جامعة السلطان مولاي سليمان
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵙⵙⵓⵍⵟⴰⵏ ⵎⵓⵍⴰⵢ ⵙⵍⵉⵎⴰⵏ
Université Sultan Moulay Slimane

ROYAUME DU MAROC

Université Sultan Moulay Slimane

Faculté Des Sciences et Techniques

Formation Doctorale : Mathématiques et Physiques Appliquées

Spécialité : Informatique et Télécommunications

THÈSE DE DOCTORAT

Par

Mme. SAHMI Imane

Sous le thème

Amélioration de la sécurité des Objets Connectés (IoT) utilisant le protocole MQTT dans le domaine de la E-santé

Soutenue le 18 Mars 2022 devant le Jury composé :

Pr. Said MELLIANI	PES	Président	Faculté de Sciences de Béni Mellal
Pr. Belaid BOUIKHALENE	PES	Rapporteur	Faculté Poly disciplinaire de Béni Mellal
Pr. Noura AKNIN	PES	Rapporteur	Faculté de Sciences de Tétouan
Pr. Norelislam ELHAMI	PH	Rapporteur	ENSA de Kénitra
Pr. Hanaa HACHIMI	PH	Examinateur	Université Sultan Moulay Slimane
Pr. Tomader MAZRI	PH	Co-directeur	ENSA de Kénitra
Pr. Nabil HMINA	PES	Directeur	Université Sultan Moulay Slimane

LABORATOIRE : GENIE DE SYSTEMES

ANNEE UNIVERSITAIRE : 2021-2022

Remerciement

Je voudrais tout d'abord exprimer mes sincères remerciements :

Au Professeur Nabil HMINA, mon Directeur de Thèse, qui a cru en mes compétences pour être une doctorante chercheuse, pour son soutien, pour sa confiance et pour sa disponibilité même avec ses grandes responsabilités.

Au Professeur Tomader MAZRI, ma co-Directrice de Thèse, qui était plus qu'une encadrante, la source du soutien continu au moment de faiblesse, Je la remercie infiniment pour la qualité de son encadrement et ses précieux conseils tout au long de ce travail.

Je tiens également à remercier les membres de Jury :

- Pr. Said MELLIANI
- Pr. Belaid BOUIKHALENE
- Pr. Noura AKNIN
- Pr. Norelislam ELHAMI
- Pr. Hanaa HACHIMI

D'avoir accepté d'évaluer ce travail et de l'enrichir par leur précieuses remarques.

Un grand merci pour les membres de deux centres de formations doctorales des universités Ibn Tofail et Sultan Moulay Slimane pour la qualité de leur formation doctorale.

Enfin je tiens à dédier ce travail :

A mes chers parents qui ont consacré leur vie pour ma réussite.

A ma petite famille, mon cher mari et mon fils.

A mes sœurs et mon frère.

Abstract

Nowadays, the Internet of Things will change our live mode, this revolution of the Information Communication Technology will be present in each domain of our life: transport, healthcare, agriculture, home, the city…. The huge volume of the transmitted confidential data is the blackhole on this technology, it must be protected and secured from malicious intention. Therefore, the security is a big challenge to be faced by the researchers. That's why, we choose to work on the security of the IoT, our contribution consists of proposing a new approach to secure the IoT applications working with the MQTT protocol, the lightweight and the most used protocol among the others application's protocols. This solution is based on the AugPAKE Algorithm to establish a secure session key between the broker and their clients (Publisher and Subscriber) and the PRESENT encryption to encrypt the payload message. This solution provides the mutual authentication, the confidentiality of the published message, the integrity, and the non-repudiation of the MQTT messages. To validate this approach, we choose to work on the smart healthcare as an application domain 'A smart healthcare monitoring' that visualize the data of the temperature sensor and pulse rate one, transmitted by Internet to the cloud 'Thingspeak', to be supervised by any part of Hospital system to make the necessary care.

Keywords: Internet of Things, Security, MQTT protocol, AugPAKE Algorithm, PRESENT encryption, Authentication, Confidentiality, Integrity, Non-repudiation, Smart healthcare monitoring, Thingspeak.

Résumé

De nos jours, l'Internet des Objets va changer notre mode de vie. Cette révolution de technologies de l'Information et de Communication va être présente dans n'importe quel domaine de la vie : le transport, la santé, l'agriculture, le ville etc… le volume énorme des données sensibles transmises présente le point noir de cette technologie, ces données doivent être protégés et sécurisés des intentions malicieuses. Par conséquent, la sécurité est un grand défi pour les chercheurs. C'est pourquoi, nous avons choisis de travailler sur la sécurité de l'IoT. Notre contribution consiste à proposer une nouvelle approche pour sécuriser les applications IoT qui se base sur le protocole MQTT, le protocole léger et le plus utilisé parmi les autres protocoles de la couche application. Cette solution est basée sur l'algorithme AugPAKE pour établir une clé de session sécurisée entre le courtier et ses clients (éditeur et abonné) et le chiffrement PRESENT pour chiffrer le message de charge utile. Cette solution fournit l'authentification mutuelle, la confidentialité du message publié, l'intégrité et la non-répudiation des messages MQTT. Pour valider cette approche, nous avons choisis de travailler sur le domaine de la santé comme domaine d'application. L'application 'Smart healthcare monitoring ' permet de visualiser les données du capteur de la température et le capteur de pulsions cardiaques, qui sera transmise par Internet vers le cloud 'Thingspeak', qui va être supervisé par le système hospitalier pour offrir les soins nécessaires.

Mots clés : L'Internet des Objets ; sécurité ; le protocole MQTT ; l'algorithme AugPAKE, chiffrement PRESENT, l'authentification, Intégrité, Non -répudiation, Surveillance intelligente des soins de santé.

Table de matière

Liste de figures

Liste de tableaux

Liste des abréviations

AAA: Authentification, Autorisation & Accounting
ABE: Attribute-Based Encryption
ACL: Access Control List
AES: Advanced Encryption Standard
AMQP: Advanced Message Queuing Protocol
AugPAKE: Augmented Password-Authenticated Key Exchange
CoAP: Constrained Application Protocol
CP-ABE: Ciphertext-Policy Attribute-Based Encryption
CPP: Chaos-based Privacy Preservation
DH: Diffie-Hellman
DTLS: Datagram Transport Layer Security
ECC: Elliptic Curve
HTTP: Hyper Text Transfer Protocol
IDS: Intrusion Detection System
IoT: Internet of Things
IP: Internet Protocol
JWT: Json Web Token
KP-ABE: Key-Policy Attribute Based Encryption
LAN: Local Area Network
LTE: Long Term Evolution
LWE: Learning With Errors
M2M: Machine to Machine
ML: Machine Learning
MQTT: Message Queue Telemetry Transport
O.Auth : Open Authorization
QoS: Quality of Service
REST: Representational State Transfer
RFID: Radio Frequency Identification
RSA: Rivest-Shamir-Adleman
SOAP: Simple Object Access Protocol
SSL: Secure Socket Layer
TCP: Transport Control Protocol
TLS: Transport Layer Security
UDP: User Datagram Protocol
URI: Uniform Resource Identifier
VPN: Virtual Private Network
WSN: Wireless Sensor Network
XMPP: Extensible Messaging and Presence Protocol

Liste de publications

Journaux internationaux

1. Sahmi I.,Abdellaoui A.,Mazri T., Hmina N.: "**MQTT-PRESENT: Approach to secure internet of things applications using MQTT protocol**" ; ***International Journal of Electrical and Computer Engineering*** *Open Access,*Volume 11, Issue 5, Pages 4577 – 4586 ,October 2021. (Indexé SCOPUS)

   DOI : 10.11591/ijece.v11i5.pp4577-4586

Conferences Internationales

2. Sahmi I.,Mazri T., Hmina N.: "**The Security of MQTT Against the Applications Protocols for IoT**", ***Lecture Notes in Networks and Systems*** ,Volume 183, Pages 1142 - 11542021 5th International Conference on Smart City Applications, SCA 2020 Karabuk,7 October 2020 through 9 October 2020 .(Indexé SCOPUS)

   DOI : 10.1007/978-3-030-66840-2_87

3. Sahmi I.,Mazri T., Hmina N.: "**Comparison between CoAP and MQTT in Smart Healthcare and Some Threats**", ***International Symposium on Advanced Electrical and Communication Technologies, ISAECT 2018 - Proceedings***18 January 2019 Article number 86186982018 International Symposium on Advanced Electrical and Communication Technologies, ISAECT 2018 Rabat 21 November 2018 through 23 November 2018. (Indexé SCOPUS)

   DOI: 10.1109/ISAECT.2018.8618698

4. Sahmi I.,Mazri T., Hmina N.:"**Study of the different security threats on the internet of things and their applications**", ***ACM International Conference Proceeding Series*** Volume Part F1481542019 Article number 682nd International Conference Networking, Information Systems and Security, NISS 2019 Rabat 27 March 2019 (Indexé SCOPUS)

   DOI: 10.1145/3320326.3320402

Book chapter

5. Sahmi I., Mazri T., Hmina N. (2019) "**Security Study of Different Threats in Internet of Things**". In: Ben Ahmed M., Boudhir A., Younes A. (eds) ***Innovations in Smart Cities Applications Edition 2.*** SCA 2018. Lecture Notes in Intelligent Transportation and Infrastructure. Springer, Cham.

   DOI: 10.1007/978-3-030-11196-0_64

Introduction générale

1. Motivation

De nos jours, L'internet des Objets (IoT) ou les objets connectés sont devenus une composante omniprésente dans notre vie quotidienne. Chaque appareil dans n'importe quel domaine est connecté à Internet afin de transférer les données nécessaires via des réseaux hétérogènes pour subvenir aux services des utilisateurs et faciliter leur vie dans plusieurs domaines à citer : le domaine du transport 'smart transportation', le domaine de la santé 'smart healthcare', le domaine de l'agriculture 'smart agriculture',etc.. Pour enfin vivre dans une ville intelligente qui rassemble l'ensemble de ces applications. Ils offrent de nouvelles opportunités dans l'utilisation des technologies du numérique et de la communication. L'Internet des Objets constitue le noyau de ce nouveau concept, il se base principalement sur les mécanismes, les protocoles, l'architecture en couches de ces objets connectés afin de garantir le bon fonctionnement. Selon une estimation de l'IHS, 50 à 80 milliards de nouveaux objets connectés seront mis en circulation au cours de l'année 2025, comme c'est mentionné dans la figure suivante :

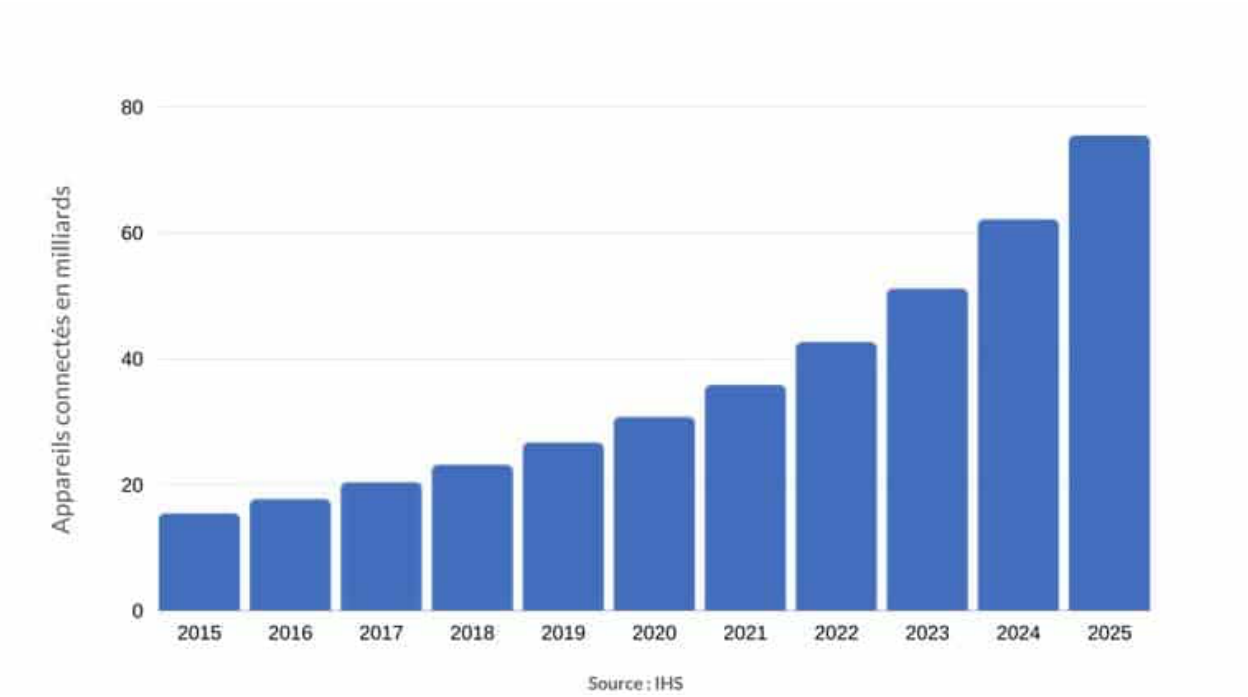

***Figure 1** : Nombre d'appareils connectés dans le monde de 2015 à 2025*

2. Problématique

Avec le nombre élevé des appareils qui seront connecté, des données énormes seront transférer via les réseaux hétérogènes, ces derniers seront traités, stockés dans des bases de données énormes, ces données qui sont de caractère personnel, elles sont sensibles et doivent être protéger et sécuriser. Les applications IoT sont présents dans plusieurs domaines ce qui a un impact direct sur la vie privée des utilisateurs, ces applications doivent respecter les normes de sécurité dont la confidentialité, l'intégrité et la protection de la vie privée. Sans ses mesures de sécurité, on ne pourrait pas faire confiance au dépoilement de ces applications. De ce fait, la sécurité est un volet primordial à prendre en considération dans le domaine de la recherche scientifique. C'est la raison pour laquelle nous avons choisi de travailler sur l'aspect de la cyber sécurité dans l'Internet des Objets ; plusieurs protocoles de la couche application dont le protocole MQTT ; le plus léger parmi eux ; sont nécessaire pour le fonctionnement de ces appareils qui ont des contraintes limitées, les mécanismes traditionnelles de sécurité ne sont pas convenable pour ces derniers , c'est la raison pour laquelle les chercheurs dans ce domaine ont un challenge primordial qui consiste à proposer des solutions convenables pour sécuriser cette technologie. C'est la problématique sur laquelle nous allons travailler lors de cette thèse qui a pour objectif de proposer une approche pour sécuriser le protocole MQTT, le protocole le plus utilisé dans les applications IoT, et cela est dû à plusieurs caractéristiques.

3. Contribution

Lors de cette thèse nous avons essayé de répondre à la problématique citée avant qui consiste à proposer une solution pour sécuriser les applications IoT en tenant compte des contraintes de ces ressources et la sensibilité des données transmises. Nos contributions sont réparties comme suit :

- Mener une étude sur les différentes menaces des applications IoT en indiquant les menaces sur les données et le réseau puis sur la protection de la vie privée et enfin sur les systèmes IoT. En citant quelques recommandations dans les différents aspects.
- Établir une étude comparative du protocole MQTT avec les autres protocoles de la couche application et étudier la vulnérabilité de ce protocole en ce qui concerne la sécurité.

- Proposer une approche pour sécuriser ce protocole basé sur l'algorithme AugPAKE et le chiffrement léger PRESENT et faire une analyse de la sécurité de cette solution.
- Faire une simulation de cette approche afin de la valider dans le domaine de la santé intelligente.

4. Organisation du manuscrit

Cette thèse est répartie en cinq chapitres :

Le premier chapitre consiste à donner un aperçu sur l'Internet des Objets qui détaille son architecture en couches, son fonctionnement, les différents protocoles de la couche application dont : CoAP, MQTT, XMPP et AMQP. Un état de l'art des différents défis vis-à-vis de cette technologie avant de mettre l'accent sur la sécurité.

Le deuxième chapitre mène une étude détaillée sur la sécurité dans les IoT et traite les différentes menaces de sécurité sur les données et le réseau, puis sur la protection de la vie privée pour enfin entamer celles sur les systèmes IoT, puis le chapitre enchaine sur un ensemble de recommandations sur les différentes catégories cités avant.

Le Troisième chapitre a pour objectif de faire une étude comparative du protocole MQTT par rapport aux autres protocoles d'applications en prenant en considération les paramètres tels que : la consommation de l'énergie, la qualité de service, la latence, cette étude nous a amené à choisir de travailler sur ce protocole vu sa légèreté et son concept de Broker /clients afin de faciliter la communication dans un environnement IoT.

Le quatrième chapitre propose une approche de sécurité de ce protocole utilisant l'algorithme AugPAKE et le chiffrement PRESENT, après avoir effectué l'état de l'art sur les différentes solutions proposées pour sécuriser ce protocole, une analyse a été effectuée en termes de sécurité et une comparaison avec les autres travaux se basant sur le même algorithme utilisé.

Pour enfin conclure avec le cinquième chapitre ; il s'agit d'une simulation dans le domaine de la santé intelligente 'Smart Health Monitoring' qui consiste à visualiser les données captées par le capteur de la température et le capteur de pulsions et de les envoyer via WIFI à l'aide du composant ESP8266 en utilisant Arduino et le visualiser dans le cloud de 'Thingspeak' pour que ça soit accessible via un utilisateur donné qui fait partie du système

hospitalier pour superviser à distance l'état du patient. Avant cela une simulation de l'approche pour valider le fonctionnement de l'approche.

Vous trouverez ci-joint le flow chart du manuscrit :

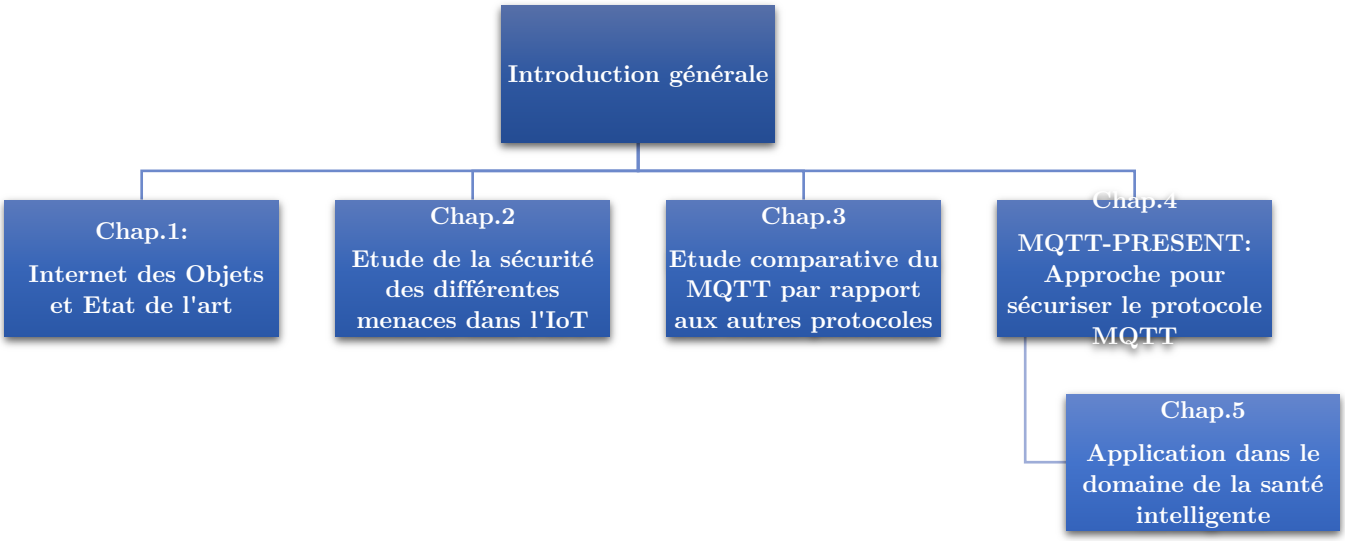

Figure 2 : *Flow chart du manuscrit*

Chapitre 1 : L'Internet des Objets et État de l'art

1. Introduction

Les objets connectés vont bientôt immerger notre quotidien, ils seront présents dans chaque domaine : la santé, le transport, l'agriculture, la maison pour le généraliser dans notre vie et parler de la 'Ville Intelligente'. Il s'agit du cœur de chaque application et chaque service développé dans le domaine de l'information et de la communication. C'est la révolution qui changera notre vie quotidienne. D'où la nécessité d'accorder de l'importance à ce nouveau concept pour mieux comprendre son fonctionnement. Dans ce chapitre, nous allons citer quelques définitions dans la littérature, qui explique le terme 'Internet des Objets', comment fonctionne cette technologie en faisant appel à d'autres concepts, son architecture en donnant une description de chaque couche et les différents protocoles qui sont utilisés et adaptés à ce contexte des applications IoTs.

2. Définition de L'Internet des objets

Les objets connectés constitue un système dont les objets sont intégrés avec des capteurs pour interagir entre eux via des communications sans fil dans le but de générer , échanger et transférer les données sans l'intervention humaine [1] . le terme 'Internet des Objets' s'est introduit pour la première fois comme idée par le chercheur Kevin Ashton en 1999 [2].Selon des prédictions de Cisco 30 milliards des objets connectés avec plus de 200 milliards de connexions vont générer un énorme volume de données en 2020. Jusqu'à maintenant il n'y a pas une définition officielle de ce que c'est l'internet des objets mais Tout d'abord on va commencer par un aperçu en littérature sur les différentes définitions qui existent de l'IoT :

L'Internet des Objets (IoT) est « un réseau qui relie et combine les objets avec l'Internet, en suivant les protocoles qui assurent leur communication et échange d'informations à travers une variété de dispositifs. »[3]

Ainsi **Weill et Souissi** ont proposé la définition de l'IoT comme étant « une extension de l'Internet actuel envers tout objet pouvant communiquer de manière directe ou indirecte avec des équipements électroniques eux-mêmes connectés à l'Internet. Cette nouvelle dimension de l'Internet s'accompagne avec de forts enjeux technologiques, économiques et sociaux, notamment avec les économies majeures qui pourraient être réalisées par l'ajout de technologies

qui favorisent la standardisation de ce nouveau domaine, surtout en matière de communication, tout en assurant la protection des droits et des libertés individuelles »[4].

Mais la définition la plus pertinente est la suivante l'IoT est « un réseau qui permet, via des systèmes d'identification électroniques normalisés et unifiés, et des dispositifs mobiles sans fil, d'identifier directement et sans ambiguïté des entités numériques et des objets physiques et ainsi, de pouvoir récupérer, stocker, transférer et traiter les données sans discontinuité entre les mondes physiques et virtuels. »[5].

L'IdO fait référence à "un réseau mondial d'objets interconnectés adressables de manière unique, basé sur des protocoles de communication standard" [6] dont le point de convergence est l'Internet.

Les objets connectés sont des objets intelligents et interconnectés capable de mesurer , déduire faire des modifications dans leur environnement dans une infrastructure de réseau ce qui implique des problèmes en terme de fiabilité , de performance , de sécurité et de confidentialité [7].

3. Fonctionnement de l'IoT :

L'Internet des Objets (IdO) permet l'interconnexion des différents objets intelligents via l'Internet. Ainsi, pour son fonctionnement, plusieurs systèmes technologiques sont nécessaires. Citons quelques exemples de ces technologies.

« L'IoT désigne diverses solutions techniques (RFID, TCP/IP, technologies mobiles, etc.) qui permettent d'identifier des objets, de capter, stocker, traiter, et transférer des données dans les environnements physiques, mais aussi entre des contextes physiques et des univers virtuels. » [5]

En effet, bien qu'il existe plusieurs technologies utilisées dans le fonctionnement de l'IoT, nous avons mis l'accent seulement sur quelques-unes qui sont, selon Han et Zhanghang, les technologies clés de l'IoT. Ces technologies sont les suivantes : RFID, WSN et M2M, et sont définies ci-dessous :

- RFID (Radio Frequency Identification) : le terme RFID englobe toutes les technologies qui utilisent les ondes radio pour identifier automatiquement des objets ou des personnes. C'est une technologie qui permet de mémoriser et de récupérer des

informations à distance grâce à une étiquette qui émet des ondes radio [6]. Il s'agit d'une méthode utilisée pour transférer les données des étiquettes à des objets, ou pour identifier les objets à distance. L'étiquette contient des informations stockées électroniquement pouvant être lues à distance [3].

- WSN (Wireless Sensor Network) : c'est un ensemble de nœuds qui communiquent sans fil et qui sont organisés en un réseau coopératif. Chaque nœud possède une capacité de traitement et peut contenir différents types de mémoires, un émetteur-récepteur RF et une source d'alimentation, comme il peut aussi tenir compte des divers capteurs et des actionneurs [8]. Comme son nom l'indique, le WSN constitue alors un réseau de capteurs sans fil qui peut être une technologie nécessaire au fonctionnement de l'IoT.
- M2M (Machine to Machine) : c'est « l'association des technologies de l'information et de la communication avec des objets intelligents dans le but de donner à ces derniers les moyens d'interagir sans intervention humaine avec le système d'information d'une organisation ou d'une entreprise » [9]

4. Architecture des Objets Connectés :

L'Internet des objets repose sur une architecture composée de plusieurs couches flexibles car on va connecter des objets hétérogènes de différentes plateformes [10] , il existe un modèle basique composé de 4 couches , ce qui est mentionné dans la figure suivante [11]:

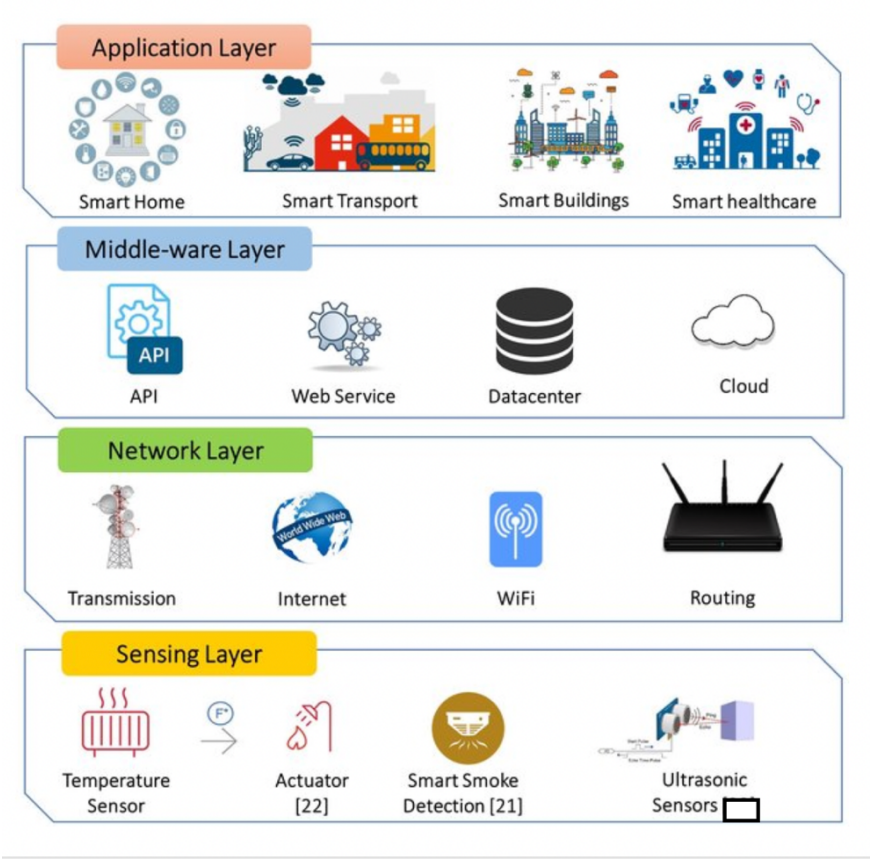

Figure 3 : *Architecture des IdOs*

Couche Perception : Cette couche présente des capteurs physiques des IoT dont l'objectif est de collecter et traiter l'information, cette couche de capteurs permet de développer plusieurs fonctionnalités telles que : la localisation, la température, l'humidité …. Elle numérise et transmet les données via des canaux sécurisés vers la couche Abstraction d'objet [12],[13].

Couche du réseau : elle se charge de transmettre les données produites par la couche inférieur vers la couche de management de service, elle va être transmise via plusieurs technologies comme : RFID, 3G, 4G, 5G,wifi, Bluetooth, zigbee etc [12].

Couche de management de service : cette couche se charge de plusieurs fonctionnalités : la réception des données, les stocker dans des bases de données, les mettre au niveau du cloud, développer des APIs et de services afin de les délivrer via le réseau. Elle permet aux programmeurs des applications IoT de travailler avec des objets hétérogènes sans considération de la plateforme [14] [13].

Couche application : Cette couche fournit les services demandés par les utilisateurs, à titre d'exemple cette couche fournit les mesures de la température et de l'humidité pour le

l'utilisateur qui demande auprès de ces données. Elle permet aussi de faire le design, l'analyse , l'implémentation , l'évaluation et le développement des systèmes de l'Internet des Objets liés aux éléments , de plus elle se charge de la surveillance et le management des couches inférieures , elle assure la confidentialité des données des utilisateurs en comparant les données reçues avec les données exceptées .L'importance de cette couche pour l'Internet des Objets c'est qu'elle fournisse une qualité supérieure des services intelligents pour répondre aux besoins des utilisateurs. Elle touche plusieurs aspects dans divers domaines tel que : la maison intelligente , le transport , l'industrie , la santé …[12] [13][14][15].

Pour le fonctionnement de cette couche, plusieurs protocoles organisent cette communication.

5. Protocoles de la couche d'application de L'Internet des Objets

5.1 Le protocole Constrained Application Protocol (CoAP)

CoAP Constrained Application Protocol : est un protocole de la couche d'application dédié pour les applications IoT. Il est lié à UDP (et non TCP) par défaut qui le rend plus approprié pour les applications IoT. En outre, CoAP modifie certaines fonctionnalités de HTTP pour répondre aux exigences IoT telles que la faible consommation d'Énergie et l'exploitation des liens avec moins de bruits [16].

CoAP est un protocole de transfert web qui a été conçu sur la base de REST (Representational state transfer) qui représente un moyen plus simple d'échanger des données entre les clients et les serveurs via http [17] . Le REST peut être considéré comme un protocole de connexion caché qui repose sur l'architecture client /serveur. Il est utilisé dans les applications de réseaux sociaux et mobiles et élimine l'ambiguïté en utilisant les méthodes HTTP : **GET**, **POST**, **PUT** et **DELETE**. Il permet aux clients et aux serveurs d'exposer et de consommer des services Web comme le protocole d'accès aux Objets simples (SOAP), mais de manière plus simple en utilisant les identificateurs de ressources uniformes (URI). CoAP permet à de minuscules appareils de faible puissance comme les objets connectés, le calcul et la capacité de communication en utilisant les interactions RESTful. Avec ce protocole les interactions deviennent bien plus simples à réaliser, une passerelle applicative assez légère

(correspondance entre les commandes REST et CoAP) se charge de l'adaptation d'un monde à l'autre [10].

Pour envoyer une donnée, un client envoie une requête CoAP ; qui contient le type de message (CON ou NON), l'identifiant du message (mid) et une action (GET, POST, PUT ou DELETE).

Si la requête de type CON, le serveur retourne une réponse qui contient, le type de message (ACK), le même mid de la requête et un code réponse (2.xx, 4.xx ou 5.xx).

La signification du code de réponse est :

- 2.xx signifie que la requête a été correctement reçue et traitée
- 4.xx signifie qu'une erreur a été rencontrée par le client
- 5.xx signifie que le serveur n'est pas capable de traiter la requête

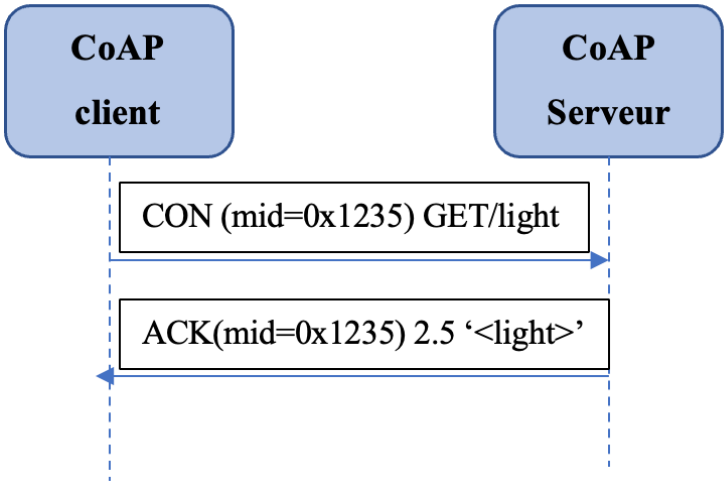

Figure 4 : *Fonctionnement de CoAP*

-Les types de requêtes utilisé dans CoAP (CRUD)

Tableau 1 : Correspondance entre RESET et CoAP-

HTTP	CoAP (CRUD)
GET	Create
PUT	Retrieve
POSTE	Update
DELETE	DELETE

Il peut être divisé sous forme de 2 sous couches :

- Sous couche de messaging
- Sous couche question/réponse

Il utilise aussi 4 types de messages :

- Confirmable : Message envoyé avec une demande d'accusé de réception, noté CON
- Non-confirmable : Message envoyé sans demande d'accusé de réception, noté NON
- Acknowledgment : Accusé de réception du message de type « confirmable », noté ACK
- Reset : Accusé de réception d'un message qui n'est pas exploitable, noté RST

Le protocole déploie 4 modes de réponse lors de son fonctionnement [16] : **le mode confirmé** (le client attend la réponse d'acquittement depuis le serveur) , **le mode non confirmé** (le client envoie la donnée sans attendre le ACK message du serveur, les IDs sont utilisé pour détecter la duplication , le serveur répond avec RST message quand les messages sont perdus) ; **le mode de réponse séparé** (il est utilisé quand le serveur a besoin d'attendre pour un moment avant de répondre au client) et **le mode biggypacked (**le client envoie un message pour demander une donnée spécifique et envoie un autre message pour indiquer si la donnée a été trouver ou non**).**

5.2 Le protocole Message Queue Telemetry Transport (MQTT)

MQTT Message Queue Telemetry Transport [10]: est un protocole de messagerie publication /souscription basé sur le protocole TCP/IP et non UDP, il a été initialement développé par Andy Stanford-Clark (IBM) et Arlen Nipper (EuroTech)et il a été standardisé par OASIS. C'est un protocole de messagerie idéal pour les communications IoT et M2M. Le modèle utilisé offre une flexibilité de transition et une simplicité d'implémentation. Il se compose de trois composants : Publisher, Subscriber et le Broker.

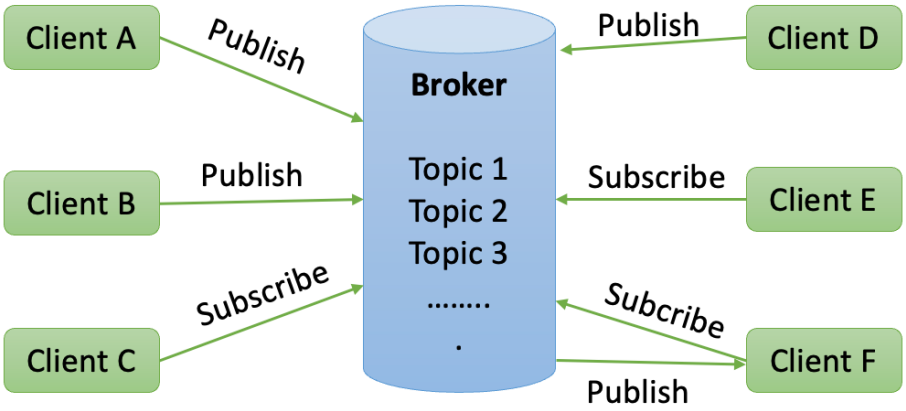

Figure 5 : *Le fonctionnement du protocole MQTT*

L'objet connecté est enregistré comme étant le subscriber qui va être informé par le Broker ; le publisher va publier le sujet intéressé, il fonctionne comme étant un générateur de données, le publisher transmet la donnée au subscriber via le broker, ce dernier s'occupe de la sécurité en vérifiant l'autorisation des éditeurs et des abonnées.

MQTT a la particularité d'être un protocole léger car le nombre des messages sont restreints et de faible taille puisque chaque message se compose d'une entête fixe de 2 octets, d'une entête variable facultatif et d'une payload (charge utile) limitée à 256 Mo. Il existe trois niveaux de qualité de service (QoS), plus le niveau de QoS est élevé et plus cela est gourmand en termes de latence et de bande passante.

De nombreuses applications utilisent MQTT telles que le domaine de la santé, la surveillance, le compteur d'énergie et la notification de Facebook. Par conséquent, le protocole MQTT permet de faire un routage pour des périphériques de petite taille qui ont une faible consommation d'énergie et une faible mémoire dans des réseaux à faible bande passante [18].

5.3 Extensible Messaging and Presence Protocol (XMPP)

L'XMPP est un protocole de messagerie instantanée IETF (IM) qui est utilisée pour les conversations multipartites, les appels vocaux et vidéo et la télé présence [19]. Il permet aux utilisateurs de communiquer entre eux en envoyant des messages instantanés sur Internet quel

que soit le système d'exploitation qu'ils utilisent. L'XMPP permet aux applications de messagerie instantanée d'accéder à l'authentification, au contrôle d'accès, à la mesure de la confidentialité, au cryptage hop-by-hop et end to end et à la compatibilité avec d'autres protocoles. Beaucoup de fonctionnalités de l'XMPP en font qu'il est l'un des protocoles préférés par la plupart des applications de messageries instantanées et pertinentes dans le cadre de l'IoT. Il fonctionne sur une variété de plateformes basées sur Internet de manière décentralisée. L'XMPP est sécurisé et permet d'ajouter de nouvelles applications au-dessus des protocoles de base [20].

L'architecture simple de XMPP se base principalement sur deux clients et un serveur. Les deux clients, avec un nom unique communiquent entre eux à travers un serveur associé, chaque client implémente sa propre forme de protocole.

Figure 6 : *Fonctionnement du protocole XMPP*

Dans un premier temps, un client établit une connexion TCP avec son serveur XMPP, qui communique la donnée au serveur XMPP du destinateur. Ce dernier transmet la donnée au destinateur s'il est connecté. S'il est déconnecté, le serveur XMPP mémorise la donnée tant que le destinateur ne s'est pas connecté, le système XMPP est décentralisé et potentiellement fonctionne en temps réel si l'émetteur et le destinateur sont connecté durant la livraison.

5.4 Advanced Message Queuing Protocol (AMQP)

L'AMQP est un protocole de Messagerie avancé de mise en file d'attente, il est un protocole de couche d'application standard ouvert pour l'IoT se concentrant sur des environnements axés

sur les messages. Il requiert un protocole de transport sécurisé comme TCP pour échanger des messages. Il prend en charge une communication fiable via des primitives de garantie de livraison de messages, en définissant un protocole au niveau du fil, les implémentations AMQP peuvent inter opérer entre elles. Les communications sont traitées par deux composants principaux : échanges et files d'attente de messages. Les échanges sont utilisés pour acheminer les messages vers les files d'attente appropriées. Le routage entre les échanges et les files d'attente des messages repose sur certaines règles et conditions prédéfinies. Les messages peuvent être stockés dans les files d'attente, puis envoyés au récepteur par la suite. AMQP prend également en charge le modèle de communication d'abonné/éditeur [21].

Le fonctionnement du protocole AMQP est basé sur le même principe que celui de MQTT, seulement le Publisher/Subscriber est remplacé par Producer/consumer. AMQP permet de router un message d'un producer vers plusieurs topics. Ainsi un même message peut être consommé par différents consumers via plusieurs topics.

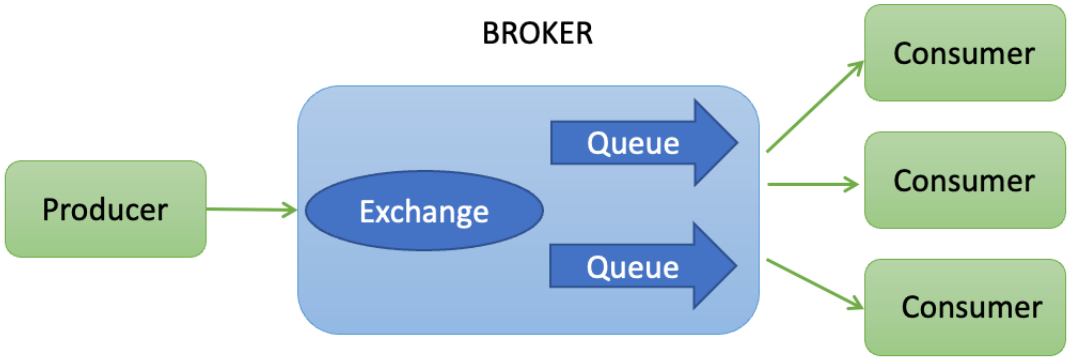

Figure 7 : *Fonctionnement du protocole AMQP*

6. État de l'art

Après avoir donné un aperçu sur le domaine de l'IoT, en détaillant ces couches, son fonctionnement et quelques protocoles de la couche d'application. Le domaine de l'IoT est un domaine mitige qui rassemble plusieurs disciplines, les chercheurs ont besoin de mettre la main dans la main pour répondre aux défis, aux opportunités offertes afin de résoudre plusieurs problématiques qui seront face aux utilisateurs de l'IoT.

6.1 La Connectivité

Comme la connectivité est un volet primordial pour le fonctionnement de cette technologie car nous avons besoin de connecter n'importe quoi, n'importe quand et n'importe où. L'importance de se concentrer sur les challenges de la connectivité et des réseaux de l'IoT. Dans l'IoT, un identifiant unique pour tous les appareils de communication exige de définir un plan d'adressage efficace pour les appareils IoT et l'intégration des standards de l'IP. Avec l'augmentation exponentielle du nombre d'appareils connectés ; des milliards et des billions d'appareils seront connectés dans le monde à cette fin, donc un besoin qui s'impose pour la gestion de l'identité qui permet de fournir une adresse efficace pour les appareils qui fonctionnent à l'échelle mondiale. De plus, un mécanisme approprié est également nécessaire pour accompagner la mobilité dans le monde de l'IoT.

La qualité du service dans le domaine de l'IoT est un autre défi pour les scientifiques. Comme il y a beaucoup d'appareils qui sont interconnectés et ayant les problèmes d'énergie en raison de leur taille et les dispositifs ou les capteurs qui ont un grand impact tout en choisissant le meilleur protocole de routage pour eux. Il faut élaborer un protocole qui répond à ces besoins. L'IoT utilise différents protocoles de routage avec peu de modifications comme utilisé dans un réseau sans fil ou un réseau ad hoc. C'est donc un autre défi pour la recherche afin de développer un protocole de routage pour l'IoT qui soit efficace en termes d'énergie, d'évolutivité et de performance.

6.2 L'efficacité énergétique

Comme nous avons constaté que les appareils IoT ont des contraintes en termes de mémoire, de puissance de traitement et d'autonomie de la batterie, etc. Par conséquent, les implémentations IoT doivent tenir compte de ces contraintes et utiliser la puissance de la batterie de ces appareils de manière efficace. Cela peut se faire en utilisant le traitement intelligent des données, c'est-à-dire qu'au lieu d'envoyer des données à tous les autres appareils du réseau, il faut faire un traitement et ensuite les envoyer aux appareils requis, et nous pouvons utiliser les fonctionnalités de parfeux au lieu de se connecter directement aux appareils. Voici quelques implémentations dans la littérature pour assurer l'utilisation efficace de l'énergie.

Des chercheurs [22] ont proposé un cadre de travail pour la surveillance et l'analyse de l'énergie des bâtiments sur la base de l'IoT. Le cadre se compose de trois couches telles que :
(1) Couche perception : utilisée pour l'acquisition de l'information intelligente sur le bâtiment,
(2) Couche réseau : utilisée pour l'intégration d'options de connectivité multiples et
(3) Couche applicative : interface pour la gestion des bâtiments, la consommation d'énergie et la surveillance via la plateforme cloud.

Liu et al. [23] ont proposé un cadre de gestion de l'énergie pour contrôler la consommation du niveau MAC des nœuds plus précisément, les décisions de contrôle sont optimisées sur la base des statistiques d'utilisation à long terme des tâches, avec des contraintes sur le délai de service de la tâche.

6.3 La sécurité

Sécuriser les applications IoT est un challenge primordial pour les chercheurs, c'est une condition sine qua non pour le déploiement de cette technologie, car ces applications touchent chaque domaine des utilisateurs dont leur vie privée et leurs données sensibles, ce qui implique que beaucoup d'effort ont été consacré à ce volet. Les mécanismes traditionnels ne sont pas adaptés aux appareils IoT vu leur contrainte en termes de ressources ce qui a poussée plusieurs recherches à se focaliser sur ce point crucial, le challenge majeur est de développer des mécanismes de sécurité légers pour des appareils à des contraintes limitées. Plusieurs études ont été mené à ce stade :

Yan et al. [24] Ont présenté une étude sur la sécurité et la protection des données personnelles pour les systèmes IoT, leur contribution se divise en quatre volets : celui de la limite des appareils IoT en terme de batterie et de puissance de calcul, ensuite ils ont exposé certains mécanismes de sécurité pour les dispositifs contraints, ils se sont concentré sur l'authentification et le contrôle d'accès pour les systèmes IoT , ainsi ils ont discuté les problèmes de sécurité dans les différentes couches sans aborder les autres points de la sécurité comme la confidentialité , la protection de la vie privée et l'intégrité .

Oracevic et al. [25] Ont considéré trois exigences principales de la sécurité qui sont la confidentialité, l'intégrité et l'authentification, puis ils ont présenté quelques solutions récentes

de la sécurité dans l'IoT, en discutant des orientations possibles de la recherche dans ce domaine.

Des auteurs dans [26] ont proposé l'utilisation des algorithmes de cryptage légers afin de fournir la protection des données et la confidentialité. D'autres ont proposé des solutions de sécurité divers à titre d'exemples : dans [27] les auteurs ont proposé un framework intelligent de sécurité pour les appareils IoT qui assure l'authentification et la confidentialité des données basés sur les chiffrement à clé symétrique et asymétrique quand la paire de clé est générée en utilisant le mécanisme Learning With Errors (LWE). Les auteurs [28] présentent un protocole de communication léger nommé Chaos-based Privacy Preservation (CPP) pour sécuriser un système de maison intelligente , les données sont chiffré par une clé symétrique qui est généré à l'aide d'un système basé sur le chaos, cette clé symétrique est mise à jour à chaque phase de transmission de données. L'analyse montre que le protocole proposé garantit la confidentialité, l'intégrité et l'authenticité des données tout en réduisant les coûts en termes de mémoire et les frais généraux de la communication.

6.4 L'extensibilité et l'évolutivité

C'est un autre challenge majeur pour les chercheurs, qui est dû aux nombres élevés des appareils qui seront connecté au réseau de l'IoT, les données qui seront générées par ces dispositifs seront énorme. Donc le problème de l'évolutivité doit être pris en considération lors du stockage des données, de la transmission des données et des protocoles existants pour l'IoT, même lorsque la taille du réseau ne cesse d'augmenter. Ci-dessous quelques implémentations considérant les problèmes d'évolutivité sont listées.

Dans [29] les auteurs ont proposé un système de stockage IoT sécurisé et évolutif pour l'agrégation des données dans les applications IoT. Pour assurer la sécurité, la méthode de partage secret est utilisée et pour l'évolutivité, multi-coefficient polynomial et le remplissage interne sont incorporés dans le système de partage secret dans lequel les blocs de données sont traités séparément. Les auteurs ont également fourni une infrastructure de système de stockage IoT distribué supportant les concepts de stockage de données.

Dans [30], les auteurs s'ont penché sur l'évolutivité des applications M2M par rapport aux réseaux mobiles LTE (Long Term Evolution) de nouvelle génération. Les auteurs ont étudié

les frais généraux de communication créés par un grand nombre d'objets sur les réseaux mobiles et ont effectué des analyses expérimentales à l'aide d'un banc d'essai de simulation LTE pour une telle étude. Sur la base de cette analyse, les auteurs ont observé que la signalisation et la charge du trafic de données s'étendent linéairement au nombre d'appareils connectés.

7. Conclusion

Ce chapitre a été consacré pour définir le concept de l'Internet des Objets , ses différentes couches , les divers protocoles de la couche application , pour enfin conclure par l'état de l'art des différents challenges parmi eux : la connectivité qui devra faire face au nombre élevé des appareils connectés, l'efficacité énergétique qui doit tenir compte des ressources contraintes de ces équipements en terme de de batterie et de puissance de calcul , la sécurité qui est un volet primordial pour faire confiance à cette technologie vu les données personnelles sensibles échangées et enfin l'évolutivité et l'extensibilité.

Nous avons choisi de travailler sur le volet de sécurité vu son importance et son degré de criticité vis-à-vis des données sensibles transmises, nous allons en premier lieu effectuer une étude sécuritaire des différentes menaces dans l'Internet des Objets.

Chapitre 2 : Étude sécuritaire des différentes menaces dans l'Internet des Objets

1. Introduction

La sécurité des systèmes IoT est un point crucial et elle menace la sécurité des systèmes traditionnels, les solutions de sécurité traditionnelles ne serviront pas aux systèmes IoT vu qu'elles exigent souvent beaucoup de ressources contrairement au systèmes IoT qui sont limités en termes de capacité de calcul, de mémoire, de durée de vie de la batterie et de la bande passante du réseau. Par conséquent, il n'est pas possible de déployer les solutions traditionnelles vu que :

- Les systèmes IoT sont fortement distribués et hétérogènes, cet aspect ajoute des difficultés et des contraintes à leur protection.
- Ils sont déployés dans un environnement physique imprévisible. Ainsi, une multitude de menaces physiques et d'attaques entravent la sécurité des IoT.
- Ils sont connectés à Internet, on peut accéder à chaque équipement via son adresse IP, donc une autre menace qui s'ajoute lier à Internet.
- Le grand nombre d'objets connectés contraints génèrent un grand volume de données, il est donc facile d'attaquer ces petits appareils et d'inonder la bande passante du réseau.
- Ils couvrent une multitude de protocoles et de technologies hétérogènes dans un même système. Par conséquent les solutions proposées doivent prendre en considération l'ensemble de ces protocoles et de technologies dans une même proposition.

2. Sécurité de l'Internet des Objets

Puisque les objets connectés ont devenu un élément clé dans l'Internet et l'infrastructure nationale et internationale, le besoin de développer l'aspect sécurité pour les infrastructures devient un point crucial et important. Beaucoup de recherches ont été mené dans différents domaines dans le but de rendre les objets connectés plus sécurisés de ces intentions malicieuses [31]. L'internet des objets coopèrent avec plusieurs réseaux hétérogènes ce qui implique qu'on devrait traiter les problèmes de compatibilité entre ces différents réseaux ; qui a un impact direct pour pallier les problèmes de la sécurité.

La sécurité dans les objets connectés est un volet primordial dans cette technologie qui va envahir le monde dans quelques années ; L'IoT est susceptible d'être attaqué plus qu'internet,

c'est la raison pour laquelle, nous allons détailler les différentes menaces en les subdivisant en catégorie : des menaces sur les données et le réseau, des menaces sur la protection de la vie privée et enfin des menaces sur les systèmes IoT.

2.1 Menaces sur les données et le réseau

En étudiant le mode de vie des utilisateurs de ces appareils, les objets communicants vont traiter des problèmes de la protection de la vie privée et des données personnelles. Par ailleurs un objet connecté implique un objet vulnérable et pose donc d'importantes questions en termes de cybercriminalité. Ce système IoT produira une grande quantité de données connues par le Big data, par ailleurs un autre volet qui s'ouvre sur la gestion et la sécurité des Big data. Sans mesures prises, les données stockées seront accessibles. La sécurité de Big data pour les objets connectés : tous les appareils conçus dans les réseaux des objets connectés vont générer des données et ils auront besoin d'espaces pour les stocker. Pour assurer la sécurité du traitement de ces données, incluant le transfert , la maintenance et la synchronisation de tous ces données depuis plusieurs appareils sans compromettre aucune partie de ce système , cela demande beaucoup d'effort et d'attention [32], plusieurs attaques sont susceptibles pour ces données, c'est ce qui est résumé dans ce tableau :

Tableau 2. Les différentes attaques physiques

Type de l'attaque	Description de l'attaque
Falsification des nœuds	L'attaquant peut causer un dommage au nœud capteur, en remplaçant entièrement tout le nœud ou une partie du matériel ou bien en interrogeant les nœuds pour avoir accès ou altérer l'information sensible [33].
Interférences RFIDs	L'attaquant envoie des signaux bruits via des radios fréquences. Ces bruits vont interférer avec les signaux RFID entravant la communication [34].

Brouillage de nœuds dans les WSN	L'attaquant va faire des interférences avec les réseaux de capteurs sans fil à l'aide des fréquences Radio, ça peut causer des bruits et entraver la communication entre les nœuds ce qui va causer un déni de service pour les objets connectés [35]
L'injection de nœuds malicieux	L'attaquant implémente de nouveaux nœuds malicieux entre les nœuds existants dans le réseau IoT , dans le but de Controller le flux de données , cette attaque est connue sous le nom de l'attaque 'Man in the Middle ' [33]
Dommage physique	L'attaquant peut causer directement un dommage physique des appareils de l'Internet des objets dans le but de causer des incidents sur la disponibilité du service [33]
Social Engineering	L'attaquant manipule les utilisateurs d'un système IoT dans le but d'extraire des informations privées. [33]
Sleep Deprivation Attack	Cette attaque, maintient les nœuds tout le temps éveillés qui se traduira par une grande consommation d'énergie et qui provoquera l'arrêt des nœuds [33]
Injection de code malicieux	L'attaquant compromet un nœud en injectant physiquement du code malveillant qui lui donnerait accès au système IoT [33]

Ces attaques se résument dans la destruction, la délocalisation et le masquage des nœuds , les attaques de répétition , d'injection et le clonage et reprogrammation des nœuds , ces derniers rendent la ressource fournie par le nœud inaccessible [36].

Il existe plusieurs types d'attaques dans la couche réseau qui nuit au bon fonctionnement des communications des objets connectés ou fraude à la sécurité des données qui circulent dans le réseau utilisé. Ces attaques se résument comme suit :

Tableau 3. Les différentes attaques du réseau

Type de l'attaque	Description de l'attaque
Traffic Analysis Attacks	Un attaquant peut renifler les informations confidentielles des technologies RFID en raison de leurs caractéristiques sans fil [37].
RFID Spoofing	Un attaquant usurpe un signal RFID pour lire et enregistrer une transmission de données à partir d'une balise RFID puis l'attaquant peut envoyer ses propres données contenant l'ID de la balise d'origine, donc l'attaquant obtient un accès complet au système prétendant être la source d'origine [38].
RFID Cloning	Un attaquant clone une balise RFID en copiant les données de la balise RFID des victimes sur une autre balise RFID [33].
RFID Unauthorized Access	L'attaquant peut lire, modifier ou même supprimer des données sur les nœuds RFID, en raison de l'absence de mécanismes d'authentification appropriés dans la majorité des systèmes RFID [39] .
Attaque de Man In the Middle	L'attaquant sur le réseau parvient à interférer entre deux nœuds de capteurs, accédant aux données restreintes, violant la confidentialité des deux nœuds en surveillant, en écoutant et en contrôlant la communication entre les deux nœuds de capteurs [40].

Denial of Service	Un attaquant peut bombarder un réseau IoT avec plus de données de trafic qu'il peut gérer, ce qui peut entraîner une attaque par déni de service réussie [33].
Routing Information Attacks	En usurpant et en modifiant ou en rejouant des informations de routage, vous pouvez saturer le réseau et créer des boucles de routage [41].
Attaque de Sybil	Un nœud malveillant est un nœud unique qui revendique les identités d'un plus grand nombre de nœuds, se faisant passer pour eux et conduit à de fausses informations étant acceptées par les nœuds WSN voisins [42].
Attaque de Sinkhole	Un adversaire compromet un nœud à l'intérieur du réseau. Ce nœud envoie les fausses informations de routage à ses nœuds voisins qu'il a le chemin de distance minimale à la station de base et attire ensuite le trafic. Il peut alors modifier les données et aussi déposer les paquets [43].

2.2 Menaces sur la protection de la vie privée

Nous allons définir tout d'abord la protection de la vie privée pour les IoT [31] : Étant donné que la plupart des informations contenues dans un système IoT peuvent être des données personnelles, il est nécessaire de soutenir l'anonymat et le traitement restrictif des informations personnelles. Ces données personnelles concernent plusieurs domaines tels que l'état de la santé : le rythme cardiaque, la tension…, de géolocalisation. Ce qui pose un problème majeur qui nuit à la confidentialité de la vie privée. Il y a de nombreuses techniques pour renforcer la protection de la vie privée :

- *Techniques cryptographiques :* qui permettent de stocker des données protégées traitées et partagées, sans que le contenu de l'information soit accessible à d'autres parties.

- *Techniques pour renforcer le concept de la confidentialité :* par la conception des données, en prenant en compte de l'identification, l'authentification et le respect de l'anonymat pour appliquer le concept de "Data minimization" qui consiste à utiliser les données collectées pour des raisons prédéfinies en avance.
- *Mécanisme de contrôle d'accès 'Fine-grain'* et d'auto-configuration émulant (simulant) le monde réel.

Les systèmes IoT produisent un volume important des données qui sont facilement accessibles à distance, la protection de la vie privée dans l'IoT devient de plus en plus difficile. L'adversaire n'a pas besoin d'être physiquement présent pour effectuer la surveillance, mais la collecte d'information peut se faire de façon anonyme avec un risque très faible. Les attaques les plus courantes contre la vie privée des utilisateurs sont les suivantes [44]:

Eavesdropping and passive monitoring : c'est l'attaque la plus connue et la plus facile à déployer, si les messages ne sont pas protégés via des mécanismes de cryptographie, l'adversaire peut facilement comprendre le contenu.

Traffic analysis **:** pour mieux réussir la violation de la vie privée, l'eavesdropping doit être combiner avec l'analyse du trafic, comme ça l'adversaire peut détecter des informations spéciales selon le rôle utilisé et dans quel secteur d'activités.

Data mining **:** Cela permet aux attaquants de découvrir des informations qui ne sont pas prévues dans certaines bases de données. C'est un problème majeur de la sécurité et de la vie privée dans l'internet des objets, l'information est disponible sans aucune demande, nous donnons peut-être plus que ce que nous avions prévu.

2.3 Menaces sur les systèmes IoTs

Les appareils de l'internet des objets ont des ressources limités (mémoire faible, capacité de calcul) qui leur rendent plus vulnérables que les autres appareils, donc cela parait nécessaire d'étudier les différents aspects de cette vulnérabilité :

La mémoire de l'IoT*:* la plupart des appareils ont une capacité limité de la mémoire ce qui demande une mémoire externe pour satisfaire le besoin , ce qui ouvrent l'opportunité à de différentes menaces dans le système [1]**.**

***Les interfaces web de IoT* :** la plupart des appareils IoT ont des interfaces web pour se connecter à la base de données de serveurs. La menace prédominante dans ce cas c'est la 'SQL injection' [45].

Les services réseaux de l'appareil IoT : L'incapacité d'exécuter des algorithmes de chiffrement de haut niveau sur les appareils IoT les rendent vulnérables aux attaques de divulgation d'informations. En raison des contraintes de ressources telles que la puissance de calcul et la capacité de stockage des données, les appareils IoT ne sont pas censés effectuer de vérification de la charge utile et de vérification de l'intégrité qui rendent le système non sécurisé [45].

La connectivité Cloud de l'appareil IoT : La majorité des appareils IoT seront connectés à l'aide d'une infrastructure Cloud qui pose de graves problèmes de sécurité. Les attaquants pourraient analyser les données en compromettant l'architecture en nuage [45]

Mise à jour logiciel des Objets connectés : La mise à jour logicielle est un processus qui ne doit jamais échouer ou compromettre dans un système basé sur l'IoT. La mise à jour manuelle des correctifs pour chaque appareil IoT peut ne pas être possible. Une approche basée sur le cloud est une solution. Mais comme le cloud a également des menaces de sécurité, la mise à jour des correctifs logiciels pour les appareils IoT fait toujours l'objet de recherches [1].

Méthode de stockage des données IdOs : En raison de la puissance de traitement et de la capacité de stockage limitées, les données stockées dans les appareils IoT peuvent être non cryptées. La plupart des appareils IoT ne prennent en charge que le chiffrement symétrique [1]

Les services AAA de l'IoT: En raison de la nature distribuée des systèmes IoT Les services d'authentification, d'autorisation et Accounting IoT (AAA) seront un grand défi pour l'appareil IoT [1].

Nous allons décrire quelques attaques logicielles qui sont résumé dans ce tableau.

Tableau 4. Les différentes attaques logicielles

Type d'attaque	Description de l'attaque
Phishing Attacks	L'attaquant peut accéder aux données confidentielles en usurpant les identifiants d'authentification d'un utilisateur, habituellement par le biais de courriels infectés ou la technique de 'phishing' [46].
Virus,Worms, Trojan Horse, Spyware and Aware	Un adversaire peut infecter le système avec un logiciel malveillant résultant vol d'information, falsification de données ou même déni de service.
Malicious Scripts	L'utilisateur qui contrôle la passerelle peut être trompé dans l'exécution de scripts exécutables Active-X qui pourrait entraîner un arrêt complet du système le vol de données. [47]
Denial of Service	Un attaquant peut exécuter des attaques DDoS ou un déni de service distribué sur le réseau IoT affecté via la couche application, affectant tous les utilisateurs du réseau [47]

3. Discussion

Après avoir détailler les différentes menaces de sécurité liée à l'IdO selon l'ordre proposé, on va se focaliser sur les attaques les plus dangereuses :

Parmi les attaques physiques ; l'injection du nœud malicieux est la plus dangereuse puisqu'elle peut arrêter définitivement le service aussi elle pourrait même modifier les données

transmises via les nœuds ce qui implique un déni de service de la transmission ou bien falsifier le contenu.

De point de vue réseau, l'attaque de Sinkhole est la plus risquée des attaques, puisque l'adversaire peut sniffer tout le trafic à partir de la station de base et il peut lancer d'autres menaces comme la modification des paquets ou les abandonner ;

De point de vue logiciel, les attaques de type Worm sont les plus dangereuse sur Internet puisqu'elle résulte soit un vol d'information, une falsification de données ou même un déni de service.

Après avoir détaillé les différentes attaques pour les systèmes IdOs, nous allons proposer quelques recommandations dans l'ordre proposé :

En ce qui concerne la partie système, les petits composants des IdOs ont des ressources limitées pour utiliser les algorithmes complexes, beaucoup de travaux de recherches sont en cours sur les batteries et la capacité de stockage pour améliorer les performances de ces appareils dans le but d'exécuter des algorithmes complexes pour prévenir les attaques, certainement il n'y a pas un système sécurisé à 100 % mais on pourra être plus vigilant en suivant ses recommandations.

En ce qui concerne la protection de la vie privée : les données personnelles utilisées au cours des communications en utilisant les appareils IdOs doivent être protégés via des mécanismes cryptographiques légères qui doivent être développé et supporté par ce type d'appareils pour qu'elles soient cachés de l'adversaire dans le cas où il aura accès.

De point de vue du réseau ; le fait d'utiliser des réseaux hétérogènes pose un problème majeur pour le sécuriser, c'est pour cela qu'il sera judicieux de choisir un réseau complet dont les appareils IdOs seront connectés au même réseau ce qui coïncide avec le déploiement du réseau 5G qui a beaucoup de ressemblance en termes d'architecture de couche et de protocoles avec les IdOs et pourra être proposé comme le réseau par excellence des Objets Connectés.

Après avoir achevé cette étude , nous allons se focaliser sur la sécurité des communications entre les IdOs utilisant le protocole MQTT qui est parmi les protocoles qui ne possèdent pas de mécanismes de sécurité et qui a seulement un mécanisme d'authentification sans capacité de chiffrement [48].

4. Conclusion

La sécurité de l'Internet des Objets est une condition sine qua non pour le dépoilement des applications IdOs, nous avons besoin de faire confiance à cette technologie avant qu'elle envahisse notre quotidien. Ce chapitre résume de façon générale l'ensemble des menaces et des vulnérabilités en les divisant par catégorie : sur les données et le réseau, sur la protection de la vie privée et enfin sur les systèmes IdOs. Nous avons aussi proposé quelques recommandations et des pistes de recherches pour chaque catégorie. Cette étude est une étape clé dans notre recherche, cela nous a permis de mettre l'accent sur l'aspect sécurité dans les systèmes IdOs et de pouvoir entamer notre recherche pour proposer des approches de sécurité dans le domaine des objets connectés.

Ce chapitre a fait l'objet d'une communication indexée dans une conférence Internationale et d'une publication d'un chapitre dans un livre prestigieux de Springer.

Chapitre 3 : Une étude comparative du protocole MQTT par rapport aux autres protocoles de la couche application dans l'Internet des Objets

1. Introduction

Dans quelques années, on pourra vivre dans une ville intelligente, ce concept qui va envahir le monde entier et qui va changer radicalement nos vies. Les villes ont devenues intelligentes non seulement en ce qui concerne la façon dont nous pouvons automatiser les tâches quotidiennes au profit des personnes, des bâtiments, des systèmes de circulation, mais aussi de la manière à pouvoir surveiller, comprendre, analyser et planifier la ville pour améliorer son efficacité, l'équité et la qualité de vie de ses citoyens en temps réel [49]. Les villes intelligentes sont la révolution pratique de l'IdO, avec l'objectif de fournir des services publics urbains efficaces, fiables et sûrs, grâce à leur gestion intelligente. Les villes intelligentes permettront aux systèmes IdOs de révolutionner notre façon de vivre et de mener nos affaires.

L'Internet des objets (IdO) est le cœur de la smart city, c'est-à-dire que l'IdO est le backbone technique des villes intelligentes. Il joue un rôle important pour relier tout l'ensemble à l'Internet à travers des protocoles spécifiques pour l'échange d'informations et de communications, la réalisation de reconnaissance intelligente, localisation, suivi, surveillance et gestion. Nous pouvons définir une ville intelligente comme « une ville intelligente et durable est une ville innovante qui utilise les technologies de l'information et de la communication (TIC) et d'autres moyens pour améliorer la qualité de vie, l'efficacité des opérations et des services urbains et la compétitivité, tout en veillant à ce qu'il réponde aux besoins des générations actuelles et futures en ce qui concerne les aspects économiques, sociaux et environnementaux [50]. L'utilisation de l'IdO peut rendre les villes intelligentes réalisables. Les smartphones, les compteurs intelligents, les capteurs intelligents et la RFID constituent essentiellement le cadre de l'IoT dans les villes intelligentes [51].

Le système IoT se compose de divers composants, y compris l'électronique, les capteurs, les réseaux, les microprogrammes et les logiciels. L'IoT est le réseau d'objets physiques interconnectés y compris les ordinateurs, les smartphones, les capteurs, les actionneurs, les appareils portables, les maisons, les bâtiments, les structures, les véhicules et les systèmes énergétiques. L'IoT assure la communication de nombreux types de systèmes et d'applications pour fournir des services toujours plus intelligents, fiables et sécurisés.

Dans ce chapitre, nous allons faire une comparaison du protocole MQTT comme étant le mieux placé pour les applications IoT avec les différents protocoles de la couche application : HTTP, CoAP et AMQP, avec une brève description pour chacun. Le reste du document est organisé comme suit : la section II donne un aperçu des communications IoT, la section III fournit une brève description de chaque protocole d'applications : MQTT, HTTP, CoAP et AMQP, la section IV comprend une étude comparative du protocole MQTT avec les autres protocoles et enfin nous proposons une nouvelle approche pour sécuriser le protocole MQTT.

2. Un Aperçu

2.1 L'intelligence de l'objet

L'UIT souscrit à la définition de l'IoT comme étant un réseau « disponible n'importe où, n'importe quand et par n'importe qui » [52]. L'Internet des objets est un réseau de « Objets » identifiables de façon unique auquel on peut accéder de n'importe où, n'importe quand et par n'importe quoi pour obtenir de l'information à partir de l'Objet. Les « Objets » sont programmables, ont des capacités de détection et d'actionnement et fournissent des services avec ou sans intervention humaine [52]. Il y a quatre composantes principales de l'IoT :

- *Objets :* sont des capteurs, des dispositifs informatiques embarqués, ou des systèmes embarqués qui peuvent transmettre et recevoir des informations sur un réseau pour contrôler un autre dispositif ou interagir avec un utilisateur,
- Le réseau local,
- L'Internet et
- Le nuage.

L'Objet peut effectuer différentes actions qui sont présentées dans la figure ci-dessous :

Figure 6 : *L'intelligence de l'objet*

Un Objet intelligent peut collecter les informations nécessaires, capable d'exécuter des commandes, transmettre et recevoir des messages, il peut jouer le rôle soit d'un capteur ou d'un actuateur ou même identifier et stocker des informations.

Vu l'utilisation de L'Internet des Objets dans plusieurs domaines tels que le transport intelligent ; la télémédecine intelligente et le management de l'énergie dans la ville intelligente [51], les appareils IoT sont une cible attrayante pour les pirates vu leur connexion à Internet [53], le fait qu'ils peuvent être contrôlée à distance [54] , l'absence de mécanismes de sécurité en raison de contraintes de ressources [53] et ils génèrent, échangent et consomment des données potentiellement sensibles et privées [55].

2.2 Les modèles de communications des IdOs

Dans [56], les auteurs ont présenté trois scénarios de communication IoT : *Scénario de base* : les appareils IoT échangent des données au sein d'un réseau local (LAN) *Scénario étendu* : les appareils IoT échangent des données sur Internet. *Scénario Cloud* : les appareils échangent des données avec les services Cloud.

De nombreux protocoles IdO ont été développés pour faciliter la communication dans ces scénarios. Dans le système IdO, nous devons envoyer et recevoir des mises à jour et des

commandes de contrôle, c'est pourquoi les protocoles de publish/subscribe sont adaptés aux communications IoT,

3. Les protocoles de la couche application dans l'Internet des Objets

Dans cette section, nous allons faire une comparaison des différents protocoles de la couche application qui va être résumé dans un tableau récapitulatif, ces protocoles sont :

- The Constrained Application Protocol (CoAP)
- Advanced Message Queuing Protocol (AMQP)
- The Hyper Text Transfer Protocol (HTTP)
- The Message Queue Telemetry Transport (MQTT)

Protocole	Développé par	Standardisation	Architecture	Protocole de transport	Propriétés	QoS	Références
CoAP	IETF	IETF	Request/response Publish/subscriber	UDP	-Faible capacité en RAM ou CPU -Réduire les frais généraux, améliorer la livraison des paquets -Sa simplicité soutenir les demandes de diffusion unicast et multicast	No QoS	[57] [58][59]
AMQP	John O'Hara at JPMorgan Chase	OASIS	Request/response Publish/subscriber	TCP	-Fiabilité, sécurité, approvisionnement -Hétérogénéité d'interopérabilité entre différents dispositifs.	2 niveaux préliminaires	[60][61] [62][63] [64]
HTTP	Tim Berners- Lee	IETF et W3C	Request/Response	TCP	-Connexions persistantes, -Demande de canalisation -Encodage de transfert morcelé	No QoS	[65] [66]
MQTT	Andy Stanford-Clark of IBM and Arlen Nipper of Arcom	OASIS	Publish/Subscriber	TCP/IP	-Modèle simple et faible utilisation de bande passante. -Capacité d'établir des communications entre les dispositifs distants. -Fournit un mécanisme de notification en cas de situation anormale. -garantit la fiabilité de la livraison des paquets.	Supporte 3 niveaux de qualité de services	[67] [68]

Tableau 5 : Comparaison des différents protocoles de la couche Application

4. Une étude comparative du protocole MQTT avec les autres protocoles

Dans cette section, nous proposons une étude comparative du protocole MQTT avec les autres protocoles définis ci-dessus : CoAP, AMQP et HTTP. Pour choisir le protocole adapté aux implémentations IoT, nous devons prendre en compte l'efficacité énergétique, les performances, l'utilisation des ressources (CPU, RAM et ROM) et la fiabilité basée sur la littérature [69],[48],[70].

Cette étude comparative est basée sur les résultats réalisés dans [70], notre contribution vise à présenter les résultats autrement pour faire une comparaison entre MQTT et les autres protocoles en tenant compte des différentes caractéristiques citées . Nous choisissons de regrouper ces caractéristiques pour réaliser notre comparaison :

- Afin de vérifier l'efficacité énergétique (consommation de l'énergie, Ressources nécessaires)
- Pour évoluer la performance (la bande passante et la latence)
- L'utilisation des ressource (CPU, RAM, ROM) c'est (taille du message et les frais généraux du message)
- Vérification de la fiabilité (QoS et la sécurité)
- Interopérabilité et l'utilisation M2M

Il est nécessaire de faire cette comparaison pour choisir le protocole approprié pour l'utiliser dans les applications de l'IoT ; le protocole conçu pour l'IoT doit être très léger en termes d'utilisation des ressources et de faibles performances utilisées mais la caractéristique la plus importante est celle qui sera désignée pour les communications IoT.

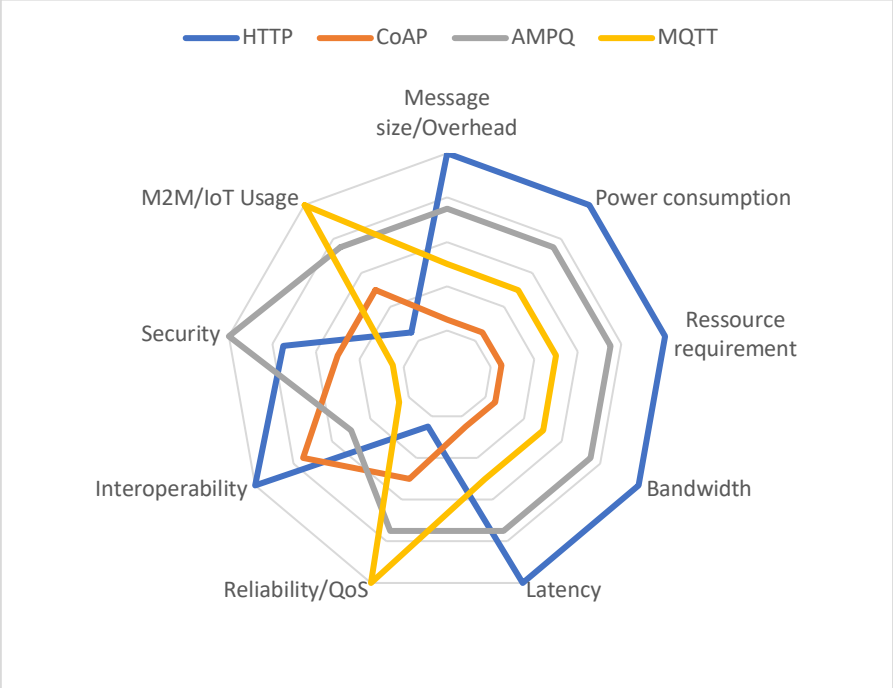

Figure 7 : *Graphe comparative des différents protocoles*

Le graphe révèle que le protocole HTTP consomme le plus de ressources, il a la taille de message et les frais généraux du transfert les plus élevés et la plus grande largeur de la bande passante et la latence que tout autre protocole, ce qui n'est pas en faveur pour les systèmes IoT qui ont des ressources limitées, alors que HTTP a été conçu pour une interopérabilité maximale sur le Web, de plus il est le standard dans le Web et ne comprenait pas la fiabilité comme caractéristique de base. Alors que, MQTT offre le plus haut niveau de qualité de service avec le moins d'interopérabilité parmi les quatre, avec moins de consommation d'énergie et d'utilisation des ressources ; il a la taille de l'entête la plus basse parmi les quatre, venant derrière le protocole CoAP qui utilise UDP au lieu de TCP qui augmente la latence, mais comme inconvénient majeur, il représente le niveau minimal de la sécurité. Alors qu'AMQP a le plus haut niveau de sécurité, de fiabilité et les services supplémentaires, mais il n'est pas léger pour les communications IdO, même si elle est plus sécurisée parmi les quatre. Enfin, le CoAP est léger comme le protocole MQTT, ouvert, simple et conçu pour être facile à mettre en œuvre.

On conclut lors de cette comparaison, que les deux protocoles CoAP et MQTT sont très légers et donc adaptés aux applications IdO, mais dans les scénarios où les nœuds capteurs

doivent communiquer avec les applications via le broker. En outre, MQTT fournit un niveau de QoS supérieur pour garantir la fiabilité des communications IdO, qui est une caractéristique importante pour sécuriser les applications IdOs.

Dans notre recherche, nous allons nous intéresser à ce type de systèmes qui ont besoin d'un broker pour effectuer les communications , c'est la raison pour laquelle nous allons nous concentrer sur le protocole MQTT car il est le plus léger parmi les protocoles IoT décrits, et les bibliothèques sont disponibles pour toutes les grandes plateformes de développement IoT, comme Arduino, pour plusieurs langages de programmation (C, Java, PHP, Python, Ruby, script Java) et pour les deux principales plateformes mobiles (iOS et Android).

Le protocole MQTT prend également en charge trois niveaux de qualité de service (QoS) qui peuvent être utilisés par les éditeurs en fonction des exigences de livraison des messages. Le tableau suivant présente ces niveaux de qualité de services [67]:

Tableau 6 : Description des différents niveaux de QoS

Niveau QoS	Comportement
QoS 1	Les messages sont transmis via le réseau, un accusé de réception n'est pas nécessaire
QoS 2	Les messages sont envoyés au moins une fois, on peut avoir des messages dupliqués, un accusé de réception est nécessaire.
QoS 3	Messages envoyés à tous les abonnés exactement une fois, aucun double, messages d'accusé de réception supplémentaires pour éviter les messages en double et la livraison garantie des messages. (Plus haut niveau de QoS)

Dans MQTT, l'éditeur envoie les données au courtier pour publication ; un abonné authentifie et s'abonne au courtier pour un sujet donné, le courtier envoie les données aux abonnés spécifiques qui sont abonnés au sujet spécifique. Les messages échangés entre les abonnés, les éditeurs et les courtiers utilisaient des paquets de contrôle MQTT. Dans ce modèle, les clients peuvent s'abonner à des sujets et recevoir des messages qui sont publiés sur sujets et, ils peuvent publier des messages à des sujets, donc les rendre disponibles à tous les abonnés à ces sujets [71].

Les caractéristiques les plus importantes du protocole MQTT sont sa simplicité et le fait qu'il ne nécessite pas d'utilisation élevée de CPU et de mémoire (protocole léger). C'est un taux d'échantillonnage élevé et une latence élevée. En outre, MQTT prend en charge un large éventail de différents appareils et plateformes mobiles [53].

La partie intéressante de la spécification MQTT est le fait qu'elle n'a pas de mécanismes de sécurité imposés, car elle est conçue pour fonctionner dans des réseaux sécurisés, développés pour des besoins spécifiques. Ainsi, nous avons identifié la nécessité d'améliorer la sécurité de ce protocole en particulier pour les applications privées des utilisateurs comme « Smart house », « Smart building » dans « smart city ».

5. La sécurité du protocole MQTT

Le protocole MQTT n'a pas encore de mécanisme de sécurité à exécuter sur les applications IdO, la plateforme doit être sécurisée, elle n'a qu'un mécanisme d'authentification sans capacités de chiffrement. Donc, c'est une grande faille pour les systèmes offrant l'opportunité aux pirates informatiques pour faire des attaques sur les réseaux. Plusieurs raisons expliquent pourquoi les implémentations IoT n'utilisent pas de mécanisme de sécurité comme [71] :

- *Dispositif à ressources limitées* : en raison des performances informatiques limitées, les dispositifs IoT ne peuvent pas utiliser le TLS pour le transport, ce qui est lourd pour le dispositif à contraintes limités.
- *Le nombre élevé d'appareils IoT* : il est très difficile à gérer si nous appliquons des mécanismes de sécurité par exemple l'authentification en utilisant le nom d'utilisateur et le mot de passe, c'est un grand défi pour l'équipe informatique de le gérer.
- *Manque de sensibilisation* à la sécurité.

De plus, les implémentations IdOs sont une cible pour les hackers. Les scénarios d'attaque se déroulent en surface d'attaque composée sur le réseau local et le réseau public. La première est analogue à une attaque interne où l'attaquant et le dispositif IdO sont dans le même réseau tandis que pour le réseau public est analogue à une attaque externe où l'attaquant est dans le réseau public pour attaquer le système IoT [72] : Par exemple, dans le réseau public, nous pouvons attaquer en faisant le déni de service, de sorte que le courtier connecté à ses clients

obtient et envoie des données incorrectes en scannant le réseau à l'aide du moteur de recherche 'Shodan'. Ensuite, dans le réseau local, un attaquant peut renifler ou modifier les données de paquets du réseau pour attaquer la confidentialité des données, l'intégrité des données et le mécanisme d'authentification MQTT [71].

Selon un ouvrage publié par l'ISACA [73] , l'objet de la sécurité de l'information comporte trois composantes :

- *Confidentialité des données :* protection du secret de la vie privée.
- *Intégrité des données* : garantir que les données transmises par la source et les données reçues sont identiques.
- *Disponibilité des données* : veiller à ce que les données soient accessibles aux utilisateurs finaux et aux applications, au moment et à l'endroit où ils en ont besoin.

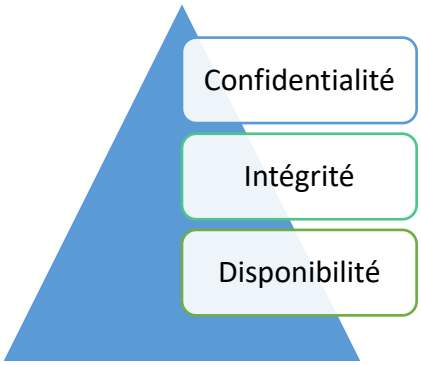

Figure 8 : *les Objectifs de la sécurité*

Voici un tableau qui résume quelques attaques qui menacent les objectifs de la sécurité :

Tableau 7 : Exemple d'attaques qui menacent les objectifs de la sécurité

Nom de l'attaque	But de l'attaque	Type de Menace
Man in the middle	-Intercepter la communication entre 2 parties et contrôler la conversation -Écouter, modifier ou supprimer l'information	Intégrité Confidentialité
Shinkhole	-Mettre un nœud à l'intérieur du réseau. -Changer ou supprimer la donnée	Intégrité Confidentialité

Sybil	-Implémenter un nœud qui prend la place d'un autre -Changer la donnée	Intégrité Disponibilité
DoS	-Saturer ou bloquer un serveur -Rendre un service non disponible	Intégrité Confidentialité Disponibilité
Flooding	-Épuiser la mémoire et l'énergie des nœuds -Saturer le réseau	Intégrité Disponibilité
Sniffing	-Capturer les trames circulantes et affiche leur contenu	Confidentialité

Il y a aussi les exigences de sécurité en terme de niveaux d'accès qui sont expliquées dans [72] :

- *Authentification* : elle garantit l'identité des nœuds appartenant au réseau MQTT, afin de prévenir les accès non autorisés (à la fois en tant qu'abonnés et éditeurs) : elle permet à un appareil IdO d'assurer l'identité du pair avec lequel il communique. Il garantit également que les utilisateurs qui sont valides d'avoir un accès aux périphériques et réseaux IdO pour les tâches administratives : reprogrammation à distance ou contrôle.
- *Autorisation* : Elle garantit que seuls les appareils autorisés et les utilisateurs ont accès aux services ou aux ressources du réseau.
- *Contrôle d'accès* : Pour garantir l'accès à l'information uniquement aux nœuds autorisés à accéder à cette information et à rien d'autre. Il s'agit de s'assurer qu'un nœud IdO authentifié ne pourra pas avoir accès que celui qui est autorisé et rien d'autre.

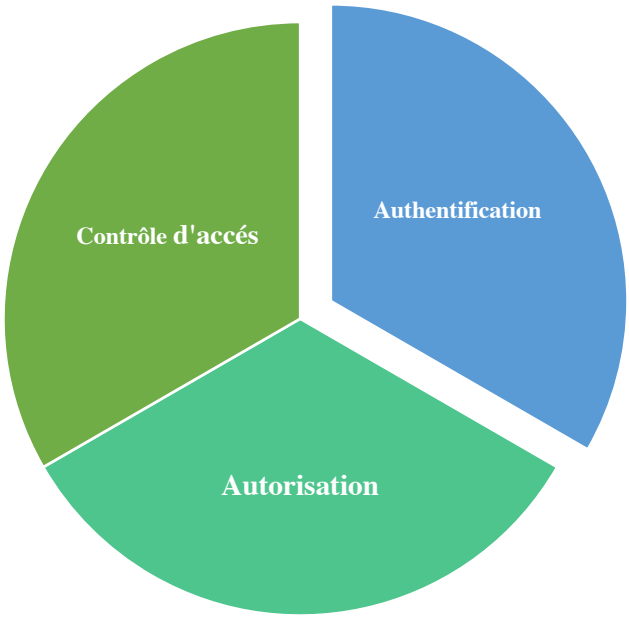

Figure 9 : *Les niveaux d'accès de la sécurité*

6. Les vulnérabilités du protocole MQTT

Le protocole MQTT comporte un certain nombre de vulnérabilités potentielles telles que les ports qui sont ouverts et peuvent être utilisés pour lancer des attaques de déni de service (DoS) aussi les attaques 'Buffer Overflow' qui peuvent se lancer à travers les réseaux et les appareils [74]. Selon le nombre des appareils IoT connectés et les cas d'utilisation pris en charge, la complexité de la structure des 'Topics' peut augmenter considérablement et entrainer des problèmes d'évolutivité. Les charges utiles des messages MQTT sont codées en binaire et les types d'applications et d'appareils doivent interagir.

Dans le système traditionnel, il existe une norme pour l'authentification pour MQTT qui s'appuie sur le chiffrement de transport avec SSL/TLS qui peuvent protéger les données lorsqu'il est correctement implémenté, il permet de protéger contre les menaces, les données sensibles, notamment les identifiants mais l'inconvénient de ce mécanisme c'est qu'il est un protocole coûteux qui ne convient pas aux dispositifs contraints. Lors de l'authentification, le nom d'utilisateur et le mot de passe seraient envoyés en texte clair, le client valide le certificat du serveur, ce qui signifie qu'il vérifie son identité pour l'authentifier. De nos jours, certains

courtiers acceptent des clients anonymes et, par conséquent, le nom d'utilisateur et le mot de passe ne sont plus requis, mais ce n'est pas souhaitable dans la plupart des cas.

Dans un réseau étendu, une mauvaise conception d'application basée sur MQTT pourrait être une opportunité aux pirates pour injecter des messages nuisibles dans le réseau d'une manière très facile [75]. Par conséquent, les mots de passe et tous les autres types d'informations d'identification doivent toujours être cryptés. Pour sécuriser le protocole MQTT, nous devons considérer la sécurité du système, du client et du courtier.

7. La sécurité basique du protocole MQTT selon les différentes couches

L'implémentation de la sécurité MQTT peut être effectuée sur les différentes couches afin d'éviter de nombreuses attaques. Il n'y a pas de système consolidé pour sécuriser l'architecture MQTT plutôt que de développer sur des normes déjà disponibles.

Couche de réseau : un réseau sécurisé ou un VPN sont une source efficace de sécurité dans la couche réseau chez le client et le courtier, ceci est utile pour fournir une connexion sécurisée et fiable. Les VPN aident à sécuriser de telles applications de passerelle lorsque les périphériques qui communiquent avec les passerelles sont dans une extrémité, tandis que VPN avec le courtier réside dans l'autre extrémité.

Couche de transport : le chiffrement est utilisé dans le codage de données afin qu'il ne soit accessible qu'aux utilisateurs autorisés et non visible aux utilisateurs non autorisés [76]. Le cryptage peut également être utilisé pour sécuriser la couche de transport. Le chiffrement du transport peut être réalisé avec DTLS (Datagram Transport Layer Security) /TLS (Transport Layer Security). TLS est attribué à TCP, tandis que DTLS est attribué à UDP [77]. Mais le TLS n'est pas pris en charge par les périphériques dotés de ressources limités, principalement en raison d'une surcharge de paquets.

Couche d'application : pour la sécurité des deux couches de Transport et d'Application, l'authentification peut être mise en œuvre. Le protocole TLS facilite l'authentification du serveur et du client avec un certificat dans la couche de transport. D'autre part, l'authentification peut être administrée en fournissant un nom d'utilisateur et un mot de passe au niveau de la couche d'application. Le client démarre une connexion sécurisée : le client transmet la demande pour créer la connexion au serveur avec une liste des options de cryptage

prises en charge. Le serveur sélectionne l'une des options et informe le client. Le serveur renvoie également la signature numérique, y compris le nom du serveur, l'autorité de certification et la clé publique du serveur. Le client le transmettra ensuite au serveur et la connexion sécurisée sera établie.

Pour s'authentifier auprès du courtier, un identifiant unique est fourni à chaque client. Cet identifiant unique est utilisé pour se connecter au courtier dans MQTT CONNECT. Il contient généralement 65 535 caractères, dont 23 dont supprimés. Il est recommandé d'inclure pour l'ID utilisateur un numéro de série ou l'adresse MAC du périphérique. Sinon, l'ID utilisateur doit comporter au moins 36 caractères. Le courtier MQTT effectue l'évaluation de l'authentification après avoir fourni le mot de passe et le nom d'utilisateur.

L'autorisation est également un moyen de sécuriser la couche d'application, dans laquelle les droits d'accès sont attribués à une ressource particulière. Dans MQTT, la liste de contrôle d'accès (ACL)est utilisée pour l'autorisation et déployée du côté du courtier. La liste de contrôle d'accès permet d'attribuer les droits d'accès et des activités autorisées à une tâche particulière. Tous les noms d'utilisateurs et les mots de passe appartenant à un utilisateur et ses rubriques publiés ou abonnées sont enregistrés dans ACL dans MQTT.

8. Conclusion

Au cours de ce chapitre, nous avons mis l'accent sur le protocole MQTT, le protocole le plus utilisé dans le monde des IoT, en faisant une étude comparative de ce protocole avec les autres protocoles de la couche d'application, pour enfin discuter sur la vulnérabilité de ce protocole vis -à vis l'aspect sécurité et détailler les mesures de sécurité pris en charge par ce protocole sur des normes déjà préétablies. Mais l'utilisation de MQTT dans des environnements contraints en termes de ressource ne s'adaptent pas avec l'utilisation du DTLS/TLS pour le sécuriser. De ce fait, les chercheurs dans ce domaine sont amenés à proposer des solutions adaptées à ces environnements pour assurer la sécurité aux applications IoT. Nous allons enchainer par un état de l'art de la sécurité de MQTT avant de proposer notre contribution qui consiste à proposer une solution basée sur le protocole AugPAKE et le chiffrement PRESENT.

Chapitre 4 : MQTT-PRESENT Proposition d'une approche pour sécuriser les applications de l'Internet des Objets utilisant le protocole MQTT

1. Introduction

Après avoir effectué une étude comparative du protocole MQTT par rapport aux autres protocoles de la couche Application et vu les vulnérabilités citées avant de ce protocole dans une réseau non sécurisé. De nombreux avantages font du protocole MQTT le protocole le plus utilisés dans les applications IoT [81][82]. Le MQTT est un protocole de messagerie simple et léger, Il existe certaines fonctionnalités de sécurité à prendre en compte dans un environnement IoT comme la confidentialité, l'intégrité et la disponibilité. C'est un grand défi pour les chercheurs de proposer des solutions qui vérifient ces caractéristiques adaptées aux dispositifs contraints. C'est dans cette perspective que nous proposons une approche de sécurité du protocole MQTT, une solution basée sur l'algorithme AugPAKE et le chiffrement PRESENT, qui assure l'authentification mutuelle entre le courtier et ses clients (éditeur/abonné), la confidentialité du message publié, l'intégrité et la non-répudiation des messages MQTT qui sont protégés pendant le processus de transmission.

Dans ce chapitre, dans la Section II, nous allons examiner les principales contributions dans la littérature concernant les solutions de sécurité proposées pour MQTT basé sur l'authentification, l'autorisation, et les mécanisme cryptographiques, et l'algorithme AugPAKE similaire à notre solution, en Section III nous allons détaillé les concepts théoriques utilisés dans notre approche (protocole MQTT et algorithme AugPAKE, chiffrement PRESENT), tandis qu'en Section IV nous décrierons la solution proposée basée sur l'algorithme AugPAKE, les différentes étapes en détail, et les messages échangés. Enfin, nous allons expliquer les principales caractéristiques approuvées par notre contribution et quelques travaux futurs.

2. L'État de l'art

La sécurité est un point crucial dans le domaine de l'IoT, beaucoup de chercheurs travaillent sur ce point afin de proposer une solution pour sécuriser les applications basées sur un dispositif contraint en termes de performances. Le grand défi consiste à appliquer des mécanismes de sécurité comme l'authentification et l'autorisation pour vérifier la confidentialité, l'intégrité et la disponibilité des données.

Certains auteurs proposent une solution basée sur l'authentification comme les auteurs dans [83] , ils proposent un mécanisme d'authentification léger basé sur un algorithme chaotique utilisant l'accord auto-clé et le chiffrement par blocs pour améliorer la sécurité MQTT, après la simulation, il a été mentionné que la confidentialité des données a été prouvée.

Dans [84] , Bhawiyuga A. propose une implémentation de l'authentification basée sur les jetons du protocole MQTT dans les périphériques contraints, mais la conception proposée se compose de quatre composants : Publisher, Subscriber, MQTT broker et Token authentication server. Ce dernier composant qui génère les jetons est un inconvénient pour la solution.

Dans [85] , les auteurs proposent une version sécurisée conçue pour les réseaux de capteurs sans fil en développant un système d'authentification à plusieurs niveaux pour garantir la confidentialité des données dans le système IoT en fonction de l'utilisation du chiffrement basé sur les attributs Ciphertext-Policy Attribute-Based Encryption (CP-ABE) ou Key-Policy Attribute Based Encryption (KP-ABE) avec la cryptographie à courbe elliptique légère (ECC).

D'autres travaux sont basés sur le mécanisme d'autorisation pour sécuriser le protocole MQTT comme dans [86] ,leur solution est basée sur O.Auth 1.0a une norme d'autorisation ouverte pour les applications Web, le système proposé est basé sur un ensemble d'informations d'identification (ID de l'appareil, Device secret, Access Token et Access Token Secret), certains de ces identifiants doivent être envoyés via un appareil qui ne doit pas être contraint et cela se fait via des messages HTTPS.

D'autres auteurs proposent un algorithme cryptographique pour sécuriser le protocole MQTT comme dans [87], ils proposent une solution de sécurité pour MQTT en combinant sur les S-box Attribute-Based Encryption (ABE) et Advanced Encryption Standard (AES). La solution est basée sur la cryptographie à clé publique et la cryptographie à clé secrète, de sorte que le double déchiffrement doit être fait par l'abonné ce qui implique l'utilisation de plus de ressources soit plus de frais généraux.

Une autre solution basée sur les attributs est dans [88] ; Singh propose une version sécurisée de MQTT et MQTT-SN pour les réseaux de capteurs (SMQTT, SMQTT-SN) avec la nouvelle commande « SPublish » qui publie des données chiffrées basées sur un ensemble de chiffrement basé sur les attributs (ABE), la politique de clés (KP-ABE) et la politique de chiffrement (CP-

ABE) utilisant la cryptographie elliptique légère (ECC). Il fonctionne lorsque l'éditeur doit proposer la politique d'accès basée sur un arbre d'accès et l'envoyer au courtier qui agit comme un générateur de clé publique, l'utilisation de (CP-ABE) provoque une surcharge qui est le principal inconvénient de l'utilisation de l'ABE.

En [89] , leur contribution consiste à sécuriser le flux des messages distribués entre les utilisateurs du protocole MQTT en utilisant un certificat d'autorité qui génère une clé privée et un certificat sur un sujet publié pour des clients certifiés .

La contribution dans [90] consiste à proposer une solution pour sécuriser MQTT en utilisant des listes de contrôle d'accès (ACL) intégrées dans un courtier Mosquitto qui agit comme un filtre permettant uniquement les données qui sont demandées, sauvant ainsi le flux de données ambiguës. La méthode (ACL) agit comme une sécurité supplémentaire à l'ensemble du processus, pour chaque nom d'utilisateur et mot de passe de données, sont créés pour accéder aux données. Cependant, cette solution peut entraîner plus de frais généraux, lorsque les données à transmettre augmente, les frais généraux aussi.

Deux travaux sont similaires à notre solution basée sur l'algorithme d'AugPAKE sont :

Dans [91], la solution est basée sur l'authentification avec mot de passe augmenté et l'échange de clés (AugPAKE) appelé 'AugMQTT'. Elle a été réalisée par l'établissement d'une clé de session entre l'éditeur et le courtier, et d'une autre clé de session entre le courtier et l'abonné, sans exiger de contrôle de validation de certificats ni de révocation de certification. Pour publier un message, l'éditeur utilise un schéma de chiffrement sécurisé à clé symétrique ou un chiffrement sécurisé authentifié avec schéma de données associé pour envoyer des données au courtier, le courtier utilise la même clé pour décrypter et stocker les données. Après demande nécessaire, le courtier transmet un message chiffré au souscripteur à l'aide de la même clé partagée avec le courtier. La solution proposée est mise en œuvre avec Eclipse Mosquitto, le broker open source qui permet d'implémenter le protocole MQTT avec ses clients, développé en langage C.

Dans [92], la solution 'MQTT-Auth' fournit deux jetons, le premier est un jeton d'authentification pour créer et publier sur un certain sujet basé sur une clé de session entre l'abonné/éditeur et le courtier à l'aide du protocole AugPAKE. Le second est un jeton

d'autorisation qui est connu uniquement par l'éditeur et certains abonnés choisis via un autre canal sécurisé secondaire. La solution utilise ActiveMQ comme un broker open source développé en JAVA.

Ce tableau résume ces solutions proposées pour sécuriser MQTT en donnant des avantages et des inconvénients de chacune.

Tableau 8 : Un résumé de certaines solutions pour sécuriser le protocole MQTT

Références	Solution proposé	Méthodes utilisés	Avantages	Inconvénients
[83]	A lightweight authentication mechanism	-Chaotic algorithm -Block cipher	Confidentialité des données	Ce n'est pas désigné pour des clients multiples.
[84]	A token-based authentication	-Json Web Token (JWT) authentication server.	Performance en termes d'authentification pour un jeton valide et expiré.	Le besoin d'une 3 partie : le serveur 'token authentication'
[88]	A secure version of lightweight MQTT	-Ciphertext-policy-based (KP/CP-ABE) -lightweight (ECC)	La protection de la vie privée Sécurisé contre l'attaque de collision et l'homme du milieu.	Le nouveau message 'SPublish' n'est pas encore déployé dans des plateformes IoT réels.
[86]	An authorization mechanism for MQTT security	-OAuth 1.0a -A set of credentials	Protection contre: eavesdropping attacks MITM attack, replay attack, node capturing attack	La partie AuthServer est nécessaire.
[87]	A composite security framework for MQTT	-Public-key cryptography -Secret-key cryptography	Fournit la confidentialité et le contrôle d'accès à grains fin des données.	Plus de frais généraux.
[89]	Secure the flow of distributed message between the users	-Certification Authority Private key	La solution offre un niveau acceptable de maturité dans différents cas pratiques.	La saturation du réseau dans le cas d'un nombre important de clients.
[90]	by using an Access Control Lists	La méthode ACL	Le courtier comme un filtre, et sauvegarde les données ambigües	Plus de frais généraux ;
[93]	Modern fuzzing based MQTT	-Fuzzer, -Sniffer,	Focalisé sur les tests.	Ne propose pas des solutions pour la sécurité

		-Mitmfuzzer		
[94]	A lightweight authentication mechanism	Based on hash and XOR operations preshared keys between sensors and servers	Faible coût de calcul, communication et frais généraux de stockage, contre les attaques suivantes : Replay, man-in-the-middle, usurpation et modification	Désigné pour les environnements IoT le protocole d'application n'est pas définie.
[91]	AugMQTT	AugPAKE algorithm	L'authentification du publishers /subscribers/broker, ainsi la confidentialité et l'intégrité des messages MQTT.	Le courtier n'est pas protégé, le message est crypté en code ASCII.
[92]	MQTT-Auth	AugPAKE algorithm, -an authentication token, -an authorization token	Garantit la confidentialité et l'autorisation pour accéder à un sujet.	Un deuxième canal est nécessaire pour les jetons.
[95]	A framework which supports 3 security levels for MQTT services	-Elliptic Curve Integrated Encryption	La sécurité des messages publiés non sensible mais ne sont pas modifiés. Des messages publiés sécurisés depuis des éditeurs authentifiés vers des abonnés authentifiés.	La solution dépend du niveau de sécurité de MQTT voulue.

L'état de l'art qui a été réaliser a pour objectif de proposer des solutions convenables pour sécuriser le protocole MQTT, notre contribution consiste à proposer une approche qui valide le maximum des avantages pour garantir une utilisation fiable des applications utilisant le protocole MQTT.

3. Les Bases Théoriques De L'approche

Dans notre approche, nous utilisons l'algorithme AugPAKE et le chiffrement PRESENT via le protocole MQTT, nous devons donc expliquer en détails chacun des concepts. Afin de proposer la nouvelle approche pour sécuriser le protocole MQTT.

3.1 Le protocole AugPAKE

Le protocole AugPAKE [95] est basé sur la modification du protocole d'échange de clés Diffie-Hellman (DH), qui est un algorithme d'échange de clés, il permet à deux entités de mettre en place un système par lequel elles pourront échanger de façon discrète. Cet algorithme permet de choisir une clé secrète de façon publique. Dans ce protocole, le client utilise une clé publique créée à partir du mot de passe pour s'authentifier sur le serveur qui n'a pas besoin de stocker les mots de passe du client. La session sécurisée est établie après une authentification réussie et un accord clé entre le client et le serveur. Le protocole AugPAKE échange 4 messages. L'AugPAKE est un protocole d'établissement de clés de session client-serveur qui utilise certains paramètres qui sont expliqués dans ce tableau :

Tableau 9 : Notation des paramètres

Variable	Description
G	The cyclic group of order q
g	Generator of G
p	A prime number such p=aq+1, with an integer
H	Hash function returning a binary string of length k
H'	Hash function returning an integer
C	Client identifier
S	Server identifier
||	Juxtaposition or concatenation of strings

L'algorithme AugPAKE est composé de 2 phase : la phase d'initialisation et la phase d'exécution actuelle.

The AugPAKE Algorithm

Inputs: C: client identifier w: client's password S: Server identifier

Initialization:

Client C computes $W = g^{w'} \bmod p$ (1)

where $w' = H(C \| S \| w)$

client C *send (C, W) to server S*

client C chooses x and computes $X = g^x \bmod p$ (2)

client C send (C, X) to server S

Traitement:

if **server S** received X equal to 0,1 or -1 (mod p)

server S discards the session

if not

server S chooses y and computes $Y = (XW^r)^y mod p$ (3)

where $r = H'(''01''\| C \| S \| X)$

server S send (S, Y) to the client C

if **client C** received Y equal to 0,1 or -1 (mod p)

client C terminates the protocol execution

If not

client C computes $K = Y^z mod\ p$ (4)

where $z = \frac{1}{x+w'r} mod\ q$

$r = H'(''01'', C, S, X)$ and $Vc = H(''02''\|C\|S\|X\|Y\|K)$ (5)

client C sent Vc to server S

if **server S** receives $Vc \neq H(''02''\|C\|S\|X\|Y\|K)$

with K= $g^y mod\ p$ (6)

server S stops the procedure

If not

server S generates an authenticator

$Vs = H("03"\|C\|S\|X\|Y\|K)$ (7)

server S sends Vs to client C

server S computes the session key

$Sk = H(''04''\|C\|S\|X\|Y\|K)$ (8)

If **client C** receives Vs $\neq H("03"\|C\|S\|X\|Y\|K)$

client C terminates the procedure.

If not

Client C computes

$Sk = H(''04''\|C\|S\|X\|Y\|K)$

3.2 Le chiffrement PRESENT

La cryptographie est un outil puissant pour garantir la confidentialité, l'intégrité et l'authentification. Le chiffrement est une technique de cryptographie qui consiste à protéger les données contre les actes malveillants afin d'assurer la confidentialité, l'intégrité et l'authenticité de l'information transmise. Vu les contraintes liées aux équipements IoT en termes de processeur, de mémoire et de batterie ; les algorithmes traditionnels de cryptographie ne sont pas adaptés aux équipement IoT de ressources restreintes. Actuellement, des primitives cryptographiques légères ont été proposer afin de sécuriser les systèmes IoT. Les algorithmes de la cryptographie légère peuvent être classifier en 4 classes : block ciphers, stream ciphers, fonctions de hachage et la cryptographie Elliptic Curve (ECC) [96] comme c'est mentionner dans la figure suivante :

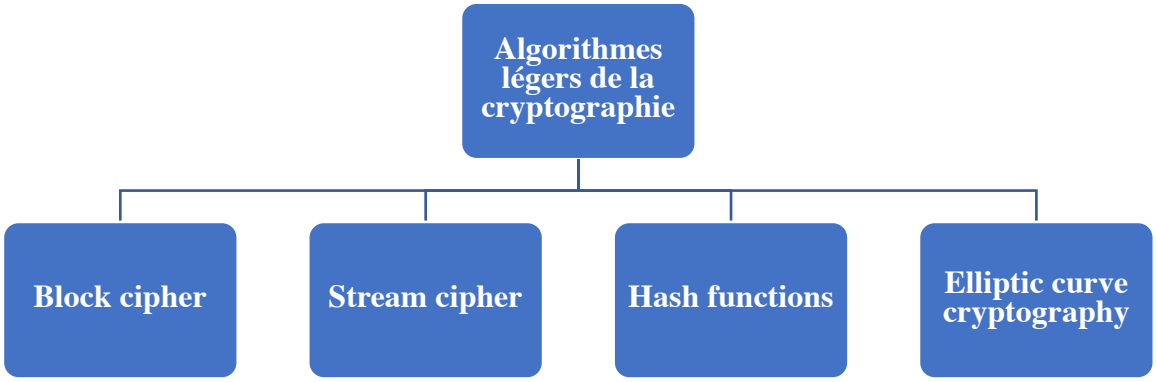

Figure 10 : *les différents algorithmes légers de la cryptographie*

Dans les chiffrements par blocs, un bloc de texte en clair est chiffré à la fois, tandis que les chiffrements par flux chiffrent/décryptent un seul bit ou octet de texte en clair/clair.

Les fonctions de hachage sont utilisées pour fournir l'intégrité des données en générant un message de longueur fixe à partir d'un message de longueur arbitraire. L'ECC est une technique cryptographique asymétrique légère qui fournit le même niveau de sécurité que Rivest-Shamir-Adelman (RSA) algorithme avec une taille de clé plus petite .

Dans notre cas, après avoir établie la session sécurisée à l'aide de AugPAKE, nous utiliserons le chiffrement par bloc de la charge utile encapsulé dans le message PUBLISH. Dans cette perspective, l'utilisation de la technique cryptographique est un choix intelligent à développer pour soutenir l'hétérogénéité, l'interopérabilité, la taille des clés, la faible consommation d'énergie, les ressources limitées des dispositifs IoT [97]. C'est pourquoi, nous utilisons le chiffrement PRESENT [98] comme un chiffrement par bloc léger, par rapport à Advanced Encryption Standard (AES), il est 2,5 plus petit que le dernier. C'est un exemple de l'algorithme cryptographique SP-Network. La longueur du bloc est de 64 bits et le chiffrement prend en charge les clés secrètes 80 et 128 bits, la procédure de cryptage est divisée en trois phases :

- Ajout de clé : les données subissent une opération XOR avec la clé à partir de laquelle seuls les derniers 64 bits du résultat sont pris pour cette opération. Ensuite, les données 64 bits sont divisées en 16 blocs contenant chacun 16 bits pour le processus suivant.
- Couche de substitution non linéaire : Chacun de ces blocs est ensuite passé par les cases de substitution où la valeur de ces cases est remplacée par les valeurs des blocs de substitution.
- Permutation binaire : Les données substituées sont ensuite passées par le bloc de permutation où tous les 64 bits des données d'entrée sont réorganisés. Après cela, la clé est mise à jour pour produire une autre clé qui est utilisée la prochaine fois dans le même processus entier et répété pour le même ensemble de données. Ceci est répété 32 fois, puis les données chiffrées sont transmises.

Algorithme de PRESENT

generateRoundKeys()

for i=1 to 31 **do**

addRoundKey(STATE;K_i)

sBoxLayer(STATE)

pLayer(STATE)

end for

addRoundKey(STATE,K_{32})

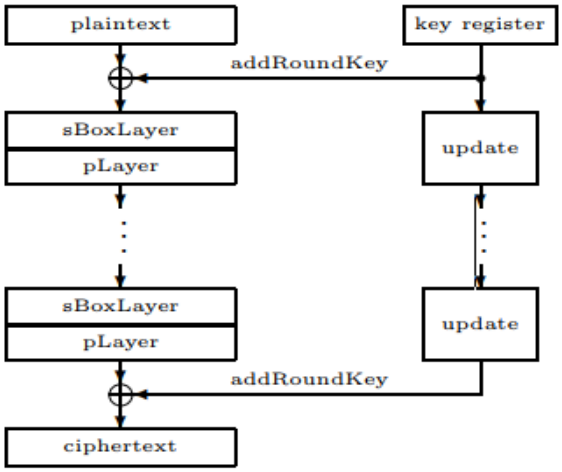

Figure 11**. Description de l'algorithme PRESENT **[98]

4. L'approche proposée

Afin de sécuriser le protocole MQTT, notre approche consiste à utiliser l'algorithme AugPAKE avec PRESENT comme chiffrement léger pour la charge utile du message PUBLISH. L'objectif de notre contribution par rapport à 'Aug-MQTT' est d'apporter plus de légèreté au processus de sécurisation de la communication MQTT. Dans 'Aug-MQTT', les auteurs n'utilisent que le chiffrement sécurisé à clé symétrique comme norme de chiffrement avancée (AES) ce qui est lourd pour les ressources limitées, aussi ils utilisent un client pour négocier le cryptage, mais dans notre contribution nous utilisons un cryptage de bout en bout que le courtier n'a aucune idée du contenu la charge utile, il est crypté par PRESENT et sera envoyé à l'abonné en toute sécurité.

Dans Aug-MQTT, le broker est une bonne cible pour les hackers, les données ne sont pas sécurisées, le message est en clair, les auteurs ont mentionné que le broker pourrait être un point d'attaque unique. Il a une visibilité complète des données échangées car il se charge du cryptage et du décryptage des charges utiles dans les messages MQTT.

Le diagramme ci-dessus présente notre approche basée sur AugPAKE et PRESENT :

La première étape est basée sur l'algorithme AugPAKE pour établir une session clé sécurisée entre l'éditeur/l'abonné et le courtier. Il est divisé en deux phases :

- *La phase d'initialisation :*

  L'éditeur envoie (Cp, Wp) au courtier où Cp : l'identifiant de l'éditeur et Wp est calculé en utilisant l'équation (1) dans l'algorithme AugPAKE :

- *La phase d'exécution <Publisher -- Broker>* est composée de plusieurs étapes :

  1) L'éditeur envoie (Cp,Xp) au courtier selon l'équation (2).

  2) Le courtier envoie (S,Yp) à l'éditeur lorsqu'il reçoit la bonne Xp ; selon l'équation (3).

  3) L'éditeur envoie (Cp,Vcp); selon l'équation (5); lorsqu'il reçoit le bon Yp.

  4) Le courtier calcule Skp selon l'équation (8) et envoie (S, Vsp), selon l'équation (7).

  5) Enfin, l'éditeur calcule le Skp; selon l'équation (8), lorsqu'il reçoit le bon Vsp. L'éditeur et le courtier partagent une clé de session Skp sécurisée.

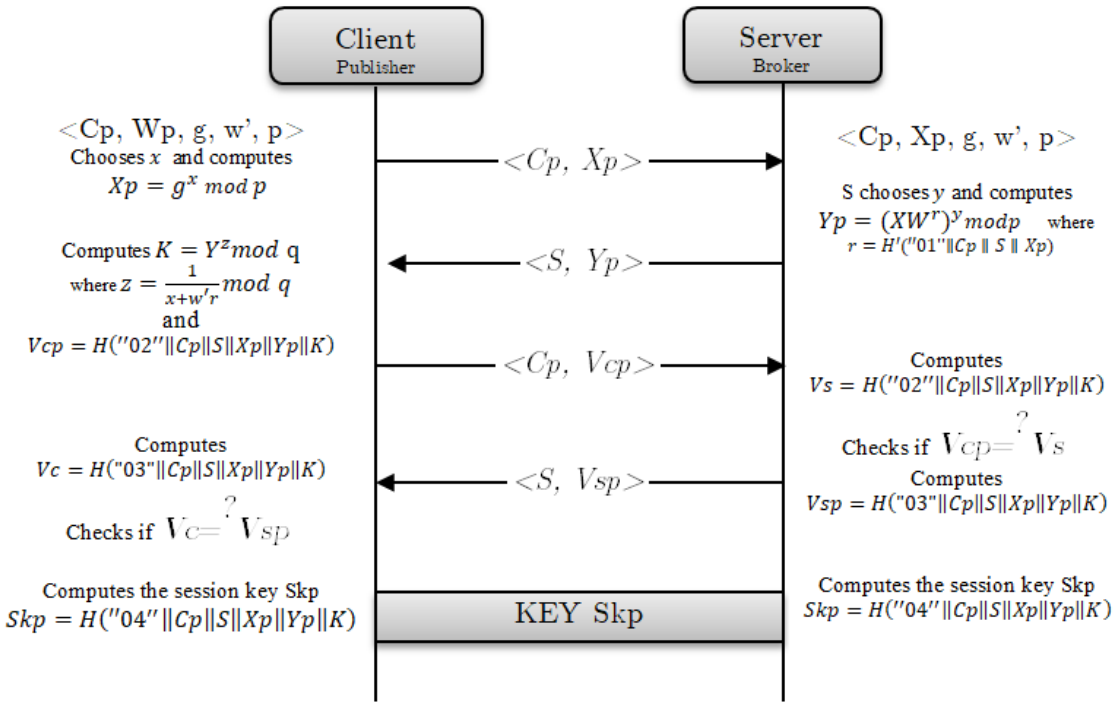

Figure 12. *Session d'établissement de clé entre Publisher/ Broker*

- *La phase d'exécution <Broker -- Subscriber>* est composée de plusieurs étapes -Figure 13-:

  1) Le Souscripteur envoie (Cs,Xs) au courtier selon l'équation (2).

  2) Le courtier envoie (S,Ys) au souscripteur lorsqu'il reçoit le bon Xs, selon l'équation (3).

  3) L'abonné envoie (Cs, Vcs); selon l'équation (5); lorsqu'il reçoit les bons Ys.

4) Le courtier calcule les S_{Ks} selon l'équation (8) et envoie (S, Vss), selon l'équation (7).

5) Enfin, l'abonné calcule les S_{Ks}; selon l'équation (8), lorsqu'il reçoit le bon V_{Cs}.

L'abonné et le courtier partagent des clés de session sécurisées.

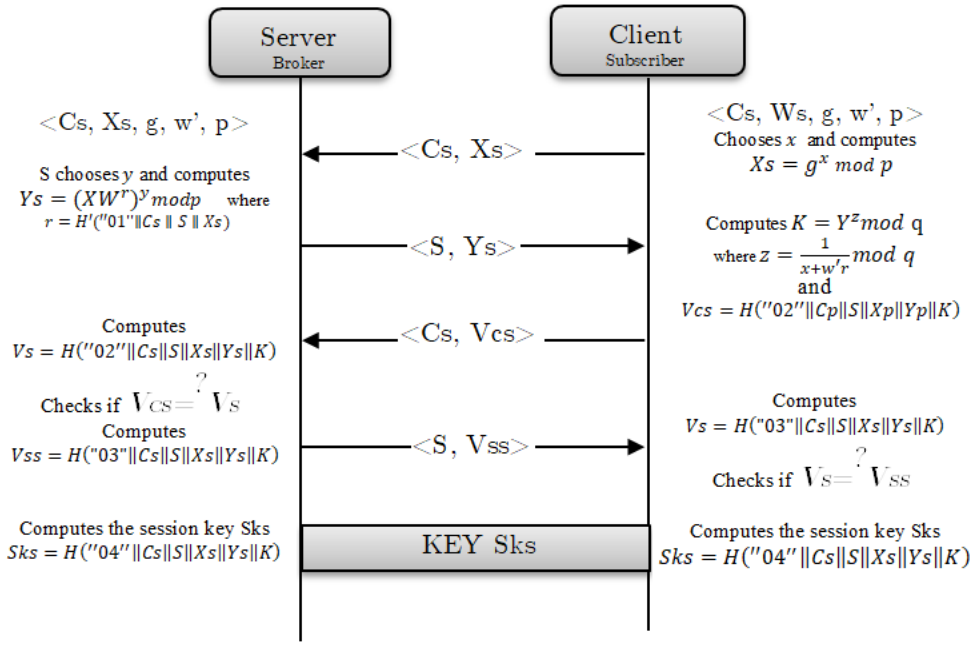

Figure 13. *Session d'établissement de la clé entre Broker/Subscriber*

Le courtier a les deux clés de session partagées par l'abonné et l'éditeur, Il calcule la clé K_s par l'équation :

$$ks = Skp \oplus Sks \quad (9)$$

La clé K_s doit être partagée avec l'éditeur et l'abonné, c'est pourquoi le courtier envoie la clé à l'éditeur (abonné) par le chiffrement PRESENT à l'aide du Skp (Sks). L'éditeur et l'abonné ont tous deux la clé partagée Ks.

Lorsque l'éditeur veut publier un message, le processus de chiffrement PRESENT appliqué à la charge utile du message PUBLISH en utilisant Ks, l'éditeur transmettra E(Ks, Data1) au courtier. Si nécessaire/demandé, Broker transmet un message chiffré E(Ks ,Data1) à l'abonné, qui

peut décrypter par le déchiffrement PRESENT avec la clé K_s pour publier le message selon son sujet correspondant.

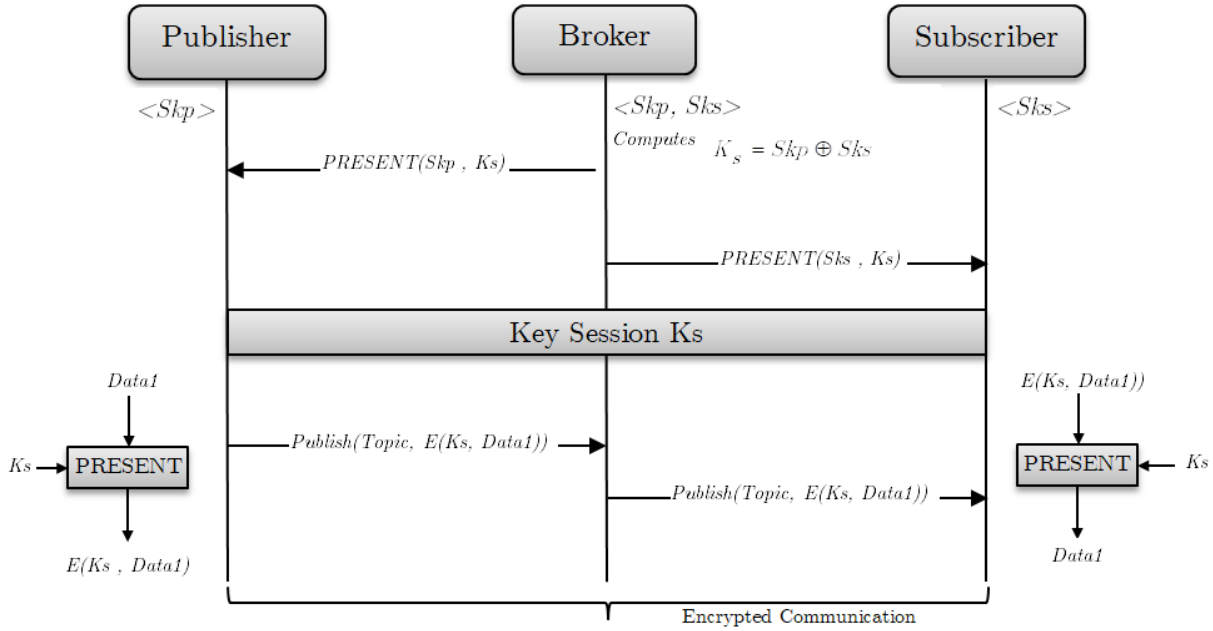

Figure 14. *Chiffrement/déchiffrement entre Publisher/Subscriber*

5. Analyse de la sécurité

Dans cette section, nous allons discuter les aspects de sécurité approuvés par notre solution :

Authentification : Tout ce qui se trouve sur le réseau de l'Internet des objets doit être identifié et authentifié avec les autres objets connectés. C'est donc une tâche difficile en raison de la nature des objets connectés. Notre solution proposée offre une authentification mutuelle entre le courtier et ses clients (éditeurs et abonnés) [99] :

- L'Authentification de l'éditeur (et de l'abonné) : l'éditeur (l'abonné) est authentifié en utilisant la clé de session Skp (Sks) approuvée par l'utilisation de l'algorithme AugPAKE.
- L'Authentification du courtier : seul le courtier qui possède la clé commune (Skp) ou (Sks) peut décrypter les messages publiés, seul les éditeurs et les abonnés qui utilisent les clés communes du courtier.

Confidentialité : Il est très important de s'assurer que les données sont sécurisées et accessibles uniquement par les utilisateurs autorisés. Autre point, les utilisateurs de l'IdO doivent

être conscients de la façon dont les données sont gérées et s'assurer que les données sont protégées par le processus [100].Dans la solution proposée, le message publié est protégé deux fois, d'abord lorsqu'il est transféré au courtier, en utilisant la session sécurisée générée par l'algorithme AugPAKE ; seul le client qui possède la clé de session peut décrypter le message et une deuxième fois côté du courtier ,il n'a pas le message en clair en raison du chiffrement PRESENT.

Intégrité et non-répudiation : Il est très important de s'assurer de l'exactitude des données et qu'elles soient créées ou modifiées que par la partie autorisée seulement. La fonction d'intégrité peut être imposée en maintenant une sécurité de bout en bout dans les communications IoTs. C'est ce que nous proposons dans notre solution ;une sécurité de bout en bout .Les données sont cryptées par le chiffrement PRESENT pendant le processus de publication du message, le courtier n'a aucune idée du message transmis [99].

Notre contribution consiste à renforcer la sécurité du courtier, les données sont stockées par le courtier dans un format crypté (chiffrement PRESENT). Le système proposé fournit l'authentification de l'éditeur et de l'abonné : ils sont authentifiés avec le courtier en utilisant la clé de session Skp (Sks) approuvée par l'utilisation de l'algorithme AugPAKE. La confidentialité, l'intégrité et la non-répudiation des messages MQTT de l'éditeur à l'abonné parce que le courtier n'a pas le message en clair en raison du chiffrement PRESENT.

En utilisant le protocole AugPAKE dans notre approche, même si un attaquant peut obtenir (Cp,Xp),(S,Yp),Vsp ou Vcp en écoutant le trafic, mais il ne peut pas calculer une clé de session authentifiée (Skp) partagée entre l'éditeur et le courtier, par conséquent, il est sécurisé contre les attaques passives et aussi si l'attaquant contrôle le message d'échange, il ne peut pas calculer une clé de session authentifiée donc il est sécurisé contre les attaques actives (Man in the middle / Replay attaque) également pour les attaques hors ligne / dictionnaire en ligne [101].

En comparant notre solution avec les travaux précédents : l'utilisation de chiffrement PRESENT de plus du protocole AugPAKE, cela offre plus de protection au courtier et le message sera crypté.

Notre solution fournit :

- ✓ L'Anonymat de l'utilisateur : pour les clients (abonné/éditeur) l'identité des clients est protégée à l'aide de Sks/Skp.

- ✓ L'authentification mutuelle : en utilisant la session sécurisée Sks (Skp) entre l'abonné/Broker et l'éditeur/Broker. La proposition valide des points de sécurité importants pour l'IoT. La solution proposée est sécurisée contre :
- ✓ l'attaque de Man In the Middle : L'attaquant sur le réseau parvient à interférer entre deux nœuds de capteurs, accédant à des données restreintes, violant la confidentialité des deux nœuds en surveillant, en écoutant et en contrôlant la communication entre les deux nœuds de capteurs [102].
- ✓ L'attaque de Replay et l'attaque *offline password guessing* : Un attaquant, qui contrôle complètement les messages échangés, ne peut pas déduire les mots de passe possibles même si l'attaquant peut deviner un mot de passe, le paramètre K dans l'équation (4) de l'algorithme AugPAKE est dérivé indépendamment du mot de passe deviné.

Nous avons réalisé une comparaison de notre solution proposée avec d'autres travaux précédents par rapport à ces avantages cités avant : Anonymat de l'utilisateur, Authentification mutuelle, Confidentialité des données, l'intégrité des données, et la robustesse contre certaines attaques comme Man in the middle, Offline Password Guessing, et le Replay attack. Les résultats semblent intéressants en comparaison avec les autres solutions travaillant sur la sécurité de MQTT.

Tableau 10. Les avantages de la solution proposée

	UA	MA	MITM	OFP attack	Replay attack	DC	DP	DI	A
[83]	×	✓	×	×	×	✓	×	×	×
[88]	×	×	✓	×	×	×	✓	×	×
[86]	×	×	✓	×	✓	×	×	×	×
[94]	×	×	✓	×	✓	×	×	×	×
[91]	×	✓	×	×	×	✓	×	✓	✓
[92]	✓	✓	✓	✓	✓	✓	×	×	✓
Notre Approche	✓	✓	✓	✓	✓	✓	✓	✓	×

UA: User Anonymity

DC: Data confidentiality

MA: Mutual authentication

DP: Data privacy

MITM: Man in the middle

DI: Data integrity

OPG: Offline password guessing attack

A: Authorization

Le deuxième paramètre important à prendre en considération est le coût estimé de cette solution, en fonction du nombre d'opérations de hachage, d'exponentiation modulaire et de chiffrement. Nous avons comparé le coût estimé de chaque solution à l'aide de AugPAKE en établissant les paramètres suivants :

- T_h = le temps de hachage
- T_{ex}= le temps de l'exponentiation modulaire
- T_{ed} = le temps de chiffrement /déchiffrement

Tableau 11. Comparaison du temps estimé avec les autres solutions

Solution	Temps estimé
Aug-MQTT	6 T_h+4 T_{ex}+4 T_{ed}
Auth-MQTT	6 T_h+ 6T_{ex} +4 T_{ed}
Notre solution	6 T_h+ 4 T_{ex} +6 T_{ed}

Le temps estimé de notre solution est prometteur par rapport aux autres solutions, en offrant une sécurité estimable pour les applications de l'internet des objets utilisant le protocole MQTT. Nous sommes sur le point de valider ces résultats par simulation.

6. Conclusion

Dans ce chapitre, notre contribution consiste à proposer une nouvelle approche qui rend le protocole MQTT, le protocole le plus utilisé dans les applications IoT, plus sûr et concurrentiel avec les autres solutions proposées par les autres recherches. Notre solution est basée sur l'algorithme AugPAKE et le chiffrement léger PRESENT. Notre contribution est plus légère et plus sécurisée par rapport aux autres travaux basés sur l'algorithme AugPAKE, la solution fournit l'anonymat de l'utilisateur, l'authentification mutuelle, la confidentialité des données, l'intégrité des données et la non-répudiation des informations également, il est robuste contre certaines attaques comme Man in the middle, Offline Password Guessing, et le Replay attack. Enfin, le coût estimatif de la solution que nous proposons semble intéressant. Pour les travaux futurs, nous sommes sur le point de simuler notre approche dans le modèle Publish/Subscribe et en particulier pour le domaine de la santé.

Ce chapitre a fait l'objet d'un article dans un journal International indexée Scopus.

Chapitre 5 : Application dans le domaine de la santé

1. Introduction

Pour l'humanité, la santé est un élément fondamental dans la vie. Beaucoup de recherches ont été réalisés pour améliorer la qualité des soins de santé et l'assistance aux patients. Mais la courbe de la population qui vieillit constamment et les maladies chroniques est dans une augmentation exponentielle, il sera donc très difficile de maintenir un bon système de santé pour la population en raison de la forte demande de ressources provenant des lits d'hôpitaux, des médecins et des infirmières. De plus, la technologie des (TIC) tente de trouver une solution pour réduire la pression sur le système de soins de la santé afin de fournir des soins de haute qualité aux patients [103][104]. Dans le domaine des soins de la santé, l'IoT joue un rôle majeur dans les systèmes de la santé intelligente pour guérir le patient dans les meilleures conditions et faciliter la tâche au corps médicale. Il s'agit d'un système complet qui communique entre les systèmes, les applications et les dispositifs connectés au réseau et qui peut aider les patients et les médecins à surveiller, suivre et enregistrer les données vitales et les informations médicales des patients [105].

Les soins de la e-santé représentent l'une des applications les plus importantes de l'IoT [106]. L'IoT peut créer de nombreuses applications médicales telles que la surveillance médicale à distance, les programmes de conditionnement physique, les maladies chroniques et les soins aux personnes âgées. Par conséquent, dans le secteur de la santé, divers dispositifs médicaux, capteurs et dispositifs d'imagerie peuvent être considérés comme des objets intelligents constituant la partie centrale de l'IoT. Les services de la santé devraient réduire les coûts, exercer une pression sur le système de santé, améliorer la qualité de vie et enrichir l'expérience de l'utilisateur [107].
L'objectif de cette partie est de donner un aperçu de certains systèmes de soins de santé intelligents, puis l'utilisation du protocole MQTT dans ce contexte , pour valider la simulation de notre approche dans un contexte virtuel et présenter à la fin une application de monitoring au profit du domaine de la santé intelligente.

2. Un aperçu sur les systèmes de la santé intelligente

Dans la littérature, beaucoup de recherches ont été effectuées pour développer des applications de soins de santé intelligentes afin d'améliorer la qualité des soins, de réduire le stress dans ce système et de faciliter la tâche du médecin et de l'hôpital pour prendre soin de ses patients dans les meilleures conditions. Le tableau ci-dessous donne un aperçu de certains systèmes existants ; en définissant les technologies utilisées, les forces et les faiblesses dans chaque système.

Système	Description	Technologies utilisés	Forces	Faiblesses
[108]	L'application permet le suivi et la surveillance du patient à distance grâce au graphe généré grâce au WSN qui recueille les informations depuis le RFID et le transmet au centre de contrôle	RFID, WSN, Smart Mobile	-Le médecin peut fournir une ordonnance et générer un rapport pour un patient depuis n'importe quel emplacement. -Le patient a la possibilité de consulter le rapport.	L'aspect de la sécurité n'est pas abordé
[109]	Architecture intelligente et sensible à l'IoT pour la surveillance et le suivi automatiques des patients, du personnel et des dispositifs biomédicaux dans les hôpitaux et les organisations de soins infirmiers.	RFID, WSN, Smart mobile technologies	-Surveillance à distance, -Gestion des urgences.	Caractéristiques de la sécurité en détail et l'analyse des performances.
[110]	Le système a pour objectif : -reconnaître les comportements nocturnes et les activités, -génère une alarme pour les opérateurs, les familles, le centre en cas d'urgence.	Capteurs passifs RFID Etiquettes portables (WT) Etiquettes ambiantes (AT) Lecteur RFID UHF	-Surveillance et contrôle à distance de l'état des enfants, des handicapés et des personnes âgées pendant la nuit.	Les mesures de sécurité sont absentes.
[111]	Le système aide à surveiller, mesurer les signes vitaux de santé du patient via le web, et indique une alarme aux	Zigbee Arduino	-Surveillez, mesurez les paramètres vitaux du patient via Internet.	L'aspect de la sécurité n'est pas abordé.

	médecins en cas de signes d'anomalie en cas d'urgence.		-Alerter le médecin en cas d'urgence	
[105]	La conception et la mise en œuvre de services médicaux d'urgence qui peuvent démontrer la collecte, l'intégration et l'interopérabilité des données de l'Internet des objets de manière flexible, ce qui peut fournir un soutien aux services médicaux d'urgence comme les unités de soins intensifs (USI).	Heart rate sensor, Temperature Sensor, Intel Galileo Board	Application mobile pour fournir de meilleurs services de santé	Les auteurs ne discutent pas les propriétés de la sécurité et ses performances.
[112]	Un cadre architectural pour décrire tout le cycle de vie de la surveillance et mettre en évidence les composantes essentielles des services	Pulse sensor, Temperature sensor, Arduino Uno	-Synchroniser les données des capteurs à travers le cloud et leur rendre accessibles via l'application mobile. -Emplacement indépendant, facile et économique -Approche pour la surveillance des patients	Les aspects de la sécurité de cette architecture ne sont pas abordés.

Tableau 12 : Aperçu de certains systèmes de la santé intelligente

Le point commun de ces systèmes c'est que l'aspect de la sécurité n'est pas discuté et représente un défaut énorme. Parce qu'un système non sécurisé n'ajoutera rien aux soins de santé. C'est pourquoi nous avons choisi de travailler sur ce point important.

1. Notre approche dans le domaine de la e-santé

L'architecture complète de notre cadre est présentée à la figure 15. L'environnement de communication comprend principalement trois parties : l'éditeur (P), le courtier (B) et l'abonné (S). Dans notre application de soins de santé proposée, les éditeurs sont les dispositifs contraints (capteurs) qui détectent les données captées de santé du patient (comme le capteur de la température et le capteur de pulsions, la pression artérielle [TA], les niveaux de glucose et le poids corporel) et produisent le message qui est stocké par sujets dans les systèmes. Ces messages sont transmis au Broker, qui transmet les messages reçus aux utilisateurs finaux. Le courtier B agit comme passerelle, qui reçoit et achemine les messages par sujet dans un arrangement de réseau. S, l'utilisateur final/serveur qui reçoit et traite le message de l'éditeur P. La topologie entière est divisée en une zone d'éditeur et d'abonné, où la zone d'éditeur considère la communication de message entre P et B, la zone d'abonné recherche la communication entre B et S.

Notre approche dans l'application de la e-santé :

La procédure de l'algorithme AugPAKE est exécuté pour chaque éditeur et chaque abonné qui va se connecter au Broker. Le courtier B va agir comme un serveur de AugPAKE ainsi que le courtier et l'abonnée comme des clients. Les messages de AugPAKE vont être encapsulé directement dans la charge utile des paquets MQTT PUBLISH. Mais avant cela une session a été établie entre l'éditeur et le courtier comme avec l'abonnée et le courtier en utilisant les messages CONNECT. Les clients vont communiquer avec le courtier avec le Topic = AuthenticationTopic, en contrepartie le courtier communique avec les clients via un Topic temporaire Topic = Identifiant du client, échangé dans le processus de connexion CONNECT message, ce sujet est créé au moment de la communication et détruit une fois la clé de session est établie.

Ces deux sujets sont utilisés en communication bidirectionnelle entre le courtier et les clients correspondants de AugPAKE (l'éditeur et l'abonné). Au niveau de ce mécanisme, 2 messages MQTT CONNECT au cours de la phase de la connexion et 5 messages encapsulés dans des messages de type ' PUBLISH'. Après cette phase, chaque client (soit un abonné ou éditeur) possède un clé secrète commune.

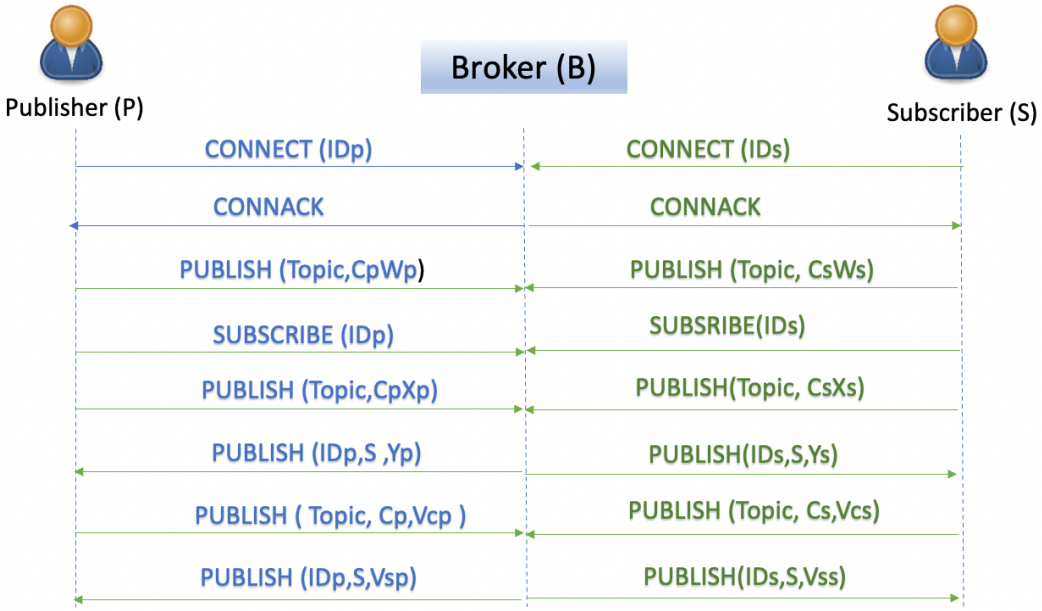

Figure 15 : *les messages échangés de MQTT lors de l'établissement de la session sécurisée*

2. La simulation de l'approche

Dans un premier temps, nous allons faire les tests sur des machines virtuelles, nous allons simuler le Broker et les clients (Publisher, Subcriber) en introduisant l'algorithme AugPAKE et le chiffrement PRESENT.

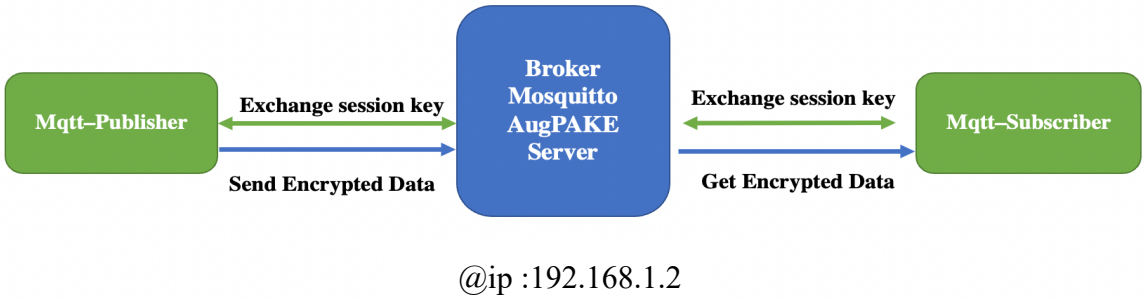

Figure 16 : *Architecture de la plateforme de la Simulation*

Nous allons utiliser le principe de Docker, qui est une plateforme permettant de lancer certaines applications dans des conteneurs logiciels. C'est un outil qui permet d'embarquer une application dans un ou plusieurs conteneurs logiciels qui pourra s'exécuter sur n'importe quel serveur machine, ce n'est pas de la virtualisation mais le principe de conteneurisation, c'est très léger, il offre plusieurs avantages tels que la flexibilité et la portabilité d'exécution d'une application sur une grande variété de machines hôtes : machine locale, un cloud …de façon fiable et prévisible et c'est Open source.

2.1 L'environnement du travail

Le tableau suivant résume l'environnement de travail, le matériel utilisé et sa configuration :

Tableau 13 : **la configuration du matériel utilisé**

Matériel	Configuration
Système d'exploitation	Windows 10
Type de processeur	i5-4350 2.6 GHZ
Nombre de cœur	6
RAM	8 GB
Disque dur	250 G SSD

Nous allons installer des containers sur les machines virtuelles suivantes :

-La machine virtuelle VM1 : Mqtt-Publisher

-La machine virtuelle VM2 : Mosquitto/AugPAKE server

-La machine virtuelle VM3 : Mqtt-Subscriber

Nous avons choisi de travailler avec le langage 'Python' puisque les deux travaux précédents utilisant l'algorithme AugPAKE, ils ont travaillé avec le langage C et le langage JAVA.

Les étapes à suivre lors de l'installation :

Nous allons préparer l'environnement, l'utilisation du Docker nous facilite la tâche puisque le fichier Dockerfile permet l'exécution d'un ensemble de commandes de façon automatique sur les machines virtuelles.

2.2 Préparation des éléments Broker, Publisher, Subscriber:

2.2.1 Build and Run l'image de AugPAKE server

```
$ cd docker-augpake-server
$ sudo docker build -t project:augpake-server -f  Dockerfile . . /. . /
$ docker run --rm -p 54321:54321 -p 9090:9090
--name augpake-server  project:augpake-server 54321
```

2.2.2 Build and Run l'image de Mosquitto

```
$ docker pull jllopis/mosquitto: latest
$ docker run –rm -p 1883:1883 -p 9883:9883
--name broker
 jllopis/mosquitto: latest
```

2.2.3 Préparation de Mqtt-Publisher

```
$ cd docker-mqtt-pub
$ sudo docker build -t  project:mqtt-pub -f ./Dockerfile ../../
Après modification du fichier de configuration du publisher
$ docker run -ti --rm --name pub
--link augpake-server:augpake-server
--link broker:broker
-v ‘pwd’/config.py:/tmp/mqtt-python/config.py
```

2.2.4 Modification du fichier de configuration du Mqtt-Publisher

```
$ vim docker-mqtt-pub/config.py
# For mqtt-pub.py
# AugPAKE server configuration
# For mqtt-pub.py
    AUGPAKE_SERVER_IP = “192.168.1.2”
    AUGPAKE_SERVER_PORT =”54321”
# MQTT broker configuration
# For mqtt-sub.py & mqtt-pub.py
  MQTT_BROKER_IP = “192.168.1.2”
```

2.2.5 Préparation de Mqtt-Subscriber

```
$ cd docker-mqtt-sub
$ sudo docker build -t project:mqtt-sub -f ./Dockerfile ../../
Après modification du fichier de configuration du subscriber
$ docker run -ti --rm --name sub
--link augpake-server:augpake-server
--link broker:broker
-v ‘pwd’/config.py:/tmp/mqtt-python/config.py
Project :mqtt-sub
```

2.2.6 Modification du ficher de configuration du Subscriber

```
$ vim docker-mqtt-sub/config.py
# For mqtt-sub.py
# AugPAKE server configuration
# For mqtt-pub.py
    AUGPAKE_SERVER_IP = “192.168.1.2”
    AUGPAKE_SERVER_PORT =”54321”
# MQTT broker configuration
# For mqtt-sub.py &mqtt-pub.py
 MQTT_BROKER_IP = “192.168.1.2”
```

Cette simulation ; dans le contexte des machines virtuelles ; nous a permis de valider la faisabilité de notre approche en appliquant l'algorithme d'AugPAKE ; en introduisant le serveur d'AugPAKE dans le Broker du protocole MQTT et en utilisant aussi le chiffrement PRESENT pour la charge utile afin de renfoncer la confidentialité des données transmises mais avec un chiffrement léger qui est efficace dans le contexte des appareils IoT qui ont de ressources limitées.

Notre simulation va s'étendre sur un choix d'un domaine d'application, celui de la santé intelligente vu son importance dans notre vie quotidienne.

3. Application de supervision de la santé intelligente

La plateforme sur laquelle nous allons travailler est une application qui vise à fournir de meilleurs soins de santé aux personnes malades et cela en enregistrant les signes vitaux tels que la fréquence cardiaque, la pression artérielle, la température corporelle via des capteurs connectés au corps de la personne malade. Plusieurs capteurs ont été fixés au corps du patient afin de mesurer et de communiquer avec le serveur situé à distance ou le médecin via le protocole MQTT. Cette technique permet au personnel médical de surveiller en permanence la santé du patient et de noter avec précision les changements dans l'état du patient en temps réel.

Plusieurs nouvelles start-ups ont été lancé dans les technologies de la santé et l'Internet des Objets jouent un rôle primordial en révolutionnant rapidement l'industrie de la santé.

Dans le cadre de ce projet, nous avons conçu un système de surveillance de la santé au profit des patients basé sur l'IoT en utilisant les composants ESP8266 et Arduino. La plateforme IoT utilisée dans ce projet est ThingSpeak. ThingSpeak est une API de l'Internet des objets (IoT) open-source pour stocker et récupérer des données à partir d'objets utilisant le protocole HTTP et le protocole MQTT sur Internet ou via un réseau local. L'appareil IoT pourrait lire le pouls et mesurer la température environnante. Il surveille en continu le pouls et la température environnante et les met à jour vers la plateforme IoT.

3.1 Block Diagram de l'application :

Figure 18 : l*e Block Diagram de l'application*

Ce bloc diagramme explique l'application de la supervision 'IoT Based Patient Health Monitoring using ESP8266 and Arduino'. Le capteur de la température LM35 et le capteur des battements de cœur mesurent respectivement la température et les battements de cœur du patient. Le module ESP8266 est connecté au WiFi afin d'envoyer les données au serveur Thingspeak, de cette façon les données peuvent être surveiller de la part des utilisateurs en se connectant aux canaux de Thingspeak.

3.2 Les composants requis :

Pour le fonctionnement de notre maquette, nous aurons besoin des composants suivants :

1. Arduino mega Board.
2. ESP8266 Wi-Fi Module.
3. Pulse Sensor.
4. LM35 Temperature Sensor.
5. Breadboard
6. Connecting Wires

3.2.1 La carte Arduino mega

Elle est parmi la gamme de la carte Arduino, qui est un circuit imprimé en matériel libre utilisant le microcontrôleur ATmega2560 et possède toute la mémoire à 256 KB, un microcontrôleur programmé peut analyser et produire des signaux électriques de manière à effectuer des tâches très diversifié, Arduino est utilisé dans beaucoup de domaine à citer le Robotique aussi dans le domaine des applications IoTs, Cette carte dispose de : (voir la figure suivante)

- 54 broches numériques d'entrées/sorties (dont 14 peuvent être utilisées en sorties PWM (MLI : Modulation de largeur d'impulsion).
- 16 entrées analogiques (qui peuvent être utilisées en broches entrées/sorties numériques).
- 4 UART (port série matériel).
- Un quartz de 16Mhz.
- Une connexion USB.
- Un connecteur d'alimentation jack.

- Un connecteur ICSP (programmation "in-circuit").
- Un bouton de réinitialisation (reset).

Elle contient tout ce qui est nécessaire pour le fonctionnement du microcontrôleur ; Pour pouvoir l'utiliser, il suffit de la connecter à un ordinateur à l'aide d'un câble USB (ou de l'alimenter avec un adaptateur secteur ou une pile, mais ceci n'est pas indispensable, l'alimentation étant fournie par le port USB)

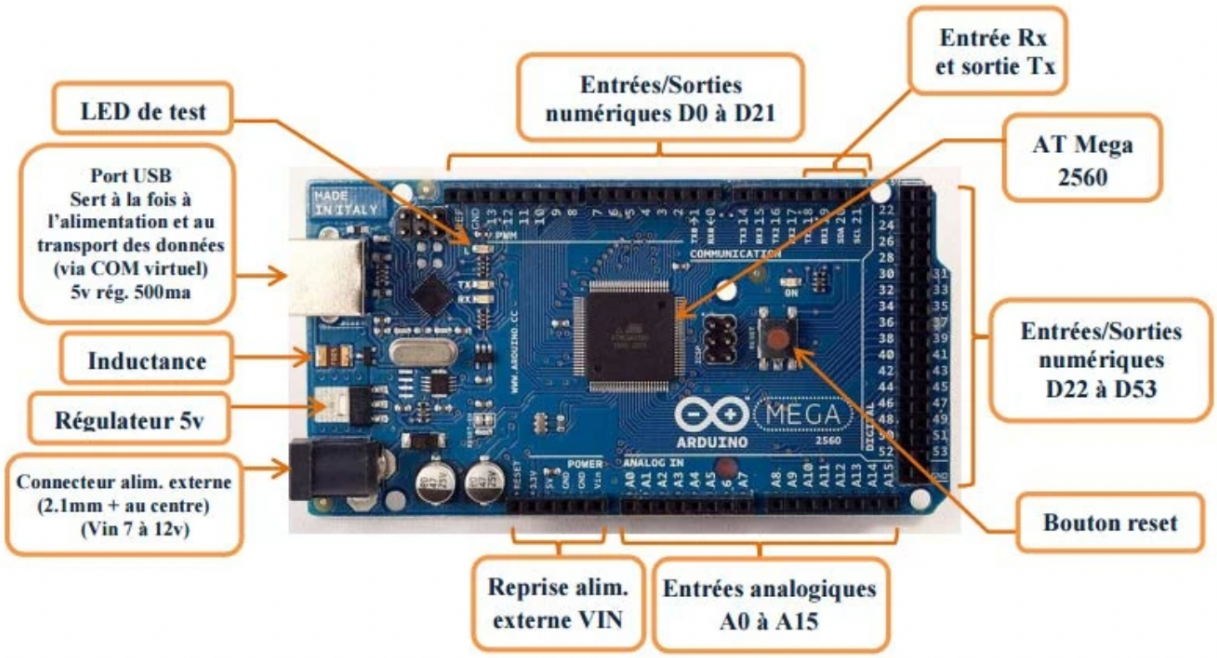

Figure 19 : *Les composants de la carte Arduino Mega*

3.2.2 Capteur de température LM35 :

Les LM35 sont des appareils de température à circuit intégré de précision avec une tension de sortie linéairement proportionnelle à la température Centigrade. L'appareil LM35 présente un avantage par rapport aux capteurs de température linéaires étalonnés en Kelvin, car l'utilisateur n'est pas tenu de soustraire une grande tension constante de la sortie pour obtenir une échelle centigrade pratique. Le dispositif LM35 ne nécessite pas d'étalonnage externe ou de réglage pour fournir des précisions types de ¼ °C à température ambiante et de ¾ °C sur une plage complète de température de 55 °C à 150 °C.

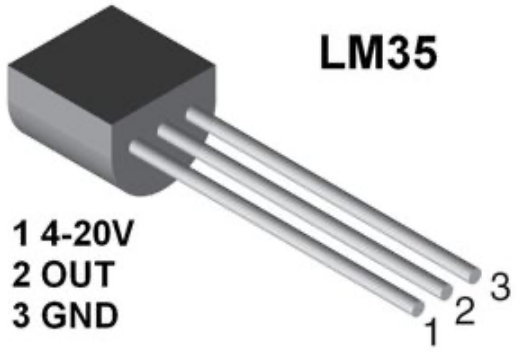

Figure 20 : *le composant LM35*

3.2.3 Capteur de pulsions cardiaques :

Le capteur de pulsion cardiaque est un capteur de fréquence cardiaque plug-and-play pour Arduino. Il peut être utilisé par les étudiants, les artistes, les athlètes pour pouvoir intégrer facilement des donnes de fréquence cardiaque dans leur projet. L'essence est un circuit amplificateur optique intégré et un détecteur de bruit. Attachez le capteur de pouls à votre lobe d'oreille ou au bout du doigt et branchez-le sur votre Arduino, vous pouvez lire la fréquence cardiaque. En outre, il a un code de démonstration Arduino qui le rend facile à utiliser. Le capteur de pulsions dispose de trois broches : VCC, GND et Analog Pin.

Il y a également une LED au centre de ce module de capteur qui aide à détecter le battement de cœur. Sous la LED, il y a un circuit d'élimination du bruit qui est censé empêcher le bruit d'affecter les lectures.

Figure 21 : *Capteur de pulsions cardiaques*

3.2.4 Le composant ESP8266 :

L'ESP8266 est un appareil très convivial et à faible coût pour fournir une connectivité Internet à vos projets. Le module peut fonctionner à la fois comme un point d'accès (peut créer hotspot) et comme une station (peut se connecter au Wi-Fi), donc il peut facilement récupérer des données et les télécharger sur Internet rendant l'Internet des Objets aussi facile que possible. Il peut également récupérer des données à partir d'Internet en utilisant des API, donc votre projet pourrait accéder à toutes les informations disponibles sur Internet, ce qui le rend plus intelligent.

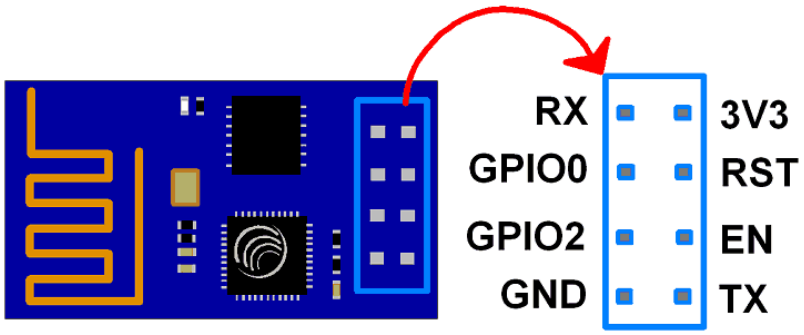

Figure 22 : *les pins de l'ESP8266*

Tableau 13 : Description des différents pins du composant ESP8266

Pin Number	Correspondent pins	Description
Pin 1	Ground	Connected to the ground of the circuit
Pin 2	Tx/GPIO -1	Connected to Rx pin of programmer/uC to upload program
Pin 3	GPIO-2	General purpose Input/output pin
Pin 4	CH_EN	Chip Enable/Active high
Pin 5	Flash/GPIO -0	General purpose Input/output pin
Pin 6	Reset	Resets the module
Pin 7	RX/GPIO - 3	General purpose Input/output pin
Pin 8	Vcc	Connect to +3.3V only

3.3 Les différentes étapes pour le fonctionnement d'ESP8266

Afin de se connecter à Thingspeak, le package ESP8266 doit être installer dans le logiciel d'Arduino IDE.

Nous avons besoin d'une connexion Internet pour télécharger le package

Ouvrir Arduino IDE, cliquer sur 'file' puis 'Preferences'

Une fenêtre qui va s'ouvrir, nous allons copier le lien suivant : http://arduino.esp8266.com/stable/package_esp8266com_index.json

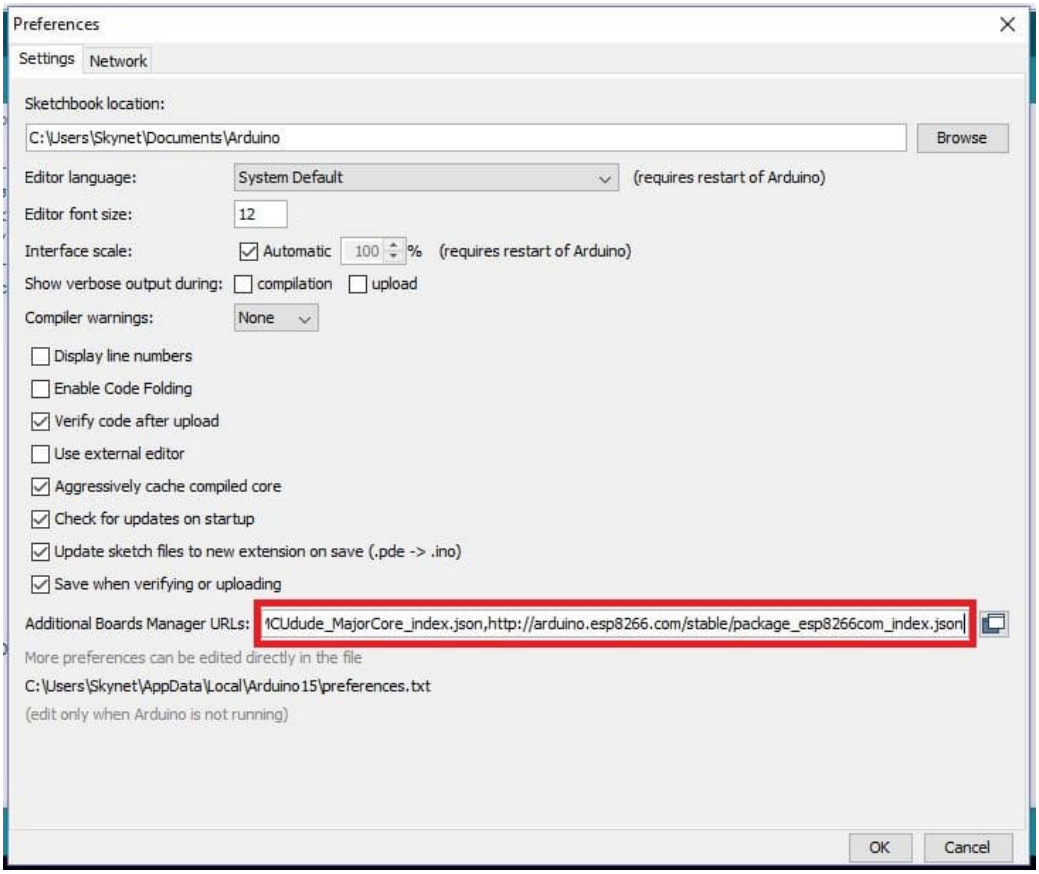

Figure 23 : *l'onglet Preferences*

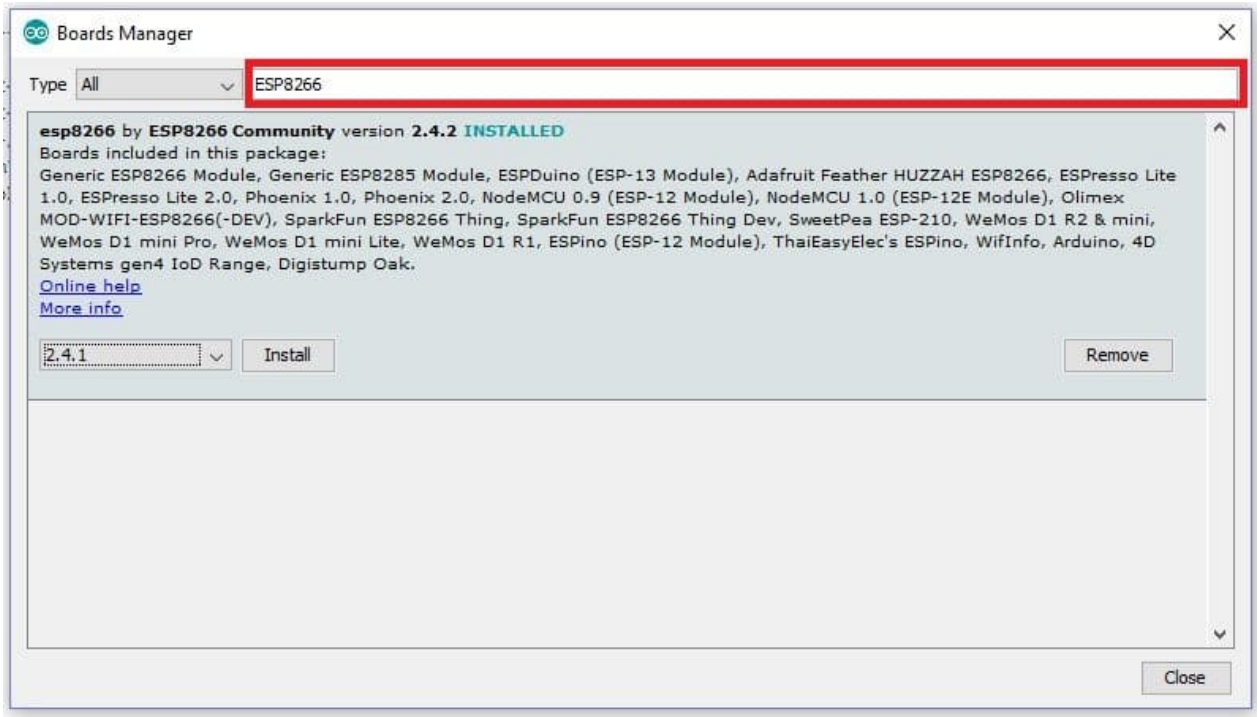

Figure 24 : *l'onglet Board Manager*

Une fois l'installation effectué du package ESP8266, en installant la version convenable, ça pourra prendre quelques minutes tout dépend de la connexion Internet.

Pour télécharger le code au composant ESP8266, nous avons besoin du branchement suivant selon le bloc diagramme :

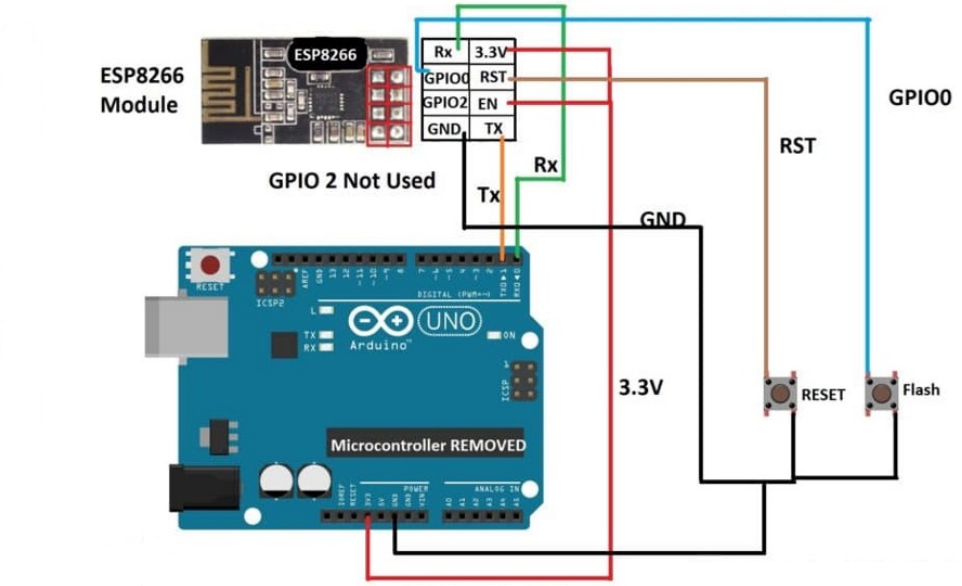

Figure 25 : *Circuit diagramme pour télécharger le code à ESP8266*

Le code à télécharger est en annexe I, pour le protocole MQTT.

Pour notre application de supervision en utilisant l'ESP8266 & Arduino Mega, nous avons besoin d'assembler le circuit suivant comme c'est mentionner dans la figure suivante :

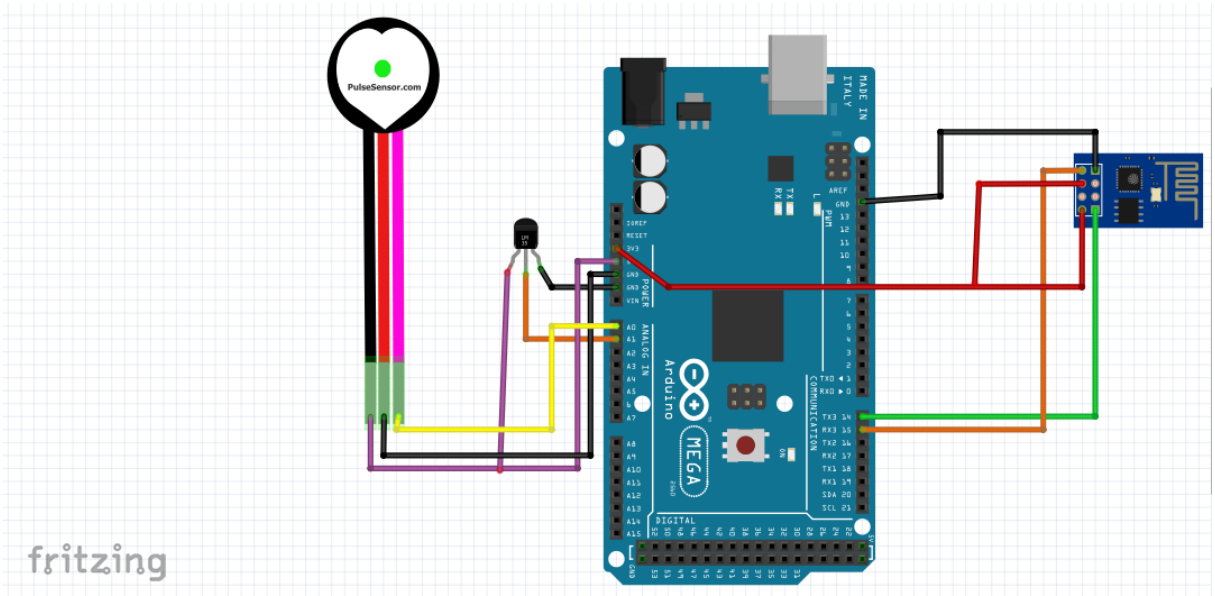

Figure 26 : *Circuit diagramme de l'application*

Le branchement s'est fait de la façon suivante :

Pour LM35 :

Pins LM35	Pins Arduino Mega
Vcc	4v
Vout	A1
GND	GND

Pour le capteur de pulsions :

Pins capteur	Pins Arduino Mega
A0	A0
Vcc	4v
GND	GND

Pour le composant ESP8266 :

Pins ESP8266	Pins Arduino Mega
Rx	Tx3 (14)
GPIO 0	---

GPIO 2	---
GND	GND
3v3	3v3
RST	---
EN	3v3
Tx	Rx3 (15)

Nous allons par la suite, télécharger le code sur Arduino IDE qui est en (Annexe II).

3.4 Paramétrage de Thingspeak

Thingspeak est un outil très puissant pour les projets basés sur IoT, en utilisant le site de 'Thingspeak', nous pouvons superviser les données, contrôler le système via l'internet, en utilisant les canaux et les pages Web offertes par Thingspeak : https://thingspeak.com

Nous avons besoin de créer un compte afin de créer des nouveaux canaux liés au capteur de pulsions et le capteur de la température LM35.

Nous avons obtenu les résultats suivant dans notre compte de Thingspeak, qui mentionne les données captées par le capteur de pulsions dans le graphe qui est à droite et dans celui qui est à gauche c'est le capteur de température.

Grâce à cette application, toute partie du système hospitalier peut consulter les résultats à distance du patient et peut intervenir en cas d'urgence.

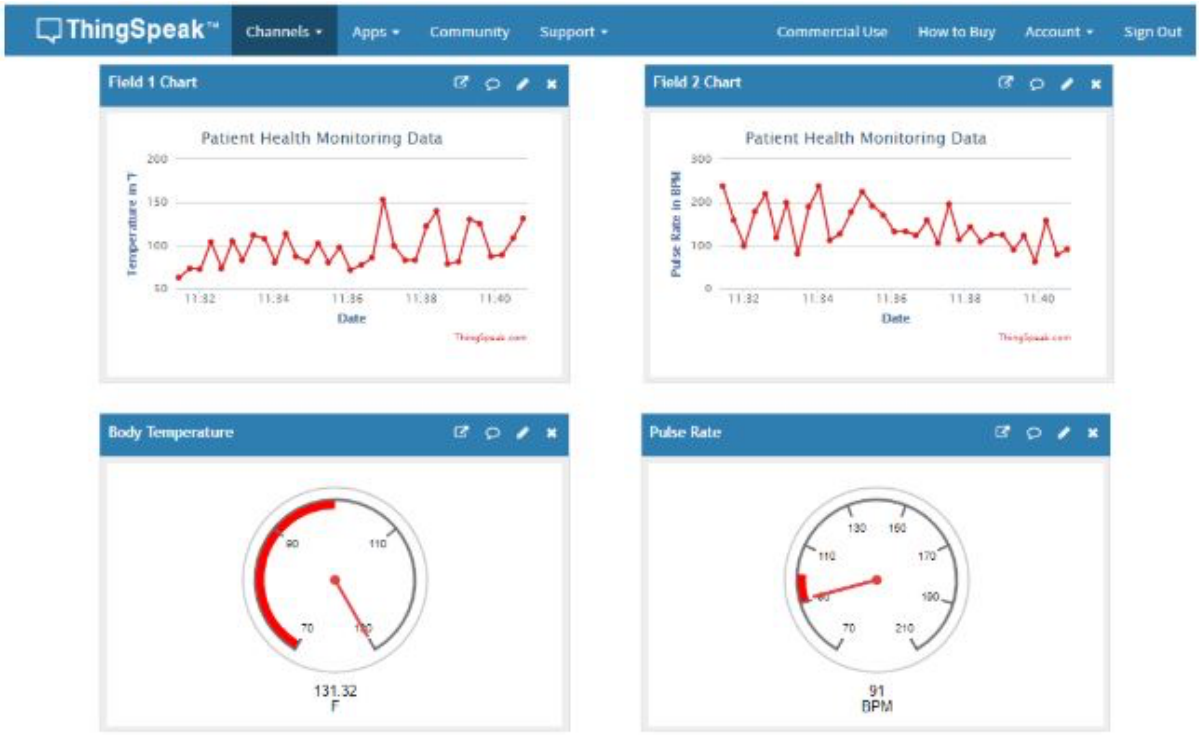

Figure 27 : *Patient Health monitoring*

Thingspeak permet de visualiser une interprétation graphique des données collectés soit sous forme de graphe ou de moniteur ce qui facilitera la tâche au corps médical.

4. Conclusion

L'objectif principal de ce dernier chapitre est de valider notre approche pour sécuriser les applications IoT utilisant le protocole MQTT, comme protocole de la couche application ; nous avons choisi l'application dans le domaine la santé vu son importance dans notre vie quotidienne. Dans un premier temps, nous avons valider l'approche proposée dans un environnement virtuel, puis nous avons implémenter l'application de supervision à distance à l'aide du capteur de la température et le capteur de pulsions pour visualiser les données envoyées sur le cloud de 'Thingspeak' et rendre les données captées accessibles par le corps médicale afin de surveiller les paramètres vitaux des patients qui ont besoin d'une longue période de convalescence ou pour les personnes âgées ayant besoin de la surveillance et l'intervention en cas d'urgence , cela permet de faire des économies pour l'hôpital et pour les patients en terme de temps et d'argent.

Conclusion Générale

L'Internet des Objets est une technologie clé dans le domaine des Technologies de l'Information et de Communication (TIC), c'est le cœur de la révolution de l'Industrie 4.0 qui va envahir le monde très prochainement. C'est la motivation derrière le choix de notre sujet de recherche ainsi le défi de la sécurité des Objets connectés, qui est condition sine qua non pour faire confiance à cette technologie. Au cours de cette thèse :

Nous avons essayé d'étudier en premier lieu, les différents aspects en termes de composants, de l'architecture en couches et des différents protocoles de la couche d'application. En deuxième lieu, les différentes vulnérabilités qui menacent les IoTs, nous avons classifié ces menaces en catégorie : tout d'abord sur les données et le réseau puis sur la protection de la vie privée et enfin sur les systèmes IoT.

Notre première contribution consistait à faire une étude comparative du protocole MQTT par rapport aux autres protocoles de la couche application afin de justifier le choix d'étudier ce protocole parmi les autres. Notre étude nous a amené à déduire que ce protocole est le plus léger et compatible pour les composants IoTs qui ont de contraintes en termes de ressources comme et il offre 3 niveaux de qualité de service.

Notre deuxième contribution visait à étudier la vulnérabilité de ce protocole et faire un état d'art sur les solutions proposées par des chercheurs pour améliorer sa sécurité , pour enfin proposer une approche basée sur l'algorithme AugPAKE et le chiffrement PRESENT, dans l'objectif est d'apporter la confidentialité , l'authentification mutuelle, l'intégrité des données et la non-répudiation des données ,également , notre solution est robuste contre certaines attaques à citer le MITM et le Replay Attack.

Notre dernière contribution a pour objectif de valider notre approche par une simulation et l'appliquer dans le domaine de la E-santé en proposant une solution de supervision des données vitales d'un patient à distance en utilisant le protocole MQTT et un cloud Thingspeak qui permet de visualiser ces données.

Notre recherche scientifique ne s'achèvera pas à ce stade , nous avons proposé une solution en amant pour améliorer la sécurité de ce protocole, le protocole utilisé par excellence dans l'Internet des Objets , en perspective , nous sommes actuellement en train de travailler sur la sécurité de ce

protocole en aval, en proposant un Intrusions Detection System (IDS) qui est un moniteur matériel ou logiciel qui analyse les données et surveille le trafic afin de rechercher des activités suspectes et émettent des alertes en cas de problème, mais les IDS traditionnels rendent le système plus complexe et moins efficace lorsqu'il s'agit d'un grand volume de données les 'Big Data', appliquée à des ressources contraintes 'IoT' car le processus d'analyse prend plus de temps et demande beaucoup de ressources, c'est la raison pour laquelle nous avons pensé à travailler sur des IDS légers adaptés à notre contexte celui des Objets Connectés utilisant toujours le même protocole MQTT, et cela en se basant sur les algorithmes de Machines Learning (ML) qui est un domaine d'étude qui permet de donner aux appareils la capacité d'apprendre et de s'améliorer à partir de l'expérience sans être programmé explicitement et automatiquement. Le processus d'apprentissage commence par des observations ou des données pour rechercher des tendances dans les données et faire de meilleures prédictions à partir des exemples fournis. L'objectif principal est de permettre aux IoTs d'apprendre sans aide humaine et d'ajuster les actions en conséquence, en traitant un Dataset spécifiquement pour le protocole MQTT-Dataset.

Annexe I

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "ThingSpeak.h"

char ssid[] = "hola";          //  Change this to your network SSID (name).
char pass[] = "123456789";    // Change this to your network password.
WiFiClient client;           // Initialize the Wi-Fi client library.
PubSubClient mqttClient(client ); // Initialize the PuBSubClient library.

char mqttUserName[] = "Patient Health Monitoring System";  // Use any name.
char mqttPass[] = "8BZHVOFJIF3WKVOR";
  // Change to your MQTT API Key from Account > MyProfile
char writeAPIKey[] = "RO0B619J68YXS90G";
  // Change to your channel write API key
long channelID = 968873;                // Change to your channel ID
int fieldNumber = 2;
const char* server = "mqtt.thingspeak.com";

static const char alphanum[] = "0123456789"
                               "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
                               "abcdefghijklmnopqrstuvwxyz";
// For random generation of client ID.

String ssid1 = "Hola";  // Wifi network SSID
String password1 = "123456789"; // Wifi network password

int status = WL_IDLE_STATUS;
unsigned long lastConnectionTime = 0;
const unsigned long postingInterval = 20L * 1000L;
// Post data every 20 seconds.

int data1, data2, data3, data4, data5, ok;
unsigned char buff[10], i;
String buffer1, buffer2;
void setup() {

  Serial.begin(9600);
  int status = WL_IDLE_STATUS;  // Set temporary Wi-Fi status.

  // Attempt to connect to Wi-Fi network.
  while (status != WL_CONNECTED)
```

```
  {
    status = WiFi.begin(ssid, pass); // Connect to WPA/WPA2 Wi-Fi network
    delay(5000);
  }
  Serial.println("Connected to wifi");
  mqttClient.setServer(server, 1883);   // Set the MQTT broker details
}
void loop() {

  if (Serial.available() > 0)
  {
    delay(100);
    while (Serial.available() > 0)
    {
      buffer1 = Serial.readString();
      if (buffer1[0] == '*')
      {
        if (buffer1[7] == '#')
        {
          Serial.println(buffer1);
          data1 = ((buffer1[1] - 0x30) * 10 + (buffer1[2] - 0x30));
          data2 = ((buffer1[4] - 0x30) * 10 + (buffer1[4] - 0x30));
        }
      }
    }
  }
  // Reconnect if MQTT client is not connected.
  if (!mqttClient.connected())
  {
    reconnect();
  }

  mqttClient.loop();
// Call the loop continuously to establish connection to the server

  // If interval time has passed since the last connection, publish data to ThingSpeak
  if (millis() - lastConnectionTime > postingInterval)
  {

    // Create data string to send to ThingSpeak
    String data = String("field1=") + String(data1, DEC) + "&field2=" + String(data1, DEC);
    int length = data.length();
    const char *msgBuffer;
    msgBuffer = data.c_str();
```

```
    Serial.println(msgBuffer);

    // Create a topic string and publish data to ThingSpeak channel feed
    String topicString = "channels/" + String( channelID ) + "/publish/" +
String(writeAPIKey);
    length = topicString.length();
    const char *topicBuffer;
    topicBuffer = topicString.c_str();
    mqttClient.publish( topicBuffer, msgBuffer );
    lastConnectionTime = millis();
  }
}

void reconnect()
{
  char clientID[9];

  // Loop until reconnected.
  while (!mqttClient.connected())
  {
    Serial.print("Attempting MQTT connection...");
    // Generate ClientID
    for (int i = 0; i < 8; i++) {
      clientID[i] = alphanum[random(51)];
    }
    clientID[8] = '\0';

    // Connect to the MQTT broker.
    if (mqttClient.connect(clientID, mqttUserName, mqttPass))
    {
      Serial.println("connected");
    } else
    {
      Serial.print("failed, rc=");
      // Print reason the connection failed.
      Serial.print(mqttClient.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void mqttPublishField(int fieldNum) {
```

```
  // Create data string to send to ThingSpeak.
  String data = String(t, DEC);
  int length = data.length();
  const char *msgBuffer;
  msgBuffer = data.c_str();
  Serial.println(msgBuffer);

  // Create a topic string and publish data to ThingSpeak channel feed.
  String topicString = "channels/" + String( channelID ) + "/publish/fields/field" +
String(fieldNum) + "/" + String(writeAPIKey);
  length = topicString.length();
  const char *topicBuffer;
  topicBuffer = topicString.c_str();
  mqttClient.publish( topicBuffer, msgBuffer );
  lastConnectionTime = millis();
}
```

Annexe II

```
#include "ThingSpeak.h"
#include <SoftwareSerial.h>
#define USE_ARDUINO_INTERRUPTS true
// Set-up low-level interrupts for most acurate BPM math.
#include <PulseSensorPlayground.h>
// Includes the PulseSensorPlayground Library.

SoftwareSerial mySerial(0, 1); // RX, TX
String apiKey = "LE7HQUHNSQT2ML82";
 // replace with your channel's thingspeak WRITE API key
String ssid = "Hola";  // Wifi network SSID
String password = "123456789"; // Wifi network password
boolean DEBUG = true; //  Variables
const int LM35 = A0;
const int PulseWire = 0;
// PulseSensor PURPLE WIRE connected to ANALOG PIN 0
int Threshold = 550;
// Determine which Signal to "count as a beat" and which to ignore.
PulseSensorPlayground pulseSensor;
// Creates an instance of the PulseSensorPlayground object called "pulseSensor"
//=============================================================
==================== showResponce
void showResponse(int waitTime) {
  long t = millis();
  char c;
  while (t + waitTime > millis()) {
    if (Serial2.available()) {
      c = Serial2.read();
      if (DEBUG) Serial.print(c);
    }
  }
}
//=============================================================
====================
boolean thingSpeakWrite(float value1, float value2) {
  String cmd = "AT+CIPSTART=\"TCP\",\"";         // TCP connection
  cmd += "184.106.153.149";                      // api.thingspeak.com
  cmd += "\",80";
  Serial2.println(cmd);
  if (DEBUG) Serial.println(cmd);
  if (Serial2.find("Error")) {
    if (DEBUG) Serial.println("AT+CIPSTART error");
```

```
    return false;
  }
  String getStr = "GET /update?api_key="; // prepare GET string
  getStr += apiKey;

  getStr += "&field1=";
  getStr += String(value1);
  getStr += "&field2=";
  getStr += String(value2);
  // getStr +="&field3=";
  // getStr += String(value3);
  // ...
  getStr += "\r\n\r\n";
// send data length
  cmd = "AT+CIPSEND=";
  cmd += String(getStr.length());
  Serial2.println(cmd);
  if (DEBUG)  Serial.println(cmd);
  delay(100);
  if (Serial2.find(">")) {
    Serial2.print(getStr);
    if (DEBUG)  Serial.print(getStr);
  }
  else {
    Serial2.println("AT+CIPCLOSE");
    // alert user
    if (DEBUG)   Serial.println("AT+CIPCLOSE");
    return false;
  }
  return true;
}
//_____________________________________________________________________________
//----------------------------------------------------------------------------
void setup()
{
  DEBUG = true;          // enable debug serial
  Serial.begin(9600);
  Serial2.begin(115200);
   // Configure the PulseSensor object, by assigning our variables to it.
  pulseSensor.analogInput(PulseWire);

  pulseSensor.setThreshold(Threshold);
```

```
  // Double-check the "pulseSensor" object was created and "began" seeing a signal.
   if (pulseSensor.begin()) {
    Serial.println("We created a pulseSensor Object !");
//This prints one time at Arduino power-up,  or on Arduino reset.
  }
  Serial2.println("AT+CWMODE=1");   // set esp8266 as client
  showResponse(1000);
  Serial2.println("AT+CWJAP=\"" + ssid + "\",\"" + password + "\"");
// set your home router SSID and password
  showResponse(5000);

  if (DEBUG)  Serial.println("Setup completed");
}
void loop()
{
 int ADC;
float temperature;
  ADC = analogRead(LM35);
  //Calculate Temperature from ADC value
  //Note that we use mV for Vref
  //Vin = ADCresult*Vref/(2^10)
  //Temp(C) = Vin/(10) = ADCresult*Vref/(1024*10)
  temperature = ADC*1100/(1024*10);
  Serial.print("Temperature = ");
  Serial.print(temperature);
  Serial.println(" *C");

  float myBPM = pulseSensor.getBeatsPerMinute();
// Calls function on our pulseSensor object that returns BPM as an "int".
    if (pulseSensor.sawStartOfBeat()) // Constantly test to see if "a beat happened".
   {
Serial.println(",ô•  A HeartBeat Happened ! "); // If test is "true", print a message "a heartbeat
happened".
    Serial.print("BPM: ");                        // Print phrase "BPM: "
    Serial.println(myBPM);            // Print the value inside of myBPM.
      }
  thingSpeakWrite(temperature, myBPM);
  delay(1000);
}
```

Références j

[1] Jinesh Ahamed, Amala V. Rajan, « Internet of Things (IoT): Application Systems and Security Vulnerabilities ». 2016.

[2] Kevin Ashton, « That Internet of Things Thing ». http://www.itrco.jp/libraries/RFIDjournal-That%20Internet%20of%20Things%20Thing.pdf (consulté le févr. 04, 2018).

[3] Han mei, 2Zhanghang, « BUSINESS INTELLIGENCE ARCHITECTURE BASED ON INTERNET OF THINGS ».

[4] M. Weill et M. Souissi, « L'Internet des objets : concept ou réalité ? », *Ann. Mines - Réal. Ind.*, vol. Novembre 2010, n° 4, p. 90, 2010, doi: 10.3917/rindu.104.0090.

[5] P.-J. Benghozi, S. Bureau, et F. Massit-Folea, « L'Internet des objets. Quels enjeux pour les Européens? », 2008.

[6] A. Bassi et G. Horn, « Internet of Things in 2020: Roadmap for the future », *Internet Things*, p. 27, 2020.

[7] E. A. Kosmatos, N. D. Tselikas, et A. C. Boucouvalas, « Integrating RFIDs and Smart Objects into a UnifiedInternet of Things Architecture », *Adv. Internet Things*, vol. 01, n° 01, p. 5-12, 2011, doi: 10.4236/ait.2011.11002.

[8] J. A. Stankovic, « Wireless sensor networks », *computer*, vol. 41, n° 10, 2008.

[9] orange, « Livre blanc Machine To Machine enjeux et perspectives ».

[10] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, et M. Ayyash, « Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications », *IEEE Commun. Surv. Tutor.*, vol. 17, n° 4, p. 2347-2376, 2015, doi: 10.1109/COMST.2015.2444095.

[11] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, et B. Sikdar, « A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures », *IEEE Access*, vol. 7, p. 82721-82743, 2019, doi: 10.1109/ACCESS.2019.2924045.

[12] Z. Yang et Y. Peng, « Study and Application on the Architecture and Key Technologies for IOT », p. 5.

[13] M. Wu, T. Lu, et F.-Y. Ling, « Research on the architecture of Internet of things », p. 4, 2010.

[14] M. A. Chaqfeh et N. Mohamed, « Challenges in Middleware Solutions for the Internet of Things », p. 6.

[15] R. Khan, S. U. Khan, R. Zaheer, et S. Khan, « Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges », p. 4.

[16] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, « Constrained Application Protocol (CoAP).draft-ietf-core-coap-18 ». Internet Eng. Task Force (IETF), Fremont, CA, USA, 2018.

[17] C. Bormann, A. P. Castellani, et Z. Shelby, « CoAP: An Application Protocol for Billions of Tiny Internet Nodes », *IEEE INTERNET Comput.*, p. 6.

[18] U. Hunkeler, H. L. Truong, et A. Stanford-Clark, « MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks », in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, Bangalore, India, janv. 2008, p. 791-798. doi: 10.1109/COMSWA.2008.4554519.

[19] P. Saint-Andre, « Extensible messaging and presence protocol (XMPP): Core ». Internet Eng. Task Force (IETF), Fremont, CA, USA, Request for Comments:, 2011.

[20] IBM developerWorks, Markham, « Meet the Extensible Messaging and Presence Protocol (XMPP) », *IBM Developer*, sept. 19, 2009. https://developer.ibm.com/technologies/messaging/tutorials/x-xmppintro/ (consulté le mai 04, 2021).

[21] Adv. Open Std. Inf. Soc. (OASIS), « OASIS Advanced Message Queuing Protocol », *Burlington, MA, USA*, 2012. http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf (consulté le mai 04, 2021).

[22] C. Wei et Y. Li, « Design of energy consumption monitoring and energy-saving management system of intelligent building based on the Internet of things », in *2011 International Conference on Electronics, Communications and Control (ICECC)*, Ningbo, China, sept. 2011, p. 3650-3652. doi: 10.1109/ICECC.2011.6066758.

[23] C. H. Liu, J. Fan, J. W. Branch, et K. K. Leung, « Toward QoI and Energy-Efficiency in Internet-of-Things Sensory Environments », *IEEE Trans. Emerg. Top. Comput.*, vol. 2, n° 4, p. 473-487, déc. 2014, doi: 10.1109/TETC.2014.2364915.

[24] Y. Yang, L. Wu, G. Yin, L. Li, et H. Zhao, « A Survey on Security and Privacy Issues in

Internet-of-Things », *IEEE INTERNET THINGS J.*, p. 11.

[25] A. Oracevic, S. Dilek, et S. Ozdemir, « Security in internet of things: A survey », in *2017 International Symposium on Networks, Computers and Communications (ISNCC)*, Marrakech, Morocco, mai 2017, p. 1-6. doi: 10.1109/ISNCC.2017.8072001.

[26] S. Babar, A. Stango, N. Prasad, J. Sen, et R. Prasad, « Proposed embedded security framework for Internet of Things (IoT) », in *2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE)*, Chennai, India, févr. 2011, p. 1-5. doi: 10.1109/WIRELESSVITAE.2011.5940923.

[27] S. Sridhar et D. S. Smys, « Intelligent Security Framework for IoT Devices », p. 5.

[28] T. Song, R. Li, B. Mei, J. Yu, X. Xing, et X. Cheng, « A Privacy Preserving Communication Protocol for IoT Applications in Smart Homes », *IEEE Internet Things J.*, vol. 4, nº 6, p. 1844-1852, déc. 2017, doi: 10.1109/JIOT.2017.2707489.

[29] H. Jiang, F. Shen, S. Chen, K.-C. Li, et Y.-S. Jeong, « A secure and scalable storage system for aggregate data in IoT », *Future Gener. Comput. Syst.*, vol. 49, p. 133-141, août 2015, doi: 10.1016/j.future.2014.11.009.

[30] J. Jermyn, R. P. Jover, I. Murynets, M. Istomin, et S. Stolfo, « Scalability of Machine to Machine systems and the Internet of Things on LTE mobile networks », in *2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Boston, MA, USA, juin 2015, p. 1-9. doi: 10.1109/WoWMoM.2015.7158142.

[31]« Converging_Technologies_for_Smart_Environments_and_Integrated_Ecosystems_IERC_Book_Open_Access_2013.pdf ». Consulté le: janv. 31, 2018. [En ligne]. Disponible sur: http://www.internet-of-things-research.eu/pdf/Converging_Technologies_for_Smart_Environments_and_Integrated_Ecosystems_IERC_Book_Open_Access_2013.pdf

[32] Md. Mahmud Hossain, Maziar Fotouhi, and Ragib Hasan, « Towards an Analysis of Security Issues, Challenges, and Open Problems in the Internet of Things ». 2015.

[33] Ioannis Andrea, Chrysostomos Chrysostomou, George Hadjichristofi, « Internet of Things: Security Vulnerabilities and Challenges ». 2015.

[34] L. Li, « Study on security architecture in the Internet of Things », in *Measurement, Information and Control (MIC), 2012 International Conference on*, 2012, vol. 1, p. 374-377.

[35] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, et G. Pantziou, « A survey on jamming attacks and countermeasures in WSNs », *IEEE Commun. Surv. Tutor.*, vol. 11, nº 4, p. 42-56, 2009, doi: 10.1109/SURV.2009.090404.

[36] J. Granjal, E. Monteiro, et J. Sa Silva, « Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues », *IEEE Commun. Surv. Tutor.*, vol. 17, nº 3, p. 1294-1312, 2015, doi: 10.1109/COMST.2015.2388550.

[37] B. Khoo, « RFID as an Enabler of the Internet of Things: Issues of Security and Privacy », oct. 2011, p. 709-712. doi: 10.1109/iThings/CPSCom.2011.83.

[38] A. Mitrokotsa, M. R. Rieback, et A. S. Tanenbaum, « Classification of RFID attacks », *G. E. N.*, vol. 15693, p. 14443, 2010.

[39] R. Uttarkar et R. Kulkarni, « Internet of things: architecture and security », *Int. J. Comput. Appl.*, vol. 3, p. 12-19, 2014.

[40] R. P. Padhy, M. R. Patra, and S. C. Satapathy, *Cloud Computing: Security Issues and Research Challenges*. Piscataway, NJ: IEEE, 2008.

[41] D. Wu, G. Hu, et G. Ni, « Research and improve on secure routing protocols in wireless sensor networks », in *Circuits and Systems for Communications, 2008. ICCSC 2008. 4th IEEE International Conference on*, 2008, p. 853-856.

[42] J. Newsome, E. Shi, D. Song, et A. Perrig, « The sybil attack in sensor networks: analysis & defenses », in *Proceedings of the 3rd international symposium on Information processing in sensor networks*, 2004, p. 259-268.

[43] J. Deogirikar et A. Vidhate, « Security attacks in IoT: A survey », in *I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), 2017 International Conference on*, 2017, p. 32-37.

[44] Sachin Babar∗, Parikshit Mahalle, Antonietta Stango, Neeli Prasad, et and Ramjee Prasad, « Proposed Security Model and Threat Taxonomy for the Internet of Things (IoT) ». 2010.

[45] Open Web Application Security Project, « Top 10 IoT Vulnerabilities (2014) Project ».

[46] Tom N. Jagatic, Nathaniel A. Johnson, Markus Jakobsson, Filippo Menczer, « Social

Phishing ». http://sci-hub.tw/10.1145/1290958.1290968 (consulté le févr. 02, 2018).

[47] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, et K. Wehrle, « Security Challenges in the IP-based Internet of Things », *Wirel. Pers. Commun.*, vol. 61, nº 3, p. 527-542, déc. 2011, doi: 10.1007/s11277-011-0385-5.

[48] S. Andy, B. Rahardjo, et B. Hanindhito, « Attack scenarios and security analysis of MQTT communication protocol in IoT system », in *Electrical Engineering, Computer Science and Informatics (EECSI), 2017 4th International Conference on*, 2017, p. 1-6.

[49] M. Batty *et al.*, « Smart cities of the future », *Eur. Phys. J. Spec. Top.*, vol. 214, nº 1, Art. nº 1, nov. 2012, doi: 10.1140/epjst/e2012-01703-3.

[50] ITU-T Focus Group on Smart Sustainable Cities, « "Smart sustainable cities: An analysis of definitions" ». Focus Group Technical Report, Geneva, Switzerland, Tech. Rep. FG-SSC-10/2014, 2014.

[51] S. P. Mohanty, U. Choppali, et E. Kougianos, « Everything you wanted to know about smart cities: The Internet of things is the backbone », *IEEE Consum. Electron. Mag.*, vol. 5, nº 3, p. 60-70, juill. 2016, doi: 10.1109/MCE.2016.2556879.

[52] R. Minerva, A. Biru, and D. Rotondi, « Towards a definition of the internet of things (iot) », 2015.

[53] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, et K. Wehrle, « Security Challenges in the IP-based Internet of Things », *Wirel. Pers. Commun.*, vol. 61, nº 3, p. 527-542, déc. 2011, doi: 10.1007/s11277-011-0385-5.

[54] Shancang Li & Li Da Xu & Shanshan Zha, « The internet of things: a survey ». 2014.

[55] A.-R. Sadeghi, C. Wachsmann, et M. Waidner, « Security and privacy challenges in industrial internet of things », 2015, p. 1-6. doi: 10.1145/2744769.2747942.

[56] Z.-K. Zhang, M. C. Y. Cho, et S. Shieh, « Emerging Security Threats and Countermeasures in IoT », 2015, p. 1-6. doi: 10.1145/2714576.2737091.

[57] Shelby, Z., Hartke, K., & Bormann, C., *The constrained application protocol*. 2014.

[58] Sangyoon Oh, Jai-Hoon Kim, Geoffrey Fox, « Real-Time Performance Analysis for Publish/Subscribe Systems ». Future Generation Computer Systems, 2010.

[59] T. Salman, « Internet of Things Protocols and Standards », 2015.

[60] A. Foster, « Messaging technologies for the industrial internet and the internet of things whitepaper ». PrismTech, 2015.

[61] HAN Ngoc Son, « Semantic service provisioning for 6LoWPAN: powering internet of things applications on web », 2015.

[62] J. L. Fernandes, I. C. Lopes, J. J. Rodrigues, et S. Ullah, « Performance evaluation of RESTful web services and AMQP protocol », in *Ubiquitous and Future Networks (ICUFN), 2013 Fifth International Conference on*, 2013, p. 810-815.

[63] T. H. Bloebaum et F. T. Johnsen, « Evaluating publish/subscribe approaches for use in tactical broadband networks », in *Military Communications Conference, MILCOM 2015-2015 IEEE*, 2015, p. 605-610.

[64] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, et M. Ayyash, « Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications », *IEEE Commun. Surv. Tutor.*, vol. 17, n° 4, p. 2347-2376, 2015, doi: 10.1109/COMST.2015.2444095.

[65] I. Grigorik, « Making the web faster with HTTP 2.0 », *Queue*, vol. 11, n° 10, p. 40, 2013.

[66] N. Naik et P. Jenkins, « Web protocols and challenges of web latency in the web of things », in *Ubiquitous and Future Networks (ICUFN), 2016 Eighth International Conference on*, 2016, p. 845-850.

[67] Andrew Banks and Rahul Gupta, « MQTT Version 3.1.1 », *Jt. Pap. Open Group OASIS OMG*, 2014.

[68] A. Niruntasukrat, C. Issariyapat, P. Pongpaibool, K. Meesublak, P. Aiumsupucgul, et A. Panya, « Authorization mechanism for mqtt-based internet of things », in *Communications Workshops (ICC), 2016 IEEE International Conference on*, 2016, p. 290-295.

[69] D.-H. Mun, M. L. Dinh, et Y.-W. Kwon, « An Assessment of Internet of Things Protocols for Resource-Constrained Applications », juin 2016, p. 555-560. doi: 10.1109/COMPSAC.2016.51.

[70] N. Naik, « Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP », in *Systems Engineering Symposium (ISSE), 2017 IEEE International*, 2017, p. 1-7.

[71] Mrunali L. Va Monali Niko, Priyanka A. Jalan, Roshan Chaudhary, « EFFECTIVE PROCESSING OF MQTT PROTOCOL IN INTERNET OF THINGS (IOT) FOR SMART SYSTEM ».

[72] Md. M. Hossain, M. Fotouhi, et R. Hasan, « Towards an Analysis of Security Issues, Challenges, and Open Problems in the Internet of Things », juin 2015, p. 21-28. doi: 10.1109/SERVICES.2015.12.

[73] E. Gelbstein, « Data Integrity—Information Security's Poor Relation », *Isaca J.*, vol. 6, p. 20, 2011.

[74] Andrew Banks and Rahul Gupta, « MQTT Version 3.1.1 », *Jt. Pap. Open Group OASIS OMG*, 2014.

[75] Imane SAHMI , Tomader MAZRI, Nabil HMINA, « Security Study of Different Threats in Internet of Things », 2018.

[76] S. Vashi, J. Ram, J. Modi, S. Verma, et C. Prakash, « Internet of Things (IoT): A vision, architectural elements, and security issues », in *I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), 2017 International Conference on*, 2017, p. 492-496.

[77] R. Khan, S. U. Khan, R. Zaheer, et S. Khan, « Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges », p. 4.

[78] L. Atzori, A. Iera, et G. Morabito, « The Internet of Things: A survey », *Comput. Netw.*, vol. 54, n° 15, Art. n° 15, oct. 2010, doi: 10.1016/j.comnet.2010.05.010.

[79] I. Yaqoob *et al.*, « Internet of Things Architecture: Recent Advances, Taxonomy, Requirements, and Open Challenges », *IEEE Wirel. Commun.*, vol. 24, n° 3, p. 10-16, juin 2017, doi: 10.1109/MWC.2017.1600421.

[80] F. A. Alaba, M. Othman, I. A. T. Hashem, et F. Alotaibi, « Internet of Things security: A survey », *J. Netw. Comput. Appl.*, vol. 88, p. 10-28, 2017.

[81] M. B. Yassein, M. Q. Shatnawi, et D. Al-zoubi, « Application layer protocols for the Internet of Things: A survey », in *2016 International Conference on Engineering & MIS (ICEMIS)*, Agadir, Morocco, sept. 2016, p. 1-4. doi: 10.1109/ICEMIS.2016.7745303.

[82] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, et W. Zhao, « A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications », *IEEE Internet Things J.*, vol. 4, n° 5, p. 1125-1142, oct. 2017, doi: 10.1109/JIOT.2017.2683200.

[83] S. and P. International Conference on Cryptography, University of Malaya in Kuala Lumpur, Association for Computing Machinery, et ICMIP (Conference), *ICCSP 2019: proceedings*

of 2019 the 3rd International Conference on Cryptography, Security and Privacy ; ICMIP 2019, with workshop 2019 the 4th International Conference on Multimedia and Image Processing : University of Malaya, Kuala Lumpur, Malaysia, January 19-21, 2019. 2019. Consulté le: mars 09, 2020. [En ligne]. Disponible sur: http://dx.doi.org/10.1145/3309074

[84] A. Bhawiyuga, M. Data, et A. Warda, « Architectural design of token based authentication of MQTT protocol in constrained IoT device », in *2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA)*, 2017, p. 1-4.

[85] A. Rahman, S. Roy, M. S. Kaiser, et M. S. Islam, « A Lightweight Multi-tier S-MQTT Framework to Secure Communication between low-end IoT Nodes », in *2018 5th International Conference on Networking, Systems and Security (NSysS)*, 2018, p. 1-6.

[86] A. Niruntasukrat, C. Issariyapat, P. Pongpaibool, K. Meesublak, P. Aiumsupucgul, et A. Panya, « Authorization mechanism for mqtt-based internet of things », in *2016 IEEE International Conference on Communications Workshops (ICC)*, 2016, p. 290-295.

[87] L. Bisne et M. Parmar, « Composite secure MQTT for Internet of Things using ABE and dynamic S-box AES », in *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, 2017, p. 1-5.

[88] Meena Singh, Rajan MA, Shivraj VL, and Balamuralidhar P, « Secure MQTT for Internet of Things (IoT) », présenté à 2015 Fifth International Conference on Communication Systems and Network Technologies.

[89] A. Mektoubi, H. L. Hassani, H. Belhadaoui, M. Rifi, et A. Zakari, « New approach for securing communication over MQTT protocol A comparaison between RSA and Elliptic Curve », in *2016 Third International Conference on Systems of Collaboration (SysCo)*, 2016, p. 1-6.

[90] Y. Upadhyay, A. Borole, et D. Dileepan, « MQTT based secured home automation system », in *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, 2016, p. 1-4.

[91] S. Shin, K. Kobara, C.-C. Chuang, et W. Huang, « A security framework for MQTT », in *2016 IEEE Conference on Communications and Network Security (CNS)*, 2016, p. 432-436.

[92] Marco Calabretta, Riccardo Pecori, Massimo Vecchio, and Luca Veltri, « MQTT-Auth: a Token-based Solution to Endow MQTT with Authentication and Authorization Capabilities », *JOURNAL OF COMMUNICATIONS SOFTWARE AND SYSTEMS*, p. VOL. 14, NO. 4,

DECEMBER 2018, 2018.

[93] S. Hernández Ramos, M. T. Villalba, et R. Lacuesta, « MQTT Security: A Novel Fuzzing Approach », *Wirel. Commun. Mob. Comput.*, vol. 2018, p. 1-11, 2018, doi: 10.1155/2018/8261746.

[94] A. Esfahani *et al.*, « A Lightweight Authentication Mechanism for M2M Communications in Industrial IoT Environment », *IEEE Internet Things J.*, vol. 6, nº 1, p. 288-296, févr. 2019, doi: 10.1109/JIOT.2017.2737630.

[95] L. Malina, G. Srivastava, P. Dzurenda, J. Hajny, et R. Fujdiak, « A Secure Publish/Subscribe Protocol for Internet of Things », in *Proceedings of the 14th International Conference on Availability, Reliability and Security - ARES '19*, Canterbury, CA, United Kingdom, 2019, p. 1-10. doi: 10.1145/3339252.3340503.

[96] S. S. Dhanda, B. Singh, et P. Jindal, « Lightweight Cryptography: A Solution to Secure IoT », *Wirel. Pers. Commun.*, vol. 112, nº 3, p. 1947-1980, juin 2020, doi: 10.1007/s11277-020-07134-3.

[97] A. Abderrahim, F. Ibtissam, C. Habiba, E. A. Hicham, et H. Nabil, « AES-PRESENT: A NEW SECURE IOT-BASED SCHEME FOR TELEMEDICINE AND E-HEALTH SYSTEMS », vol. 13, nº 24, p. 6, 2018.

[98] A. Bogdanov *et al.*, « PRESENT: An ultra-lightweight block cipher », in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2007, p. 450-466.

[99] R. Mahmoud, T. Yousuf, F. Aloul, et I. Zualkernan, « Internet of Things (IoT) Security: Current Status, Challenges and Prospective Measures », p. 6.

[100] R. Roman, J. Zhou, et J. Lopez, « On the features and challenges of security and privacy in distributed internet of things », *Comput. Netw.*, vol. 57, nº 10, p. 2266-2279, juill. 2013, doi: 10.1016/j.comnet.2012.12.018.

[101] S. Shin et K. Kobara, « Efficient augmented password-only authentication and key exchange for IKEv2 », *IETF RFC 6628 Exp.*, 2012.

[102] I. Sahmi, T. Mazri, et N. Hmina, « Study of the Different Security Threats on the Internet of Things and their Applications », in *Proceedings of the 2nd International Conference on Networking, Information Systems & Security - NISS19*, Rabat, Morocco, 2019, p. 1-6. doi: 10.1145/3320326.3320402.

[103] Australian Institute of Health and Welfare, *Australia's health 2016.* 2016.

[104] E. Perrier, « Positive Disruption: Healthcare, Ageing and Participation in the Age of Technology ». Sydney, NSW, Australia: The McKell Institute, 2015.

[105] P. Gupta, D. Agrawal, J. Chhabra, et P. K. Dhir, « IoT based smart healthcare kit », in *Computational Techniques in Information and Communication Technologies (ICCTICT), 2016 International Conference on*, 2016, p. 237-242.

[106] Z. PANG, « Technologies and Architectures of the Internet-of-Things (IoT) for Health and Well-being », p. 91.

[107] S. M. Riazul Islam, Daehan Kwak, M. Humaun Kabir, M. Hossain, et Kyung-Sup Kwak, « The Internet of Things for Health Care: A Comprehensive Survey », *IEEE Access*, vol. 3, p. 678-708, 2015, doi: 10.1109/ACCESS.2015.2437951.

[108] S. Sivagami, D. Revathy, et L. Nithyabharathi, « Smart Health Care System Implemented Using IoT », *Int. J. Contemp. Res. Comput. Sci. Technol.*, vol. 2, nº 3, 2016.

[109] L. Catarinucci *et al.*, « An IoT-Aware Architecture for Smart Healthcare Systems », *IEEE Internet Things J.*, vol. 2, nº 6, Art. nº 6, déc. 2015, doi: 10.1109/JIOT.2015.2417684.

[110] C. Occhiuzzi, C. Vallese, S. Amendola, S. Manzari, et G. Marrocco, « NIGHT-Care: A Passive RFID System for Remote Monitoring and Control of Overnight Living Environment », *Procedia Comput. Sci.*, vol. 32, p. 190-197, 2014, doi: 10.1016/j.procs.2014.05.414.

[111] H. Mansor, M. H. A. Shukor, S. S. Meskam, N. Q. A. M. Rusli, et N. S. Zamery, « Body temperature measurement for remote health monitoring system », in *Smart Instrumentation, Measurement and Applications (ICSIMA), 2013 IEEE International Conference on*, 2013, p. 1-5.

[112] D. S. Rajput et R. Gour, *An IoT Framework for Healthcare Monitoring System.* LAP LAMBERT Academic Publishing, 2017.