étude, en raison de sa capacité à garantir la transparence, la sécurité et la confiance dans les systèmes d'information tout en réduisant les risques d'attaques de confiance.

Nous avons également consacré une partie de ce chapitre à la présentation détaillée de la technologie blockchain, en insistant sur les différents protocoles de consensus utilisés. Cette exploration nous a permis de comprendre les enjeux liés à l'adaptation de cette technologie à notre domaine d'étude, en particulier pour la prévention des attaques de confiance dans l'IoT social. Nous avons ainsi pris conscience de la nécessité de proposer un nouveau protocole de consensus qui soit spécifiquement adapté à notre contexte.

En outre, nous avons examiné de près les différents moteurs de traitement de flux qui ont été proposés dans la littérature pour l'identification des attaques et des intrusions en temps réel. Notre analyse approfondie nous a permis de conclure qu'Apache Spark était la technologie la plus adaptée pour notre contexte. En effet, Spark nous permet de surveiller les transactions en temps réel, d'analyser rapidement les flux de données continus, et de construire des modèles d'apprentissage automatique pour détecter les transactions malveillantes et les annuler. En l'intégrant à notre protocole de consensus de la blockchain, nous pourrons ainsi garantir la sécurité des transactions effectuées par les utilisateurs de notre système IoT social.

Enfin, nous avons conclu ce chapitre en présentant le framework Apache Spark, qui est une plate-forme de traitement distribué capable de traiter des données massives en temps réel. Nous avons expliqué les principes fondamentaux de Spark et ses avantages pour la gestion des données en temps réel, en particulier pour la détection des comportements malveillants en combinant ses deux bibliothèques Spark MLlib et Spark Streaming.

En nous appuyant sur les connaissances acquises dans les chapitres précédents, nous allons dans le chapitre suivant exposer notre contribution, qui consiste en un nouveau mécanisme de gestion de confiance pour prévenir en temps réel tous les types d'attaques de confiance, tout en tenant compte des propriétés de sécurité requises. Cette approche novatrice repose sur la combinaison de deux technologies : la blockchain et Apache Spark.

3 Solution innovante de prévention des attaques de confiance en temps réel dans l'IoT social basée sur la blockchain et Apache Spark

Sommaire

3.1 Introduction

Dans les environnements dynamiques et complexes comme celui de l'IoT social, il est essentiel de pouvoir déterminer à qui accorder sa confiance et à qui ne pas l'accorder (Ramanathan, 2015). Donc, la gestion de la confiance revêt une importance cruciale. En effet, la présence d'un utilisateur malveillant peut entraîner la transmission d'informations incorrectes ou malveillantes, causant ainsi une altération inappropriée du système (Masmoudi et al., 2020). Il est donc impératif de mettre en place un mécanisme de gestion de la confiance efficace dans l'IoT social qui vise à faire face aux attaques de confiance.

C'est pourquoi, dans le premier chapitre, nous avons expliqué les quatre phases principales du mécanisme de gestion de la confiance, tel que décrit dans le premier chapitre. La première phase est la phase de composition, qui consiste à sélectionner les facteurs permettant de décrire le comportement d'un utilisateur dans le réseau. La deuxième phase est la phase d'agrégation, qui permet de combiner les valeurs de ces différents facteurs pour obtenir des scores permettant de classer les transactions effectuées par les utilisateurs. Les deux dernières phases, à savoir la phase de propagation et de mise à jour de la confiance, permettent de propager et de mettre à jour ces scores de confiance dans le réseau IoT social. En outre, nous avons également présenté une étude comparative des différents mécanismes de gestion de la confiance dans les environnements IoT social. Ces mécanismes sont basés sur deux approches différentes : la première permet de détecter les attaques de la confiance, tandis que la deuxième vise à les prévenir.

Dans le deuxième chapitre, nous avons effectué un survol des différentes techniques et technologies utilisées dans la littérature pour la prévention des attaques dans les domaines similaires à l'IoT social. Parmi les technologies identifiées, la blockchain a émergé comme la plus pertinente pour notre étude. Nous avons également examiné de près les différents moteurs de traitement de flux utilisés dans la littérature pour l'identification des attaques et des intrusions en temps réel. Notre analyse approfondie nous a permis de conclure qu'Apache Spark était la technologie la plus adaptée pour notre contexte. Nous prévoyons donc de l'intégrer à notre protocole de consensus de la blockchain afin de garantir la sécurité des transactions effectuées par les utilisateurs de notre réseau IoT social. En combinant la technologie de la blockchain et le framework Apache Spark, nous pouvons détecter les comportements malveillants en temps réel

En capitalisant sur ces connaissances, nous présentons dans ce chapitre notre contribution qui consiste en une nouvelle architecture en couches permettant de combiner la technologie de la blockchain, Apache Spark avec l'IoT social. Nous proposons ensuite un nouveau mécanisme de gestion de la confiance, destiné à prévenir efficacement en temps réel tous les types d'attaques de confiance, tout en prenant en compte les propriétés de sécurité requises. Pour ce faire, nous avons opté pour la combinaison de la technologie de la blockchain avec le framework Apache Spark. Cette approche nous permet de surveiller instantanément les transactions en cours et de détecter les transactions malveillantes.

Le reste de ce chapitre est structuré comme suit : nous commençons par présenter un scénario de motivation dans la section 3.2 pour illustrer les problèmes que notre solution vise à résoudre. Ensuite, dans la section 3.3, nous exposons la modélisation du système que nous avons utilisée pour notre étude. Dans la section 3.4, nous présentons notre architecture en couches, qui intègre les différentes technologies utilisées, à savoir la blockchain et le framework Apache Spark avec notre domaine de recherche IoT social. Dans la section 3.5, nous exposons en détail notre nouveau mécanisme de gestion de la confiance, qui permet de prévenir en temps réel tous les types d'attaques de confiance. Nous expliquons la première phase de composition de notre approche et les différents facteurs proposées. Les autres phases seront détaillées dans le chapitre suivant. Enfin, nous récapitulons par une conclusion.

3.2 Scénario de motivation

Avant d'entrer dans les détails de notre méthode, nous souhaitons tout d'abord présenter un scénario de motivation afin de situer notre système dans son contexte. Nous avons choisi de proposer un scénario dans le domaine de l'apprentissage en ligne (E-learning), qui est particulièrement pertinent dans le contexte actuel de la pandémie mondiale de COVID-19, car il s'agit d'un domaine en plein essor. Nous avons ainsi opté pour ce scénario, car il permettra d'expliquer de manière concrète les conséquences des attaques de confiance dans un environnement tel que d'apprentissage en ligne. Par conséquent, grâce à ce scénario, nous souhaitons souligner l'importance cruciale d'un mécanisme de gestion de la confiance fiable et efficace pour garantir la sécurité et l'exactitude des données et des transactions dans cet environnement en pleine croissance.

Avec la propagation de la pandémie de COVID-19 à travers le monde, le confinement est devenu la solution la plus efficace pour contrôler et endiguer la maladie. Cependant, cette mesure a eu des impacts négatifs sur l'éducation, car les établissements scolaires ont dû fermer leurs portes, laissant les étudiants sans moyen d'assister aux cours. Pour éviter une interruption de l'année scolaire, le ministère de l'Éducation a encouragé les enseignants et les étudiants à se tourner vers l'apprentissage en ligne, qui s'est avéré être la meilleure alternative dans ces circonstances.

C'est dans ce contexte que notre scénario de motivation prend place. Une étudiante nommée "Bella" a rencontré des difficultés pour réviser son cours pendant la période de révision. Elle s'est donc tournée vers l'apprentissage en ligne pour comprendre ses leçons et réussir ses examens. Cependant, n'ayant pas de cartes internationales telles que PayPal, MasterCard ou Visa pour payer les frais, elle a préféré chercher des cours gratuits pour économiser de l'argent.

Après avoir exprimé son besoin de cours en ligne, l'étudiante "Bella" va effectuer une requête personnalisée, notée Req, en utilisant son appareil électronique. Cet appareil peut être un smartphone, une tablette, un tableau interactif ou encore un outil de vidéoconférence, selon ses préférences. La requête est composée

d'informations telles que le nom du cours ou des mots-clés associés à ses intérêts d'apprentissage. En réponse à cette requête, Bella recevra une liste de services sociaux pertinents, basée sur sa requête et ses relations sociales, comme illustré dans la figure 3.1. Cette liste peut inclure une variété de plateformes d'apprentissage en ligne telles que edx, Udemy, Mooc, des forums d'étudiants, des sites web éducatifs tels qu'une chaîne YouTube qui propose des cours et des formations en ligne, etc. En effet, parmi ces services, certains peuvent être malveillants et les utilisateurs malintentionnés cherchent à propager des comportements nuisibles via ces services. Ainsi, l'étudiante peut se retrouver dans une situation difficile où elle doit choisir parmi plusieurs options sans être sure de la qualité et de la sécurité de chacune d'entre elles. En choisissant un service inapproprié, l'étudiante, risque de compromettre la sécurité et l'exactitude des données.

Les utilisateurs malveillants présents dans la liste recommandée à Bella peuvent diffuser de fausses informations et des idées trompeuses qui peuvent induire en erreur les apprenants, sans se soucier de leurs besoins. Leur motivation première est de gagner de l'argent, sans considération pour les apprenants. De plus, ils peuvent utiliser des virus pour endommager les appareils ou compromettre les informations personnelles des utilisateurs. Cette présence d'utilisateurs malveillants dans la liste recommandée peut être attribuée à leur haute valeur de confiance dans le réseau. Cependant, cette valeur de confiance peut être le résultat de votes malhonnêtes, également connus sous le nom d'attaques de confiance. Certains utilisateurs, tels que les tuteurs, peuvent manipuler la valeur de confiance des autres utilisateurs en utilisant des attaques telles que SPA ou BMA, comme illustré dans la figure 3.1. En conséquence, les services de ces utilisateurs malveillants pourraient être choisis par d'autres utilisateurs.

Pour éviter les risques liés à la recommandation de services sociaux malveillants, il est essentiel d'instaurer un mécanisme de gestion de la confiance efficace pour éviter les risques liés à la recommandation de services sociaux malveillants. Ce modèle de confiance vise à prévenir les attaques potentielles afin de garantir la sécurité et la fiabilité des services sociaux proposés aux utilisateurs. Il est particulièrement important de protéger les utilisateurs tels que Bella en leur évitant les risques associés à l'utilisation de services sociaux malveillants. En adoptant un tel modèle, les utilisateurs peuvent être assurés de faire le bon choix lorsqu'ils choisissent un service en ligne, évitant ainsi les conséquences négatives qui peuvent résulter de l'utilisation de services sociaux malveillants.

3.3 Modélisation du système IoT social

Cette section se concentre sur la modélisation des systèmes IoT social. Ces systèmes sont constitués de deux types d'entités : les utilisateurs et les dispositifs IoT. Ces entités possèdent des caractéristiques distinctes, sont interconnectées par divers types de relations et peuvent interagir de différentes manières.

Nous étudions dans ce travail un système IoT social qui comprend des utilisateurs $\sum_{i=1}^{n} u = \{u_1, u_2, u_3, ..., u_n\}$ et des dispositifs $\sum_{j=1}^{m} d = \{d_1, d_2, d_3, ...,$

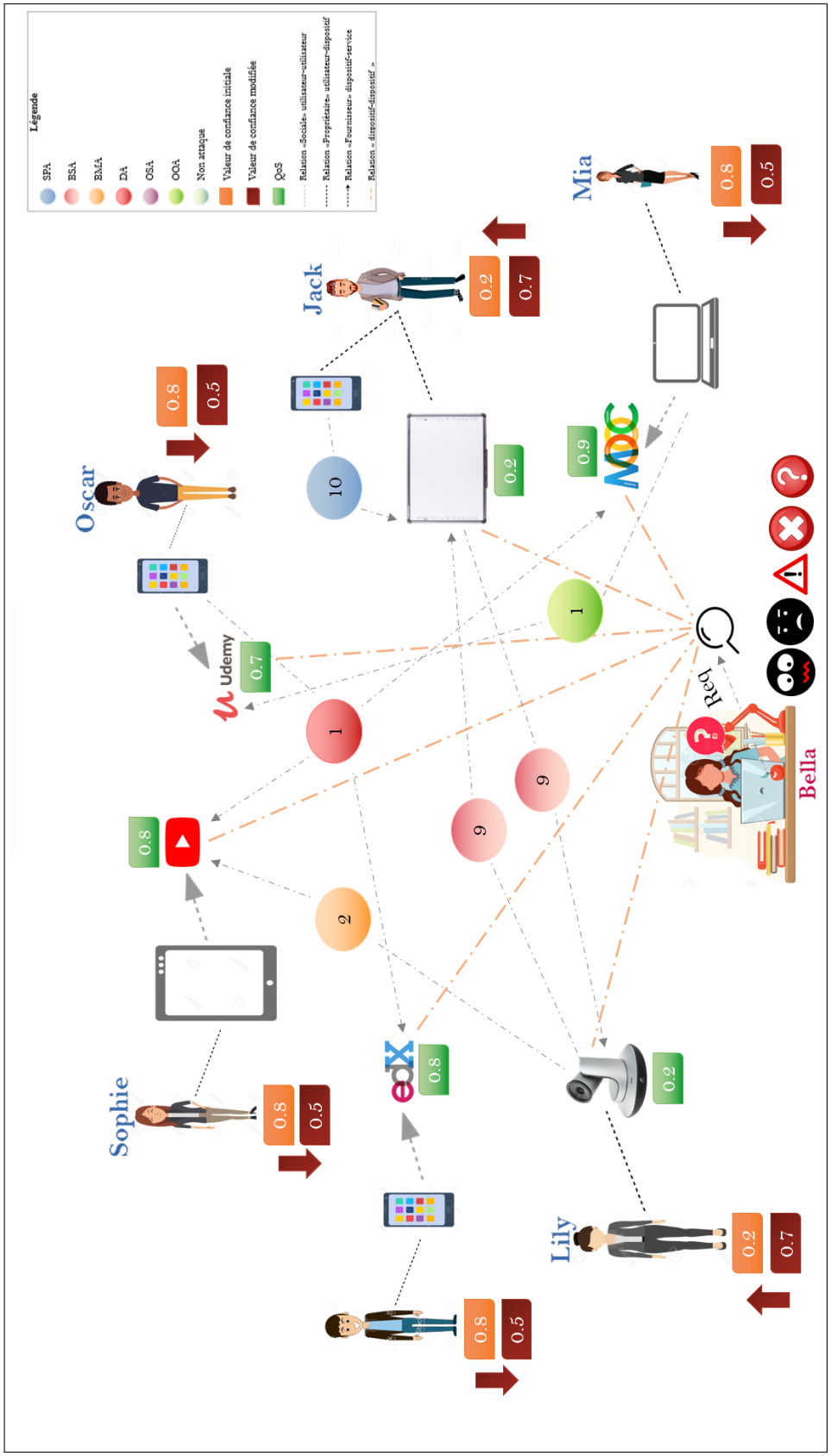

FIGURE 3.1 – *Scénario de motivation*

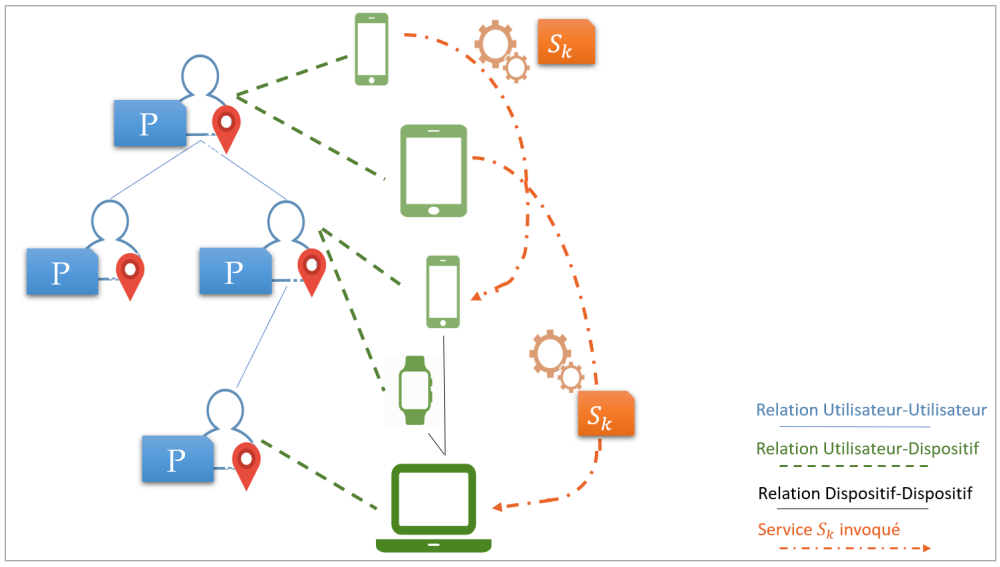

FIGURE 3.2 – *Modélisation du système IoT social*

d_m } comme montre la figure 3.2. Dans ce système, chaque utilisateur peut posséder un ou plusieurs dispositifs, et chaque dispositif appartient à un utilisateur spécifique. Chaque dispositif dans ce système peut générer et demander un ou plusieurs services à différentes entités, représentés par $\sum_{k=1}^{p} s = \{s_1, s_2, s_3, ..., s_p\}$. Chaque utilisateur $u_i \in u$ et dispositif IoT $d_j \in d$ peut intéragir dans ce système et invoque un service $s_k \in s$ et lui donne un vote.

3.3.1 Modélisation d'un utilisateur

La modélisation d'un utilisateur $u_i \in u$ repose sur plusieurs critères, tels que son identifiant unique, son profil, ses intérêts, ses préférences et sa position, qui sont utilisés pour personnaliser ses interactions au sein du système. En outre, chaque utilisateur peut établir des liens sociaux avec d'autres utilisateurs, ce qui leur permet de partager des informations et de collaborer sur différents services. En outre, un utilisateur peut posséder un ou plusieurs dispositifs IoT, et peut utiliser ces derniers pour offrir des services ou pour invoquer des services proposés par d'autres utilisateurs du réseau IoT social. Enfin, l'utilisateur peut également donner des votes à l'issue de chaque interaction qu'il a avec un autre utilisateur.

3.3.2 Modélisation d'un dispositif IoT

Un dispositif $d_j \in d$ est un objet connecté qui peut être possédé par un seul utilisateurs, noté $u_i \in u$. Ces dispositifs interagissent avec d'autres dispositifs en utilisant soit les protocoles du réseau social de superposition, soit les protocoles du réseau de communication standard sous-jacents (câblés ou sans fil). Chaque dispositif d_j est identifié par un identifiant unique tel qu'un code barre, une adresse IP ou une balise RFID. En outre, il possède des caractéristiques matérielles telles que sa capacité de stockage (ROM, RAM, Flush), sa stratégie de communication (Normalement-off, Toujours-sur, Basse-puissance, ...), ses limites

d'utilisation de l'énergie (par exemple, l'énergie disponible sur une période donnée ou la durée de vie de la batterie), et ses exigences de sécurité (confidentialité, intégrité, disponibilité, ...).

Les dispositifs IoT dj peuvent inclure des capteurs et des actionneurs qui leur permettent d'accomplir leurs fonctions. En fournissant des services à d'autres périphériques et en établissant différents types de relations avec eux, les dispositifs ont un impact sur l'environnement dans lequel ils opèrent. En outre, chaque dispositif possède un état (state) St_j (t) qui décrit des informations clés telles que son emplacement actuel, sa connectivité et son niveau d'énergie au temps t.

3.3.3 Modélisation d'un service

Un service $s_k \in s$ est caractérisé par son identifiant unique, sa description fonctionnelle et sa description technique. Il est fourni par un ou plusieurs dispositifs et est invoqué par d'autres dispositifs. Il est également classé selon sa qualité. De plus, il peut produire des événements qui sont transmis aux utilisateurs concernés. Ces derniers peuvent également évaluer le service en lui attribuant un vote qui reflète leur niveau de satisfaction ou d'insatisfaction.

3.3.4 Modélisation des interactions

Lors d'une interaction, deux entités sont impliquées : le dispositif fournisseur $d_i \in d$ qui fournit un service $s_k \in s$ au dispositif invocateur $d_j \in d$. Le dispositif invocateur d_j quant à lui, invoque le service s_k et donne un vote (rate) rt_{jk}. Cette interaction a lieu à un moment précis t et est représentée par la notation $I_{n,s_k,rt_{jk}}$ (d_i, d_j) où $n \in I$ (d_i, d_j) (ensemble des interactions entre d_i et d_j) et $s_k \in s_p$ (d_j) (ensemble des services fournis par d_j).

3.3.5 Modélisation des relations

Dans un environnement IoT social, les relations R entre les différents utilisateurs et dispositifs peuvent être de différents types en fonction des entités impliquées, telles que les relations utilisateur-utilisateur, utilisateur-dispositif et dispositif-dispositif :

— Relation utilisateur-utilisateur Ruu : est définie comme une relation de suivi mutuel entre deux utilisateurs u_i et u_j, ce qui signifie que chaque utilisateur suit l'autre. Elle est représentée comme u_i suit u_i et u_j suit u_i. Cette relation est notée $Ruu(u_i,u_j) = Ruu(u_j,u_i)$.

— Relation utilisateur-dispositif Rud : est définie comme une relation entre un utilisateur et un dispositif qu'il possède. Si un utilisateur u_i possède un dispositif d_j, alors la relation entre eux est représentée comme u_i possède d_j. Cette relation est notée $Rud(u_i,d_j)$.

— Relation dispositif-dispositif Rdd : Cette relation symétrique est définie comme une relation entre deux dispositifs d_i et d_j dans un environnement IoT social. Cette relation est représentée comme d_i est connecté à d_j et d_j est connecté à d_i. Elle est notée $Rdd(d_i,d_j) = Rdd(d_j,d_i)$.

Après avoir présenté la modélisation d'un réseau IoT Social, notre recherche poursuit son évolution vers l'étape suivante : la proposition d'une architecture innovante pour répondre aux défis actuels des systèmes IoT social dans la section suivante. Cette architecture novatrice repose sur la combinaison du réseau IoT social avec des technologies avancées telles que la Blockchain et Apache Spark. La combinaison de ces technologies avancées permet de relever les enjeux majeurs tels que la sécurité, la confiance et l'analyse en temps réel, qui sont des enjeux majeurs dans le domaine de l'IoT social.

3.4 Architecture en couches basée Blockchain et Apache Spark pour les systèmes IoT social

Cette section vise à introduire une nouvelle architecture en couches pour l'intégration de la blockchain et Apache Spark avec l'IoT social. Cette architecture est composée de cinq couches, comprenant les couches de l'architecture de l'IoT social présentées dans le chapitre 1, auxquelles nous avons ajouté une nouvelle couche. Cette nouvelle couche, appelée "couche de IoT social-Blockchain-Spark. Elle est positionnée comme une couche distincte placée entre la couche de gestion de l'IoT social et la couche d'application. Cette disposition est illustrée dans la figure 3.3, qui présente les différentes couches et leur interconnexion. L'ajout de cette nouvelle couche permet de surmonter les problèmes de sécurité et de confiance du réseau IoT social. Ainsi, elle assure la gestion des transactions en utilisant la technologie de la blockchain, ainsi que l'analyse des données en temps réel en utilisant Apache Spark.

3.4.1 Modèle en couches de base de l'IoT social

Le modèle en couches de l'IoT social se compose de quatre couches principales comme illustré dans la figure 1.2. La première est la couche de perception, qui regroupe les capteurs et les actionneurs permettant de collecter des données pertinentes pour mieux comprendre l'environnement. La deuxième couche est la couche de communication, qui gère la connectivité et le routage des données entre les différents objets de l'IoT social. Cette couche inclut également des dispositifs de sécurité pour garantir la confidentialité et l'intégrité des données. La troisième couche est la couche de gestion de l'IoT social qui rassemble les éléments clés pour la gestion des services de l'IoT Social. Cette couche comprend la gestion des identités (ID), le profilage, le Contrôle par le Propriétaire (CP), la Gestion Relationnelle (GR), la Découverte de Services (DS), la Composition de Services (CS) et la Gestion de la Confiance (GC), comme présenté dans le tableau 1.1 du chapitre 1.

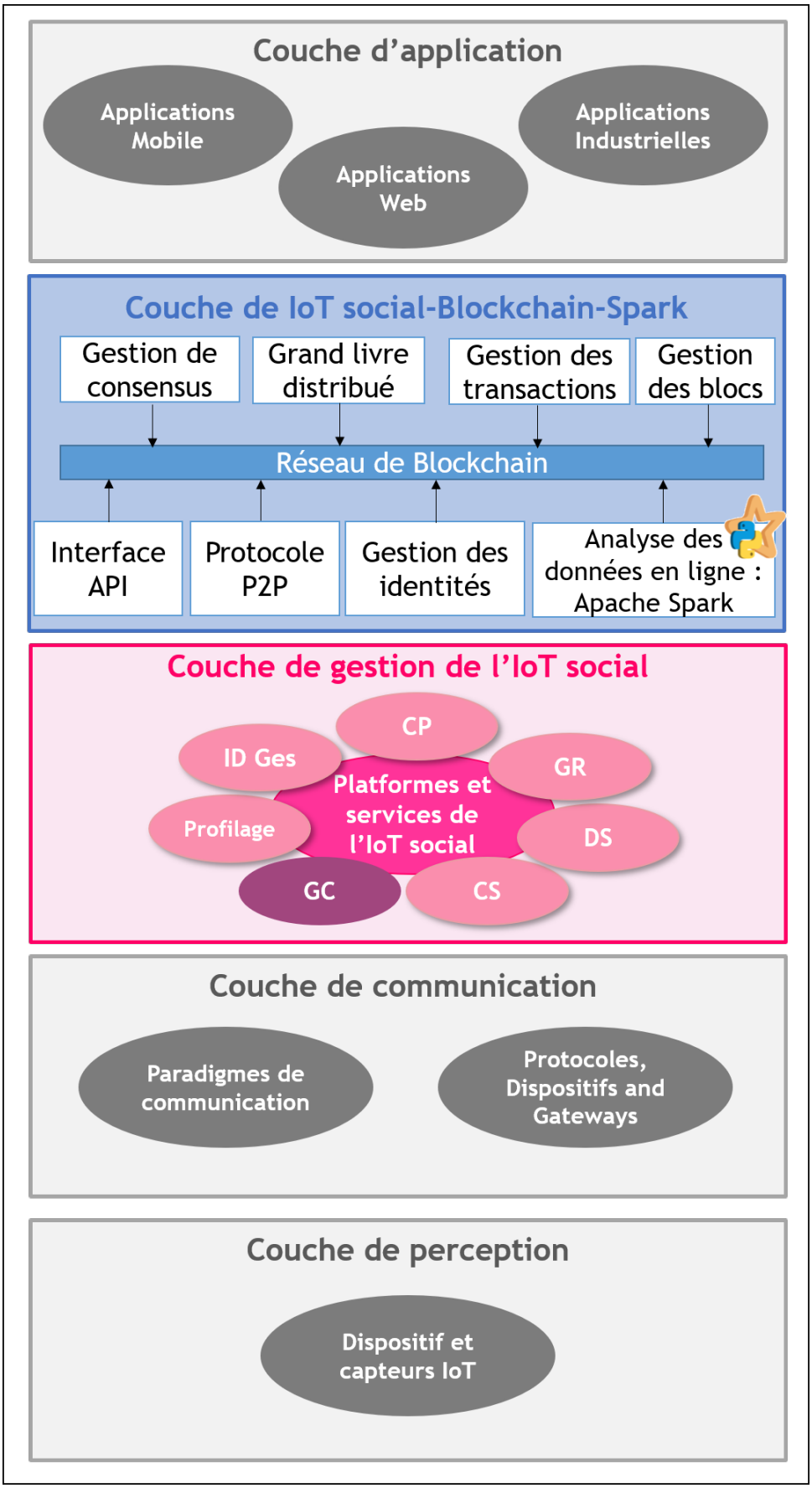

FIGURE 3.3 – *Une nouvelle architecture en couches pour l'intégration de la blockchain et Apache Spark avec l'IoT social*

Et finalement, la couche d'application, qui constitue la couche supérieure de cette architecture, englobe une variété d'applications IoT social et fournit des outils de visualisation de données pour créer des services numérisés. Ces outils permettent aux décideurs d'accéder à des données précises et fiables collectées à partir des dispositifs connectés, ce qui facilite la prise de décisions éclairées.

3.4.2 Nouvelle couche de IoT social-Blockchain-Spark

La nouvelle couche ajoutée, nommée **couche de IoT social-Blockchain-Spark**, regroupe plusieurs modules nécessaires pour mettre en place les fonctionnalités des technologie blockchain et spark dans l'environnement Social IoT. Cette couche intègre diverses fonctionnalités, telles que le protocole de communication P2P, l'interface de programmation d'applications (API), le grand livre distribué, la gestion des identités, l'analyse des données en temps réel basé Apache Spark, la gestion des consensus, la gestion des transactions et la gestion des blocs. L'objectif de cette couche est d'améliorer la sécurité et la confiance des données échangées au sein du réseau IoT social grâce aux avantages offerts par les technologies blockchain et Spark.

Les protocoles P2P jouent un rôle clé dans la mise en place d'une communication décentralisée entre les différents noeuds de l'environnement IoT social. Par ailleurs, **l'interface de programmation d'applications API** offre un accès aux services de la blockchain pour les applications de l'IoT social tels que la gestion des identités, la gestion des transactions et la gestion des consensus. Elle permet ainsi une intégration plus étroite et une utilisation plus facile des fonctionnalités de la blockchain dans les applications de l'environnement IoT social.

Dans le contexte du Social IoT, **la fonction de grand livre distribué** est cruciale pour garantir l'intégrité des données et maintenir une vision cohérente des transactions effectuées sur le réseau. Chaque objet connecté garde une copie du grand livre distribué afin de mettre à jour son état en fonction des changements survenus dans le réseau. Le choix du type de grand livre dépend fortement du contexte de l'IoT social, y compris le nombre d'utilisateurs dans le réseau et les exigences de sécurité et de confidentialité. Les types courants de grands livres distribués comprennent les grands livres permissionnés et non permissionnés. Les grands livres permissionnés sont contrôlés par un groupe d'utilisateurs autorisés, tandis que les grands livres non permissionnés sont ouverts à tous les utilisateurs du réseau.

Afin d'assurer un traitement rapidement les transactions générées sur le réseau Social IoT, nous avons intégré **un module d'analyse de données en ligne** dans la couche de Social IoT-Blockchain. Les méthodes de traitement de données traditionnelles ne sont pas suffisantes à un environnement aussi dynamique et nécessitent un traitement en temps réel des transactions pour garantir une expérience utilisateur fiable. Pour cette raison, nous avons opté pour l'utilisation du framework Apache Spark pour analyser et traiter en temps réel les transactions

générées par le réseau. Cette approche garantit une gestion efficace des transactions sur la blockchain, tout en offrant une expérience utilisateur fiable au sein de l'IoT social.

La gestion du consensus est une fonctionnalité essentielle de l'intégration de la blockchain dans l'environnement Social IoT. Elle agit comme un serveur central qui maintient la confiance entre les différents utilisateurs communicants du réseau en garantissant que toutes les transactions sont valides et cohérentes. De plus, **la gestion des identités** est une autre fonctionnalité importante qui est utilisée pour contrôler et identifier les différents utilisateurs du réseau IoT social. Elle permet de garantir que les utilisateurs communiquant sur le réseau sont authentiques et autorisés, ce qui contribue à assurer la sécurité et la confidentialité des données transmises.

La gestion des blocs est une autre fonctionnalité clé qui est également importante dans l'environnement de l'IoT social. Cette fonctionnalité permet de stocker de manière sécurisée les transactions effectuées sur le réseau et de les organiser en blocs qui sont ensuite ajoutés au grand livre distribué. Cela garantit l'intégrité et la sécurité des transactions enregistrées sur la blockchain. De plus, la gestion des blocs permet de suivre les modifications apportées aux transactions et d'assurer l'exactitude des données stockées sur la blockchain. Elle joue également un rôle crucial dans la scalabilité de la blockchain en gérant efficacement les blocs de transactions à mesure que le nombre de transactions sur le réseau IoT social augmente, garantissant ainsi un traitement rapide et efficace des transactions.

La gestion des transactions dans la couche de IoT social-Blockchain-Spark implique la validation et la vérification des transactions sur le réseau IoT social. Cette fonctionnalité permet de garantir que toutes les transactions sont correctes et sécurisées avant d'être enregistrées sur la blockchain. Elle implique également la gestion de la cohérence de la base de données distribuée, en veillant à ce que toutes les transactions soient synchronisées et valides sur tous les utilisateurs du réseau.

3.5 SparkChain : Notre mécanisme de gestion de la confiance

La littérature présente une variété de mécanismes de gestion de confiance pour assurer la fiabilité des services et des interactions. Cependant, ces mécanismes ne peuvent pas prévenir toutes les attaques en temps réel, malgré leur sophistication. En effet, le système est constamment vulnérable aux attaques malveillantes, ce qui compromet sa fiabilité. Dans ce contexte, il est essentiel que les mécanismes de gestion de confiance remplissent leur rôle de garants de la fiabilité du système en assurant la prévention des différents types d'attaques de confiance en temps réel.

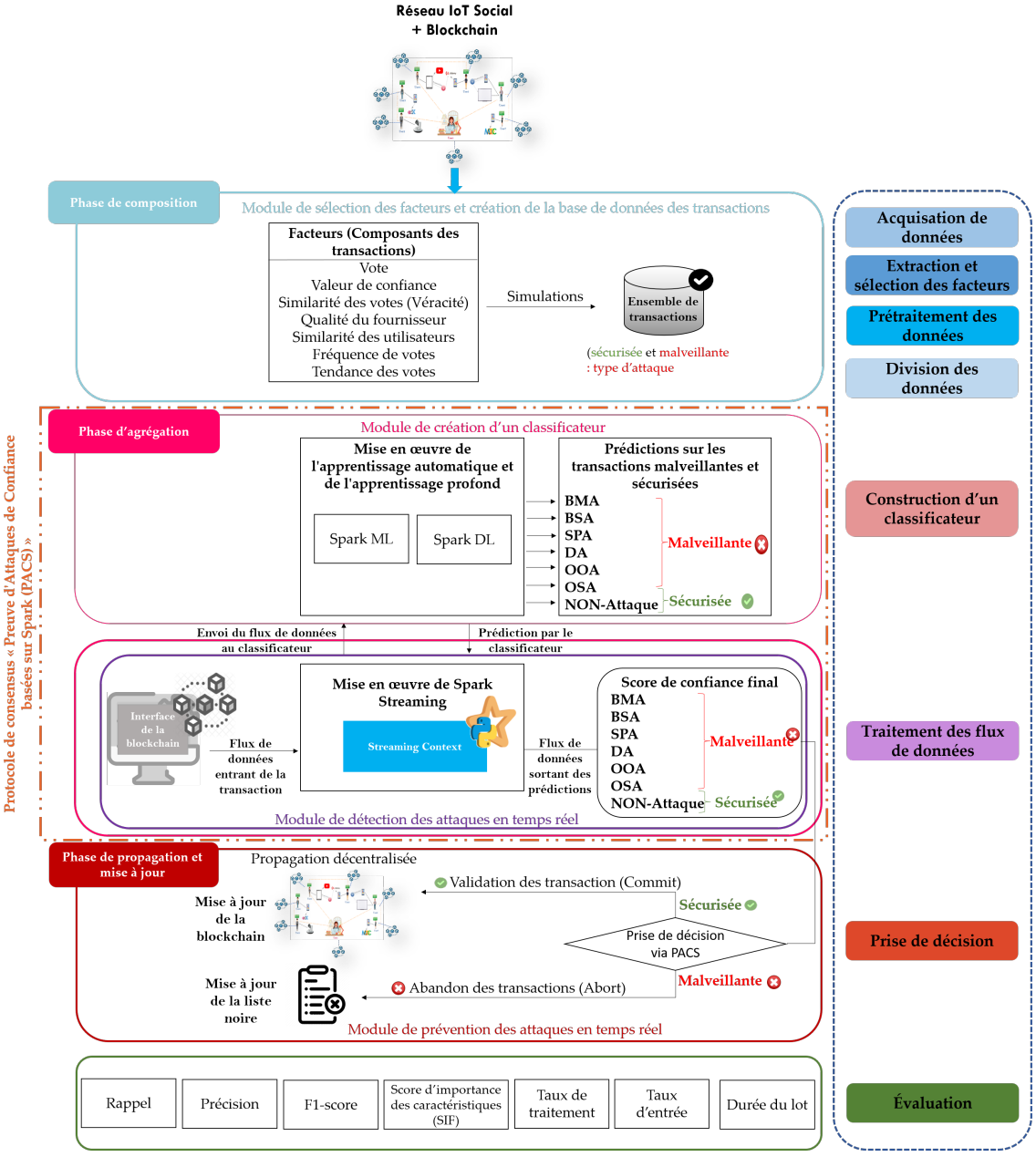

FIGURE 3.4 – *SparkChain : Notre mécanisme de gestion de la confiance innovant de prévention des attaques de confiance en temps réel dans l'IoT social, basé sur Spark et la blockchain*

Ainsi, nous proposons un mécanisme de gestion de confiance qui vise à analyser en temps réel les transactions effectuées sur le réseau, détecter les transactions malveillantes et déterminer le type d'attaque effectué par l'utilisateur malveillant. En outre, ce mécanisme permet d'interrompre et d'annuler les transactions malveillantes, empêchant ainsi leur propagation sur le réseau.

3.5.1 Architecture de SparkChain

Notre solution innovante qui repose sur la proposition d'un nouveau mécanisme de gestion de la confiance consiste à intégrer la technologie de la blockchain et ses étapes de validation de transactions avec un nouveau protocole de consensus appelé Preuve d'Attaques de Confiance basées sur Spark (PACS) qui s'appuie sur Apache Spark. Cette adaptation permettra de renforcer la sécurité et la fiabilité du système en détectant et en prévenant les attaques de confiance en temps réel. Plus précisément, le protocole de consensus PACS est conçu pour analyser les transactions et détecter les comportements malveillants afin de prendre des mesures appropriées pour garantir l'intégrité de la blockchain. Les transactions bienveillantes sont validées et ajoutées à la blockchain, tandis que les transactions malveillantes sont annulées et ajoutées à une liste noire qui joue un rôle essentiel dans la prévention des attaques futures, car elle permet d'identifier et de signaler rapidement les transactions suspectes. En outre, PACS garantit la transparence et la cohérence de toutes les transactions enregistrées sur la blockchain.

La Figure 3.4 expose notre mécanisme de gestion de la confiance, appelée "SparkChain", qui intègre Spark et la blockchain. Il se compose de quatre modules principaux, chacun dédié à une étape spécifique du processus de gestion de la confiance : la phase de composition, qui consiste à identifier les différents facteurs proposés, la phase d'agrégation, qui permet de rassembler les différents facteurs, la phase de propagation, qui vise à diffuser la confiance dans le réseau, et la phase de mise à jour, qui assure la mise à jour continue de la confiance.

La phase initiale de notre mécanisme de gestion de la confiance est la phase de composition, qui inclut le module de sélection des facteurs et de création de la base de données de transactions. La phase suivante est la phase d'agrégation de la confiance, qui utilise les différents facteurs de confiance sélectionnés et se compose de deux modules principaux : (1) le module de création d'un classificateur et (2) le module de détection des attaques en temps réel. Enfin, les phases de propagation et de mise à jour intègrent le module de prévention des attaques en temps réel pour garantir la sécurité du réseau.

Dans la suite, nous allons détailler les différents phases et modules de notre approche visant à prévenir les attaques de confiance. Cette approche repose sur l'utilisation conjointe des technologies blockchain et Apache Spark.

3.5.2 Phase de composition

La première phase de notre mécanisme de gestion de la confiance est la phase de composition détaillée dans la figure 3.5, qui comprend le module de sélection

des facteurs et de création de la base de données de transactions. Ce module a pour objectif de choisir et sélectionner les différents composants des transactions (appelés facteurs) qui permettent d'analyser les transactions effectuées par les utilisateurs dans le réseau IoT social. Ces composants permettent de prédire si une transaction est malveillante, en identifiant le type d'attaque effectué par l'utilisateur malveillant, ou si elle est sécurisée. Cette phase consiste également à collecter et stocker toutes les transactions effectuées sur le réseau dans une base de données. Cette base ainsi créée est ensuite divisée en sous-ensembles pour faciliter les étapes suivantes de l'analyse des transactions et la prévention des attaques de confiance.

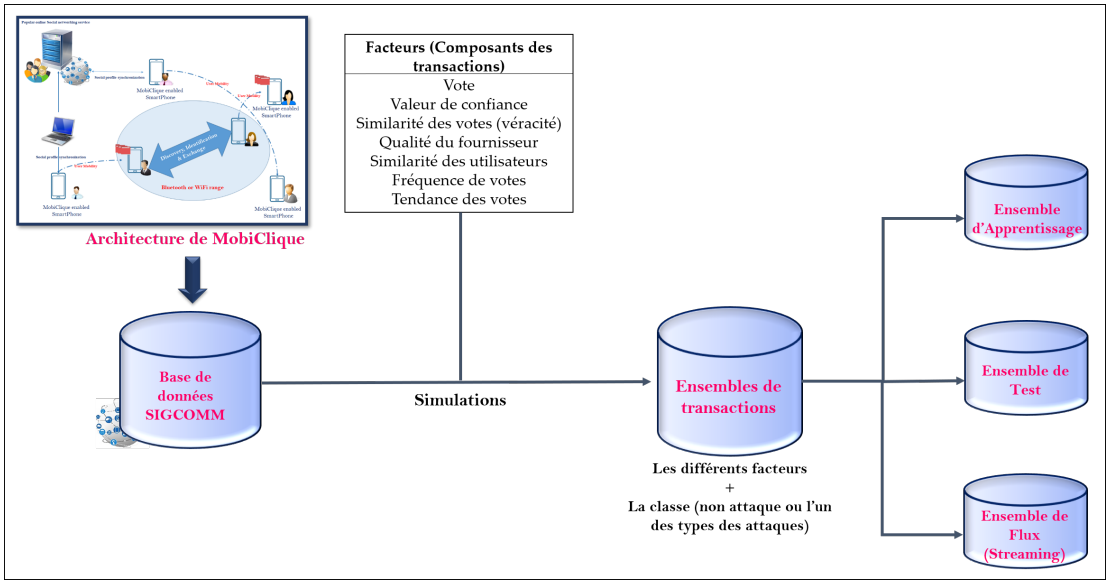

FIGURE 3.5 – *Phase de compositon*

Les facteurs proposés :

Cette étape consiste à définir les facteurs permettant d'analyser les transactions effectuées par les utilisateurs dans le réseau IoT social. Dans ce travail, à la suite d'une étude dans la littérature nous avons remarqué que les auteurs ont utilisé des facteurs plus généraux tels que la centralité de l'utilisateur ou le nombre d'amis en commun entre deux utilisateurs, qui n'ont aucune relation sémantique avec les attaques de confiance en question. Cependant, pour prédire le type d'attaque effectué par l'utilisateur malveillant, les facteurs doivent être dérivés de la description de chaque type d'attaque et doivent décrire les comportements des utilisateurs. A cet égard, dans nos travaux antérieurs (Abdelghani et al., 2018; Masmoudi et al., 2020), nous avons proposé sept nouveaux facteurs dérivés de la description de chaque type d'attaque de confiance (BMA, BSA, SPA et DA). L'objectif était de détecter ces quatre types d'attaques en mode hors ligne. Ces travaux ont donné des résultats satisfaisants avec des valeurs de f1-score égales à 92,8% pour (Abdelghani et al., 2018) et 95,03% pour (Masmoudi et al., 2020). Cependant, notre objectif actuel est de prévenir en temps réel tous les types d'attaques de confiance (BMA, BSA, SPA, DA, OSA et OOA). Par conséquent, les

facteurs proposés dans nos travaux antérieurs doivent être modifiés et améliorés pour atteindre notre objectif et améliorer la précision de la prédiction des transactions malveillantes.

Nous présentons chaque facteur comme suit :

— **La valeur de la confiance :** ce facteur $ValConf(u_i)$ représente la valeur de confiance globale d'un utilisateur $u_i \in u$ dans le réseau et se calcule en divisant le nombre d'interactions positives de u_i par le nombre total d'interactions effectuées, comme illustré par l'équation 3.1. Les interactions positives correspondent à des interactions ayant reçu des valeurs de vote élevées. Les noeuds ayant une valeur de confiance élevée sont plus vulnérables aux attaques de la part d'autres noeuds, tandis que les noeuds ayant une valeur de confiance faible sont plus susceptibles d'initier des attaques de confiance.

$$ValConf(u_i) = \frac{1}{N_i} \sum_{k=0}^{N_i} rt_i \tag{3.1}$$

(Avec N_i est le nombre de votes attribués à l'utilisateur u_i et $rt_i \in [1, 10]$ la valeur du vote.)

— **Le vote (rate) :** Nous définissons le vote $rt(u_i, u_j)$ comme la valeur donnée par l'utilisateur $u_i \in u$ au service $s_k \in s$ fourni par l'utilisateur $u_j \in u$.
Dans le cadre de cette étude, nous avons décidé de limiter la plage de valeurs possibles pour les votes entre 1 et 10. Les votes faibles, compris entre 1 et 3, ainsi que les votes élevés, entre 7 et 10, sont particulièrement pertinents pour notre analyse. Il est important de souligner que les attaques visant à manipuler la confiance impliquent souvent des votes malhonnêtes (Abdelghani et al., 2018; Masmoudi et al., 2020). Les utilisateurs malveillants peuvent augmenter leur propre valeur de confiance dans le réseau en donnant des votes élevés ou réduire la valeur de confiance des autres utilisateurs en donnant des votes faibles. Nous avons donc considéré le vote comme une composante essentielle de notre système de détection d'utilisateurs malveillants qui ont effectués des attaques de confiance. En prenant en compte les votes et d'autres facteurs pertinents, notre système peut identifier les utilisateurs qui ont des comportements malveillants dans le réseau IoT social.

— **Similarité de vote ou véracité :** Le terme de véracité est souvent utilisé dans la littérature en lien avec la notion de confiance, mais il n'existe pas de définition commune à ce terme et sa mesure varie d'un travail à l'autre. Dans notre étude, nous considérons qu'un utilisateur est digne de confiance s'il est véridique dans les votes qu'il attribue, c'est-à-dire que ses votes reflètent son opinion réelle et ne sont pas biaisés pour augmenter sa propre valeur de confiance ou diminuer celle des autres utilisateurs. Ainsi, nous pouvons considérer un utilisateur comme fiable s'il n'essaie

pas de donner de mauvais votes pour se présenter sous une fausse image dans le but d'être sélectionné comme fournisseur de service. Par conséquant, dans notre système, la véracité des votes données par un utilisateur est cruciale pour évaluer la fiabilité des transactions effectuées par les utilisateurs. Lorsqu'elle est combinée à d'autres facteurs, elle permet de détecter divers types d'attaques. Par exemple, dans l'attaque SPA, un utilisateur malveillant tente d'augmenter sa propre valeur de confiance en se donnant de bon votes, même si ses services sont de mauvaise qualité. Dans l'attaque BMA, un utilisateur malveillant donne des faibles valeurs de vote à un utilisateur fournissant des services de bonne qualité, afin de ruiner sa valeur de confiance.
Nous utilisons la distance euclidienne selon l'équation 3.2 pour mesurer la véracité et la similarité des votes d'un utilisateur $u_i \in u$, notée $Ver(u_i)$. Cette mesure est effectuée en comparant la valeur de vote rt_{ik} de l'utilisateur u_i avec la moyenne des autres votes $\overline{rt_k}$ pour le service $s_k \in s$ de l'utilisateur $u_j \in u$. Si la valeur de vote est significativement différente de la majorité des autres votes, elle est considérée comme malhonnête. Si la distance de véracité est grande, le vote est classé comme une attaque, tandis que si elle est faible, le vote est considéré comme similaire aux autres votes.
Cependant, il est important de noter que des opinions différentes peuvent mener à des votes différents, mais si les votes sont très différents des autres utilisateurs, ils ne sont pas représentatifs. La véracité doit être combinée avec d'autres facteurs pour déterminer si un utilisateur est malveillant ou non.

$$Ver(u_i) = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{p}\left(rt_{ijk} - \overline{rt}_k\right)^2} \forall u_i \in u \tag{3.2}$$

(Avec Le terme rt_{ijk} représente la valeur de vote donnée par l'utilisateur u_i pour le service s_k de l'utilisateur u_j et le terme $\overline{rt}_k$ correspond à la moyenne des votes données par l'ensemble des utilisateurs du réseau pour le service s_k.)

La figure 3.6 représente une matrice des votes de tous les utilisateurs u sur tous les services s dans le réseau. Pour évaluer la similarité entre les votes de l'utilisateur $u_i \in u$ et la moyenne des votes de tous les utilisateurs pour chaque service, nous extrayons le vecteur $\overline{rt}$, qui est un vecteur colonne de dimension p contenant la moyenne des votes attribués à chaque service. Ensuite, nous utilisons la distance euclidienne pour calculer la similarité entre les votes de l'utilisateur u_i et les votes des autres utilisateurs.

— **Qualité du fournisseur (Quality of Provider) :** La mesure $QoP(u_i)$ évalue la qualité des services $QoS(s_k)$ fournis par l'utilisateur u_i, permettant de déterminer s'ils sont de bonne ou de mauvaise qualité. Cette mesure est importante car les utilisateurs malveillants ont tendance à propager des services de mauvaise qualité, tandis que les services de bonne qualité ac-

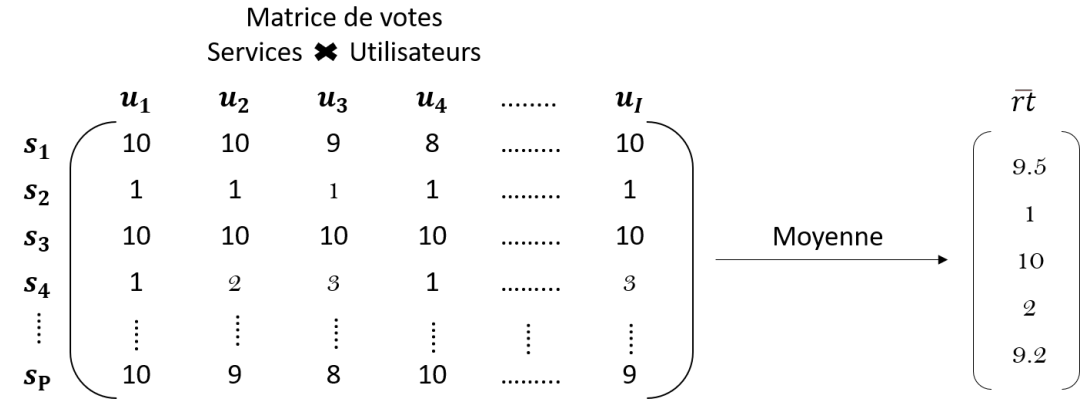

FIGURE 3.6 – *Matrice des votes de tous les utilisateurs sur tous les services dans le réseau*

quièrent une valeur de confiance favorable dans le réseau. Les utilisateurs malveillants doivent adopter des comportements malveillants pour propager des services de mauvaise qualité.
Ce facteur est calculé selon l'équation 3.3 :

$$QoP(u_i) = \frac{\sum d_j \in D(u_i), s_k \in S(d_j) QoD(d_j) \times QoS(s_k)}{\sum d_j \in D(u_i) QoD(d_j)} \tag{3.3}$$

(Avec $QoP(u_i)$ l'ensemble des services fournis par l'utilisateur u_i et $QoS(s_k)$ est la valeur QoS du service s_k.)

— **Similarité des utilisateurs :** La mesure de similarité $SimU(u_i, u_j)$ évalue le degré de similarité entre deux utilisateurs, $u_i et u_j \in u$, en se basant sur plusieurs caractéristiques telles que les profils, les intérêts, les services fournis, les dispositifs utilisés, ainsi que la fréquence de proximité. L'objectif de cette mesure est de détecter l'affinité entre les utilisateurs. Toutefois, elle peut également révéler des attaques de type SPA où un même utilisateur tente de promouvoir sa propre valeur de confiance en utilisant une fausse identité. Pour estimer cette similarité, nous utilisons l'indice de Jaccard selon l'équation 3.4. Cet indice est compris entre 0 et 1, où 0 signifie que les utilisateurs n'ont aucune similarité et 1 signifie qu'ils sont identiques.

$$SimU(u_i, u_j) = \alpha_1 \times SimP_{ij} + \alpha_2 \times SimI_{ij} + \alpha_3 \times SimD_{ij} + \alpha_4 \times SimS_{ij} + \alpha_5 \times FreqP_{ij} \tag{3.4}$$

(Avec $SimP_{ij}$ fait référence à la similarité des profils ; $SimI_{ij}$ fait référence à la similarité des intérêts ; $SimD_{ij}$ fait référence à la similarité des périphériques ; $SimS_{ij}$ fait référence à la similarité de service et $FreqP_{ij}$ se réfère à la fréquence de proximité entre u_i et u_j et est calculée par l'équation 3.5)

$$FreqP_{ij} = \frac{nb_{prox}(u_i, u_j)}{nb_{prox}(u_i, u_k)} \tag{3.5}$$

— **Fréquence des votes :** Ce facteur correspond à la fréquence à laquelle l'utilisateur u_i donne des votes à l'utilisateur u_j et est noté $FreqV(u_i, u_j)$. Il est calculé en divisant le nombre de votes donnés par l'utilisateur u_i à l'utilisateur u_j par le nombre total de votes donnés par u_i. Ce facteur repose sur la typologie des différentes attaques possibles. Par exemple, dans l'attaque SPA, un utilisateur malveillant fournissant de mauvais services et ayant une mauvaise valeur de confiance s'attribue un grand nombre de votes élevés pour augmenter sa propre valeur de confiance. L'attaque BMA, vise à discréditer un utilisateur bienveillant bien connu en lui attribuant un grand nombre de mauvais votes, tandis que l'attaque BSA cherche à promouvoir la valeur de confiance d'un utilisateur malveillant en lui donnant un grand nombre de votes élevés. Dans tous les cas, un grand nombre de votes attribués est révélateur car un petit nombre de votes ne suffirait pas à modifier la valeur de confiance de l'utilisateur ciblé. Donc, ce facteur révèlera probablement un grand nombre de votes donnés par u_i à u_j.

— **Tendance des votes :** Ce facteur est noté $TendV(u_i)$. Il consiste à évaluer la tendance d'évaluation d'un utilisateur u_i en examinant le rapport entre le nombre de votes élevé qu'il a donné et le nombre total de votes qu'il a émis. Cette mesure permet de déterminer si cet utilisateur est enclin à émettre des votes mauvais ou élevés. Elle est également utile pour détecter l'attaque discriminatoire (DA), où un utilisateur fournit des mauvais votes de manière aléatoire, sans justification.

Les facteurs Qualité du fournisseur, Similarité des utilisateurs, Fréquence et Tendance des votes sont expliqués en détail dans nos travaux antérieurs (Abdelghani et al., 2018; Masmoudi et al., 2020).

Division de la base de données :

Après avoir sélectionné et choisi les différents éléments des transactions (appelés facteurs) qui permettent d'analyser les actions effectuées par les utilisateurs dans le réseau IoT social, l'étape suivante consiste à collecter et stocker toutes les transactions effectuées dans une base de données selon ces facteurs comme montre la figure 3.5. Cette base de données est ensuite divisée en sous-ensembles pour faciliter les étapes ultérieures d'analyse et de prévention des attaques de confiance, en permettant une analyse plus ciblée et précise. L'origine des transactions collectées et stockées dans notre base de données seront détaillées dans le chapitre 5.

Chaque ligne de la base de données est structurée de la manière suivante :

Le vote (u_i, s_k), la valeur de la confiance de l'utilisateur (u_i), la valeur de la confiance de l'utilisateur (u_j), la similarité des votes ou la véracité de l'utilisateur (u_i), la qualité du fournisseur de l'utilisateur (u_i), la qualité du fournisseur de l'utilisateur (u_j), la tendance des votes de l'utilisateur (u_i), la tendance des votes de l'utilisateur (u_j), la fréquence des votes de l'utilisateur u_i à l'utilisateur u_j (u_i, u_j), la similarité entre les deux utilisateurs (u_i, u_j) et la classe, soit sécurisée, soit l'un des types d'attaques de confiance (BMA, BSA, SPA, DA, OSA ou OOA).

Après avoir créé la base de données contenant les transactions et les différents types d'attaques de confiance, nous procédons à sa division en trois sous-ensembles distincts. Tout d'abord, nous avons l'ensemble d'apprentissage qui nous permet de générer un modèle de classification, également appelé un classificateur, pour effectuer une classification précise des transactions. Ce modèle sera utilisé pour prédire la classe des nouvelles transactions en temps réel. Ensuite, nous avons l'ensemble de test qui sert à évaluer les performances de notre classificateur. Enfin, nous avons l'ensemble de flux (ou streaming) qui nous permet de générer les transactions en temps réel et de prédire leur classe en utilisant le classificateur préalablement entraîné.

Les autres étapes de notre approche SparkChain seront explicitées dans le prochain chapitre. Nous expliquerons en détail comment les facteurs seront agréger pour analyser les transactions afin de prédire si une transaction est malveillante ou sécurisée en identifiant le type d'attaque de confiance effectuée par l'utilisateur malveillant.

3.6 Conclusion

Tout au long de ce chapitre, nous avons exposé la modélisation du système IoT social que nous avons utilisée pour notre étude. Ensuite, nous avons proposé une architecture en couches qui fusionne les avantages de la blockchain et d'Apache Spark avec l'IoT social, dans le but de fournir une sécurité et une confiance maximales pour les transactions effectuées dans le réseau IoT social. En effet, notre architecture est conçue pour garantir la traçabilité, la transparence et la sécurité de chaque transaction, ce qui représente un défi majeur pour les systèmes IoT sociaux.

Par ailleurs, nous avons introduit notre solution novatrice, SparkChain, conçue pour prévenir les attaques de confiance en temps réel dans l'IoT social. Elle permet de détecter les transactions malveillantes à temps, en empêchant leur propagation dans le réseau IoT social.

En outre, nous avons présenté en détail les facteurs, ou les composants des transactions, qui sont utilisés pour la phase de composition de la confiance. Ces facteurs sont déduits de chaque modèle d'attaque de confiance, et ils permettent de détecter rapidement si un utilisateur a lancé une transaction malveillante, qui constitue en fait une attaque de confiance. L'utilisation de ces facteurs est donc

essentielle pour garantir la sécurité et la confiance des transactions effectuées dans l'IoT social.

Dans le chapitre suivant, nous allons poursuivre notre présentation de SparkChain, en décrivant les méthodes proposées pour agréger ces facteurs et les transformer en scores de confiance. Ce score aide notre protocole de consensus (PACS) de prendre la décision de valider et sauvegarder les transactions sécurisées ou interrompre et annuler les transactions malveillantes. Nous allons détailler également notre protocole qui s'appuie sur Apache Spark. De plus, nous aborderons les méthodes de propagation et de mise à jour de ces scores de confiance dans le réseau IoT social, ainsi que leur gestion.

4 Gestion de la confiance avec SparkChain : agrégation, propagation et mise à jour

Sommaire

4.1 Introduction

Dans le chapitre précédent, nous avons introduit SparkChain, notre architecture innovante visant à prévenir les attaques de confiance dans l'IoT social en temps réel. SparkChain est basée sur la combinaison de plusieurs technologies avancées, notamment la blockchain et Apache Spark, et utilise notre propre protocole de consensus de la blockchain pour surveiller les transactions en temps réel et détecter les attaques de confiance et produire une réponse immédiate en interrompant et annulant ces transactions malveillantes.

Afin d'atteindre cet objectif, nous avons commencé dans le chapitre précédent par la phase de composition qui consiste à identifier les différents facteurs qui permettent de détecter les transactions malveillantes. Nous avons décrit ces facteurs en détail et avons montré comment ils peuvent être utilisés pour identifier rapidement les transactions malveillantes, si elles existent.

Dans ce chapitre, nous allons poursuivre notre présentation de SparkChain. Pour cela, nous allons détailler la phase d'agrégation. Cette phase consiste à agréger les différents facteurs identifiés dans la phase de composition pour générer un score de confiance pour chaque transaction. Ce score de confiance aide notre protocole de consensus à prendre une décision éclairée sur la validité de la transaction en temps réel. Pour calculer les scores de confiance, nous avons développé plusieurs méthodes d'agrégation basées sur des techniques d'apprentissage automatique et d'apprentissage profond. Nous avons également utilisé Apache Spark pour mettre en œuvre ces méthodes et les intégrer dans notre architecture SparkChain.

Nous allons également détailler notre nouveau protocole de consensus pour la blockchain, qui utilise Apache Spark, en particulier Spark Streaming, pour l'analyse en temps réel des transactions générées par les utilisateurs de l'IoT social. Ce protocole permet une détection rapide des attaques de confiance et une réponse immédiate en les interrompant et en les annulant.

En plus de l'agrégation des scores de confiance, nous détaillerons également notre proposition pour les phases de propagation et de mise à jour de ces scores de confiance dans le réseau IoT social, ainsi que les stratégies pour gérer ces scores de manière sécurisée et efficace.

Ce chapitre est organisé en plusieurs sections. Tout d'abord, nous présentons en détails la phase d'agrégation des facteurs de confiance, qui est essentielle pour générer des scores de confiance précis pour chaque transaction dans le réseau IoT social. Pour ce faire, nous décrivons plusieurs méthodes d'agrégation que nous avons conçues pour calculer efficacement ces scores, en utilisant des techniques d'apprentissage automatique et d'apprentissage profond basées sur Apache Spark. Nous présentons également notre protocole de consensus, qui permet de valider les transactions en utilisant les scores de confiance générés par notre approche. Dans la deuxième partie de ce chapitre, nous présentons les

méthodes de propagation et de mise à jour de la confiance. Nous expliquons comment les transactions et les scores de confiance générés par notre approche sont propagés à travers le réseau IoT social et comment ils sont mis à jour en temps réel. Nous fournissons également un scénario concret qui met en évidence les mesures préventives contre les attaques de confiance dans le réseau IoT social en utilisant la technologie de la blockchain. Enfin, nous concluons ce chapitre en résumant les points clés de notre approche et en soulignant les principales contributions de notre travail. Nous insistons sur l'importance de notre approche pour prévenir les attaques de confiance dans l'IoT social en temps réel et sur les avantages de la combinaison de la blockchain, d'Apache Spark et de l'IoT social pour fournir une solution efficace et sécurisée.

4.2 Phase d'agrégation

La phase d'agrégation de la confiance permet d'agréger les différents facteurs de confiance considérés afin de générer un score de confiance pour chaque transaction. Elle se compose de deux modules principaux. Le premier module, appelé module de création d'un classificateur, se concentre sur l'analyse des transactions pour construire des classificateurs de transactions en utilisant des techniques d'apprentissage automatique et d'apprentissage profond. Pour mettre en œuvre ces classificateurs, nous avons également utilisé Apache Spark. Le deuxième module, appelé module de détection des attaques en temps réel, utilise ces classificateurs pour faire des prédictions sur un flux de transactions en utilisant le cadre de traitement en temps réel appelé Spark Streaming. Ce module permet de prédire la classe de chaque nouvelle transaction générée dans le réseau en temps réel (soit sécurisée soit l'un des types d'attaque). En se basant sur cette étiquette, notre nouveau protocole appelé Preuve d'Attaques de Confiance basées sur Spark (PACS) prend la décision de valider ou d'annuler la transaction.

Dans cette section, nous allons fournir une description détaillée des deux modules qui composent la phase d'agrégation de la confiance, à savoir le module de création de classificateur et le module de détection des attaques en temps réel. Nous allons également présenter les différentes méthodes que nous avons proposées pour agréger les différents facteurs de confiance, afin de prévenir efficacement les attaques de confiance en temps réel dans le réseau IoT social via notre protocole de consensus PACS.

4.2.1 Module de création d'un classificateur

Comme indiqué précédemment, la prévention des attaques de confiance dans l'IoT social est essentielle pour garantir la sécurité et la fiabilité des transactions effectuées par les utilisateurs. Ainsi, nous avons élaboré une méthode novatrice qui repose sur l'adaptation de la technologie blockchain, associée à un nouveau protocole de consensus nommé Preuve d'Attaques de Confiance basées sur Spark (PACS) qui se sert d'Apache Spark.

En effet, la détection des attaques de confiance constitue une technique clé pour prévenir les transactions malveillantes en temps réel. En particulier, un utilisateur est considéré comme malveillant s'il tente d'effectuer l'une des attaques suivantes : BMA, BSA, SPA, DA, OSA ou OOA. En revanche, s'il ne réalise aucune de ces attaques, il est considéré comme bienveillant. Pour ce faire, il est nécessaire de réaliser une analyse en profondeur des transactions générées par les utilisateurs dans le but de détecter toute transaction malveillante et de l'interrompre rapidement via notre protocole PACS.

Ainsi, PACS est conçu comme un système de classification qui permet de classer les transactions en sept classes comme suit :

1. la classe bma-attack, lorsqu'un utilisateur effectue une attaque de type BMA.
2. la classe spa-attack, lorsqu'un utilisateur effectue une attaque de type SPA.
3. la classe bsa-attack, lorsqu'un utilisateur effectue une attaque de type BSA.
4. la classe da-attack, lorsqu'un utilisateur effectue une attaque de type DA.
5. la classe osa-attack, lorsqu'un utilisateur effectue une attaque de type OSA.
6. la classe ooa-attack, lorsqu'un utilisateur effectue une attaque de type OOA.
7. la classe sécurisée, lorsqu'un utilisateur n'a effectué aucune des attaques citées.

Ce protocole PACS utilise des techniques d'apprentissage automatique et d'apprentissage profond pour analyser les transactions et générer des classificateurs de transactions à l'aide d'Apache Spark. Ces classificateurs sont ensuite utilisés par le deuxième module de détection d'attaques en temps réel, qui fait des prédictions sur un flux de transactions en utilisant Spark Streaming. A l'aide de cette prédiction, notre protocole PACS sera capable de prévenir les attaques de confiance en temps réel, en détectant les transactions malveillantes et en les interrompant. Il constitue une méthode innovante et efficace pour améliorer la sécurité et la fiabilité des transactions dans l'IoT social.

Dans le but de choisir le modèle le plus optimal pour la classification précise des transactions, nous avons décidé d'utiliser deux types de techniques d'apprentissage supervisé basés sur Spark : MLlib pour l'apprentissage automatique (Machine Learning) (Masmoudi et al., 2023) et Elephas pour l'apprentissage profond (Deep Learning) (Jmal et al., 2023). Grâce à ces deux bibliothèques, nous avons entraîné plusieurs modèles utilisant diverses méthodes de classification, dans le but de déterminer le modèle le plus performant parmi eux.

Ainsi, dans le cadre de notre étude, nous considérons chaque paire d'utilisateurs (u_i , u_j) et nous récupérons toutes les transactions passées. Nous calculons ensuite la valeur des différents facteurs qui permettent de caractériser les transactions entre (u_i , u_j), comme présenté dans le tableau 4.1. L'ensemble de ces valeurs de facteurs est ensuite utilisé comme entrée dans les modèles d'apprentissage automatique et profond lors de la phase d'apprentissage.

TABLE 4.1 – *Illustration d'une observation pour l'entrée d'un modèle d'apprentissage automatique et profond*

Désignation	Facteur	Lié à
Valeur de la confiance	$ValConf(u_i)$	u_i
Veracité ou Similarité des votes	$Ver(u_i)$	
Qualité du fournisseur	$QoP(u_i)$	
Tendance des votes	$TendV(u_i)$	
Valeur de la confiance	$ValConf(u_j)$	u_j
Qualité du fournisseur	$QoP(u_j)$	
Tendance des votes	$TendV(u_j)$	
Vote	$rt(u_i, u_j)$	(u_i, u_j)
Similarité des utilisateurs	$SimU(u_i, u_j)$	
Fréquence des votes	$FreqV(u_i, u_j)$	

4.2.1.1 Classificateur basé sur Spark MLlib

L'un des objectifs clés de l'apprentissage automatique consiste à classer chaque transaction entre deux utilisateurs u_i et u_j en lui attribuant une étiquette ou une classe. La classification est généralement réalisée en utilisant des techniques telles que l'apprentissage supervisé. Le processus d'apprentissage automatique se déroule en deux phases distinctes, comme illustré dans la figure 4.1. La première phase est la phase d'apprentissage. Pendant cette phase, le modèle est ajusté à partir des transactions étiquetées en utilisant différentes techniques de traitement de données et d'optimisation pour améliorer la performance du classificateur. La seconde phase est la phase de test, qui permet de prédire l'étiquette de nouvelles transactions (bma-attack, bsa-attack, spa-attack, da-attack, osa-attack, ooa-attack ou sécurisée) à l'aide du classificateur préalablement construit. La performance de cet classificateur est évaluée en comparant les prédictions aux étiquettes réelles. Si le classificateur est performant, il peut être utilisé pour effectuer des prédictions sur des futures transactions non étiquetées en temps réel.

4.2.1.1.1 Choix du modèle d'apprentissage automatique

Dans notre étude, nous avons entrepris une exploration de différents modèles d'apprentissage supervisé couramment utilisés, tels que les machines à vecteurs de support (SVM), les Naive Bayes, la table de décision et la régression linéaire (RL). Suite à une évaluation comparative des résultats obtenus, nous avons choisi d'adopter le modèle de régression linéaire (RL) pour notre approche, car il a démontré les performances les plus prometteuses.

Le modèle de régression linéaire est une méthode d'apprentissage supervisé qui permet de prédire la valeur d'une variable dépendante (Y) en fonction d'une variable indépendante donnée (X). Il établit une relation linéaire entre l'entrée (X) et la sortie (Y), d'où son nom de régression linéaire. Cette technique est particulièrement adaptée lorsque la relation entre les variables peut être approximée

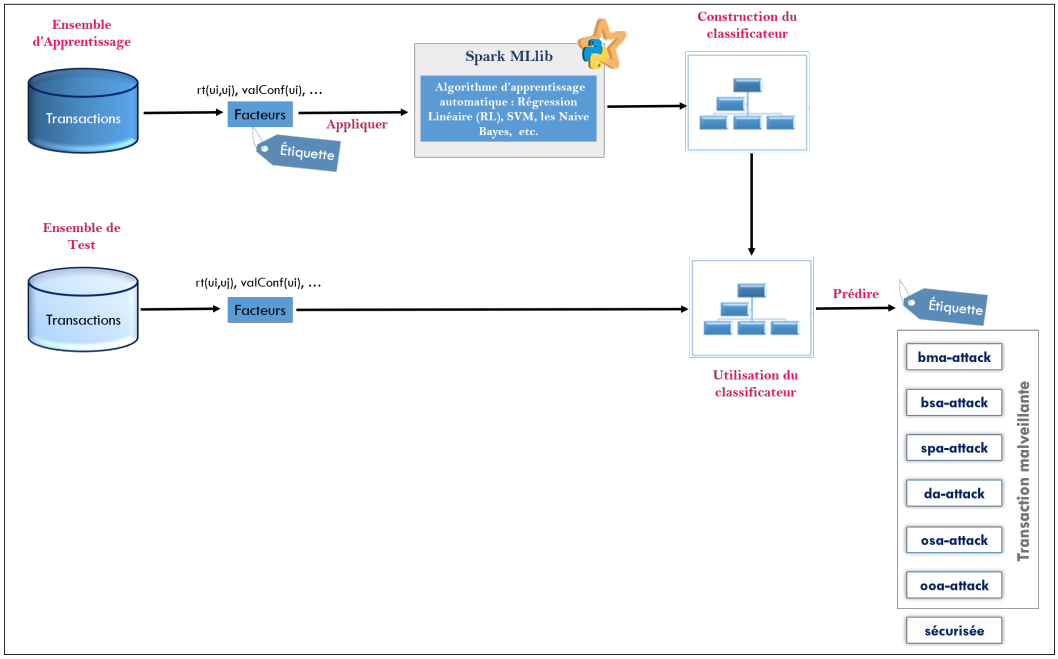

FIGURE 4.1 – *Phase d'agrégation : Création d'un classificateur d'apprentissage automatique basé sur Spark MLlib*

par une ligne droite. L'équation de la fonction de régression linéaire s'exprime comme suit 4.1 :

$$Y = \theta 1 + \theta 2 \cdot X \tag{4.1}$$

où Y : la variable dépendante qui représente l'étiquettes de données, X : la variable indépendante correspond aux données d'apprentissage en entrée, et $\theta 1$ et $\theta 2$ représentent les coefficients de régression qui sont estimés à partir des données d'entraînement et jouent un rôle essentiel dans la modélisation de la relation entre X et Y.

Au cours de la phase d'apprentissage, le modèle de régression linéaire (LR) cherche à trouver les valeurs optimales des coefficients θ_1 et θ_2 afin de tracer une ligne qui correspond au mieux aux données. Son objectif est de minimiser l'erreur entre les valeurs prédites (pred) de Y et les valeurs réelles (Y) pour un ensemble donné de valeurs de X. Cette erreur est évaluée à l'aide d'une fonction de coût(J), qui mesure la différence entre les valeurs prédites et les valeurs réelles. Dans le cas de la régression linéaire, la fonction de coût la plus couramment utilisée est l'erreur quadratique moyenne (RMSE), qui calcule la moyenne des carrés des différences entre les valeurs prédites et les valeurs réelles.

Ainsi, en minimisant la fonction de $cot(J)$, la méthode de régression linéaire cherche à ajuster les coefficients θ_1 et θ_2 de manière à réduire au maximum l'erreur entre les valeurs prédites et les valeurs réelles. Ce processus d'optimisation

permet d'obtenir la meilleure ligne de régression qui représente au mieux la relation linéaire entre X et Y. La fonction de $cot(J)$ peut être présentée comme suit dans l'équation 4.2 :

$$J = \sum_{i=1}^{n} (pred_i - Y_i)^2 \frac{1}{n} \tag{4.2}$$

Le processus de descente de gradient commence par l'initialisation des valeurs des coefficients θ_1 et θ_2 avec des valeurs aléatoires. Ensuite, de manière récursive, les valeurs des coefficients sont mis à jour itérativement en fonction de la pente du coût par rapport à chaque coefficient. Cette mise à jour répétée permet au modèle de converger progressivement vers les valeurs optimales des coefficients θ_1 et θ_2 qui minimisent le coût.

Par conséquant, en utilisant ce modèle de Régression Linéaire (RL) dans notre protocole de consensus PACS, nous sommes en mesure de faire des prédictions sur de nouvelles transactions non étiquetées en utilisant la relation linéaire établie entre la variable indépendante (données d'entraînement) et la variable dépendante (étiquette). Cela nous permet d'obtenir des estimations ou des prédictions pour ces nouvelles transactions, contribuant ainsi à notre approche basée sur la classification.

4.2.1.1.2 L'algorithme de notre classificateur basé sur RL

Après avoir sélectionné le modèle pour notre protocole de consensus PACS, nous présentons désormais l'algorithme 1 pour classer les transactions en utilisant les étiquettes (bma-attack, bsa-attack, spa-attack, da-attack, osa-attack, ooa-attack ou sécurisée). Notre classificateur est basé sur un apprentissage supervisé, où nous utilisons un ensemble de données d'entraînement étiquetées.

Dans un premier temps, à la ligne 1, nous initialisons une session Spark, car chaque application Spark nécessite une session Spark. Ensuite, à la ligne 2, l'algorithme prend en entrée un ensemble de transactions étiquetées "T". Ce dernier contient les valeurs de chaque facteur, tels que la valeur de confiance, le vote, la similarité, la similarité des votes, la qualité du fournisseur, la fréquence des votes, ainsi que l'étiquette associée à chaque ligne de l'ensemble des transactions. L'étiquette peut correspondre à l'un des types d'attaques ou sécurisée.

Ensuite, à la ligne 3, l'algorithme regroupe les facteurs en une seule colonne nommée "features" en tant que vecteur à l'aide d'une fonction de la bibliothèque ML de Spark, puis les transmet à une autre fonction pour les normaliser. De plus, aux lignes 4 à 7, l'algorithme convertit les étiquettes en étiquettes numériques allant de 0,0 à 6,0 à l'aide de la bibliothèque d'indexation de chaînes de la bibliothèque ML de Spark. Cette transformation est réalisée car lors de la phase d'apprentissage, les résultats sont toujours de type numérique. Ensuite, à la ligne

Algorithme 1 : Algorithme du classificateur basé sur RL

Require: **Ensemble de transactions étiquetées "T"**
Ensure: **Classificateur entraîné "CML"**

▷ **Créer une session SparkSession**

$spark \leftarrow SparkSession.builder()$

▷ **Charger les données du fichier "T"**

$df \leftarrow spark.read(T)$ ▷ df : est un vecteur qui contient les données chargées

▷ **Combiner les facteurs en une seule colonne sous forme de vecteur "requiredFeatures"**

$requiredFeatures \leftarrow (rt, ValConf_i, ValConf_j, ver_i, qop_i, qop_j, tendV_i, tendV_j, simU_{ij}, freqV_{ij}, label)$

▷ **Transformer les étiquettes en étiquettes numériques de 0,0 à 6,0 à l'aide de la bibliothèque d'indexation de chaînes de la bibliothèque Spark ML**

for *each label in df* **do**

$classEncoder \leftarrow StringIndexer(inputCol = label, outputCol = newLabel).fit(df)$

$df \leftarrow classEncoder.transform(df)$

END for

▷ **Créer un nouveau vecteur de données df2 avec la nouvelle étiquette numérique**

$df2 \leftarrow df.select(rt, ValConf_i, ValConf_j, ver_i, qop_i, qop_j, tendV_i, tendV_j, simU_{ij}, freqV_{ij}, newlabel)$

▷ **Inclusion des données d'entrée dans la colonne des "features"**

$vecAssembler \leftarrow VectorAssembler(inputCols = requiredFeatures, outputCol = features)$

▷ **Construire un pipeline pour combiner le vecteur de facteurs et l'étiquette numérique**

$vecDF \leftarrow vecAssembler.transform(df2)$

▷ **Diviser le pipeline en deux parties aléatoirement, l'une pour la phase d'apprentissage A et l'autre pour la phase de test T**

$A, T \leftarrow vecDF.randomSplit(0.7, 0.3)$

▷ **Appeler le modèle de régression linéaire pour la phase d'apprentissage**

$lr \leftarrow LinearRegression(featuresCol = features, labelCol = newLabel)$

▷ **Commencer la phase d'apprentissage avec les classes numériques en utilisant le modèle de régression linéaire**

$lrModel \leftarrow lr.fit(A)$

▷ **Évaluer les performances du modèle entraîné en utilisant la fonction de transformation sur les données de test "T"**

$predictions \leftarrow lrModel.transform(T)$

$LREvaluator \leftarrow RegressionEvaluator(metricName = "fMeasure")$

$LRFMeasure \leftarrow LREvaluator.evaluate(predictions)$

$print("FMeasure = " + str(fMeasure))$

▷ **Produire le classificateur "CML" à la fin de la phase d'évaluation**

$lrModel.save("CML")$

8, l'algorithme construit un pipeline pour regrouper toutes ces transformations. Les étapes du pipeline sont également spécifiées sous la forme d'un tableau ordonné, ce qui facilite le prétraitement des données. Ensuite, aux lignes 9 et 10, nous sélectionnons uniquement la colonne des facteurs et la colonne des étiquettes pour l'optimisation. Enfin, à la ligne 11, nous répartissons aléatoirement les données en un ensemble d'apprentissage "A" et un ensemble de test "T".

Ensuite, aux lignes (12-13), l'algorithme invoque le modèle de régression linéaire pour la phase d'apprentissage. Il appelle la fonction d'ajustement et l'applique à nos données d'apprentissage. Ce modèle entraîné sera utilisé dans la phase de test pour évaluer ses performances en termes de classification des transactions. Cette évaluation sera réalisée à l'aide de la fonction de transformation sur les données de test "T" comme montre les lignes de 13 à 17.

Une fois que nous obtenons des résultats satisfaisants, nous conservons le classificateur "CML" pour son utilisation dans le deuxième module comme indique la ligne 18. Ce module a pour objectif d'étiqueter en temps réel les flux de transactions avec différentes catégories, telles que les attaques (bma-attack, bsa-attack, spa-attack, da-attack, osa-attack, ooa-attack ou sécurisée). L'étiquetage en temps réel permet une détection proactive des attaques de confiance, ce qui permet de prendre des mesures immédiates par notre protocole de consensus PACS pour interrompre les transactions malveillantes et prévenir les attaques. Cette méthode assure une protection continue en surveillant activement les flux de transactions et en identifiant les transactions suspectes.

4.2.1.2 Classificateur basé sur Spark Elephas

Après avoir étudié différentes recherches sur Apache Spark dans le chapitre 2, il est observé que toutes les approches se basent sur l'apprentissage automatique avec Spark, sans utiliser l'apprentissage profond. Cependant, l'apprentissage profond est une méthode avancée de l'intelligence artificielle qui dépasse l'apprentissage automatique en exploitant les réseaux neuronaux artificiels pour obtenir des performances remarquables dans diverses tâches telles que la classification d'objets, la reconnaissance d'images, de vidéos, de textes, et bien d'autres.

Malgré cela, la plupart des chercheurs dans la littérature ont utilisé des modèles classiques tels que les réseaux neuronaux convolutifs (CNN) pour le traitement des images, les réseaux neuronaux récurrents (RNN) pour la reconnaissance vocale et la traduction, ainsi que le perceptron multicouche (MLP) pour les données numériques (Masmoudi et al., 2020). Cependant, dans le cadre de notre étude, nous souhaitons construire un modèle d'apprentissage profond (ou un classificateur) en utilisant Apache Spark. Pour ce faire, nous utiliserons une extension de Keras appelée Elephas, qui est spécialement conçue pour construire des modèles d'apprentissage profond avec Spark en utilisant l'API PySpark.

L'intégration d'Elephas avec Spark facilite également le déploiement des modèles d'apprentissage profond entraînés. Une fois qu'un modèle est entraîné avec

Elephas, il peut être utilisé pour effectuer des prédictions sur de nouvelles données en utilisant Spark, ce qui permet d'exploiter les capacités de traitement distribué de Spark pour une exécution rapide et évolutive.

Les modèles d'apprentissage profond, tels que les réseaux neuronaux profonds, sont constitués de plusieurs couches interconnectées. Ces couches sont organisées de manière ascendante, allant des couches d'entrée aux couches de sortie. Lors du traitement des données dans chaque couche, le réseau neuronal calcule la moyenne pondérée des entrées, qui est ensuite transmise à une fonction d'activation pour produire la sortie de la couche. Entre la couche d'entrée et la couche de sortie, il existe des couches intermédiaires appelées couches cachées, qui appliquent la même fonction de transformation aux données qu'elles reçoivent en entrée. Les sorties d'une couche donnée servent d'entrées à la couche suivante du réseau. Les poids synaptiques, attribués à chaque connexion entre les neurones, déterminent l'importance relative de chaque entrée dans le calcul des sorties de la couche. Pour déterminer les poids optimaux, un algorithme appelé rétropropagation est utilisé. Cet algorithme est itéré des milliers de fois pour chaque neurone du réseau neuronal, en commençant par la dernière couche et en remontant jusqu'à la première couche. L'algorithme de rétropropagation calcule l'erreur entre la sortie réelle et la sortie prédite par le réseau, puis propage cette erreur en arrière en ajustant les poids pour la réduire. Ainsi, grâce à l'algorithme de rétropropagation, les réseaux neuronaux profonds peuvent apprendre à partir des données en ajustant les poids de manière itérative, ce qui leur permet de s'adapter aux caractéristiques des données et de produire des prédictions précises.

Afin de créer et d'entraîner un modèle d'apprentissage profond, il est nécessaire de fixer une combinaison de valeurs de paramètres. Ces paramètres sont configurés en fonction d'une série d'expériences menées sur l'ensemble des données. Les détails de ces paramètres seront présentés dans le prochain chapitre consacré à l'expérimentation. Une fois le modèle entraîné et le classificateur sauvegardé, celui-ci peut être utilisé dans la phase de test pour évaluer ses performances en termes de classification des transactions. Si le classificateur présente de bonnes performances, il peut être utilisé pour effectuer des prédictions sur de futures transactions non étiquetées en temps réel.

4.2.1.2.1 L'algorithme de notre classificateur basé sur Elephas

Dans cette sous-section, nous présentons l'algorithme 2 qui permet de classifier les transactions en utilisant des étiquettes telles que "bma-attack", "bsa-attack", "spa-attack", "da-attack", "osa-attack", "ooa-attack" ou "sécurisée". Cet algorithme repose sur l'apprentissage profond supervisé utilisant Spark et est spécifiquement basé sur l'utilisation de la bibliothèque Elephas (voir Figure 4.2). L'algorithme se base sur un ensemble de données d'entraînement préalablement étiquetées pour entraîner notre classificateur.

Tout d'abord, nous initialisons une session Spark à la ligne 1. Ensuite, nous procédons à la création de notre application Spark et à la lecture de la base de

Algorithme 2 : Algorithme du classificateur basé sur Elephas

Require: **Ensemble de transactions étiquetées "T"**
Ensure: **Classificateur entraîné "CDeep"**

▷ **Prétraitement des données : les mêmes étapes du premier algorithme de la ligne 1 à 8**

1: $spark \leftarrow SparkSession.builder()$
2: $df \leftarrow spark.read(T)$
3: $requiredFeatures \leftarrow (rt, ValConf_i, ValConf_j, ver_i, qop_i, qop_j, tendV_i, tendV_j, simU_{ij}, freqV_{ij}, label)$
for *each label in df* **do**
_
4: $classEncoder \leftarrow StringIndexer(inputCol = label, outputCol = newLabel).fit(df)$
5: $df \leftarrow classEncoder.transform(df)$
6: **END for**
7: $df2 \leftarrow df.select(rt, ValConf_i, ValConf_j, ver_i, qop_i, qop_j, tendV_i, tendV_j, simU_{ij}, freqV_{ij}, newlabel)$
8: $vecAssembler \leftarrow VectorAssembler(inputCols = requiredFeatures, outputCol = features)$
9: $vecDF \leftarrow vecAssembler.transform(df2)$
10: $nbclasses \leftarrow vecDF.select("newlabel").distinct().count()$

▷ **Diviser le pipeline en "trainData" et "testData"**

11: $trainData, testData \leftarrow vecDF.randomSplit(0.7, 0.3)$

▷ **Construire le modèle d'apprentissage profond**

12: $inputdim \leftarrow len(trainData.select("features").first()[0])$
13: $DLmodel = Sequential()$
14: $DLmodel.add(Dense(256, inputshape = (inputdim)))$
15: $DLmodel.add(Activation('activationFunction'))$
16: $DLmodel.add(Dropout(0.5))$
17: $DLmodel.add(Dense(256))$
18: $DLmodel.add(Activation('activationFunction'))$
19: $DLmodel.add(Dropout(0.5))$
20: $DLmodel.add(Dense(nbclasses))$
21: $DLmodel.add(Activation('activationFunction'))$
22: $DLmodel.compile(loss =' lossfunction', optimizer =' optimizer')$

▷ **Intégrer le modèle Keras dans Spark**

23: $estimator \leftarrow ElephasEstimator(DLmodel, epochs = epochs, batchsize = batchsize, frequency = batch, mode = synchronous, categorical = True, nbclasses = nbclasses)$
24: $KerasModel \leftarrow Pipeline(stages = [estimator])$

▷ **Entraîner le modèle**

25: $DLpipeline \leftarrow KerasModel.fit(trainData)$

▷ **Évaluer les performances du modèle**

26: $predTrain \leftarrow DLpipeline.transform(testData)$
27: $pnl \leftarrow predTrain.select("label","prediction")$
28: $predictionAndLabel \leftarrow pnl.rdd.map(lambdarow : (row.label, row.prediction))$
29: $metrics \leftarrow MulticlassMetrics(predictionAndLabel)$
30: $print(metrics.fMeasure())$

▷ **Sauvegarder le modèle en un classificateur CDeep**

31: $DLpipeline.save("CDeep")$

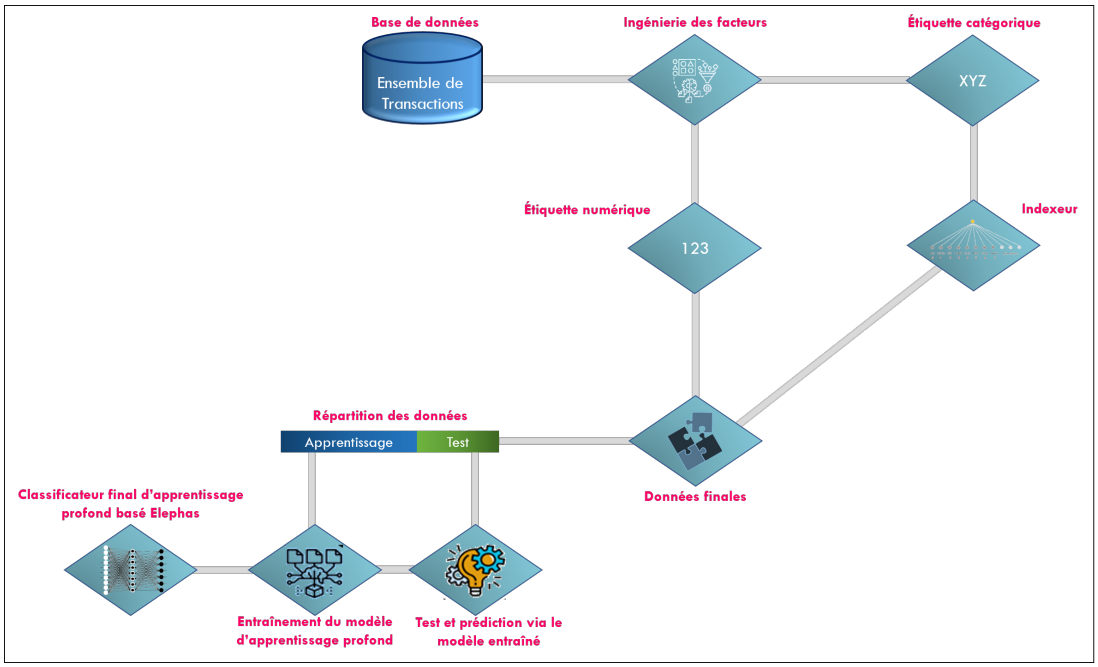

FIGURE 4.2 – *Phase d'agrégation : Création d'un classificateur d'apprentissage profond basé sur Spark Elephas*

données sous forme de DataFrame à l'aide de la bibliothèque Spark SQL dans la ligne 2. Dans le contexte de notre modèle d'apprentissage profond, seules les caractéristiques et les étiquettes numériques sont prises en compte. Ainsi, dans la ligne 3 nous combinons nos caractéristiques en une seule colonne appelée "requiredFeatures" à l'aide d'une fonction fournie par la bibliothèque Spark ML. En ce qui concerne la colonne "label", elle contient sept classes différentes dans notre base de données. Pour les représenter numériquement, dans les lignes (4-6), nous effectuons un encodage qui attribue les valeurs de 0.0 à 6.0 à chacune de ces classes, utilisant une fonction fournie par la bibliothèque Spark ML.

Après cela, de la ligne 7 à ligne 9, nous mettons en place un pipeline pour regrouper toutes ces étapes de transformation. Les étapes du pipeline sont définies dans un ordre séquentiel, ce qui facilite le prétraitement des données. Ensuite, nous sélectionnons uniquement la colonne des caractéristiques et la colonne des étiquettes numériques. Après, dans la ligne 11, afin de garantir la reproductibilité des résultats, nous divisons aléatoirement le pipeline en données d'entraînement et en données de test.

Aux lignes (12-22), nous procédons à la construction du modèle d'apprentissage profond en utilisant Keras en sélectionnant une série de couches denses successives, en utilisant la technique de dropout et en spécifiant la fonction d'activation appropriée. Pour pouvoir intégrer ce modèle Keras dans Spark, nous le convertissons en un "estimateur" en utilisant la classe ElephasEstimator dans la ligne 23. Dans le contexte de Spark, un estimateur est une représentation d'un modèle qui doit encore être entraîné.

Il existe deux méthodes de parallélisme des données. La méthode synchrone de parallélisme des données attend que tous les travailleurs aient terminé le mini-lot actuel avant de passer à l'itération suivante. Cependant, en raison des variations entre les machines, cette méthode ne sera jamais véritablement synchrone, ce qui peut affecter la vitesse d'apprentissage des travailleurs. En revanche, la méthode asynchrone permet aux travailleurs de récupérer la variable centrale et de mettre à jour le serveur de paramètres avec le gradient calculé dès qu'ils sont prêts. Étant donné que nous n'utilisons qu'un seul travailleur, nous optons pour la méthode synchrone, qui traite les tâches selon le principe du Premier Entré, Premier Sorti (PEPS). Nous définissons également d'autres configurations d'entraînement, notamment la création d'un pipeline où nous ajoutons l'estimateur en tant qu'étape du pipeline dans la ligne 24. Ensuite, dans la ligne 25 nous ajustons ce pipeline sur les données d'entraînement "trainData" pour effectuer l'apprentissage. Après avoir appliqué la méthode "fit" sur notre pipeline, nous obtenons un modèle entraîné. Ce dernier peut ensuite être utilisé pour effectuer des prédictions en utilisant la méthode "transform" sur les données de test "testData" afin d'évaluer ses performances comme le montrent les lignes (26-30).

Une fois que nous obtenons des performances satisfaisantes du modèle, il sera sauvegarder comme un classificateur pour une utilisation ultérieure en utilisant la méthode save() de la classe ElephasModel, comme montre la ligne 31.

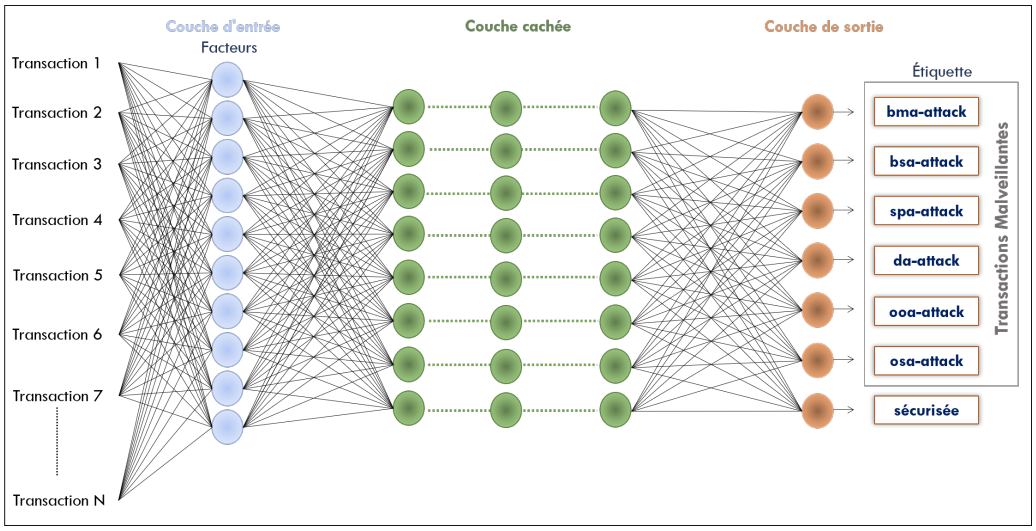

FIGURE 4.3 – *Classificateur d'apprentissage profond*

4.2.2 Module de détection d'attaques en temps réel basé sur Spark Streaming

Dans cette partie, nous allons présenter en détails le fonctionnement général de ce module basé sur Spark Streaming, en mettant en évidence son rôle dans la détection proactive des attaques de confiance.

Dans ce deuxième module, notre objectif est de prédire l'étiquette des transactions générées en temps réel. Pour cela, nous mettons en place un processus

de prédiction continu de flux de données transactionnelles. En effet, en intégrant la détection des attaques dans notre approche, nous créons un système résilient et réactif qui protège activement notre système contre les attaques potentielles.

Tout d'abord, nous créons de nouvelles transactions en utilisant l'ensemble de flux (streaming) qui contient des transactions préparées à l'avance via l'interface de l'application de la blockchain comme le montre la figure 3.4. Ces transactions sont ensuite acheminées vers un nouveau flux de données transactionnelles qui sera analysé en temps réel. Pour cela, nous exploitons la puissance de la bibliothèque Spark Streaming. Cela nous permet de traiter les flux de données transactionnelles de manière efficace et parallèle, en tirant parti du traitement distribué offert par Spark.

Le cœur de notre présent module réside dans l'utilisation du classificateur que nous avons développé dans le premier module. Ce classificateur est entraîné sur un ensemble de transactions étiquetées dans le cadre de l'apprentissage supervisé. Il est capable d'étiqueter les transactions en fonction des modèles d'attaques appris pendant la phase d'apprentissage. Donc, en exploitant la bibliothèque Spark Streaming, nous appliquons le classificateur sur le nouveau flux de transactions pour prédire l'étiquette de chaque transaction en temps réel. Ces prédictions permettent d'identifier et de signaler les transactions potentiellement malveillantes au fur et à mesure qu'elles se produisent, renforçant ainsi la sécurité et la confiance dans l'environnement de l'IoT social.

4.2.2.1 L'algorithme de détection des attaques en temps réel basé sur Spark Streaming

Dans cette section, nous présentons l'algorithme 3 de détection des attaques en temps réel basé sur Spark Streaming. Cet algorithme permet d'analyser en continu les flux de données transactionnelles générées, d'appliquer un modèle de classification préalablement entraîné et de prédire l'étiquette des transactions.

Avant de démarrer le traitement en temps réel, une étape cruciale de prétraitement des données a été réalisée pour assurer la fiabilité des résultats. Dans cette phase, un ensemble de transactions individuelles a été généré, chaque transaction étant sauvegardée dans un fichier distinct. Il convient de souligner que ces transactions n'ont pas été utilisées précédemment lors de la phase d'apprentissage, garantissant ainsi l'intégrité et la validité des données en temps réel.

De plus, afin d'assurer une interprétation correcte des données, un schéma détaillé a été créé. Ce schéma spécifie les noms et les types des facteurs utilisés dans les transactions, garantissant ainsi la correspondance adéquate entre les données lues à partir de la source des transactions générées par la blockchain et les types de données requis pour l'algorithme de détection des attaques en temps réel. Cette étape de prétraitement essentielle assure la cohérence des données et établit des fondations solides pour la phase suivante de l'algorithme.

Après avoir effectué le prétraitement des données, notre algorithme de détection des attaques en temps réel entre dans une nouvelle étape cruciale. Aux lignes

Algorithme 3 : Algorithme de détection des attaques en temps réel basé sur Spark Streaming

Require: **Classificateur entraîné "C"**
Ensure: **Etiquette prédite**

▷ **Créer un StreamingContext local avec un intervalle de 1 seconde**

1: $sc \leftarrow SparkContext(''local[*]'', ''attackDetection'')$
2: $ssc \leftarrow StreamingContext(sc, 1)$

▷ **Charger le classificateur C**

3: $loadedModel \leftarrow ssc.Model.load(C)$

▷ **Créer un DStream qui se connectera à hostname :port "8080" afin de recevoir des flux de transactions de notre application blockchain**

4: $DStream \leftarrow spark.readStream(''localhost'', 8080))$

▷ **Utiliser la fonction de transformation pour faire des prédictions**

5: $streaming \leftarrow loadedModel.transform(DStream).select('label', 'probability', 'prediction')$

▷ **Démarrage du traitement**

6: $ssc.start()$

▷ **Attendre la fin de traitement**

7: $ssc.awaitTermination()$

(1-2), nous créons un contexte Spark Streaming qui nous permet de gérer le flux continu de transactions. Cela nous permettra de traiter les transactions en temps réel et de prendre des décisions instantanées. Ensuite, à la ligne 3, nous chargeons le classificateur "C" que nous avons précédemment créé lors du premier module. Ce classificateur est essentiel pour prédire l'étiquette des transactions et détecter les potentielles attaques de confiance. À la ligne 4, nous utilisons la fonction "readStream" pour lire les flux de transactions provenant de notre application blockchain. Ces flux de données contiennent les transactions récemment générées et sont structurés conformément au schéma que nous avons préalablement défini. Cette étape garantit que les données sont interprétées correctement lors de leur lecture.

Une fois que le flux de transactions est lu, nous procédons à la fonction de transformation à la ligne 5. Cette fonction est responsable d'appliquer le classificateur "C" aux données du flux, ce qui nous permet d'effectuer des prédictions instantanées sur chaque transaction. Grâce à cette méthode proactive, nous pouvons identifier les transactions potentiellement malveillantes, avant même qu'elles ne causent des dommages. Cette capacité de prédiction en temps réel est essentielle pour garantir la sécurité et l'intégrité de notre système. De plus, nous sommes en mesure de prendre des mesures immédiates et appropriées par notre protocole PACS pour contrer et prévenir les attaques détectées dans la phase suivante de notre approche.

Finalement, il est important de noter que lors de l'exécution de ces lignes, Spark Streaming prépare simplement le calcul qu'il effectuera une fois démarré, et aucun traitement réel n'a encore commencé. Pour déclencher le traitement

une fois que toutes les transformations ont été configurées, nous utilisons finalement la méthode "start" sur le contexte Spark Streaming comme l'indique la ligne 6. Cela lance le traitement en temps réel des flux de données, permettant à l'algorithme d'effectuer les prédictions sur chaque transaction à mesure qu'elles arrivent.

4.3 Phases de propagation et de mise à jour

La phase de propagation et de mise à jour est une étape essentielle dans notre mécanisme de gestion de confiance, visant à garantir la sécurité et la fiabilité des transactions dans l'IoT social. Cette phase est conçue pour gérer de manière sécurisée et efficace les scores de confiance générés lors de l'évaluation des transactions en temps réel. En utilisant des stratégies spécifiques, nous veillons à prendre des mesures appropriées pour prévenir les attaques de confiance et de garantir que les scores de confiance soient correctement propagés et mis à jour dans notre système.

Dans cette section, nous détaillerons le module de prévention des attaques en temps réel et les méthodes utilisées pour la propagation et la mise à jour des scores de confiance, en mettant l'accent sur la sécurité, l'efficacité et la reproductibilité de notre approche.

4.3.1 Module de prévention des attaques en temps réel

Après l'analyse en temps réel des flux de transactions, notre protocole de consensus PACS prendra des mesures immédiates et appropriées pour contrer et prévenir les attaques détectées. En fonction des résultats de l'analyse, deux actions sont possibles : valider les transactions sécurisées ou annuler les transactions malveillantes.

Dans le cas des transactions sécurisées, notre protocole génère un nouveau bloc de blockchain en regroupant les transactions vérifiées avec d'autres transactions fiables. Ce nouveau bloc est ensuite ajouté à la blockchain existante de manière permanente et immuable, et la transaction est validée et finalisée. Cela permet d'assurer l'intégrité et la sécurité de la blockchain en incluant uniquement des transactions légitimes et fiables. Cependant, lorsqu'une transaction est identifiée comme malveillante, elle n'est pas validée. Au lieu de cela, elle est ajoutée à une liste noire spécifique qui conserve les traces des transactions suspectes et malveillantes détectées dans le réseau. Cette action permet d'isoler les transactions non sécurisées et d'éviter qu'elles n'affectent l'intégrité globale de la blockchain. En conséquence, ces transactions malveillantes sont annulées et exclues du processus de validation, contribuant ainsi à prévenir les attaques et à maintenir la fiabilité de la blockchain.

Cette approche dynamique et réactive du protocole de consensus PACS garantit que seules les transactions sécurisées sont validées et ajoutées à la blockchain, renforçant ainsi la confiance dans le réseau IoT social. Les transactions

malveillantes sont identifiées, isolées et rejetées, ce qui maintient l'intégrité et la sécurité de la blockchain tout en préservant son caractère immuable.

4.3.2 Méthode de propagation des scores de confiance dans le réseau

Dans cette étape, nous déterminons la méthode de propagation des scores de confiance, obtenus lors de la phase précédente. Nous avons la possibilité d'utiliser une méthode centralisée ou décentralisée, comme mentionné dans le chapitre 1, en fonction des exigences du système et des transactions en cours. Cependant, il est important de souligner les inconvénients de la méthode centralisée. Cette méthode implique la présence d'un nœud central chargé de calculer et de stocker les paramètres de confiance des nœuds participants du réseau. Bien que simple à mettre en place, elle présente des limitations en termes de scalabilité et de faisabilité, notamment lorsque des milliards d'appareils sont impliqués et que de multiples échanges et interactions se produisent chaque seconde. C'est pourquoi la méthode décentralisée est préférable, car elle offre une plus grande extensibilité et permet de résoudre les problèmes inhérents à la méthode centralisée. Elle se réfère à des nœuds qui propagent de manière autonome des informations de confiance aux autres nœuds rencontrés ou avec lesquels ils interagissent, sans avoir besoin d'une entité centrale. C'est ici que l'utilisation de la blockchain se révèle avantageuse.

En optant pour une méthode décentralisée dans notre mécanisme SparkChain, nous évitons le risque que l'ensemble du processus soit contrôlé par une entité potentiellement malveillante. Dans cette approche, tous les nœuds sont responsables du calcul et du partage des données de confiance lors des échanges entre eux, en utilisant un schéma décentralisé fourni par la blockchain. Cette décision est justifiée par le souci d'assurer une confiance accrue et une meilleure sécurité, tout en exploitant les avantages offerts par la technologie de la blockchain.

En utilisant la blockchain, nous bénéficions de plusieurs avantages importants : (i) **la décentralisation** de la blockchain implique que chaque nœud du réseau participe au calcul et à la propagation des scores de confiance, éliminant ainsi la dépendance à l'égard d'une entité centrale et réduisant les risques de manipulation ; (ii) **la transparence et l'immuabilité** de la blockchain permettent de stocker de manière permanente les scores de confiance, garantissant ainsi l'intégrité des informations ; (iii) **la traçabilité** offerte par la blockchain permet de retracer l'historique complet des transactions et des mises à jour des scores de confiance, facilitant ainsi la vérification et la détection des comportements frauduleux ; (iv) **la fiabilité et la résilience** de la blockchain garantissent le bon fonctionnement du système même en cas de défaillance ou d'attaque sur certains nœuds ; (v) **la sécurité renforcée** offerte par la blockchain protège les scores de confiance contre les attaques et les falsifications ; (vi) **l'interopérabilité** de la blockchain permet l'échange de données de confiance entre différentes applications et systèmes, favorisant ainsi la collaboration et l'utilisation étendue des scores de confiance, etc.

Ainsi, en choisissant une méthode de propagation décentralisée basée sur la blockchain dans notre mécanisme SparkChain, nous bénéficions d'une infrastructure solide, sécurisée et transparente, garantissant l'intégrité des données et renforçant la confiance dans l'IoT social.

4.3.3 Méthode de mise à jour en temps réel du niveau de confiance

Dans cette sous-section, nous allons exposer la méthode de mise à jour de la confiance dans le réseau IoT social. Comme mentionné précédemment dans le chapitre 1, il existe deux approches générales pour mettre à jour la confiance : l'approche basée sur le temps et l'approche basée sur les événements.

L'approche basée sur le temps pose des difficultés, voire même peut s'avérer impossible, lorsqu'il s'agit de définir la granularité de la période de mise à jour. En revanche, l'approche basée sur les événements est plus adaptée à un système aussi imprévisible que l'IoT social. Cependant, en raison de la taille et de la dynamique de ce réseau, l'approche basée sur les événements peut être coûteuse pour les systèmes de propagation centralisés et décentralisés.

C'est pourquoi notre mécanisme de gestion de confiance, SparkChain, propose une méthode de mise à jour en temps réel. Cette approche permet de mettre à jour les scores de confiance en temps réel à la suite de chaque transaction générée dans le réseau. Ainsi, elle garantit que les scores de confiance sont constamment actualisés et reflètent les changements dynamiques dans le réseau IoT social. Grâce à cela, les nœuds peuvent prendre des décisions basées sur des informations de confiance à jour, ce qui renforce la fiabilité et la sécurité du système dans un environnement en constante évolution.

L'utilisation du temps réel dans la mise à jour de la confiance dans le réseau IoT social présente plusieurs avantages. Tout d'abord, cela permet une actualisation constante des scores de confiance, fournissant ainsi des informations de confiance actualisées en temps réel et en phase avec les changements dynamiques du réseau. En outre, l'utilisation du temps réel renforce la fiabilité du système en évitant les problèmes liés à l'utilisation de méthodes basées sur le temps ou les événements qui pourraient entraîner des retards ou des incohérences dans la mise à jour de la confiance. Enfin, les utilisateurs bénéficient d'une expérience améliorée grâce à la visibilité immédiate sur la confiance dans le réseau, leur permettant de prendre des décisions éclairées et de participer à des interactions sécurisées avec des utilisateurs de confiance. Vu qu'ils disposent d'informations précises sur la fiabilité et la crédibilité des utilisateurs dans le réseau, ce qui facilite la sélection des utilisateurs de confiance et la participation à des interactions sécurisées.

En somme, l'utilisation du temps réel dans la mise à jour de la confiance dans le réseau IoT social confère une expérience utilisateur plus enrichissante et favorise la confiance dans ce réseau.

4.4 Un scénario illustratif de prévention des attaques de confiance dans l'IoT social basé sur blockchain

Nous présenterons dans cette partie, un scénario illustratif dans la figure 4.4 qui met en évidence les mesures de prévention des attaques de confiance dans l'IoT social grâce à l'utilisation de la technologie de la blockchain. L'IoT social constitue un environnement dynamique où les appareils interconnectés et les utilisateurs collaborent au sein d'un réseau distribué. Malheureusement, certains interactions peuvent être des attaques malveillantes qui compromettent la confiance entre les parties impliquées. Cependant, à travers notre scénario, nous explorerons comment les mécanismes de confiance basés sur la blockchain sont en mesure de détecter et de prévenir ces attaques, garantissant ainsi l'intégrité et la sécurité des interactions au sein de l'IoT social.

Dans notre scénario, nous avons deux utilisateurs dans le réseau IoT social basé sur la blockchain : Jack et Oscar. Chacun d'eux possède un profil, des dispositifs connectés et offre des services. Jack décide d'effectuer une transaction spécifique et souhaite également donner un vote au service d'Oscar. Il soumet sa demande de transaction au réseau, ce qui entraîne la création et l'enregistrement de la transaction dans le système, en associant les informations de Jack à celles d'Oscar. Le système prend en compte les valeurs de confiance des utilisateurs, la similarité entre les profils et la qualité des services de chaque utilisateur pour évaluer la confiance et la fiabilité de la transaction. Ensuite, la transaction est soumise à notre protocole de consensus PACS (Preuve d'attaque de confiance basée sur Spark), qui analyse en profondeur les données associées à la transaction. Ce protocole utilise des algorithmes avancés basés sur Apache Spark pour vérifier la légitimité de la transaction et détecter toute activité malveillante potentielle. En fonction des résultats de l'analyse effectuée par notre protocole PACS, deux scénarios sont possibles :

Scénario 1 : Le protocole PACS confirme la sécurité de la transaction, ce qui entraîne sa validation et son ajout, avec d'autres transactions vérifiées, dans un nouveau bloc. Ce bloc est ensuite intégré à la blockchain existante, garantissant ainsi l'intégrité et la sécurité des données. Par conséquent, la transaction est considérée comme terminée et le service d'Oscar reçoit le vote de Jack. Les informations sur la transaction, ainsi que le score de confiance calculé à la suite de celle-ci, sont propagées de manière décentralisée dans le réseau grâce à l'utilisation de la blockchain dans l'IoT social. Ces informations sont également mises à jour en temps réel. Ce scénario 1 illustre la validation réussie d'une transaction sécurisée grâce au protocole PACS.

Scénario 2 : Le protocole PACS détecte des éléments de risque dans la transaction, tels qu'une identité douteuse ou une tentative d'attaque. Par conséquent, la transaction est identifiée comme non valide et malveillante où elle présente des caractéristiques suspectes. Dans cette situation, elle est répertoriée dans une liste noire spécifique qui conserve les traces des transactions malveillantes. Par conséquent, elle ne peut pas être finalisée ni considérée comme valide. De plus, des

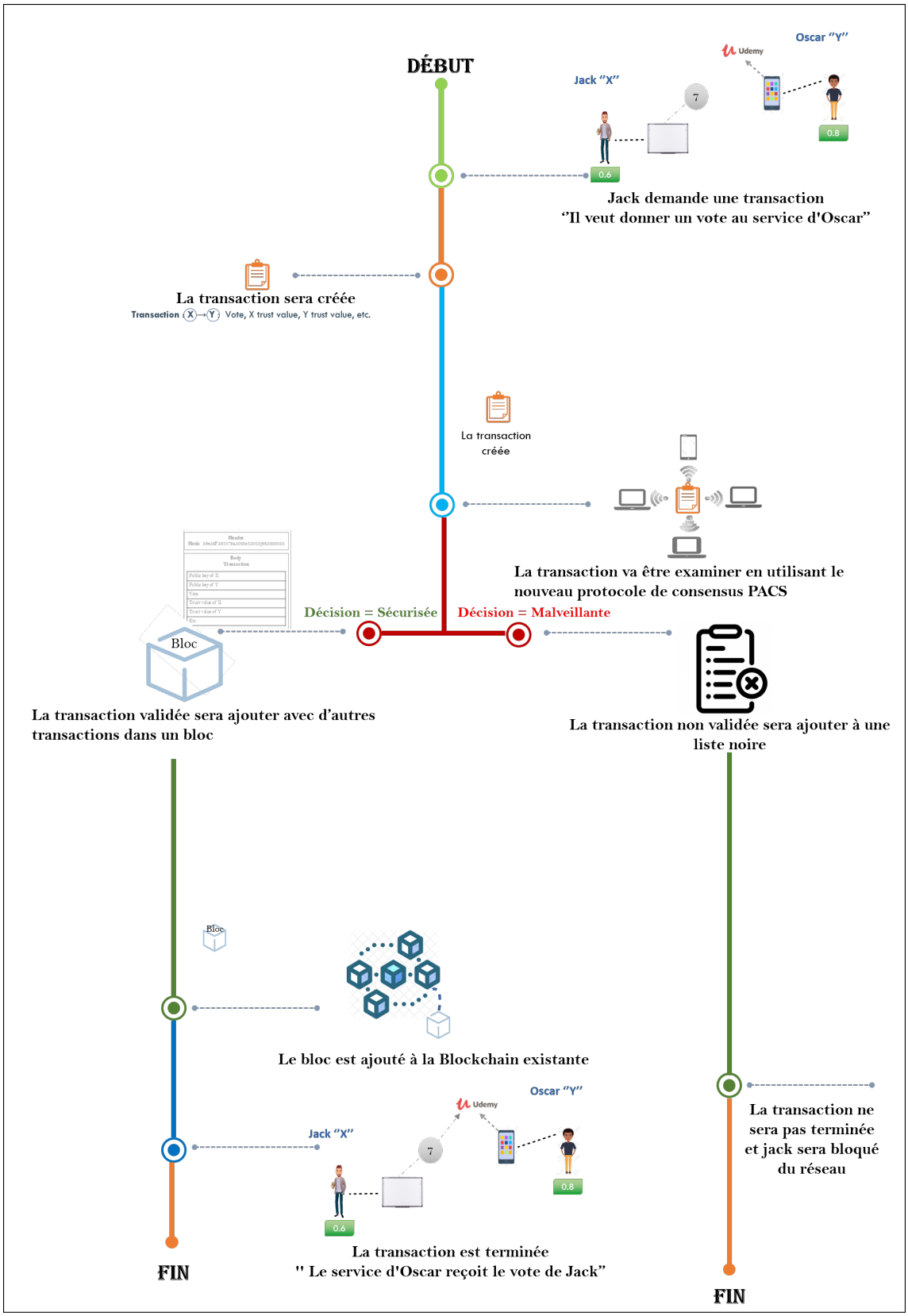

FIGURE 4.4 – *Un scénario illustratif de prévention des attaques de confiance dans l'IoT social basé sur Blockchain*

mesures de précaution sont prises pour garantir la sécurité du réseau. Jack étant impliqué dans une transaction malveillante, est bloqué et ne peut plus interagir dans le réseau. Cette action permet de prévenir toute autre interaction potentiellement dangereuse et de protéger l'intégrité du réseau et des utilisateurs.

Grâce à ce scénario illustratif, nous pouvons apprécier comment les mécanismes de confiance basés sur la technologie de la blockchain, associés au protocole PACS, prennent en compte les profils et les services des utilisateurs pour évaluer et prévenir les attaques de confiance dans l'IoT social, renforçant ainsi la sécurité et la fiabilité du réseau.

4.5 Conclusion

Tout au long de ce chapitre, nous avons présenté les différentes phases et les modules de notre mécanisme de gestion de confiance appelé "SparkChain", conçu pour prévenir en temps réel les différentes attaques de confiance dans l'IoT social.

SparkChain est composé de quatre phases distinctes : la phase de composition, la phase d'agrégation, la phase de propagation et la phase de mise à jour. Dans la phase de composition, nous avons introduit de nouveaux facteurs de confiance et amélioré ceux proposés par notre équipe de recherche. Ces facteurs sont utilisés pour évaluer la confiance associée à chaque transaction. Dans la phase d'agrégation, nous avons combiné les différents facteurs de confiance afin de générer un score de confiance pour chaque transaction en temps réel. Ce score est ensuite utilisé par notre protocole PACS pour prendre des décisions sur la validité des transactions. Le calcul des scores de confiance est effectué en utilisant les techniques d'apprentissage automatique et profond de Spark, tandis que l'analyse en temps réel des transactions est réalisée à l'aide de la bibliothèque Spark Streaming et en exploitant les techniques d'apprentissage également. Les phases de propagation et de mise à jour décrivent en détail les stratégies que nous avons mises en place pour gérer de manière sécurisée et efficace les transactions et les scores de confiance. Ces stratégies garantissent la diffusion appropriée des transactions et scores et leur mise à jour régulière pour maintenir la précision du mécanisme de gestion de confiance. En résumé, SparkChain constitue une approche complète pour prévenir les attaques de confiance dans l'IoT social, en utilisant des techniques avancées d'apprentissage automatique et profond, ainsi que des mécanismes de propagation et de mise à jour bien conçus pour garantir la fiabilité et la sécurité du système.

Dans le prochain chapitre, nous aborderons l'implémentation concrète de notre approche, en détaillant les différentes étapes de mise en œuvre. Nous examinerons également plusieurs cas de simulation pour évaluer la pertinence et l'efficacité de notre proposition. Cette évaluation nous permettra d'analyser les résultats obtenus et de déterminer dans quelle mesure notre approche répond aux exigences et aux objectifs fixés.

Évaluation de SparkChain 5

Sommaire

5.1 Introduction

Dans ce chapitre, nous présentons en détail l'implémentation de notre approche SparkChain novatrice basée sur la blockchain et Apache Spark pour prévenir les attaques de confiance en temps réel dans l'IoT social, ainsi que les résultats obtenus pour valider notre proposition.

Dans la première partie, nous allons décrire l'environnement de travail logiciel que nous avons utilisé pour mettre en place notre système. Nous expliquerons également les différentes étapes de la mise en œuvre de notre solution, en détaillant les technologies utilisées et les choix réalisés. Dans la deuxième partie, nous décrirons la base de données utilisée pour notre étude, en expliquant comment nous avons collecté et préparé les données nécessaires pour notre analyse. La troisième partie sera consacrée à la présentation de l'implémentation pratique de la blockchain, les classificateurs basés sur Spark et les différentes interfaces graphiques utilisées. Nous montrerons comment ces interfaces graphiques permettent aux utilisateurs de visualiser et de surveiller en temps réel les transactions dans le réseau. Ensuite, dans la quatrième partie, nous exposerons les métriques d'évaluation que nous avons utilisées pour mesurer la performance de notre approche. Nous expliquerons comment nous avons choisi ces métriques et comment nous les avons utilisées pour évaluer notre approche. La cinquième partie sera consacrée à la présentation des résultats de l'évaluation, en détaillant les performances de notre système par rapport aux travaux de la littérature. Nous montrerons comment notre approche SparkChain est plus efficace que les approches existantes pour prévenir les attaques de confiance en temps réel dans l'IoT social. Enfin, nous conclurons ce chapitre par une synthèse des résultats obtenus et une discussion sur les perspectives d'amélioration de notre approche. Nous soulignerons les limites de notre étude et les directions futures que nous envisageons pour améliorer encore la performance et l'efficacité de notre approche SparkChain.

5.2 Environnement de travail

Après avoir décrit les concepts de notre système, nous passons maintenant à sa mise en œuvre concrète. Dans cette section, nous présentons l'implémentation de notre système, qui a été réalisée à l'aide du langage de programmation Python, en utilisant l'environnement de développement PyCharm. Nous détaillons également les différentes bibliothèques que nous avons exploitées pour la mise en place de notre système.

5.2.1 Environnement logiciel

Après avoir présenté les moyens matériels dans la partie précédente, nous nous concentrerons ici sur les différentes technologies logicielles utilisées pour la mise en œuvre de notre solution innovante.

5.2.2.1 Langage de programmation

Nous avons choisi de mettre en place notre méthode en utilisant le langage de programmation Python, pour sa grande flexibilité, sa simplicité d'utilisation et ses bibliothèques spécialisées très riches. Ces bibliothèques nous ont permis de développer notre système rapidement et efficacement, en utilisant Pandas, NumPy, Scikit-learn, TensorFlow, PySpark et Flask pour la gestion de notre blockchain. Elles nous ont permis d'implémenter facilement les différentes étapes de notre méthode, de la préparation des données à la création de la blockchain, en passant par l'apprentissage automatique et l'analyse des résultats. Nous avons également pu profiter de la grande communauté Python pour résoudre les problèmes éventuels rencontrés lors de la mise en œuvre de notre méthode.

5.2.2.2 Outils de développement

Dans cette sous-section, nous exposons les différentes ressources et outils que nous avons utilisés pour la création de notre approche SparkChain.

1. **Pycharm** : PyCharm est un environnement de développement intégré (EDI ou IDE – Integreted Development Environment). Il est spécifique au langage Python et a été développé par la société JetBrains. C'est une multiplateforme qui regroupe un éditeur de texte, un compilateur et un débogueur Python intégré. De plus, il facilite la navigation entre les projets et le code grâce à une vue de structure de fichier et saut rapide entre fichiers.
2. **Framework Spark** : Pour mettre en œuvre notre approche, nous avons choisi d'utiliser la version 3.0.1 de Spark. Cette version est plus avancée et propose des fonctionnalités étendues par rapport aux versions précédentes, telles que les versions "1.x" et "2.x". Elle offre ainsi une puissance accrue pour le traitement de données à grande échelle. Dans notre cas, nous avons opté pour l'utilisation de l'interface PySpark, qui permet de développer des applications Spark. Cette interface présente plusieurs avantages, notamment la facilité d'utilisation et la familiarité du langage de programmation Python. Elle offre également une prise en charge complète des fonctionnalités et des bibliothèques de Spark, telles que Spark SQL, Streaming et ML.
3. **Anaconda Navigator** : Anaconda est un gestionnaire d'environnement virtuel des langages de programmation Python et R pour les applications liées à l'apprentissage automatique. Il vise à simplifier la gestion et le déploiement des librairies.
4. **Tensorflow** : Tensorflow est un outil open source le plus utilisé dans le domaine de l'apprentissage automatique. Il permet de créer et d'entrainer les modèles d'apprentissage profond. En effet, il permet de créer les différentes couches des réseaux de neurones profonds.
5. **Interface Spark Web UI** : est une console Web qui permet de surveiller et d'analyser l'état et l'utilisation des ressources d'une application Spark à partir d'un navigateur Web. Lorsque nous lançons SparkContext, une instance de l'interface Web est automatiquement démarrée et est accessible

par défaut à l'adresse http ://[driver] :4040. De plus, elle offre plusieurs fonctionnalités et paramètres qui facilitent le suivi et la gestion des jobs Spark. Elle affiche l'état, la durée et la progression de chaque job, ainsi que la chronologie globale des événements. Elle permet également de visualiser les détails spécifiques de chaque job, y compris les étapes qui le composent. En outre, elle possède un onglet dédié aux étapes et permet de consulter l'état actuel de toutes les étapes de tous les travaux, qu'ils soient actifs, en attente, terminés, ignorés ou en échec.

6. **Flask** : est un framework Web léger et flexible en Python qui permet de créer des applications Web. Il peut être utilisé comme outil d'interaction avec la blockchain en utilisant des requêtes HTTP. Lorsqu'il est utilisé en conjonction avec la blockchain, Flask peut être utilisé pour développer des interfaces utilisateur ou des API permettant aux utilisateurs d'interagir avec la blockchain via des requêtes HTTP. Par exemple, il est possible de créer des endpoints (points d'extrémité) qui autorisent les utilisateurs à envoyer des transactions, à récupérer des informations sur les blocs ou les transactions, ou à effectuer d'autres actions liées à la blockchain.

5.2.2.3 Librairies

Dans cette section, nous examinerons plusieurs librairies qui jouent un rôle essentiel dans la mise en œuvre de notre approche.

1. **Pandas** : La bibliothèque Pandas, écrite pour le langage de programmation Python, s'est avérée être un choix judicieux pour la manipulation et l'analyse des données dans notre étude. Cette bibliothèque offre une grande variété d'outils pour la lecture et l'écriture de données structurées, ainsi que des outils de manipulation et d'analyse des données pour traiter des volumes de données importants. Parmi les différents formats de fichiers supportés par Pandas, nous avons utilisé des fichiers CSV, des fichiers textuels et des fichiers du tableur Microsoft Excel pour stocker et accéder à nos données de test et de validation.
2. **Scikit-learn** : Scikit-learn est une bibliothèque open source dédiée à l'apprentissage automatique pour le langage de programmation Python. Elle offre une grande variété d'algorithmes d'apprentissage automatique, de prétraitement de données, de validation de modèle et de mesure de performance. Elle permet également la mise en place de pipelines de traitement de données et d'apprentissage automatique. Scikit-learn est une bibliothèque très utilisée dans le domaine de l'apprentissage automatique en raison de sa simplicité d'utilisation, de sa documentation complète et de sa grande communauté de contributeurs actifs.
3. **Keras** : est une bibliothéque open source écrite par le langage Python. Elle permet d'interagir avec les outils et les algorithmes de réseaux de neurones profonds et de l'apprentissage automatique, notamment Tensorflow. Elle facilite le développement des modèles d'apprentissage profond. Keras est un choix optimal pour notre approche. Il sera facile d'y construire un modèle de réseau neuronal en seulement quelques lignes de code. De plus, nous

utilisons un estimateur basé sur Keras appelé Elephas afin de construire un classificateur.

4. **Elephas** : Elephas est une extension de Keras qui nous permet d'exécuter des modèles d'apprentissage profond distribués à l'échelle en utilisant Spark. Il vise à maintenir la simplicité et la haute disponibilité de Keras, permettant le prototypage rapide de modèles distribués qui peuvent fonctionner sur de grands ensembles de données. Elephas met en œuvre une classe d'algorithmes parallèles aux données sur Keras en utilisant les RDD et les data-frames de Spark. Nous effectuons dans notre étude la classification en testant un modèle d'apprentissage basé sur Spark et utilisé pour le streaming, qui n'a pas été déployé auparavant dans la littérature.

5.2.2 Base de données utilisée

Avant de passer à l'étape d'implémentation de notre approche, une préparation minutieuse des données est nécessaire. Cette préparation englobe la collecte d'un ensemble de données représentatif, incluant un grand nombre de transactions, dans le but d'obtenir des modèles aptes à distinguer les attaques de confiance des transactions sécurisées. Ces modèles seront obtenus grâce à de nombreuses expériences sur les données. Une fois que nous aurons obtenu des résultats satisfaisants et fiables dans la détection des attaques de confiance dans le réseau IoT social basé sur la blockchain, nous serons en mesure de créer un classificateur. Mais avant d'en arriver là, nous devons nous concentrer sur la base de données utilisée.

Étant donné que les données réelles sont souvent indisponibles dans la plupart des travaux de recherche, de nombreux travaux se basent sur des simulations pour évaluer les performances de leurs modèles. Dans notre étude, nous avons également utilisé des simulations pour évaluer les performances de notre approche. Cependant, nous avons eu la chance d'avoir accès à une base de données réelle appelée "Sigcomm" [1], sur laquelle nous avons appliqué nos simulations pour générer des transactions et des interactions entre les utilisateurs dans le réseau IoT social basé sur la blockchain. Cela nous a permis d'évaluer les performances de notre approche dans des conditions proches de la réalité et d'obtenir des résultats pertinents et fiables dans la détection et la prévention des attaques de confiance en temps réel.

La base de données "Sigcomm" est une ressource riche en informations sur les utilisateurs, leurs profils et leurs intérêts. Elle contient également des données sur les relations sociales entre les utilisateurs, leurs interactions et les fréquences de proximité entre chaque paire d'utilisateurs. Pour chaque utilisateur, nous générons un ou plusieurs appareils qui sont associés à son profil. Le tableau 5.1 fournit des statistiques détaillées sur l'ensemble de données que nous avons obtenu à partir de la base "Sigcomm".

En utilisant l'ensemble de données que nous avons obtenu, nous avons procédé à des simulations afin de générer différents formes de transactions, qu'elles

1. https ://crawdad.org/thlab/sigcomm2009/20120715/

TABLE 5.1 – *Statistiques sur la base de données réelles "Sigcomm"*

Utilisateurs	75
Intérêts des utilisateurs	711
Relations sociales entre les utilisateurs	531
Interactions entre les utilisateurs	32000
Appareils	300
Services	364
Proximités	285788

soient sécurisées ou malveillantes, représentant ainsi les différentes types d'attaques décrites précédemment. Ces simulations nous ont permis d'évaluer les performances de notre approche et de mesurer son efficacité dans la détection et la prévention de ces attaques dans le réseau IoT social.

Prenons l'exemple des attaques de type SPA. Dans ce scénario, un utilisateur malveillant u_i, possède deux appareils distincts : un smartphone et une tablette intelligente. L'objectif de u_i est d'améliorer sa propre valeur de confiance en donnant des votes élevés à son propre service. En visualisant la figure illustrative 5.1, nous pouvons observer cette tentative d'attaque de confiance. u_i ne se contente pas d'évoquer le service fourni par son smartphone, mais il attribue également des votes élevés, tels que 9 ou 10, en utilisant sa tablette intelligente. Cette stratégie de l'attaquant vise à créer une image trompeuse de la qualité de son service, dans le but de tromper le système de confiance et d'inciter les futurs demandeurs de service à choisir son service.

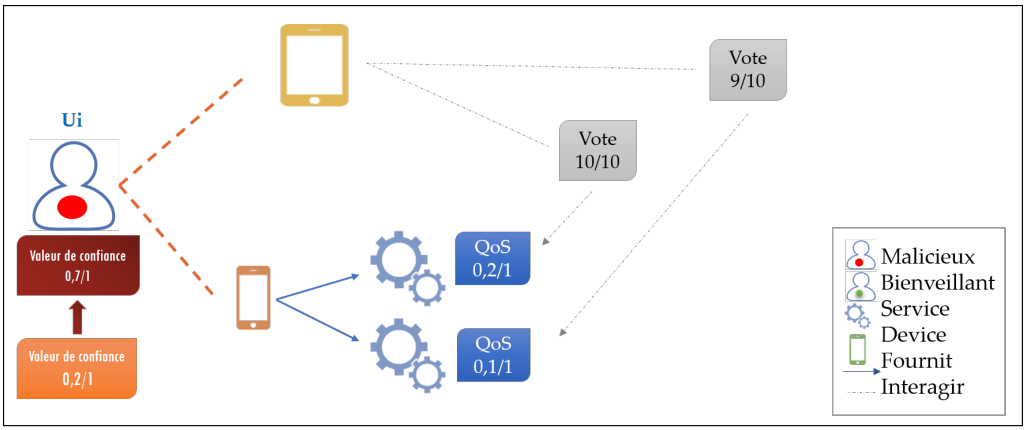

FIGURE 5.1 – *Exemple des attaques de type SPA*

Une fois les simulations terminées, nous avons créé un fichier au format CSV, qui est l'un des formats les plus couramment utilisés pour l'apprentissage automatique et profond. Ce fichier contient un total de 6590 instances, qui correspondent aux transactions collectées lors des simulations. Chaque transaction est associée à une étiquette indiquant son type, qu'il s'agisse d'une transaction sécurisée ou d'une transaction malveillante avec leur type (SPA, BMA, BSA, DA, OSA ou OOA). Le tableau 5.2 présente des statistiques détaillées de ce fichier CSV. Ces statistiques fournissent des informations précieuses sur la répartition des différentes attaques au sein de notre ensemble de transactions.

Table 5.2 – *Ensemble de transactions*

Type d'attaque	Nombre de transactions
BMA	150
BSA	150
SPA	145
DA	110
OSA	310
OOA	165
Sécurisée	5560

5.3 Implémentation

Après avoir présenté la base de données en détail, ainsi que les différents outils et bibliothèques utilisés, nous nous concentrons maintenant sur les paramètres essentiels pris en compte lors de l'implémentation de la blockchain, la création des interfaces graphiques et des classificateurs basés sur Spark.

5.3.1 Implémentation de la blockchain

Cette sous-section présente les résultats clés de l'expérience de simulation visant à évaluer l'efficacité de la blockchain dans la prévention des attaques de confiance. Pendant la phase de mise en œuvre, nous avons utilisé divers outils et ressources, notamment l'éditeur PyCharm pour le développement basé sur le langage de programmation Python, ainsi que le navigateur Anaconda pour gérer les bibliothèques requises telles que Crypto, SHA, etc. De plus, nous avons exploité le framework Flask pour faciliter l'interaction avec la blockchain via des requêtes GET et POST. Ces outils ont contribué à la réalisation de l'implémentation et à la mise en place des fonctionnalités nécessaires pour mener à bien notre expérience de simulation.

1. **Création de paires de clés pour l'expéditeur et le destinataire** :
   Nous avons procédé à la génération d'une adresse publique, également connue sous le nom de "clé publique", pour l'expéditeur. Cette clé est représentée par une séquence de caractères alphanumériques, par exemple : 30819f300d06092a864886f70d010101050003818d0030818902818100a93119a... Parallèlement, nous avons également généré une autre séquence de caractères appelée "clé privée". Cette clé est utilisée pour signer et sécuriser la transaction. Pour permettre la réception de la transaction, nous avons associé une clé publique au destinataire. La figure 5.2 présente l'interface utilisée pour générer ces clés publiques et privées.
2. **Génération de transactions :**
   Nous avons inclus les éléments essentiels de la transaction tels que la valeur du vote, les valeurs de confiance, la similarité des votes, et d'autres éléments décrits dans la phase de composition de notre approche SparkChain proposée. Ensuite, nous avons procédé à la création de la transaction.

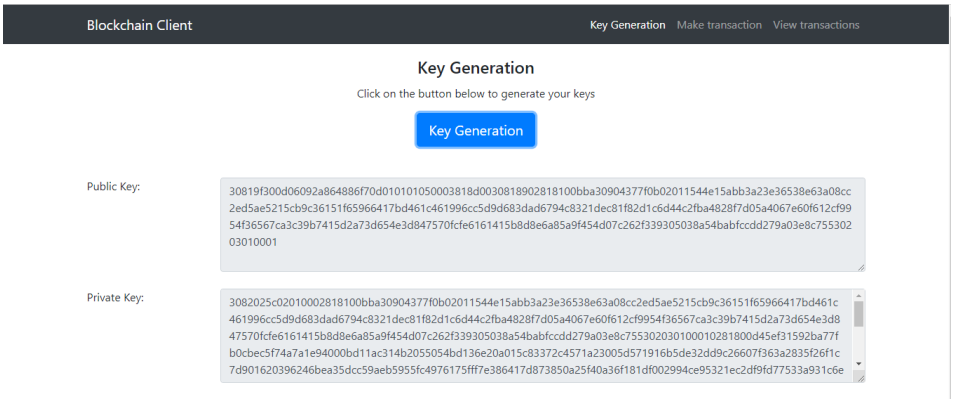

FIGURE 5.2 – *Exemple de clés générées*

L'interface du formulaire de génération de la transaction est illustrée dans la Figure 5.3. Une fois les données nécessaires pour la nouvelle transaction saisies, un formulaire de confirmation apparaît, comme présenté dans la figure 5.4.

3. **Vérification des transactions**
   Nous avons soumis cette nouvelle transaction à notre protocole de consensus PACS nouvellement développé. Ce protocole est conçu pour vérifier l'intégrité et la fiabilité des transactions en examinant leur nature (attaque ou non) et en identifiant les types d'attaques potentiels qu'elles pourraient représenter. Sur la base des résultats de cette vérification, PACS prendra une décision finale quant à la validité de la transaction. Si la transaction est jugée valide, elle sera regroupée avec d'autres transactions vérifiées pour former un nouveau bloc qui sera ensuite ajouté à la blockchain existante. En revanche, si la transaction est identifiée comme suspecte, elle sera ajoutée à une liste noire pour maintenir une trace des transactions malveillantes. L'interface de validation des transactions est présentée dans la figure 5.5, où les utilisateurs peuvent soumettre leurs transactions et déclencher le processus de vérification par le protocole PACS.

4. **Prise de décision**

   (a) **Cas d'une transaction vérifiée : Rassembler les transactions vérifiées pour former un nouveau bloc :**
   Après avoir passé avec succès la vérification, la transaction est prête à être intégrée dans un nouveau bloc. Cependant, dans notre approche, un simple bloc ne ne serait pas suffisant pour atteindre notre objectif. Nous proposons une structure de blocs améliorée. Chaque bloc nouvellement créé contient non seulement la transaction vérifiée, mais aussi une collection d'autres transactions vérifiées, formant ainsi un ensemble cohérent de données. Cela permet d'atteindre un niveau supérieur de sécurité. Pour assurer l'intégrité et la traçabilité des blocs, chaque nouveau bloc est accompagné d'un résumé cryptographique appelé hachage. Ce hachage est calculé à l'aide d'un algorithme de hachage qui prend en compte les données du bloc précédent, créant

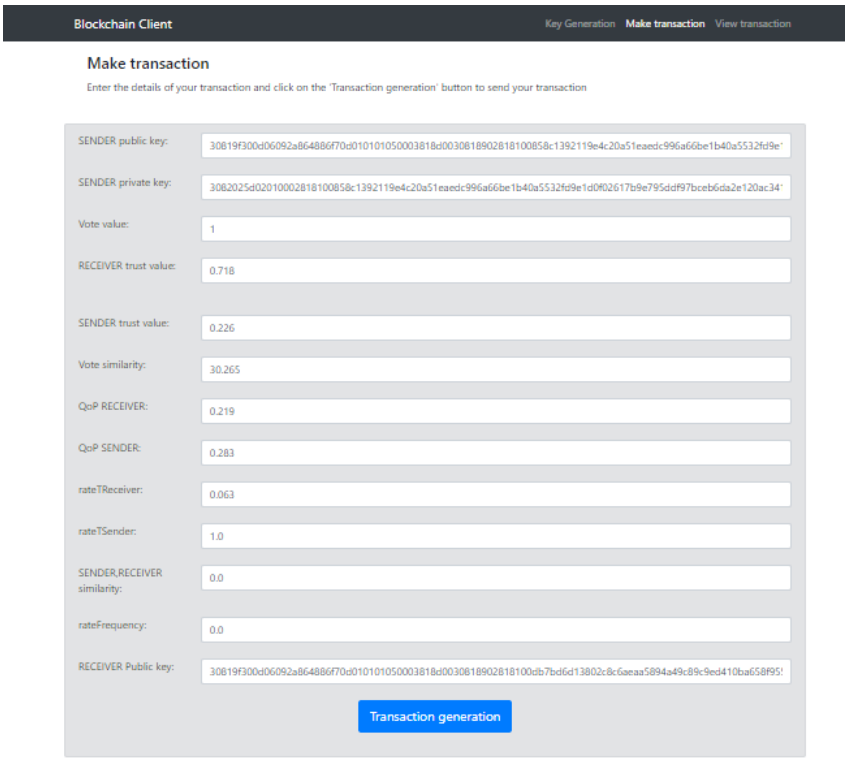

FIGURE 5.3 – *Formulaire de génération de transaction*

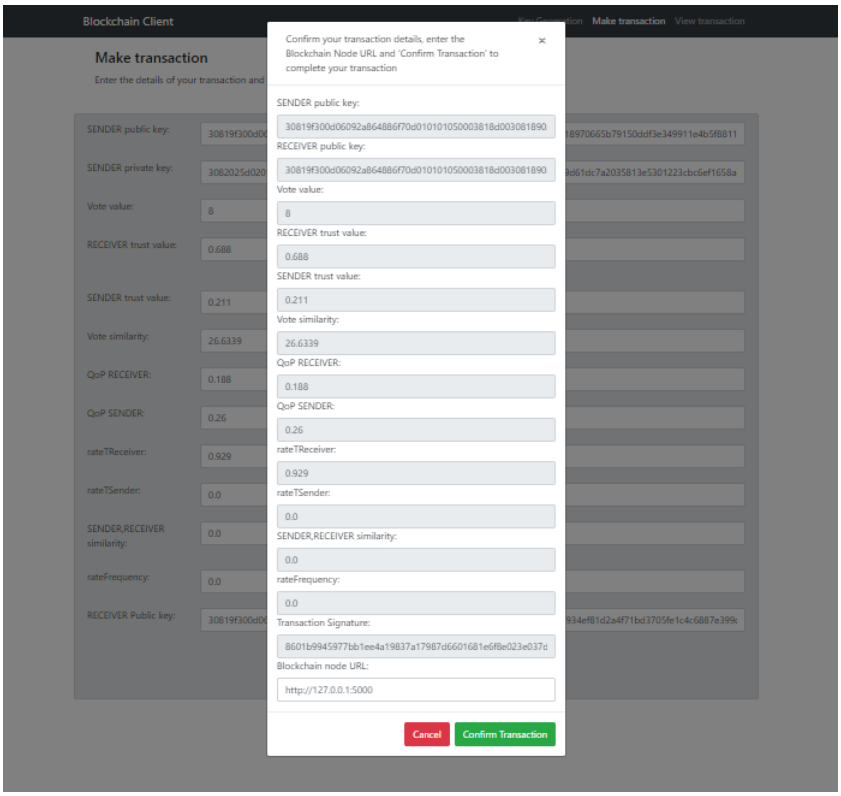

FIGURE 5.4 – *Formulaire de confirmation de la transaction*

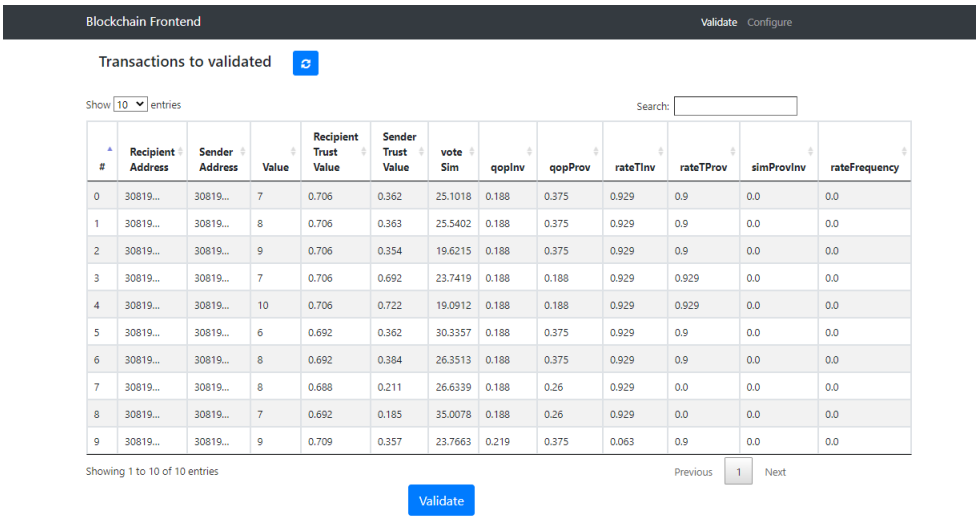

FIGURE 5.5 – *Interface de validation des transactions*

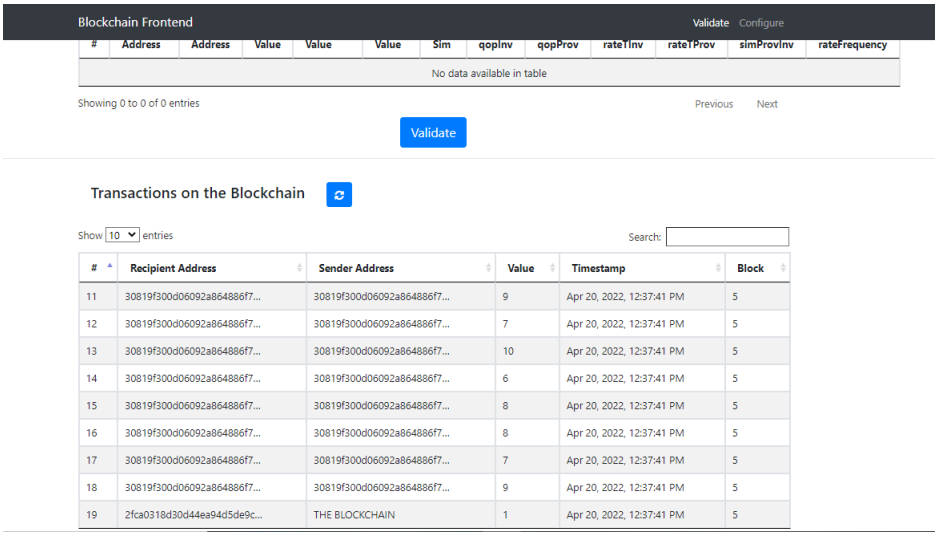

FIGURE 5.6 – *Transactions vérifiées ajoutée à la blockchain*

ainsi une chaîne immuable de blocs liés entre eux. Cette structure en chaîne garantit que toute modification ou altération d'un bloc précédent se répercutera sur tous les blocs suivants, ce qui la rend extrêmement résistante à la falsification des données. Une fois que le nouveau bloc est formé avec toutes les transactions vérifiées et le hachage correspondant, il est ajouté à la blockchain existante, comme illustré dans la figure 5.6. Cette opération renforce la sécurité globale du système en assurant la cohérence et l'intégrité de toutes les transactions précédentes. En ajoutant régulièrement de nouveaux blocs à la blockchain, le système se construit une base de confiance solide, où chaque transaction est vérifiée et enregistrée de manière permanente, établissant ainsi une source fiable de données pour les utilisateurs et les applications de l'IoT social.

(b) **Cas de transaction non vérifiée (attaque de confiance) : Transactions malveillantes enregistrées dans la liste noire :**
Dans le cas où la transaction est identifiée comme étant une attaque, elle est immédiatement interrompue, annulée et ajoutée à une liste noire dédiée. Cette liste noire est spécifiquement conçue pour conserver une trace des transactions malveillantes détectées dans le réseau. En ajoutant la transaction à cette liste, nous assurons que toutes les informations pertinentes concernant les activités malveillantes sont enregistrées et conservées de manière sécurisée. En effet, cette liste joue un rôle essentiel dans la prévention des attaques futures, car elle permet d'identifier et de signaler rapidement les transactions suspectes. Elle constitue une ressource précieuse pour les mécanismes de sécurité et les protocoles de confiance du système, leur permettant de prendre des mesures préventives appropriées pour contrer les attaques potentielles.

5.3.2 Implémentation du classificateur de l'apprentissage automatique basé sur Spark : MLlib

Pour commencer, nous avons utilisé l'outil Weka afin d'explorer différentes algorithmes d'apprentissage supervisé. Nous avons testé diverses algorithmes, dont la table de décision, le SVM, le perceptron, la Régression Linéaire (RL), et d'autres encore. Suite à une analyse approfondie des résultats obtenus, nous avons décidé d'opter pour l'algorithme de Régression Linéaire (RL) en raison de ses performances supérieures dans notre contexte. Le choix de la Régression Linéaire s'est avéré judicieux pour plusieurs raisons. Tout d'abord, notre base de données semblait présenter une relation linéaire entre les variables d'entrée et de sortie. En utilisant RL, nous pouvions modéliser cette relation de manière précise, ce qui nous a permis d'obtenir des prédictions plus fiables. De plus, la Régression Linéaire est un modèle relativement simple et interprétable, ce qui facilite la compréhension de ses décisions par les parties prenantes. Cela est particulièrement important dans notre domaine où la transparence du modèle est essentielle. Enfin, la Régression Linéaire est un algorithme bien établi avec de nombreuses ressources disponibles pour l'optimisation et la mise en œuvre. Nous avons pu exploiter ces ressources pour obtenir des performances optimales sur notre base de données.

De ce fait, nous avons mis en pratique l'algorithme de RL en utilisant le framework Spark à l'aide de l'outil PyCharm, qui est basé sur le langage de programmation Python. PyCharm nous a fourni un environnement de développement convivial pour construire notre classificateur. Pour gérer efficacement les bibliothèques nécessaires, notamment pandas, pyspark, et bien d'autres, nous avons utilisé le navigateur Anaconda.

Après avoir entraîné l'algorithme conformément aux instructions du chapitre 4 sur notre ensemble de données, nous avons créé le classificateur. Celui-ci a été sauvegardé avec l'extension de fichier ".sav" pour pouvoir être réutilisé ultérieurement. Cette étape de sauvegarde nous offre la possibilité de recharger aisément

le classificateur et de l'utiliser pour analyser de nouvelles transactions en temps réel, en prédisant avec précision si elles sont malveillantes ou sécurisées. Ce processus contribue ainsi à la prévention des attaques de confiance au sein du réseau IoT social.

5.3.3 Implémentation du classificateur de l'apprentissage profond basé sur Spark : Elephas

Comme nous avons exploré précédemment dans le chapitre 4, le modèle d'apprentissage profond nécessite une configuration précise des paramètres pour sa création et son entraînement. Ces paramètres sont établis à travers une série d'expérimentations menées sur l'ensemble des données disponibles.

Les paramètres essentiels sont détaillés dans les deux tableaux suivants, accompagnés des valeurs spécifiques qui leur sont attribuées. Le tableau 5.4 présente les paramètres propres à chaque couche du modèle Keras. Le tableau 5.3 concerne les paramètres de l'estimateur Elephas, utilisé pour l'entraînement du modèle. Ces paramètres sont soigneusement sélectionnés en vue de créer un modèle classificateur, qui sera ensuite enregistré en conservant les différentes configurations choisies. Ces deux ensembles de paramètres, combinés de manière appropriée, permettent de créer un modèle d'apprentissage profond performant et adapté aux besoins spécifiques de notre besoin.

TABLE 5.3 – *Les paramètres de l'estimateur Elephas*

Nom	**Description**	**Valeur**
Optimiseur	L'algorithme de rétropropagation qui détermine les meilleurs poids applicables à tous les neurones	Adam
Taux d'apprentissage	contrôle l'ampleur de la modification du modèle en réponse à l'erreur estimée à chaque fois que les poids du modèle sont mis à jour	0.001
Fonction de calcul de l'erreur	Calcule l'erreur entre la sortie réelle et la sortie délivrée par le modèle	categorical_crossentropy
Epoque	Le nombre de forward et backward pour tous les exemples d'entrainement	20
Batch-size	Le nombre d'exemples d'entrainement pour une seule forward et backward	32
Mode	Effectue l'apprentissage par répartition avec parallélisme des données	Synchroniser
Nombre de travailleurs	Définit le nombre d'exécuteurs qui seront utilisés dans le cluster spark	1

5.3.4 Implémentation du flux (streaming)

Afin d'optimiser le traitement du flux de transactions, nous avons minutieusement ajusté les paramètres en nous appuyant sur de nombreuses expériences menées sur l'ensemble des transactions. Le tableau 5.5 présente en détail les paramètres spécifiques du flux de travail en continu, qui ont été soigneusement

TABLE 5.4 – *Les paramètres du modèle Keras*

Les couches	Paramètre des couches	Description	Valeur
Couche d'entrée	Nombre de neurones	• Défini selon le nombre d'entrées (le nombre de facteurs)	128
	Fonction d'activation	• Défini selon une série d'expérimentations • Calcule la sortie délivrée par cette couche vers la couche cachée	Unité de Rectification Linéaire (ReLu)
	Fonction d'exclusion	• Défini selon une série d'expérimentations • L'exclusion est une technique qui permet d'éviter que les réseaux neuronaux ne soient surajoutés.	0.5
Couche cachée	Nombre de couche cachées	• Défini selon une série d'expérimentations	1
	Nombre de neurones	• Défini selon une série d'expérimentations	256
	Fonction d'activation	• Défini selon une série d'expérimentations • Calcule la sortie délivrée par cette couche vers la couche de sortie	Unité de Rectification Linéaire (ReLu)
	Fonction d'exclusion	• Défini selon une série d'expérimentations	0.5
Couche de sortie	Nombre de neurones	• Défini selon le nombre de classes	7
	Fonction d'activation	• Défini selon une série d'expérimentations • Calcule la sortie finale du modèle	Softmax

TABLE 5.5 – *Paramètres de transmission en continu (streaming)*

	Nom	Description	Valeur
Lecture de flux	MaxFilesPerTrigger	Nombre maximum de nouveaux dossiers à prendre en compte dans chaque micro-lot	1
Ecriture du flux	Mode de sortie	Pour écrire chaque nouvelle ligne dans le flux dataFrame dans le flux du récepteur	Ajouter
	Format	Le récepteur où la requête de streaming a été enregistrée	Mémoire
	Temps de traitement	L'intervalle de temps pour chaque déclencheur dans la requête de diffusion en continu	3 s
	Attendre la fin	Une fonction de productivité qui fait que le streaming fonctionne tout le temps	100 s (le temps est une option verser la visualisation)

définis pour assurer un traitement optimal du flux de transactions. Ces paramètres, déterminés grâce à une méthodologie rigoureuse, permettent d'atteindre une performance optimale et une gestion efficace du flux continu de données transactionnelles.

5.4 Métriques d'évaluation

Dans la section précédente, nous avons détaillé l'implémentation du modèle, où nous avons sélectionné les paramètres des couches du modèle Keras, de l'estimateur Elephas et du flux de travail en continu. Ces paramètres ont été choisis après de nombreuses expériences visant à trouver le meilleur classificateur possible.

Dans cette section, nous allons évaluer la pertinence de notre approche en utilisant différentes mesures. Notre objectif est d'analyser chaque transaction en temps réel afin de prédire si elle est malveillante, en identifiant également le type d'attaque de confiance, ou si elle est sécurisée afin de prévenir les attaques en temps réel. Pour atteindre cet objectif, notre approche se divise en trois phases distinctes : la classification des attaques en utilisant le classificateur, la prédiction en temps réel et la prévention des attaques en temps réel en utilisant notre protocole de consensus PACS.

Pour évaluer l'efficacité de notre approche à chaque phase, nous utiliserons des métriques spécifiques. Voici comment nous les définissons :

5.4.1 La phase de classification

Dans la phase de classification, nous évaluerons les performances du classificateur en utilisant des mesures telles que la précision, le rappel et le F1-score. Ces mesures nous permettront de quantifier la capacité du modèle à apprendre à partir des données d'entraînement et à classifier de manière précise et cohérente les transactions malveillantes et sécurisées. Voici comment nous les définissons :

— **La précision** mesure le nombre de transactions correctement identifiées comme malveillantes, qui correspond aux vrais positifs (VP), par rapport au nombre total de transactions identifiées comme malveillantes, qui est la somme des vrais positifs et des faux positifs (FP). Cela peut être calculé à

l'aide de l'équation 5.1. Une précision élevée indique que le classificateur est capable d'identifier avec précision les transactions malveillantes.

$$Precision = \frac{VP}{VP + FP} \tag{5.1}$$

— **Le rappel** mesure le nombre de transactions correctement identifiées comme malveillantes, qui sont les vrais positifs (VP), par rapport au nombre total de transactions réellement malveillantes, qui est la somme des vrais positifs et des faux négatifs (FN). Le rappel est calculé à l'aide de l'équation 5.2. Un rappel élevé signifie que le classificateur est capable de détecter la plupart des transactions malveillantes présentes dans le réseau.

$$Rappel = \frac{VP}{VP + FN} \tag{5.2}$$

— **Le F1-score ou F1**, également appelée F-mesure, est une mesure combinée de la précision et du rappel, qui prend en compte les deux aspects pour fournir une mesure globale des performances du classificateur. Une valeur élevée de F1-score indique un équilibre entre la précision et le rappel, ce qui signifie que le classificateur est capable de bien identifier les transactions malveillantes tout en minimisant les faux positifs et les faux négatifs. Il est calculé comme suit 5.3 :

$$F1 = 2 \cdot \frac{Precision \cdot Rappel}{Precision + Rappel} \tag{5.3}$$

De plus, nous évaluerons également le Score d'Importance des Caractéristiques (SIC). Cette métrique permet de déterminer l'importance de chaque caractéristique dans la prédiction du classificateur. En identifiant les caractéristiques les plus importantes, nous pouvons comprendre quelles informations sont les plus pertinentes pour détecter les transactions malveillantes afin d'aider à améliorer le classificateur en optimisant la précision des prédictions.

5.4.2 La phase de prédiction en temps réel

Dans la phase de prédiction en temps réel, nous évaluerons la capacité du modèle à prédire avec précision si une transaction est malveillante ou sécurisée en utilisant des métriques de traitement en temps réel. Voici les métriques que nous utiliserons :

— **Taux d'entrée** : Cette métrique mesure le taux global d'arrivée des données. Elle indique le nombre d'enregistrements chargés par seconde entre le début du dernier déclenchement et le début du déclenchement actuel. Un taux d'entrée élevé peut indiquer une forte demande de transactions à traiter.

— **Taux de traitement** : Cette métrique mesure le taux auquel Spark traite les données en temps réel. Elle décrit le nombre d'enregistrements chargés par seconde dans chaque déclencheur. Un taux de traitement élevé indique que le modèle est capable de traiter efficacement les transactions en temps réel.

— **Durée des lots** : Cette métrique mesure la durée de traitement de chaque lot de transactions. Elle indique combien de temps il faut au modèle pour traiter un ensemble de transactions. Une durée de lot plus courte est souhaitable car elle permet un traitement plus rapide des transactions en temps réel.

5.4.3 La phase de prévention des attaques en temps réel avec le protocole PACS

Dans la phase de prévention des attaques en temps réel avec le protocole PACS, nous souhaitons évaluer l'efficacité de ce protocole en utilisant différentes mesures. L'une de ces mesures clés est **le taux de validation des transactions**. Cette mesure nous permet de quantifier l'efficacité du protocole dans la détection et la prévention des attaques de confiance en temps réel. Le taux de validation des transactions mesure la capacité du protocole PACS à identifier avec précision les transactions malveillantes et à les bloquer avant qu'elles ne puissent causer des dommages. Un taux élevé de validation des transactions indique que le protocole n'est pas capable de détecter efficacement les attaques de confiance en temps réel et de prendre les mesures appropriées pour les prévenir.

En utilisant ces métriques spécifiques à chaque phase, nous serons en mesure d'évaluer l'efficacité et la pertinence de notre approche dans la détection, la prédiction et la prévention des attaques de confiance en temps réel dans le réseau IoT social.

5.5 Résultats

Au cours des sections précédentes, nous avons discuté en détail l'implémentation de notre approche proposée, qui se compose de la création de la blockchain et de notre protocole PACS basé sur un classificateur, ainsi que de la configuration du flux de traitement en ligne utilisant Spark avec l'ajustement des paramètres appropriés. De plus, nous avons défini notre ensemble de données et identifié les métriques que nous utiliserons pour évaluer les performances de notre proposition.

Dans cette section, nous nous concentrons sur l'évaluation de notre approche dans la détection et la prévention en temps réel des attaques de confiance, ainsi que sur la capacité de traitement du flux de transactions. Notre objectif principal est d'analyser les transactions générées et de détecter les attaques de confiance à l'aide du moteur de Spark Streaming, ce qui nous permet de prévenir les attaques en temps réel en annulant les transactions malveillantes. Ces attaques sont représentées par différentes classes prédites par notre classificateur, telles que "bma-attack", "bsa-attack", "spa-attack", "da-attack", "osa-attack", "ooa-attack" pour les transactions malveillantes, et "sécurisée" pour les transactions sûres.

Nous commencerons en évaluant l'importance des caractéristiques proposées dans notre approche. Ensuite, nous testerons chaque type d'attaque individuellement afin d'évaluer la précision de sa détection par notre modèle. Nous poursuivrons ensuite avec des simulations en combinant différentes attaques pour obtenir une vision plus globale des performances de notre système. Parallèlement, nous réaliserons des simulations pour évaluer l'efficacité du traitement des flux de transactions dans un environnement en temps réel. Enfin, nous évaluerons la capacité de notre protocole PACS à prévenir les attaques de confiance en temps réel.

Veuillez noter que les résultats et les analyses présentés dans cette section sont basés sur des simulations et des expérimentations approfondies pour garantir la fiabilité de l'évaluation.

5.5.1 Résultats expérimentaux des facteurs proposés

Pour évaluer l'importance de chaque facteur proposé, nous avons utilisé le score d'importance des caractéristiques. Ce score nous permet de quantifier l'importance relative de chaque facteur dans la prédiction des étiquettes des transactions. Il joue un rôle crucial dans la conception d'un modèle prédictif en identifiant les facteurs les plus pertinents pour la variable cible et en mettant en évidence ceux qui ont moins d'impact. De plus, le score d'importance des caractéristiques offre des critères de sélection pour améliorer l'efficacité et l'efficience d'un modèle prédictif. Nous avons employé la méthode de calcul du score d'importance des caractéristiques [2] afin d'obtenir des informations précieuses sur la contribution de chaque facteur à nos prédictions.

La figure 5.7 illustre le score d'importance des caractéristiques pour chaque facteur proposé dans notre approche. Nous observons que le facteur *rt* ou vote obtient le score le plus élevé parmi toutes les caractéristiques. Cela confirme que ce facteur joue un rôle crucial dans l'identification des transactions malveillantes. Ces résultats valident notre hypothèse selon laquelle "le vote peut être considéré comme un facteur primaire".

D'autre part, les facteurs Similarité des votes ou véracité Ver, valeur de confiance $ValConf$, et fréquence de votes $FreqV$ ont le deuxième impact le plus

2. https ://machinelearningmastery.com/calculate-feature-importance-with-python/

significatif dans l'évaluation de la confiance des transactions entre les utilisateurs de l'IoT social. Cette importance découle de leur capacité à décrire des comportements malveillants survenant de manière aléatoire, dans le but de propager de fausses recommandations et évaluations nuisant au système de confiance. De plus, nous notons que Similarité des utilisateurs *SimU*, Qualité du fournisseur *QoP*, et tendance des votes *TendV* ont un impact réduit sur l'évaluation de la confiance, mais ne peuvent être négligés, car ils sont considérés comme des informations complémentaires avec d'autres métriques pour la détection des transactions malveillantes. Ils contribuent de manière significative à la capacité de notre classificateur à identifier les transactions malveillantes.

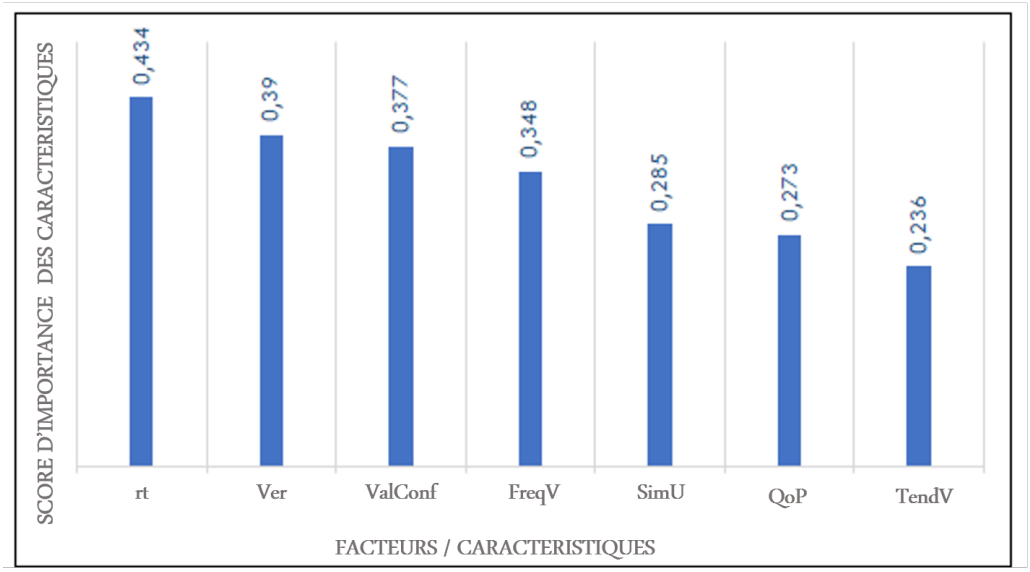

FIGURE 5.7 – *Évaluation des scores d'importance des caractéristiques pour chaque facteur proposé*

5.5.2 Résultats expérimentaux du classificateur basé sur Spark MLlib

Dans cette sous-section, nous allons évaluer l'efficacité de notre classificateur en utilisant la bibliothèque d'apprentissage automatique Spark MLlib pour classifier les transactions en tant que malveillantes en identifiant le type d'attaque effectuée, ou bien sécurisées. Nous débuterons par présenter les résultats obtenus pour chaque type d'attaque pris individuellement. Par la suite, nous exposerons les résultats obtenus en regroupant tous les types d'attaques tout en comparant les performances de notre classificateur, en termes de F1-score, avec celles d'autres études citées dans la littérature Abdelghani et al. (2018); Masmoudi et al. (2020, 2021); Magdich et al. (2022).

5.5.2.1 Prédiction des étiquettes pour chaque type d'attaque séparément

Ces expérimentations ont pour objectif d'évaluer la capacité prédictive de notre classificateur à prédire de manière individuelle les labels de chaque type d'attaque (BMA, BSA, SPA, DA, OOA ou OSA). Les résultats dévoilés dans la

figure 5.8 mettent en lumière les performances significativement améliorées de notre classificateur en termes de F1-score, comparativement à nos travaux précédents dans Masmoudi et al. (2020, 2021). Il est important de noter que dans ces précédents travaux, nous avions recours à des méthodes d'apprentissage automatique standard, et que dans Masmoudi et al. (2020), les attaques de types OSA et OOA n'ont pas été abordées. Ces avancées notables témoignent des progrès accomplis dans le perfectionnement de la conception et de la mise en œuvre de notre classificateur, en capitalisant sur les capacités puissantes mises à disposition par Apache Spark.

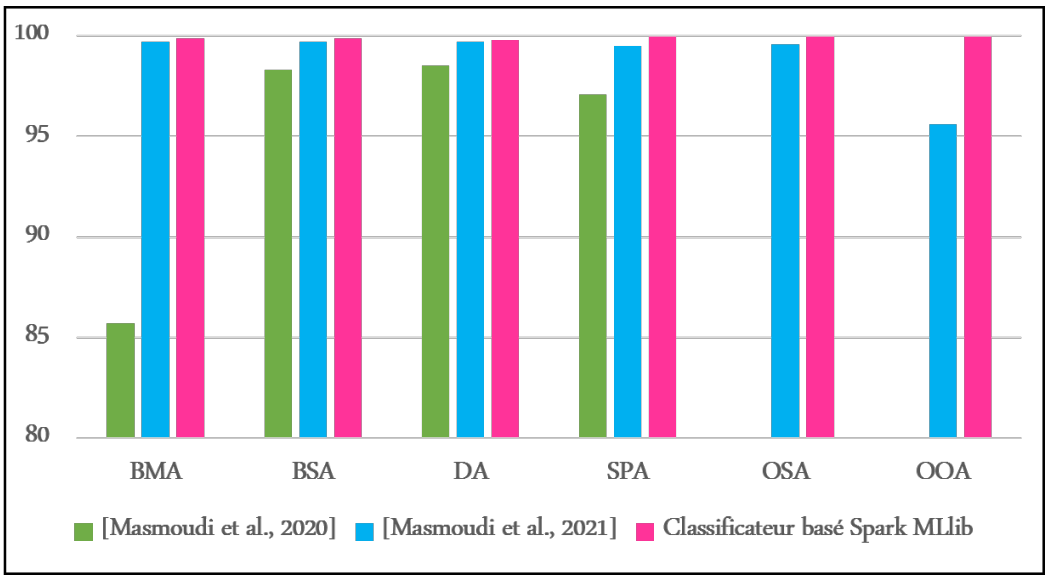

FIGURE 5.8 – *Performance du classificateur basé sur Spark MLlib par type d'attaque individuel avec F1-Score*

5.5.2.2 Prédiction des étiquettes pour tous les types d'attaques

Ces expériences visent à évaluer la capacité du classificateur à classifier et à identifier avec précision les types d'attaques présents dans les transactions effectuées.

Pour valider les performances de notre nouveau classificateur, nous avons réalisé une analyse comparative avec les résultats obtenus dans nos travaux précédents Masmoudi et al. (2020, 2021), ainsi qu'avec d'autres travaux de la littérature Abdelghani et al. (2018, 2022); Magdich et al. (2022), dont nous avons utilisé la même base de données (Sigcomm). Les résultats sont présentés dans la Figure 5.9. De plus, il convient de mentionner que le travail de Abdelghani et al. (2022) a démontré la pertinence de leur approche par rapport à plusieurs autres travaux de la littérature Jayasinghe et al. (2018); Truong et al. (2016); Chen et al. (2016). Par conséquent, il n'est pas nécessaire d'inclure ces travaux dans la figure de comparaison.

Sur la base des résultats obtenus, notre nouveau classificateur Spark MLlib surpasse nos travaux précédents et les travaux proposés dans Abdelghani et al.

(2018, 2022); Magdich et al. (2022) en termes de F1-score avec une valeur de 99,6%. Cette amélioration met en évidence la précision accrue de notre classificateur pour l'identification et la prévention des attaques liées à la confiance. De plus, l'utilisation de notre classificateur du framework Apache Spark, en particulier la bibliothèque Spark Machine Learning (MLlib), offre des performances supérieures par rapport à d'autres approches qui reposent sur des techniques d'apprentissage automatique standard.

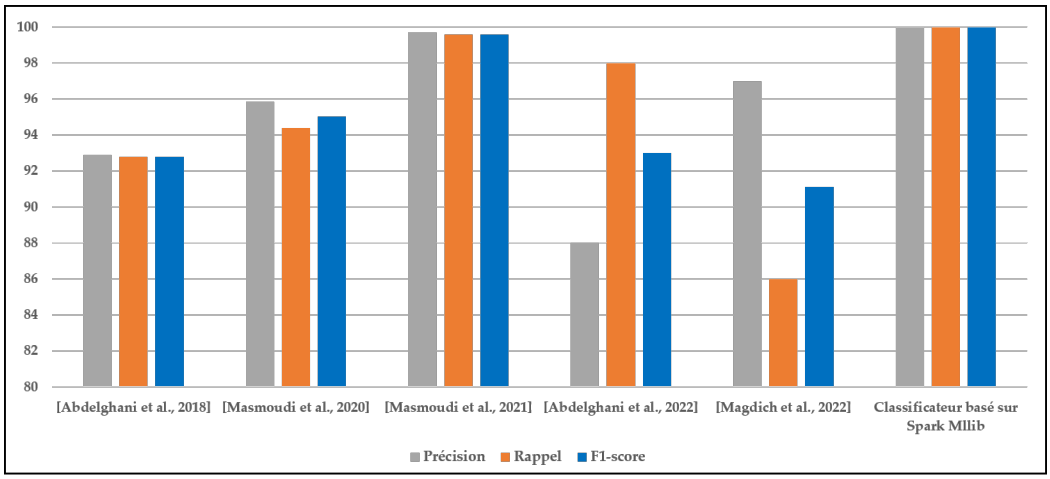

FIGURE 5.9 – *Performance du classificateur basé sur Spark MLlib pour tous les types d'attaques*

5.5.3 Résultats expérimentaux du classificateur basé sur Spark Elephas

Au sein de cette partie, nous entreprendrons l'évaluation de l'efficacité de notre classificateur qui utilise la bibliothèque d'apprentissage profond Spark Elephas pour catégoriser les transactions comme malveillantes, en identifiant précisément le type d'attaque en cours, ou comme sécurisées. Notre démarche débutera par la présentation des résultats obtenus pour l'ensemble des types d'attaques. Nous procéderons ensuite à une comparaison des performances de notre classificateur basé sur Spark Elephas avec celui basé sur Spark MLlib.

5.5.3.1 Résultats obtenus pour l'ensemble des types d'attaques

Dans cette sous-section, nous procédons à des tests visant à identifier tous les types d'attaques de confiance possibles. La représentation graphique des résultats obtenus par notre classificateur basé sur Spark Elephas est particulièrement convaincante. Les indicateurs de performance, incluant la précision, le rappel et la mesure F1-score, révèlent des performances exceptionnelles : une précision de 100%, un rappel de 99,46% et un F1-score de 99,73% comme indique la Figure 5.10. Ces résultats mettent en évidence de façon convaincante l'efficacité indiscutable de notre classificateur dans l'identification des transactions malveillantes camouflées en attaques de confiance.

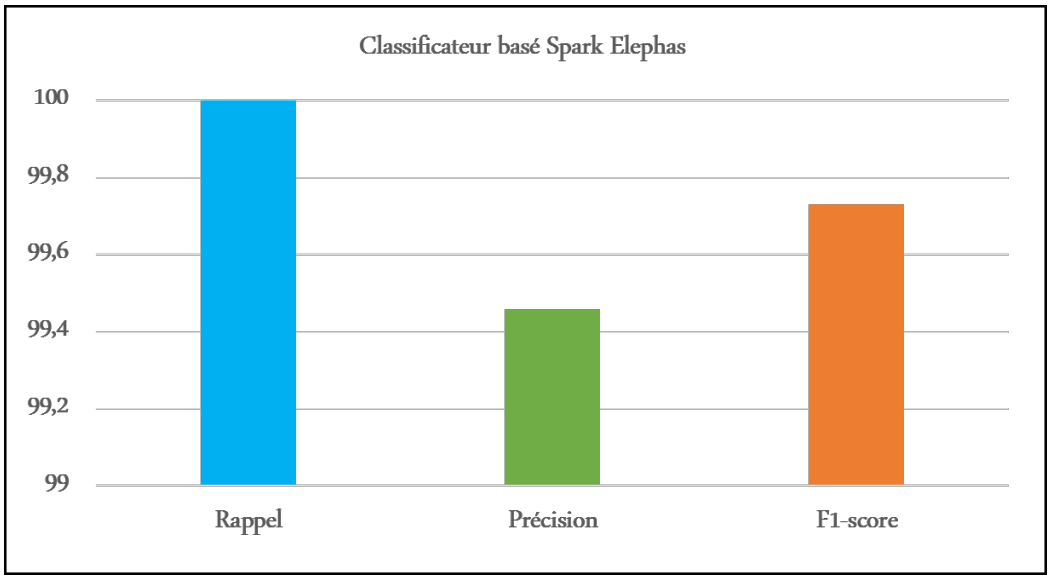

FIGURE 5.10 – *Résultats obtenus pour l'ensemble des types d'attaques*

5.5.3.2 Comparaison des Performances : Classificateur Spark Elephas vs. Classificateur Spark MLlib

Dans le cadre de la comparaison des résultats obtenus entre le classificateur basé sur Spark MLlib et celui basé sur Spark Elephas, des performances exceptionnelles ont été observées pour les deux approches. Le classificateur de Spark MLlib a présenté des performances impressionnantes avec un rappel de 99,6%, une précision de 99,7% et un F1-score de 99,6%. De manière encore plus saisissante, le classificateur basé sur Spark Elephas a affiché des performances exemplaires avec une précision de 100%, un rappel de 99,46% et un F1-score de 99,73% comme indique la Figure 5.11. Cette amélioration significative des performances est attribuée à l'utilisation de l'apprentissage profond offert par Spark Elephas. En effet, les modèles de l'apprentissage profond tels que ceux formés par Spark Elephas sont capables de saisir des structures de données complexes, permettant ainsi une identification plus précise des schémas dans les transactions. Cela se traduit par des performances améliorées en matière de prédiction et de classification, ce qui se reflète dans les résultats obtenus. Par conséquent, notre choix d'utiliser Spark Elephas s'est avéré judicieux pour les tâches de détection des attaques, offrant ainsi une meilleure précision et une robustesse accrue dans la classification des transactions malveillantes déguisées en attaques de confiance. Dans cette situation, nous choisissons d'intégrer le classificateur basé sur Spark Elephas au sein de notre protocole de consensus PACS. Cette décision vise à renforcer la prévention des attaques de confiance en temps réel au sein de l'environnement de l'IoT Social.

5.5.4 Résultats expérimentaux du traitement des flux des transactions

Dans cette section, nous réalisons une évaluation du traitement en temps réel en utilisant les métriques fournies par l'interface utilisateur de Spark UI. La

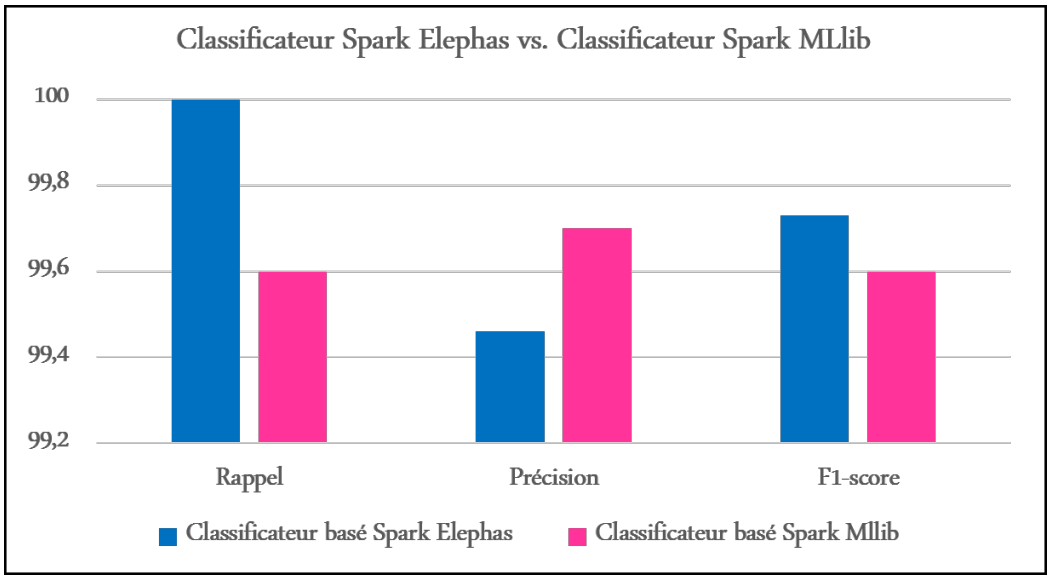

FIGURE 5.11 – *Comparaison des Performances : Classificateur Spark Elephas vs. Classificateur Spark MLlib*

visualisation graphique présentée dans la Figure 5.12 illustre trois courbes distinctes représentant le taux d'entrée (Input Rate), le taux de traitement (Process Rate) et la durée des lots(Batch Duration).

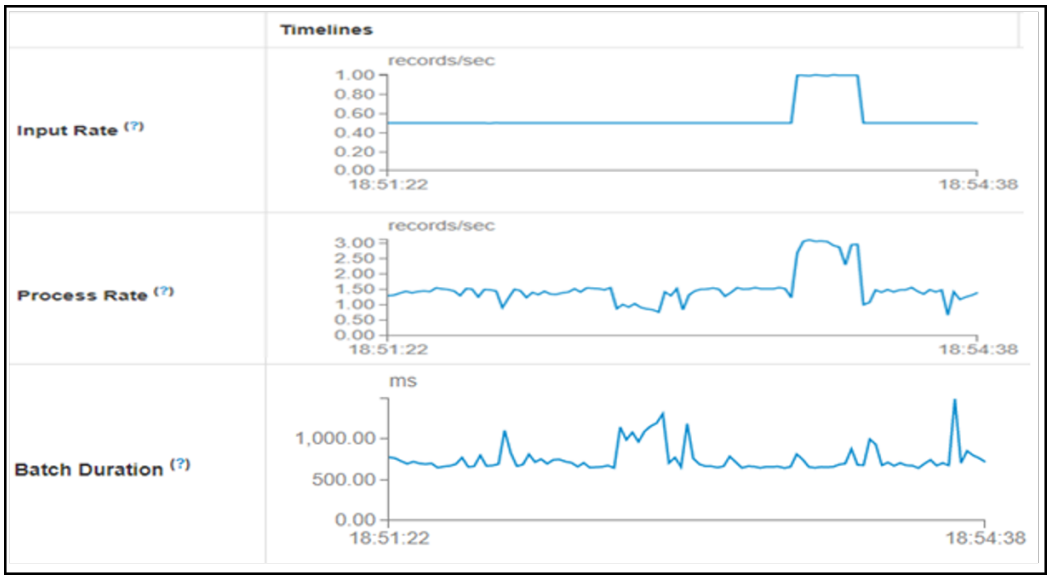

FIGURE 5.12 – *Résultats expérimentaux du traitement des flux des transactions*

Pour évaluer la détection des attaques en temps réel, nous avons généré 150 transactions au format CSV, dont aucune n'a été utilisée auparavant dans la phase d'apprentissage, afin de valider efficacement les prédictions. Comme l'illustre la Figure 5.12, le taux d'entrée spécifie la quantité de données reçu à partir d'un système de traitement de flux ou d'un répertoire local, comme c'est notre cas. Le taux de traitement indique à quelle vitesse nous avons pu analyser ces données.

Dans le cas idéal, les deux taux devraient varier de manière cohérente ensemble ; cependant, ils varieront en fonction de la quantité de données d'entrée disponibles lorsque le traitement démarre. Dans notre cas, nous constatons que le taux de traitement peut rester stable avec une moyenne de 1,4 enregistrements par seconde au même taux d'entrée avec une moyenne de 0,5 enregistrements par seconde. Cela signifie que la capacité de traitement d'une tâche est suffisante pour traiter les données d'entrée. En ce qui concerne la durée du lot (Batch Duration), l'augmentation des données dans chaque lot entraînera une latence élevée. Certains systèmes de traitement de flux proposent une option de latence élevée en échange d'un débit plus faible. Dans la Figure 5.12, nous pouvons constater que notre Durée de Lot oscille de manière cohérente autour de 700 ms. En réalité, le traitement en temps réel parvient à combiner la latence et le débit. En fonction du traitement des données, il oscille à mesure que le traitement structuré en temps réel traite des nombres variables d'événements au fil du temps. Par conséquent, si nous utilisons un cluster plus important, nous aurons un taux de traitement bien plus rapide ainsi qu'une durée de lot nettement plus courte.

De plus, nous comparons les performances de notre traitement en temps avec l'évaluation de deux travaux de recherche Abid et Jemili (2020); Zhou et al. (2018). Le système proposé par Abid et Jemili (2020) a réussi à traiter 60635 enregistrements en une seconde, tandis qu'il est capable de collecter 613027 enregistrements par seconde. Ces résultats sont illogiques étant donné que les lignes arrivées dépassent la capacité des tâches à traiter les données d'entrée, ce qui entraînerait une latence élevée. Pour notre évaluation du traitement en temps réel, notre taux de traitement est trois fois supérieur au taux d'entrée, ce qui le rend meilleur que Abid et Jemili (2020) avec une faible latence. Le deuxième modèle proposé par Zhou et al. (2018) peut collecter en moyenne 200 000 enregistrements par seconde, tandis que le système a une moyenne de 546 ms en tant que temps de traitement. Par rapport à notre classificateur, nous obtenons de meilleurs résultats, puisque nous pouvons traiter les données d'entrée en seulement 333 ms en moyenne pour le temps de traitement.

5.5.5 Résultats expérimentaux de notre protocole de consensus (PACS) basé sur Apache Spark

Pour examiner les performances du protocole PACS que nous proposons, nous comparerons ses capacités en termes de taux de traitement (nombre de transactions traitées au cours d'une période donnée) et de prédiction des transactions malveillantes par rapport au protocole de référence Preuve de Travail (PdT).

5.5.5.1 Taux de traitement

La comparaison réalisée dans la Figure 5.13 confirme l'efficacité remarquable de notre protocole PACS dans le traitement d'un plus grand nombre de transactions dans un laps de temps plus court, surpassant ainsi les performances du protocole PdT. Malgré une charge de travail accrue avec un plus grand nombre de

transactions, notre protocole de consensus PACS maintient de manière constante sa capacité à traiter un volume plus important de transactions rapidement. Ce résultat met en évidence l'efficacité de Spark Streaming en tant qu'outil puissant pour gérer des charges transactionnelles plus élevées de manière rapide et efficace sur le plan temporel.

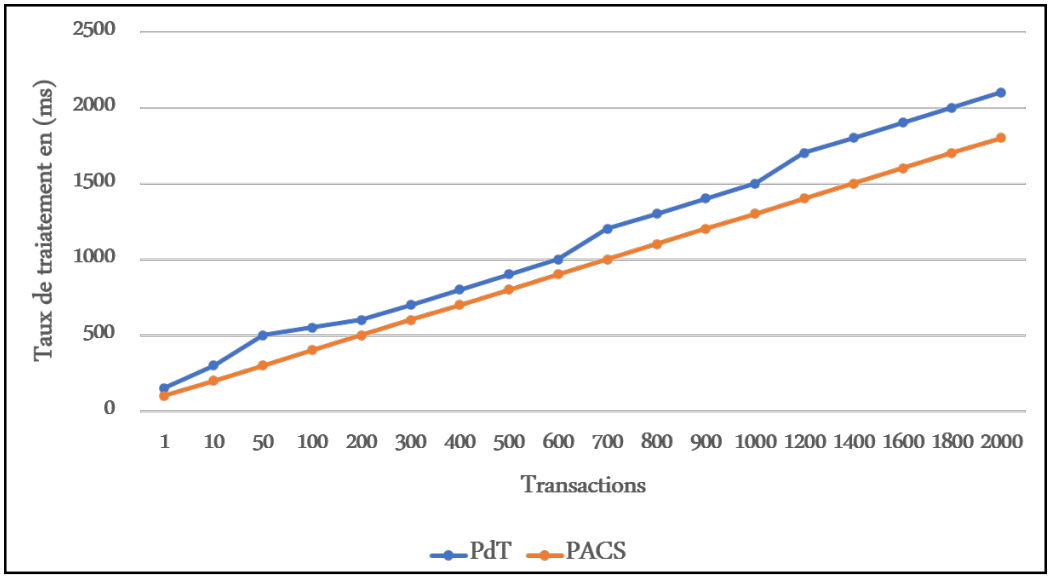

FIGURE 5.13 – *Taux de traitement des transactions en ms*

5.5.5.2 Taux de validation des transactions

La Figure 5.14 illustre clairement que le protocole PdT ne parvient pas à anticiper les attaques examinées, entraînant ainsi la validation et l'ajout de toutes les transactions générées à la blockchain existante. En revanche, notre protocole PACS se distingue par sa remarquable efficacité dans la prédiction des transactions malveillantes qui, en conséquence, ne sont pas validées ni intégrées à la blockchain. Cette observation témoigne du rôle crucial joué par notre classificateur dans la classification précise des attaques, renforçant ainsi la capacité de notre protocole à éviter la validation de transactions malveillantes. Par conséquent, notre protocole s'affirme comme une solution puissante dans la prévention des attaques de confiance au sein de l'IoT Social.

Suite à ces comparaisons, nous pouvons synthétiser les principales avancées de notre recherche comme suit : À notre connaissance, notre étude est la première à entreprendre une prévention en temps réel contre toutes les types d'attaques de confiance. Cette approche novatrice contribue non seulement à la progression de la sécurité des transactions, mais établit également un nouveau standard en matière de prévention globale des attaques. En exploitant les capacités d'Apache Spark et de sa bibliothèque Elephas, nous avons réalisé une solution innovante qui parvient à détecter et prévenir avec efficacité les attaques de confiance en temps réel dans l'IoT Social.

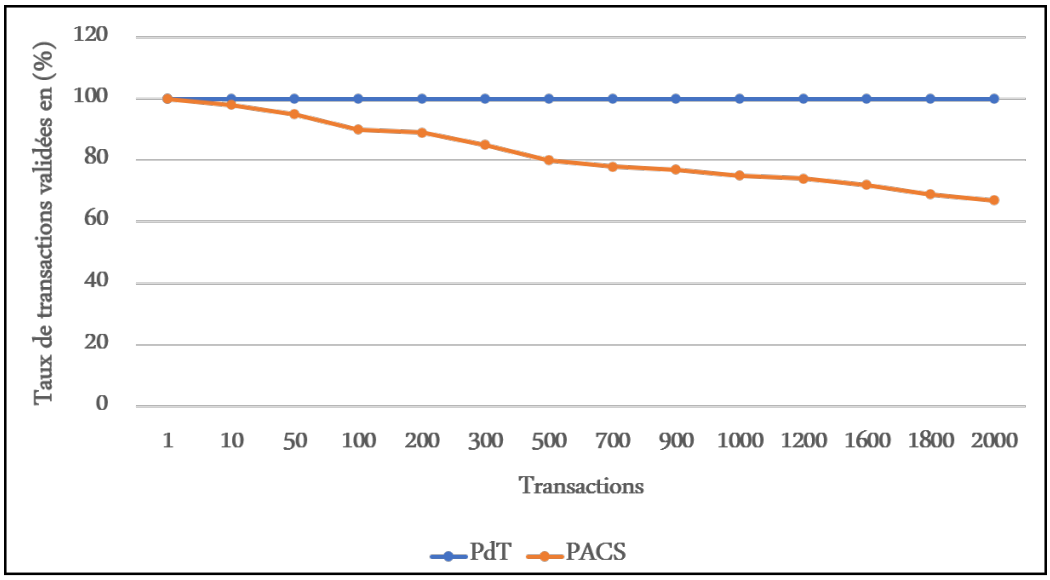

FIGURE 5.14 – *Taux de validation des transactions*

5.6 CONCLUSION

Tout au long de ce chapitre, nous avons présenté en détail la réalisation et l'évaluation de notre approche, SparkChain. Nous avons commencé par décrire l'environnement de développement matériel et logiciel que nous avons utilisé, notamment le choix du langage de programmation et des outils pour la mise en œuvre de notre solution. Ensuite, nous avons détaillé la base de données que nous avons utilisée comme point de départ pour nos expérimentations. Par la suite, nous avons exposé l'implémentation de la blockchain et les classificateurs basés sur Spark, en mettant en évidence les fonctionnalités clés utilisées pour l'analyse et l'apprentissage des données. Nous avons également mentionné les différentes interfaces graphiques que nous avons développées pour notre blockchain. Ensuite, nous avons présenté les différentes mesures d'évaluation que nous avons utilisées pour évaluer les performances de notre approche. Nous avons expliqué comment ces métriques nous ont permis de mesurer l'efficacité de SparkChain dans la prévention des attaques de confiance en temps réel dans l'IoT social. Enfin, nous avons présenté les résultats obtenus à partir de nos expérimentations et les avons comparés aux travaux existants dans la littérature. Nous avons démontré comment SparkChain se révèle plus efficace que les approches précédentes pour prévenir les attaques de confiance dans l'IoT social, en fournissant des résultats prometteurs et en mettant en évidence les avantages de notre approche.

Conclusion générale

«Il est facile de manquer le but et difficile de l'atteindre »
Aristote

Dans les environnements intelligents tels que l'IoT social, la communication et l'interaction entre un grand nombre d'utilisateurs et d'objets intelligents peuvent entraîner des problèmes de confiance. Les comportements malveillants tels que les "attaques de confiance" peuvent influencer les services recommandés aux utilisateurs, les incitant à faire confiance à des services malveillants. En conséquence, les utilisateurs peuvent douter de la fiabilité des services et des objets connectés qu'ils utilisent, ainsi que des informations qu'ils partagent avec eux. Cette situation peut susciter un sentiment de méfiance et de vigilance à l'égard du réseau. Ainsi, la gestion de la confiance devient un enjeu majeur pour garantir des expériences fiables pour les utlisateurs.

Cette thèse effectuée s'inscrite dans le cadre de cette problématique. Dans cette perspective, nous avons proposé un nouveau mécanisme de gestion de la confiance efficace qui permet de prévenir en temps réel tous les types d'attaques de confiance dans l'IoT social nommé SparkChain. Pour cela, nous avons commencé par une étude de la littérature et une analyse des approches existantes des mécanismes de gestion de la confiance. Cette analyse a montré l'inexistence d'un mécanisme qui permet de prévenir en temps réel tous les types d'attaques de confiance et qui peut prendre en considération les différents propriétés de sécurité. C'est la raison pour laquelle nous nous sommes penchés sur le problème de développement d'un mécanisme pour la prévention en temps réel les attaques de confiance. Ce mécanisme se repose sur l'utilisation des technologies avancées telles que la blockchain et Apache Spark. Dans cette optique, nous avons établi une comparaison approfondie de ces technologies. A la lumière de cet état de l'art, nous avons conclu que la combinaison de la blockchain et Apache Spark nous permet d'avoir des résultats satisfaisants en terme de prévention en temps réel des attaques de confiance. En effet, la blockchain présente de nombreux avantages par rapport aux autres technologies, notamment en termes de sécurité, de transparence et de résilience. De même, Apache Spark présente des avantages significatifs en termes de vitesse de traitement, de facilité de mise en œuvre et de compatibilité avec d'autres technologies telles que la blockchain. C'est pourquoi, nous avons opté pour l'utilisation d'Apache Spark dans notre protocole de consensus PACS de la blockchain afin d'améliorer la vitesse et la précision de la détection des transactions malveillantes, ce qui à son tour améliore la précision de la prévention des attaques de confiance en temps réel.

Après avoir mené une évaluation exhaustive de notre approche,SparkChain, de prévention en temps réel des attaques de confiance, nous avons démontré ses performances en utilisant plusieurs métriques d'évaluation telles que la précision, le rappel, le f1-score, le taux de validation des transactions, la capacité de traitement des transactions par secondes, etc. Les résultats ont montré que notre approche offre une haute performance. Ces résultats encourageants confirment l'efficacité de notre approche de prévention des attaques de confiance en temps réel tout en maintenant les propriétés de sécurité dans les environnements IoT social.

Perspectives

Dans la continuité directe de notre travail de thèse, nous pouvons :

— Définir un autre mécanisme de gestion de la confiance niveau dispositif. Ce niveau dispositif permet de mesurer la confiance pour les nœuds de type dispositif.
— Approfondir l'étude de l'utilisation de la blockchain et d'Apache Spark pour la prévention des attaques de confiance en SIoT, en explorant d'autres aspects tels que la scalabilité et l'efficacité énergétique.
— Étendre la recherche aux autres types d'attaques dans les environnements IoT social, en utilisant des techniques similaires pour les prévenir en temps réel.
— Étudier l'impact de l'application de la solution proposée sur les performances des systèmes IoT social, en tenant compte des coûts en termes de traitement, de stockage et de communication.
— Tester l'efficacité de la solution proposée dans des scénarios d'utilisation réels en utilisant des environnements IoT social existants ou en créant des environnements de test simulés.

En conclusion, il convient de souligner que les problématiques liées à la sécurité et à la confiance dans les environnements de l'IoT social restent toujours des problèmes ouverts. En effet, avec l'augmentation constante du nombre d'objets connectés et d'utilisateurs interagissant dans cet écosystème, les risques potentiels d'attaques malveillantes et de violations de la vie privée continuent de croître. Par conséquent, de nouvelles approches et technologies devront être développées pour renforcer la confiance et la sécurité dans l'IoT social.

Bibliographie

Wafa Abdelghani, Ikram Amous, Corinne Amel Zayani, Florence Sèdes, et Geoffrey Roman-Jimenez. Dynamic and scalable multi-level trust management model for social internet of things. *The Journal of Supercomputing*, 78(6) :8137–8193, 2022.

Wafa Abdelghani, Corinne Amel Zayani, Ikram Amous, et Florence Sèdes. Trust management in social internet of things : a survey. Dans *Conference on e-Business, e-Services and e-Society*, pages 430–441. Springer, 2016.

Wafa Abdelghani, Corinne Amel Zayani, Ikram Amous, et Florence Sèdes. Trust evaluation model for attack detection in social internet of things. Dans *International Conference on Risks and Security of Internet and Systems*, pages 48–64. Springer, 2018.

Oumaima Ben Abderrahim, Mohamed Houcine Elhdhili, et Leila Saidane. Tmcoi-siot : A trust management system based on communities of interest for the social internet of things. Dans *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 747–752. IEEE, 2017.

Ahlem Abid et Farah Jemili. Intrusion detection based on graph oriented big data analytics. *Procedia Computer Science*, 176 :572–581, 2020.

Ilya Afanasyev, Alexander Kolotov, Ruslan Rezin, Konstantin Danilov, Alexey Kashevnik, et Vladimir Jotsov. Blockchain solutions for multi-agent robotic systems : Related work and open questions. Dans *Proceedings of the 24th Conference of Open Innovations Association FRUCT*, page 76. FRUCT Oy, 2019.

Sana Alam, Shehnila Zardari, et Jawwad Ahmed Shamsi. Blockchain-based trust and reputation management in siot. *Electronics*, 11(23) :3871, 2022.

Yara Alghofaili et Murad A Rassam. A dynamic trust-related attack detection model for iot devices and services based on the deep long short-term memory technique. *Sensors*, 23(8) :3814, 2023.

Meraj Farheen Ansari, Pawan Kumar Sharma, et Bibhu Dash. Prevention of phishing attacks using ai-based cybersecurity awareness training. *Prevention*, 2022.

Malihe Asemani, Fatemeh Abdollahei, et Fatemeh Jabbari. Understanding iot platforms : towards a comprehensive definition and main characteristic description. Dans *2019 5th International Conference on Web Research (ICWR)*, pages 172–177. IEEE, 2019.

Mehdi Assefi, Ehsun Behravesh, Guangchi Liu, et Ahmad P Tafti. Big data machine learning using apache spark mllib. Dans *2017 ieee international conference on big data (big data)*, pages 3492–3498. IEEE, 2017.

Luigi Atzori, Antonio Iera, et Giacomo Morabito. Siot : Giving a social structure to the internet of things. volume 15, pages 1193–1195. IEEE, 2011.

Luigi Atzori, Antonio Iera, Giacomo Morabito, et Michele Nitti. The social internet of things (siot)–when social networks meet the internet of things : Concept, architecture and network characterization. volume 56, pages 3594–3608. Elsevier, 2012.

Mazhar Javed Awan, Umar Farooq, Hafiz Muhammad Aqeel Babar, Awais Yasin, Haitham Nobanee, Muzammil Hussain, Owais Hakeem, et Azlan Mohd Zain. Real-time ddos attack detection system using big data approach. *Sustainability*, 13(19) :10743, 2021.

Sheikh Tahir Bakhsh, Saleh Alghamdi, Rayan A Alsemmeari, et Syed Raheel Hassan. An adaptive intrusion detection and prevention system for internet of things. volume 15, page 1550147719888109. SAGE Publications Sage UK : London, England, 2019.

Imran Bashir. *Mastering blockchain*. Packt Publishing Ltd, 2017.

Bouziane Beldjilali. *Gestion de Confiance dans le Cloud Computing*. PhD thesis, Université d'Oran, 2016.

Davide Calvaresi, Alevtina Dubovitskaya, Jean Paul Calbimonte, Kuldar Taveter, et Michael Schumacher. Multi-agent systems and blockchain : Results from a systematic literature review. Dans *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 110–126. Springer, 2018.

Rajanpreet Kaur Chahal, Neeraj Kumar, et Shalini Batra. Trust management in social internet of things : A taxonomy, open issues, and challenges. *Computer Communications*, 150 :13–46, 2020.

Ray Chen, Fenye Bao, et Jia Guo. Trust-based service management for social internet of things systems. *IEEE transactions on dependable and secure computing*, 13(6) :684–696, 2015.

Yun Chen, Hui Xie, Kun Lv, Shengjun Wei, et Changzhen Hu. Deplest : A blockchain-based privacy-preserving distributed database toward user behaviors in social networks. volume 501, pages 100–117. Elsevier, 2019.

Zhikui Chen, Ruochuan Ling, Chung-Ming Huang, et Xu Zhu. A scheme of access service recommendation for the social internet of things. volume 29, pages 694–706. Wiley Online Library, 2016.

Nathalie Dagorn. Détection et prévention d'intrusion : présentation et limites. 2006.

Gholamhossein Ekbatanifard et Omid Yousefi. A novel trust management model in the social internet of things. volume 5, pages 57–70. Science and Research Branch, Islamic Azad University, 2019.

Kanav Raj Farishta, Vivek Kumar Singh, et D Rajeswari. Xss attack prevention using machine learning. *World Review of Science, Technology and Sustainable Development*, 18(1) :45–50, 2022.

Bo Feng, Qiang Li, Yuede Ji, Dong Guo, et Xiangyu Meng. Stopping the cyberattack in the early stage : assessing the security risks of social network users. volume 2019. Hindawi, 2019.

Darshankumar Vinubhai Gorasiya. Comparison of open-source data stream processing engines : spark streaming, flink and storm. 2019.

Barbara Guidi. When blockchain meets online social networks. volume 62, pages 101–131. Elsevier, 2020.

Nancy Gulati et Pankaj Deep Kaur. When things become friends : A semantic perspective on the social internet of things. Dans *Smart Innovations in Communication and Computational Sciences*, pages 149–159. Springer, 2019.

Jia Guo, Ray Chen, et Jeffrey JP Tsai. A survey of trust computation models for service management in internet of things systems. volume 97, pages 1–14. Elsevier, 2017.

Dezhi Han, Kun Bi, Han Liu, et Jianxin Jia. A ddos attack detection system based on spark framework. *Computer Science and Information Systems*, 14(3) :769–788, 2017.

Rahmeh Fawaz Ibrahim, Qasem Abu Al-Haija, et Ashraf Ahmad. Ddos attack prevention for internet of thing devices using ethereum blockchain technology. *Sensors*, 22(18) :6806, 2022.

Razi Iqbal, Talal Ashraf Butt, Muhammad Afzaal, et Khaled Salah. Trust management in social internet of vehicles : Factors, challenges, blockchain, and fog solutions. volume 15, pages 155–177. SAGE Publications Sage UK : London, England, 2019.

U. Jayasinghe, G. M. Lee, T. W. Um, et Q. Shi. Machine learning based trust computational model for iot services. volume 4, pages 39–52. IEEE, 2018.

Raouf Jmal, Mariam Masmoudi, Ikram Amous, Corinne Amel Zayani, et F Sèdes. Apache spark based deep learning for social transaction analysis. Dans *19th International Conference on Web Information Systems and Technologies*, 2023.

V Srinivas Jonnalagadda, P Srikanth, Krishnamachari Thumati, Sri Hari Nallamala, et K Dist. A review study of apache spark in big data processing. *International Journal of Computer Science Trends and Technology (IJCST)*, 4(3) :93–98, 2016.

Nisanthan Kathirkamanathan, Bahinthan Thevarasa, Gukajan Mahadevan, Nimalaprakasan Skandhakumar, et Nuwan Kuruwitaarachchi. Prevention of ddos attacks targeting financial services using supervised machine learning and stacked lstm. Dans *2022 IEEE 7th International conference for Convergence in Technology (I2CT)*, pages 1–5. IEEE, 2022.

Solomon Damena Kebede, Basant Tiwari, Vivek Tiwari, et Kamlesh Chandravanshi. Predictive machine learning-based integrated approach for ddos detection and prevention. *Multimedia Tools and Applications*, 81(3) :4185–4211, 2022.

Poonam Sinai Kenkre, Anusha Pai, et Louella Colaco. Real time intrusion detection and prevention system. Dans *Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing : Theory and Applications (FICTA) 2014*, pages 405–411. Springer, 2015.

Adnan Shahid Khan, Kuhanraj Balan, Yasir Javed, Seleviawati Tarmizi, et Johari Abdullah. Secure trust-based blockchain architecture to prevent attacks in vanet. volume 19, pages 49–54. Multidisciplinary Digital Publishing Institute, 2019.

Wazir Zada Khan, Saqib Hakak, Muhammad Khurram Khan, et al. Trust management in social internet of things : Architectures, recent advancements, and future challenges. *IEEE Internet of Things Journal*, 8(10) :7768–7788, 2020.

A Meena Kowshalya et ML Valarmathi. Trust management for reliable decision making among social objects in the social internet of things. volume 6, pages 75–80. IET, 2017a.

A Meena Kowshalya et ML Valarmathi. Trust management in the social internet of things. volume 96, pages 2681–2691. Springer, 2017b.

J Senthil Kumar, G Sivasankar, et S Selva Nidhyananthan. An artificial intelligence approach for enhancing trust between social iot devices in a network. Dans *Toward Social Internet of Things (SIoT) : Enabling Technologies, Architectures and Applications*, pages 183–196. Springer, 2020.

Zhiting Lin et Liang Dong. Clarifying trust in social internet of things. *IEEE Transactions on Knowledge and Data Engineering*, 30(2) :234–248, 2017.

Zhaojun Lu, Wenchao Liu, Qian Wang, Gang Qu, et Zhenglin Liu. A privacy-preserving trust model based on blockchain for vanets. volume 6, pages 45655–45664. IEEE, 2018.

Rim Magdich, Hanen Jemal, et Mounir Ben Ayed. Context-awareness trust management model for trustworthy communications in the social internet of things. *Neural Computing and Applications*, 34(24) :21961–21986, 2022.

Naila Marir, Huiqiang Wang, Guangsheng Feng, Bingyang Li, et Meijuan Jia. Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark. *IEEE Access*, 6 :59657–59671, 2018.

Mariam Masmoudi, Wafa Abdelghani, Ikram Amous, et Florence Sèdes. Deep learning for trust-related attacks detection in social internet of things. Dans *International Conference on e-Business Engineering*, pages 389–404. Springer, 2020.

Mariam Masmoudi, Ikram Amous, Corinne Amel Zayani, et Florence Sèdes. Real-time mitigation of trust-related attacks in social iot. Dans *International Conference on Model and Data Engineering*, pages 303–318. Springer, 2023.

Mariam Masmoudi, Corinne Amel Zayani, Ikram Amous, et Florence Sèdes. A new blockchain-based trust management model. Dans *25th International Conference on Knowledge-Based and Intelligent Information Engineering Systems*, pages 389–404. Elsevier, 2021.

Carolina Veronica Lezama Mendoza et João Henrique Kleinschmidt. A distributed trust management mechanism for the internet of things using a multiservice approach. *Wireless Personal Communications*, 103 :2501–2513, 2018.

Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. Mllib : Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1) :1235–1241, 2016.

Alexander Mikroyannidis, Allan Third, John Domingue, Michelle Bachler, et Kevin A Quick. Blockchain applications in lifelong learning and the role of the semantic blockchain. Dans *Blockchain Technology Applications in Education*, pages 16–41. IGI Global, 2020.

Satoshi Nakamoto et A Bitcoin. A peer-to-peer electronic cash system. pages 8–9, 2008.

Koichi Nakayama, Yutaka Moriyama, et Chika Oshima. An algorithm that prevents spam attacks using blockchain. volume 9, pages 204–208. SCIENCE & INFORMATION SAI ORGANIZATION LTD 19 BOLLING RD, BRADFORD, WEST . . . , 2018.

Michele Nitti, Roberto Girau, et Luigi Atzori. Trustworthiness management in the social internet of things. volume 26, pages 1253–1266. IEEE, 2013.

Alfonso Panarello, Nachiket Tapas, Giovanni Merlino, Francesco Longo, et Antonio Puliafito. Blockchain and iot integration : A systematic survey. volume 18, pages 25–75. Multidisciplinary Digital Publishing Institute, 2018.

Nilesh Vishwasrao Patil, C Rama Krishna, et Krishan Kumar. Ss-ddos : : Spark-based ddos attacks classification approach. Dans *Security and Resilience of Cyber Physical Systems*, pages 81–90. Chapman and Hall/CRC, 2022a.

Nilesh Vishwasrao Patil, C Rama Krishna, et Krishan Kumar. Ssk-ddos : distributed stream processing framework based classification system for ddos attacks. *Cluster Computing*, pages 1–18, 2022b.

Nilesh Vishwasrao Patil, C Rama Krishna, et Krishan Kumar. S-ddos : Apache spark based real-time ddos detection system. *Journal of Intelligent & Fuzzy Systems*, 38(5) :6527–6535, 2020.

Shelan Perera, Ashansa Perera, et Kamal Hakimzadeh. Reproducible experiments for comparing apache flink and apache spark on public clouds. *arXiv preprint arXiv :1610.04493*, 2016.

Anusha Ramanathan. *A multi-level trust management scheme for the Internet of Things*. PhD thesis, 2015.

MS Roopa, Santosh Pattar, Rajkumar Buyya, Kuppanna Rajuk Venugopal, SS Iyengar, et LM Patnaik. Social internet of things (siot) : Foundations, thrust areas, systematic review and future directions. *Computer Communications*, pages 32–57, 2019.

Jeremy Rubin. Btcspark : Scalable analysis of the bitcoin blockchain using spark. *Dec*, 16 :1–14, 2015.

S Saravanan et al. Performance evaluation of classification algorithms in the design of apache spark based intrusion detection system. Dans *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pages 443–447. IEEE, 2020.

Eman Shaikh, Iman Mohiuddin, Yasmeen Alufaisan, et Irum Nahvi. Apache spark : A big data processing engine. Dans *2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM)*, pages 1–6. IEEE, 2019.

Besfort Shala, Ulrich Trick, Armin Lehmann, Bogdan Ghita, et Stavros Shiaeles. Novel trust consensus protocol and blockchain-based trust evaluation system for m2m application services. volume 7, pages 39–58. Elsevier, 2019.

Bc Branislav Smik. *Blockchain technologies adapted for data manipulation in IoT*. PhD thesis, Masaryk University Faculty of Informatics, 2018.

Said Talbi et Abdelmadjid Bouabdallah. Interest-based trust management scheme for social internet of things. volume 11, pages 1129–1140. Springer, 2020.

Nguyen B Truong, Tai-Won Um, et Gyu Myoung Lee. A reputation and knowledge based trust service platform for trustworthy social internet of things. 2016.

Leon Wirz, Rinrada Tanthanathewin, Asipan Ketphet, et Somchart Fugkeaw. Design and development of a cloud-based ids using apache kafka and spark streaming. Dans *2022 19th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 1–6. IEEE, 2022.

Hui Xia, Fu Xiao, San-shun Zhang, Chun-qiang Hu, et Xiu-zhen Cheng. Trustworthiness inference framework in the social internet of things : A context-aware approach. Dans *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 838–846. IEEE, 2019.

Youcef Ould Yahia. *A proposal for a security model for the protection of personal data in systems based on the internet of things*. PhD thesis, 2019.

Hao Zhang, Shumin Dai, Yongdan Li, et Wenjun Zhang. Real-time distributed-random-forest-based network intrusion detection system using apache spark. Dans *2018 IEEE 37th international performance computing and communications conference (IPCCC)*, pages 1–7. IEEE, 2018.

Lirong Zheng, Hui Zhang, Weili Han, Xiaolin Zhou, Jing He, Zhi Zhang, Yun Gu, et Junyu Wang. Technologies, applications, and governance in the internet of things. *Internet of Things-Global Technological and Societal Trends From Smart Environments and Spaces to Green ICT*, page 143, 2011.

Lirong Zheng, Hui Zhang, Weili Han, Xiaolin Zhou, Jing He, Zhi Zhang, Yun Gu, et Junyu Wang. Technologies, applications, and governance in the internet of things. Dans *Internet of Things-Global Technological and Societal Trends from Smart Environments and Spaces to Green Ict*, pages 143–177. River Publishers, 2022.

Baojun Zhou, Jie Li, Jinsong Wu, Song Guo, Yu Gu, et Zhetao Li. Machine-learning-based online distributed denial-of-service attack detection using spark streaming. Dans *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.

Jun Zou, Bin Ye, Lie Qu, Yan Wang, Mehmet A Orgun, et Lei Li. A proof-of-trust consensus protocol for enhancing accountability in crowdsourcing services. volume 12, pages 429–445. IEEE, 2018.

Notations

IoT	Internet of Things
IoT social	Social Internet of Things
SN	Social Networks
ID Mgmt	ID management
OC	Owner Control
RM	Relationship Management
SD	Service Discovery
SC	Service Composition
TM	Trustworthiness Management
SPA	Self-Promoting Attack
DA	Discriminatory Attack
BMA	Bad-Mouthing Attack
BSA	Ballot-Stuffing Attack
OSA	Opportunistic Service Attack
OOA	On-Off Attack
IDPIoT	Intrusion Detection and Prevention system for the Internet of Things
DoS	Denial-of-Service attacks
DDoS	Distributed Denial-of-Service attacks
XSS	Cross-Site Scripting
SQL	Structured Query Language
PoTA	Proof of Trust Attacks
PACS	Preuve d'Attaques de Confiance basées sur Spark
MAC	Medium Access Control layer
BARS	Blockchain-based Anonymous Reputation System
VANETs	Vehicular Ad-hoc NETworks
SAGA-BC	SPAM Attack Guard Algorithm Using Blockchain
CoI	Community of Interest
OSN	Online Social Network
DOSN	Decentralized OSN
ML	Machine Learning
MLlib	Machine Learning library
RF	Random Forest algorithm
LR	Logistic Regression algorithm
SVM	Support Vector Machine algorithm

SGD	Stochastic Gradient Descent
DT	Decision Tree algorithm
S-DDoS	Spark-based real-time DDoS detection system
MLP	Multi-Layer Perceptron
IDS	Intrusion Detection System
UDP	User Datagram Protocol
UDP flooding	DDoS attack type
TCP	Transmission Control Protocol
TCP flooding	DDoS attack type
ICMP	Internet Control Message Protocol
ICMP flooding	DDoS attack type
SSK-DDoS	Spark Streaming and Kafka to classify different types of DDoS attacks
RFID	Radio Frequency Identification
NFC	Near Field Communication
GPS	Global Positioning System

Résumé

L'IoT social est un nouveau paradigme qui améliore la navigabilité des réseaux IoT et stimule la découverte de services en intégrant les contextes sociaux. Néanmoins, ce paradigme est confronté à plusieurs défis qui réduisent la qualité de ses performances. La confiance, en particulier les attaques de confiance, est l'un des défis les plus importants. Certains utilisateurs adoptent des comportements malveillants et lancent des attaques pour propager des services malveillants. Un mécanisme de gestion de la confiance est devenu une exigence majeure dans l'IoT social pour prévenir ces attaques en temps réel et garantir des expériences dignes de confiance pour les utilisateurs finaux.

Cependant, peu de travaux ont abordé les questions de gestion de la confiance pour prévenir les attaques de confiance dans les environnements de l'IoT social. La plupart des études ont été menées pour détecter les attaques en mode hors ligne avec ou sans spécification du type d'attaque réalisée. En outre, elles n'ont pas pris en compte les propriétés de sécurité, telles que la cryptographie, la transparence et l'immutabilité, etc. A cet ègard, nous devons traiter les transactions en continu pour prévenir ces attaques au niveau de la génération des transactions en temps réel tout en maintenant les propriétés de sécurité. Pour ce faire, nous avons comparé les techniques et technologies utilisées précédemment, dont le point commun est la prévention des attaques dans les contextes sociaux et l'IoT. Sur la base de ces comparaisons, nous avons indiqué que la technologie blockchain peut aider à développer un mécanisme de gestion de la confiance qui peut prévenir les attaques de confiance tout en maintenant la sécurité. Pour le temps réel, nous avons proposé de combiner un moteur de traitement de flux distribué, connu sous le nom d'Apache Spark, avec la technologie blockchain. Notre choix est basé sur une comparaison des moteurs de traitement de flux de données open-source.

En conséquence, nous proposons un nouveau mécanisme de gestion de la confiance, basé sur la blockchain et Apache Spark. Ce mécanisme permet de prévenir en temps réel tous les types d'attaques de confiance effectuées par des nœuds malveillants, afin d'obtenir un environnement fiable. L'expérimentation réalisée sur un jeu de données réelles nous permet de prouver la performance de notre proposition.

Mots clés

Internet des Objets (IoT) ; IoT social ; Mécanisme de gestion de la confiance ; Prévention des attaques de confiance ; Temps réel ; Blockchain ; Apache Spark ; Apprentissage Automatique ; Apprentissage Profond.

Abstract

The social IoT is a new paradigm that enhances the navigability of IoT networks and boosts service discovery by integrating social contexts. Nonetheless, this paradigm faces several challenges that reduce its performance quality. Trust, particularly trust attacks, is one of the most significant challenges. Some users resort to malicious behaviors and launch attacks to propagate malicious services.

A trust management mechanism has become a major requirement in Social IoT to prevent these attacks in real-time and ensure trustworthy experiences for end-users.

However, few studies have addressed trust management issues to prevent trust attacks in Social IoT environments. Most studies have been conducted to detect offline attacks with or without specifying the type of attack performed. Moreover, they did not consider security properties, such as cryptography, transparency, and immutability, etc. In fact, we must continuously process transactions to prevent these attacks at the transaction generation level while maintaining security properties. For this, we compared the previously used techniques and technologies, whose common point is attack prevention in the SN and IoT areas. Based on these comparisons, we indicated that blockchain technology can assist in developing a trust management mechanism that can prevent trust attacks while maintaining security. For real-time prevention, we proposed the combination of a distributed stream processing engine, known as Apache Spark, with blockchain technology. Our choice is based on a comparison of open-source data-stream processing engines.

As a result, we propose a new trust management mechanism, based on blockchain and Apache Spark. This mechanism permit to prevent in real-time all trust attack types performed by malicious nodes, in order to obtain a reliable environment. Experimentation made on a real data-set enable us to prove the performance of our proposition.

Keywords

Internet of things (IoT); Social IoT; Trust Management Mechanism; Trust attacks Prevention; Real-Time; Blockchain; Apache Spark ; Machine Learning ; Deep learning.