

THESE DE DOCTORAT EN COTUTELLE ENTRE L'UNIVERSITE BOURGOGNE FRANCHE-COMTE ET L'UNIVERSITE DE LA MANOUBA

École doctorale n°37

Présentée par

Maissa DAMMAK

Pour obtenir le

Grade de Docteur de l'Université de Bourgogne

Spécialité: **Informatique**

Authentication and Authorization Security Solutions for the Internet of Things

Thèse présentée et soutenue à l'ISAT, le 17 Juillet 2021 devant le Jury composé de :

Pascal URIEN	Président de jury	Professeur, Telecom-paris, France
Sidi-Mohammed SENOUCI	Directeur de thèse	Professeur, Université de Bourgogne, France
Leila SAIDANE	Co-Directeur de thèse	Professeur, Université de la Manouba, Tunis
Christophe GRANSART	Co-encadrant	Charge de Recherche, IFSTTAR, France
Mohamed Houcine ELHDHILI	Co-encadrant	Maitre Assistant, Université de la Manouba, Tunis
Hakima CHAOUCHI	Rapporteur	Professeur, Institut Telecom, Telecom Sud Paris, France
Abdelmadjid BOUABDALLAH	Rapporteur	Professeur, Université de Technologie de Compiègne, France
Yacine GHAMRI-DOUDANE	Examineur	Professeur, Université de La Rochelle, France
Isabelle CHRISMENT	Examineur	Professeur, Telecom Nancy, France

“No research is ever quite complete. It is the glory of a good bit of work that it opens the way for something still better, and this repeatedly leads to its own eclipse.”

-Mervin Gordon-

Acknowledgment

A successful project big or small is always achieved due to the effort of a group of helpful people who have always given their valuable advice or lend a helping hand. I have the honor and the pleasure to express my gratitude and appreciation to all those who have guided, assisted and supervised me during the development of this thesis.

I would like to express my sincere gratitude to my thesis supervisor Professor Sidi-Mohammed SENOUCI for his continuous support, and guidance throughout the realization of this thesis and beyond. His availability, constant encouragement, and his patience made for a great working relationship and the motivation for me to finish. Big thanks once again to him for giving me the chance to do this PhD and offering me the best conditions and opportunities to succeed this work and making it as it is today. Thank you Sidi!

I would also like to thank all the jury members, Abdelmadjid BOUABDALLAH, Hakima CHAOUCHI, Yacine GHAMRI-DOUDANE, Isabelle CHRISMENT, and Pascal URIEN, for their participation in my PhD defense.

I am grateful to my joint supervisor, Professor Leila SAIDANE for her advice and guidance. I greatly appreciate her valuable support and cheerful encouragement. Without her praise, endorsement and valuable help, I would have never gotten the chance to undertake this thesis.

I will be forever grateful to my co-supervisor Christophe GRANSART for his advice, guidance, and deep insights that helped me at various stages of my research. I really appreciate his useful suggestions so that I could improve my research and writing skills.

I would like to extend my thanks to Mohamed Houcine ELHDAHILI who co-supervised this work. His valuable suggestions and expertise throughout this thesis definitely help improving the quality and presentation of this work. I appreciate his patience during all those long meetings that we had together in order to come up with answers and provide solutions for each problem.

I gratefully acknowledge the funding sources that made my Ph.D. work possible. This work is achieved as part of the European project ITEA PARFAIT, which is partially funded by FEDER (European Regional Development Fund), BPIFRANCE, the BFC region (Bourgogne-Franche Comté), and l'Agglomération de Nevers.

My heartfelt thanks go to my family for always believing in me. I thank my mother for her blessing and tireless support all the way through my adult life, I would have not be what I am today. A special thank goes to my great role model of resilience, and strength, my father who sacrifices his life for me keeping encouraging and motivating me. I feel so blessed and thankful to have such devoted parents who always expressed how proud they are of me. I shall always be in their debt, without them, I would have achieved nothing. I would never forgot to thank my sister for her caring, her help and support throughout my struggle and for being there for me in good and bad times. I am also thankful for my little brother who is the light and joy of my life for his joyful humor, which gave me the strength and made this arduous journey easier.

My deepest thanks to all my friends who were by my side and made this hard adventure easier, for me, especially Amel, Awatef, Wided, Mariem, Aicha and Rahma who have always been

there despite the thousands of kilometers that separate us. Special thanks goes for my friend Amina who has stood by me especially during writing the manuscript and with the vicissitudes moments of the pandemic that helped me a lot to reach my goal.

Contents

<i>Introduction</i>	<i>1</i>
1.1. Context and Motivation.....	1
1.2. Opportunities and Challenges.....	2
1.3. Dissertation Scope and Methodology.....	3
1.4. Thesis Structure and Contributions Overview	5
<i>Authentication and Access Control in IoT Environment.....</i>	<i>9</i>
2.1 Introduction.....	9
2.2 Internet of Things (IoT).....	10
2.2.1 IoT Architecture.....	11
2.2.1.1 Perception Layer.....	11
2.2.1.2 Network Layer.....	12
2.2.1.3 Service Management Layer	13
2.2.1.4 Application & Business layers	13
2.2.2 IoT Applications	13
2.2.2.1 Smart Home.....	14
2.2.2.2 Smart Grid.....	14
2.2.2.3 Transportation Systems.....	14
2.2.2.4 Healthcare.....	15
2.2.2.5 Smart Cities	15
2.2.2.6 Manufacturing and Industrial IoT.....	15
2.2.3 IoT Challenges	15
2.3 Security in the Internet of Things.....	18
2.3.1 IoT Security Challenges and Security High Requirements.....	21
2.3.2 Authentication in IoT	25
2.3.2.1 Centralized IoT Authentication Architecture	26
2.3.2.2 Distributed IoT Authentication Architecture	30
2.3.2.2.1 Distributed Trusted Authentication IoT Infrastructure	30
2.3.2.2.2 Distributed Authentication Architecture Based on Blockchain	31
2.3.2.2.2.1 Blockchain Overview.....	31
2.3.2.2.2.2 Authentication in IoT based on Blockchain	32
2.3.3 Access Control in IoT	34
2.3.3.1 Traditional Access Control Solutions.....	34
2.3.3.2 Group Key Management Solutions in IoT	36

2.4	Summary & Discussion.....	40
2.5	Conclusion	41
	<i>Token-based Lightweight Authentication for IoT environment.....</i>	<i>42</i>
3.1.	Introduction.....	42
3.2.	Related Works	43
3.3.	Background.....	46
3.3.1.	One-way Hash Function.....	46
3.3.2.	Symmetric Key Cryptography	47
3.4.	System Model and Security Requirements	47
3.4.1.	System Model	48
3.4.2.	Security and Threat Model.....	49
3.5.	Proposed Token Based Lightweight Authentication for IoT Environment (<i>TBLUA</i>) 49	
3.5.1.	Offline Smart Device and GW Registration phase	50
3.5.2.	User Registration Phase	51
3.5.3.	Token Distribution Between GW and Smart Device Phase.....	52
3.5.4.	Login, Authentication, and Key Agreement Phase	54
3.5.5.	Password Change Phase.....	57
3.6.	Security Evaluation.....	59
3.6.1.	Security Analysis.....	59
3.6.2.	Formal Verification Using AVISPA Tool.....	62
3.7.	Performance Analysis.....	64
3.7.1.	Functionality Comparison	64
3.7.2.	Computation Costs Comparison	65
3.7.3.	Communication Costs Comparison	66
3.8.	Proof of Concept Within Smart Hotel Use Case	67
3.8.1.	Smart Hotel Use Case.....	67
3.8.2.	Risk Management and Vulnerability.....	69
3.8.3.	Design of a Smart Lock Prototype	70
3.9.	Conclusion	74
	<i>DGKM-AC: Decentralized Group Key Management for Access Control in IoT.....</i>	<i>76</i>
4.1.	Introduction.....	76
4.2.	Related Works	77
4.3.	Background.....	81
4.3.1.	Group Key Management (GKM).....	81
4.3.2.	Master Key Encryption (MKE)	81

4.3.3.	Logical Key Hierarchy (LKH).....	82
4.3.4.	One-Time Pad (OTP) key.....	83
4.4.	DLGKM-AC General Overview.....	83
4.4.1.	System Architecture	83
4.4.2.	Threat Model	85
4.4.3.	System Requirements	85
4.5.	DLGKM-AC Detailed Description.....	86
4.5.1.	Overview	86
4.5.2.	Initialization of the System	87
4.5.3.	Registration Phase.....	89
4.5.3.1.	Device Groups Registration.....	89
4.5.3.2.	User Groups Registration.....	90
4.5.4.	Key Update Scenarios.....	92
4.5.4.1.	User joins/leaves Events Scenarios	92
4.5.4.2.	IoT Devices Joins/Leaves Events	96
4.6.	Security Analysis.....	100
4.6.1.	Forward Secrecy.....	101
4.6.2.	Backward Secrecy	102
4.6.3.	Collusion Attack Analysis Using Random Oracle Model	103
4.7.	Performance Analysis and Evaluation.....	104
4.7.1.	Performance Analysis	104
4.7.1.1.	Storage Overhead.....	104
4.7.1.2.	Computation Overhead	105
4.7.1.3.	Communication Overhead	105
4.7.2.	Performance Evaluation.....	106
4.7.2.1.	Storage Costs.....	106
4.7.2.2.	Computation Cost.....	108
4.7.2.2.1.	When a User Leaves a Group	109
4.7.2.2.2.	When a User Joins a Group	111
4.7.2.2.3.	When an IoT Device Joins a Group.....	114
4.7.2.2.4.	When an IoT Device Leaves a Group.....	114
4.7.2.3.	Communication Cost	115
4.8.	Conclusion	116
	<i>DiGABlock: Distributed Group Authentication based on Blockchain Technology</i>	117
5.1.	Introduction.....	117
5.2.	Related Works	119

5.3.	Background.....	121
5.3.1.	Elliptic Curve Cryptography (ECC).....	121
5.3.2.	Review on Shamir’s Secret Sharing Scheme.....	122
5.3.3.	Blockchain - Practical Byzantine Fault Tolerance Consensus Algorithm (PBFT) 123	
5.4.	System Model.....	124
5.4.1.	System Architecture.....	124
5.4.2.	Adaptation to the Smart Hotel Scenario.....	125
5.4.3.	Threat Model.....	126
5.4.4.	Security Goals.....	127
5.5.	DiGABlock Description.....	127
5.5.1.	Setting up the Blockchain Network.....	129
5.5.2.	Initialization Phase.....	129
5.5.3.	User Registration Phase.....	130
5.5.4.	Distributed Group Authentication Phase.....	134
5.5.5.	Consensus Phase.....	138
5.5.6.	Service Delivery Phase.....	141
5.6.	Security Evaluation.....	142
5.6.1.	Correctness Proof.....	142
5.6.2.	Security Analysis.....	144
5.6.2.1.	Forgery Attack.....	144
5.6.2.2.	Denial of Service Attack:.....	144
5.6.2.3.	Man-in-The-Middle Attack.....	145
5.6.2.4.	Replay Attack.....	145
5.6.2.5.	User Impersonation Attack.....	145
5.6.2.6.	Perfect Forward and Backward Secrecy.....	145
5.7.	Performance Analysis and Evaluation.....	146
5.7.1.	Experimental Settings.....	146
5.7.2.	Computation Costs.....	147
5.7.2.1.	User Overhead.....	147
5.7.2.2.	System Response Time.....	148
5.7.3.	Energy Consumption.....	149
5.7.4.	Communication Costs.....	150
5.7.5.	Improvement Rate of Communication Costs.....	152
5.8.	Conclusion.....	153
	Conclusion.....	154

6.1	Summary of the Contributions	154
6.2	Future Research Directions	156
	<i>Bibliography</i>	158
	<i>Appendix</i>	167

List of Figures

Figure.1. 1: Dissertation outline diagram	8
Figure.2. 1: Internet of Things.....	11
Figure.2. 2: IoT 5-layers architecture [11]	12
Figure.2. 3: IoT main applications	14
Figure.2. 4: Taxonomy of IoT authentication schemes	25
Figure.2. 5: Group Key Management Taxonomy	37
Figure.3. 1: Symmetric key cryptography	47
Figure.3. 2: Proposed Network model.....	48
Figure.3. 3: Security model	49
Figure.3. 4: User registration phase.....	52
Figure.3. 5: Token distribution phase.....	54
Figure.3. 6: Login, authentication and key agreement phase	58
Figure.3. 7: Results reported by the OFMC backend.....	63
Figure.3. 8: Computation costs.....	66
Figure.3. 9: Communication costs.....	67
Figure.3. 10: Smart Hotel architecture	68
Figure.3. 11: Smart Hotel prototype.....	71
Figure.3. 12 : Reservation phase	71
Figure.3. 13: Token generation	72
Figure.3. 14: Authorized access	73
Figure.3. 15: Unauthorized access	73
Figure.3. 16: (a) Successful access and (b) Unsuccessful access	74
Figure.4. 1: The logical key hierarchy tree structure.....	82
Figure.4. 2: Proposed system model	84
Figure.4. 3: Key distribution in our system model.....	92
Figure.4. 4 : Structure inside UG1 when U4 joins	94
Figure.4. 5: Structure inside UG1 when U3 leaves	96
Figure.4. 6: Examples of LKH structure updates for device join.....	98
Figure.4. 7: Examples of LKH structure updates for device leave.....	98
Figure.4. 8: Rekeying procedure based on OTP when a device joins a group	99
Figure.4. 9: Rekeying procedure when a device leaves a group	100
Figure.4. 10: Users' storage overhead while varying the number of devices.....	107
Figure.4. 11: Users' storage overhead while varying the number of users	107
Figure.4. 12: Devices storage overhead	108
Figure.4. 13: Remaining user computation overhead varying devices' number (user leave)	109
Figure.4. 14: Remaining user computation overhead varying users' number (user leave)	110
Figure.4. 15: Server time update on the user-leaving event	111
Figure.4. 16: Old user computation overhead varying the devices' number (user join)	112
Figure.4. 17: Old user computation overhead varying the users' number (join).....	112
Figure.4. 18: New user computation overhead varying devices.....	113
Figure.4. 19: Server time update on the joining event	113
Figure.4. 20: Computation cost: device join	114

Figure.4. 21: Computation overhead: device leave	115
Figure.4. 22: Communication costs.....	115
Figure.5. 1: Network model of the proposed scheme.....	124
Figure.5. 2: IoT environment of a Smart Hotel	126
Figure.5. 3: Workflow model of the proposed scheme	128
Figure.5. 4: Secret shared authenticator recovering	132
Figure.5. 5: Authentication & Service delivery phases	137
Figure.5. 6: User authentication data block.....	139
Figure.5. 7: Computation overhead on users.....	147
Figure.5. 8: System Response Time	148
Figure.5. 9: Energy consumption	149
Figure.5. 10: Communication costs.....	151
Figure.5. 11: Improvement Rate.....	152

List of Tables

Table 2. 1: Main security issues vs. Applications	18
Table 2. 2: Security services.....	20
Table 2. 3: Security IoT challenges.....	23
Table 2. 4: Access Control Solutions Comparison.....	36
Table 3. 1: Evaluation of IoT Authentication Schemes	44
Table 3. 2: Symbols and their descriptions.....	50
Table 3. 3: Functionality Features Comparison	64
Table 3. 4: Computation Costs Comparison.....	65
Table 3. 5: Vulnerability analysis of a smart hotel.....	69
Table 4. 1: Comparison of existing GKM Schemes.....	80
Table 4. 2: Keys' description	87
Table 4. 3: Summary of symbols and their description.....	88
Table 4. 4: Communication analysis.	105
Table 5. 1 : Comparison of existing Authentication Schemes	120
Table 5. 2: Comparison of consensus algorithms [27]	123
Table 5. 3: List of acronyms.....	129
Table 5. 4: Energy costs	149
Table 5. 5: Communication Parameters	150

List of Publications

❖ Accepted papers:

- M. Dammak, O. R. M. Boudia, M. A. Messous, S. M. Senouci and C. Gransart, "Token-Based Lightweight Authentication to Secure IoT Networks," *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA, 2019, pp. 1-4, doi: 10.1109/CCNC.2019.8651825.
- M. Dammak, S. -M. Senouci, M. A. Messous, M. H. Elhdhili and C. Gransart, "Decentralized Lightweight Group Key Management for Dynamic Access Control in IoT Environments," in *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1742-1757, Sept. 2020,
- M. Dammak *et al.*, "A Secure and Interoperable Platform for Privacy Protection in the Smart Hotel Context," *2020 Global Information Infrastructure and Networking Symposium (GIIS)*, Tunis, Tunisia, 2020, pp. 1-6, doi: 10.1109/GIIS50753.2020.9248483.

❖ Under review papers:

- M. Dammak, S. -M. Senouci, M. H. Elhdhili, C. Gransart and L. Saidane "DiGABlock:Distributed Group Authentication based on Blockchain technology for IoT," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Under review.
- M. Dammak, S. -M. Senouci, "Group Lightweight Authentication Based on Blockchain in IoT Environments" in *IEEE Internet of Thing Magazine*, Under review.

Glossary

Letter	Acronym	Description
A	ABAC	Attribute-Based Access Control
	ABE	Attribute-Based Encryption
	ACL	Access Control List
	ACM	Access Control Matrix
	AES	Advanced Encryption Standard
	AMI	Advanced Metering Infrastructure
	API	Application Programming Interface
	ATIS	Alliance for Telecommunications Industry Solutions
C	CA	Certification Authority
	CapBAC	Capability-Based Access Control
	CBA	Certificate-Based Authentication
	CBC	Cipher Block Chaining
	CIA	Confidentiality, Integrity, and Availability
	CoAP	Constrained Application Protocol
	CRP	Challenge-Response Pair
	CRT	Chinese Remainder Theorem
D	D2D	Device to Device
	DDoS	Distributed Denial of Service
	DEP	Dual-Encryption Protocol
	DES	Data Encryption Standard
	DoS	Denial of Service
	dPoS	delegated Proof of State
	DTLS	Datagram Transport Layer Security
E	ECC	Elliptic Curve Cryptography
	ECDSA	Elliptic Curve Digital Signature Algorithm
	EPID	Enhanced Privacy ID
	E-SAP	Efficient-Strong Authentication Protocol
	ETSI	European Telecommunications Standards Institute
F	FCAPS	Fault, Configuration, Accounting, Performance, and Security
G	GCRT	Generalized Chinese Remainder Theorem
	GKM	Group Key Management
H	HLPSL	High-Level Protocol Specification Language
	HMAC	Hash-based Message Authentication Code
	HTTP	Hypertext Transfer Protocol
	HVAC	Heating, Ventilation and Air-Conditioning
I	IBE	Identity-Based Encryption
	IIoT	Industrial Internet of Things
	IoT	Internet of Things
	IP	Internet Protocol
	IT	Information Technology
	ITS	Intelligent Transportation Systems
	ITS	Intelligent Transportation Systems
K	KDC	Key Distribution Center
	KEK	Key Encryption Key
L	LAN	Local Area Network
	LKH	Logical Key Hierarchy
	LoRaWAN	Low Radio Wide Area Networks
	LTE	Long-Term Evolution

	LWM2M	Lightweight Machine-to-Machine
M	MAC	Medium-Access Control
	MD5	Message Digest 5
	MKE	Master Key Encryption
	mMTC	massive Machine Type Communication
	MFA	Multi-Factor Authentication
	MQTT	Message Queuing Telemetry Transport
	M2M	Machine-to-Machine
N	NFC	Near Field Communication
O	OAuth	Open Authorization
	OBU	On-Board Unit
	OFT	One-way Function Tree
	OTP	One-Time Pad
P	PAuthKey	Pervasive Authentication Protocol and Key establishment
	PBFT	Practical Byzantine Fault Tolerance
	PFS	Perfect Forward Secrecy
	PKI	Public Key Infrastructure
	PUF	Physical Unclonable Functions
	PoET	Proof of Elapsed Time
	PoS	Proof of State
	PoW	Proof of Work
Q	QoE	Quality of Experience
	QoS	Quality of Service
R	RA	Registration Authority
	RBAC	Role-Based Access Control
	RFID	Radio-Frequency Identification
	ROM	Random Oracle Model
	ROR	Real-Or-Random
	RPL	IPv6 Routing Protocol
	RSA	Rivest, Shamir, Adleman (3 inventors of the RSA protocol)
	RSU	Roadside Unit
S	SDN	Software Defined Networking
	SHA	Secure Hashing Algorithm
	SKDCs	Sub Key Distribution Centers
T	TIA	Telecommunications Industry Association
	TEK	Traffic Encryption Key
	TPM	Trusted Platform Module
	TRNG	True Random Number Generator
	TTP	Trust Third-Party
U	UCON	Usage Control
V	VANETs	Vehicular Ad hoc Networks
	V2C	Vehicle to Cloud
	V2G	Vehicle to Grid
	V2I	Vehicle to Infrastructure
	V2P	Vehicle to Pedestrian
	V2V	Vehicle to Vehicle
W	WBAN	Wireless Body Area Network
	WiFi	Wireless Fidelity
	WSNs	Wireless Sensor Networks.
Z	ZigBee	Zonal Intercommunication Global-standard
	ZKP	Zero Knowledge Proof

Introduction

1.1. Context and Motivation

The Internet of Things (IoT) is a global new paradigm that considers connecting objects, intelligent systems, and applications in order to gather data from the physical world and offer IoT services to IoT consumers [42]. The IoT has emerged as a prominent solution that allows anyone to access anything from anywhere and anytime. In particular, it enables several physical objects prepared to collect data through the sensing and actuation capabilities, process, and exchange this data over the network transparently and seamlessly. The communication through the IoT network provides an entirely connected smart world in which objects collaborate to achieve a high-level new service dimension. Thus, the capacity to monitor and manage things in the physical world develops the spectrum of IoT applications that directly impacts the economics' increasing and the quality of our daily life. IoT supports numerous and massive IoT applications, including smart home, smart manufacturing, smart building, smart transportation, smart grid, and smart healthcare [1]. According to the published study [2], the anticipated number of connected IoT devices during this year, 2021, will reach 27.1 billion IoT devices. Despite the attractive promises of the developing IoT network, security presents a real issue that hinders its full deployment. In fact, IoT security is not efficiently established as it has not gained sufficient attention proportional to the IoT growth. The US Intelligence Community classifies the IoT as a significant cyber technology that can endanger data privacy, data integrity, and service availability. Besides, the IoT network's open nature makes it composed of many heterogeneous smart devices and characterized by a dynamic structure. Thus IoT security could be a disaster and more severe than traditional security problems in the Internet.

Moreover, IoT intensifies existing cyber-security issues and introduces a whole new degree of potential threats. We give in the following the some well-known attacks that occurred recently in the IoT network: (i) In December 2014, attackers penetrated a German steel mill facility by using booby-trapped emails to steal logins and obtained control access to the mill's control systems. Through this attack, they lead to a furnace explosion [3]. (ii) In October 2016, the Mirai [4] launched a malicious program that infected numerous IoT devices by taking control of connected objects such as surveillance cameras and routers and then initiated massive

distributed denial of service attacks (DDoS) by flooding servers. This attack leads to a botnet, which results in transforming the large-scale Internet network paralyzed. (iii) In 2017, a medical malware named BrickerBot damaged the healthcare application. Indeed, attackers used the brute force password and compromised medical IoT devices [5], destroyed their memory, and deleted their data. This attack has dangerous consequences on users of healthcare IoT applications.

To sum up, the IoT's attractiveness by the massive number of connected devices into IoT systems increases the attack surface and the hackers' possibilities to get unauthorized access and damage these systems. Therefore, adequate security mechanisms should be deployed to mitigate risks and respond to the dynamic environment's security requirements.

1.2. Opportunities and Challenges

Despite the encouraging advances of the IoT environment in our daily life, security and privacy challenge the way of its full development. Even though the IoT networks present the same security concerns with the Internet as the 4/5G security, WiFi security, and Internet Protocol (IP)-based security, the traditional security solutions cannot be directly implemented for the IoT environment [6]. Indeed, IoT's ability to connect billions of smart things, collecting sensitive and personal data, creates new degrees of security and privacy issues, especially authentication and authorization problems. According to the massive number of connected objects, IoT network records high volume communication traffic of exchanged/collected data that potentially threaten IoT. Therefore, this environment, characterized by high **scalability**, should apply an effective security solution to mitigate attackers' exploitation. Furthermore, another crucial challenge is related to the diversity in security requirements and resource availability. Explicitly, the potential number of communication standards and information system technologies with **heterogeneous** security configuration requirements will generate a complex networking model and impact the IoT systems' security. Besides, the remote access mechanisms and the sensitive exchanged data over the wireless channel attract many intruders' attention through physical and wireless access and increases the probability of **threats**. In particular, IoT systems are susceptible to denial-of-service (DoS) and distributed DoS (DDoS), in which an adversary may exploit network protocols with massive traffic [47] and degrade the system's **availability**. Adding to the fundamental illustrated challenges, the IoT environment is likely to face other silent challenges that stand in the way of its deployment. The **resource constrained** IoT devices cannot support the excessive computational requirements in cryptography and will be subject to high energy consumption. Besides, with the **distributed** and the high heterogeneity nature of an IoT environment, many IoT services are offered. However, devices may be added and removed, and users may subscribe and unsubscribe from these IoT services **dynamically**.

Consequently, dynamically unstable situations may impact authentication/authorization in IoT systems. The current preventive and security countermeasures solutions are inadequate and insufficient to successfully address these characteristics and mitigate threats. In fact, most of the designed authentication and authorization schemes rely on a centralized trusted third party, which might lead to a bottleneck in the IoT system due to the scalability issue and the dynamic

changes. Moreover, using a server-client model establishing one-to-one connections is not scalable with broadcasting communications and leads to repeated requests or authentication and access authorization. Therefore, frequent authentication and authorization require a more appropriate authentication and authorization architecture model with dynamic defense to address the security requirements. Some distributed authentication trusted frameworks were proposed to handle the scalability issue and eliminate the third party's load and trust. However, these mechanisms suffer from the necessity of having distributed trusted servers that are a great point of attackers' attraction. Hence, blockchain technology-based on trustless distributed nodes might be beneficial to deal very well with scalability and heterogeneity issues. However, since blockchain technology is energy and time-consuming due to the proof of work mechanism, it become important to think how to take advantage of this technology while taking into account resource-limited IoT devices.

In addition to authentication, granting the corresponding permissions with the dynamic IoT environment is a high challenge that needs to be addressed. The exiting solutions for access control are not suitable for a large dynamic environment and limited resources IoT environments. They need to rely on a connected third party to get access permissions continuously. Moreover, regarding the continuous growth of connected objects in IoT, group-based applications have emerged. Thus, authorization frameworks should also address the group security requirement to control the permissions' assignment. The Group Key Management (GKM) has been used as a prominent solution to achieve a secure and efficient access control in IoT. The group members in a heterogeneous IoT network could subscribe to IoT services and then change their subscriptions dynamically. In this context, members join and leave the IoT services group (subscribe/unsubscribe to an IoT service). Therefore, the group key must be changed whenever a member leaves or joins the group to ensure secure group communication, especially both forward and backward secrecy with respect to the resources and capacity of IoT objects.

1.3. Dissertation Scope and Methodology

According to the previously discussed IoT challenges and the security requirements, we introduce in this section how to build a secure IoT system to address the earlier mentioned issues. The large IoT environment connecting billion of things enumerates many security challenges that need to be overcome to deploy the IoT system efficiently. More specifically, authentication and authorization are essential security features for building a secure IoT environment. In this thesis, we focus on proposing authentication and access control protocols for IoT environments with respect to the dynamic nature of group communication, the security requirements, and the constrained features of IoT devices. We summarize in what follows our research objectives:

- Design a mutual lightweight authentication for a period of time that responds to the users' dynamic changes in the IoT environment and the resource-limited IoT devices.

- Deploy an efficient access control system using group key management that responds to the dynamic changes features and the group communications requirements and eliminates the third-party load.
- Design a distributed group authentication protocol for authenticating the user with many IoT services to eliminate the trust in the third-party using a trustless environment, meet the requirement of a large scale and heterogenous environment, and ensure a secure and efficient authentication for the group-based communication.

To design our secure IoT system regarding the security requirements, including scalability, heterogeneity, dynamic changes, automated authentication, limited resources, and security features, we summarize the adopted approaches in Table 1.1.

Table 1.1: Methodology of building our IoT secure system

Requirement	Design Principle	Approach	Chapter
Scalability	Distributed group authentication architecture	Blockchain	5
		Edge server	
		Shamir secret sharing	
	A decentralized access control architecture	Group Key Management	4
Heterogeneity	Ubiquitous network with heterogeneous entities	GKM for users	4
		GKM for IoT devices	
Dynamic changes	Dynamic access control	Logical key Hierarchical	4
		Master Token Encryption	
		One Time Pad encryption	
	Periodic authentication	Token of identification	3
Automated authentication	Distributed authentication for users with many IoT services	Trustless environment based on Blockchain	5
Limited resources	Considering the consumption from computation, Storage, Bandwidths, and Power	Lightweight authentication protocol	3
			5
Security & Privacy	Confidentiality	Secure Authentication and Access Control Management	3 4 5
	Integrity		
	Availability		
	Non-repudiation		
	Authentication		

1.4. Thesis Structure and Contributions Overview

We addressed two crucial security features in the IoT environment all over this thesis, including authentication and authorization. These security services are challenging regarding the peculiar characteristics of the progressing IoT network. To be more practical, we describe in the following a storyline scenario that uses our secure IoT system. We adopt a smart-hotel scenario equipped with various modern IoT technologies that make the guests' stay more comfortable, lower the energy consumption, and help the staff and management with their tasks. Our system gives the freedom for guests to choose a smart room or a standard room. Suppose two different guests named Bob and Jenny, where Jenny wants to book a smart room and Bob wants to book a standard room.

Bob, the first smart-hotel guest, passes by our smart hotel rapidly and wants to book a room at a low price for five days. After checking the hotel's availability on the given days, Bob uses the TBLUA system (chapter 3) to make a reservation. The hotel booking system uses TBLUA that generates a token of identification, which uses high-security standards to authenticate the guest during his accommodation and open the reserved room's smart lock. At Bob's arrival, Bob has not to pass by the reception for the check-in. Indeed, Bob uses his smartphone to connect to the hotel application, uses the received token and then opens his room by approaching his smartphone to the smart lock. During Bob's stay, his friend has joined him, and he preferred to stay with him in the same room. Therefore, Bob's friend needs to get access to Bob's room at any time during the accommodation. At this level, it is essential to share the entrance to the same room securely between Bob and his friend. The hotel ensures a secure share of access by using the DLGKM-AC system (chapter 4). In particular, the DLGKM-AC system controls the access to the same room and gives another token for Bob's friend, who could enter the room freely. At Bob's departure, the DLGKM-AC system updates Bob's room's smart lock by revoking the two tokens used during the last reservation.

Jenny is the second guest, who is impressed by the advanced technology of nowadays, has preferred a reservation to a smart room with full smart objects. Besides, after checking the hotel's availability on the given days, Jenny chooses to book a full smart room with many other IoT services offered by the hotel (smart cleaning, smart food operations, smart tourism, etc.). The hotel system uses DiGABlock (chapter 5) to reserve the room and register Jenny for the requested hotel services. At Jenny's arrival, the environment in the room is prepared on her preferences (such as the temperature, lighting, etc.), and Jenny needs to get access to her room and all objects in the room. Therefore, DiGABlock authenticates Jenny by making a full authentication to secure access to the smart room. At this level, Jenny's information is stored in the hotel's blockchain network to be used during her stay and for her next visit. During Jenny's accommodation, she enjoyed the offered IoT services, which are accessible based on her reservation request. In fact, DiGABlock ensures rapid and secure access to these IoT services through delivering these services without the need to re-authenticate. For instance, Jenny gets access to the cleaning robots and programmed them with her non-presence in the room. Thus, if Jenny is not in the room, robots could clean it. Knowing that Jenny has two kids with her, aged 10 and 18, the hotel uses the DLGKM-AC system to securely share the access between Jenny and her kids. Indeed, Jenny and her kids have different access permissions, they

all could access the room, but only Jenny and the adult kid could control the heater and all room's accessories, while only Jenny could program the robots to clean the room. Once Jenny and her kids check out the hotel, the DLGKM-AC updates all IoT devices used during the accommodation. Furthermore, the blockchain network store all information related to Jenny and her kids about their access to all IoT services and their preferences that are useful for further reservation.

The rest of this dissertation is organized as follows, as illustrated in Figure 1.1, representing our dissertation's diagram. Chapter 2 reviews the existing security solutions in the IoT environment based on the unconventional characteristics and the IoT security challenges, including heterogeneity, scalability, dynamic changes, and limited resources. Chapter 3 describes a lightweight authentication protocol to meet IoT's resource-constrained requirements and design a proof-of-concept representing a smart hotel use case. Chapter 4 addresses the problem of granting permissions access to users and IoT devices for a large-scale environment. Chapter 5 improves the authentication process in such a large-scale environment through designing distributed group authentication based on blockchain technology. Conclusions and future directions of the research are presented in Chapter 6. More details and contributions of each chapter are given in the following:

- **Chapter 2:**

Chapter 2 surveys the most prominent literature related to authentication and access control in IoT environments. Throughout this chapter, we first provide a detailed study of the IoT network by presenting its architecture and the challenges standing in the way of its deployment. Then, we point out the security challenges and requirements related to its development. We continue by presenting the existing solutions that address the fundamental security requirements, including the IoT authentication and access control schemes. We also give an overview of the different approaches used to handle the selected security requirements, such as blockchain, token, group key management, etc.

- **Chapter 3:**

This chapter proposes a new lightweight mutual authentication for a one-to-one scenario in IoT. In fact, using passwords or pre-defined keys is insufficient to authenticate legitimate users in a dynamic environment. For instance, in the smart hotel application that involves different IoT devices, the users dynamically change their reservation status. Meanwhile, the users who reserve the IoT services in the smart hotel should get authenticated only during their accommodation. Hence, a temporary authentication is required to give access to the hotel during the reservation period. In this context, and to enhance the robustness of authentication, the chapter proposes a new protocol named Token-Based Lightweight User Authentication (TBLUA). This protocol is achieved by adding a new security layer using the software token of identification mechanisms. In fact, adding to the password and the login, the token is used to identify the legitimate user during a specific period securely. This token is mainly designed to respond to the limitation of the resources of IoT devices. Both security and performance analysis show that the proposed scheme is a strong competitor among existing ones for user

authentication in IoT environments. Furthermore, we describe the smart hotel use case reservation system composed of one smart lock and study its vulnerability.

- **Chapter 4:**

In addition to the authentication requirement described in the previous chapter, it is crucial to give adequate permissions to legitimate users and IoT devices. In particular, in a large-scale dynamic IoT environment characterized by subscribers (users/IoT devices) that frequently change interest to IoT services, it is significant to maintain secure data distribution to legitimate subscribers. Therefore, we elaborate a novel Decentralized Lightweight Group Key Management architecture for Access Control in the IoT environment (DLGKMP-AC) that manages the dissemination of keys of access control and secure data distribution. This solution aims to address the scalability challenge introduced by the massive scale of IoT devices and the increased number of subscribers. This, thanks to a hierarchical architecture composed of one Key Distribution Center (KDC) and several Sub Key Distribution Centers (SKDCs), enhancing subscribers' management groups, and alleviating the rekeying overhead on the KDC. Furthermore, the solution removes the dependency of symmetric group keys per subgroup communication, which is inefficient when managing access control for subscribers with highly dynamic behavior. Hence, a new master token management protocol was introduced through this chapter to succeed in keys dissemination across a group of subscribers. This protocol reduces storage, computation, and communication overheads during join/leave events. Likewise, DLGKM-AC guarantees secure group communication by preventing collusion attacks and ensuring backward/forward secrecy. Simulation results and analysis show considerable resource gain in storage, computation, and communication overheads.

- **Chapter 5:**

The growing IoT environment offers many IoT services that might be composed of many IoT devices allowing group-based communication. Indeed, we can recognize that controlling unauthorized access to group communication is achieved through our solution in the previous chapter 4. However, before granting permission access, users need to authenticate themselves with all requested IoT services by authenticating with each IoT device composing these IoT services. These frequent and redundant authentication actions may lead both to exploit exchanged data of authentication mechanisms by intruders and signaling congestion. Therefore, to secure the communication in an environment with a large number of devices, we propose a novel Distributed Group Authentication system based on Blockchain technology (DiGABlock) to build an efficient and secure distributed group authentication system in an IoT environment based on group communication. In particular, we design a group authentication algorithm based on the threshold secret sharing technique through the Blockchain edge layer to allow users to authenticate securely within many groups of IoT devices in a distributed manner. In fact, users have to achieve only one full authentication process with an IoT service (a group of IoT devices), and then they need to complete a service delivery process to get authenticated with the rest of the required IoT services. Security analysis shows that DiGABlock resists man-in-the-middle and DDoS attacks. Furthermore, simulation results show that DiGABlock

outperforms exiting schemes by 75%-80% in terms of communication costs and conducts a considerable computation and energy consumption gain.

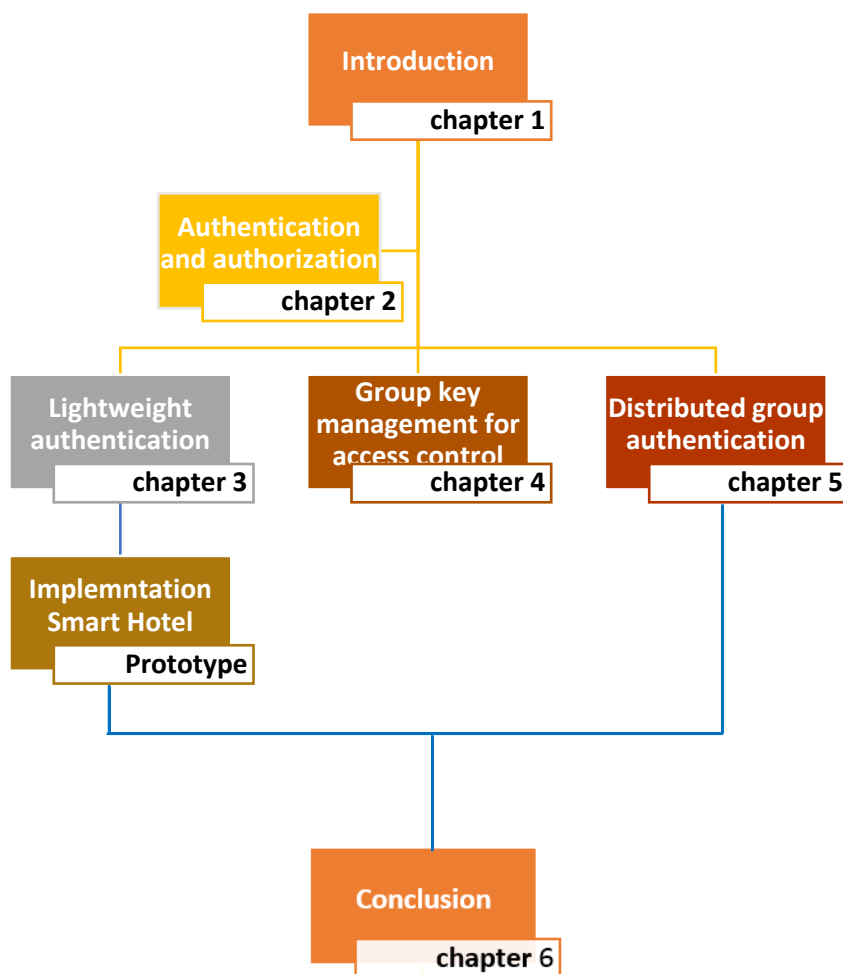


Figure.1. 1: Dissertation outline diagram

Authentication and Access Control in IoT Environment

2.1 Introduction

As we deliberated in the introduction chapter, the Internet of Things (IoT) has witnessed a tremendous evolution with the significant increase in the number of smart devices. These intelligent devices around us are increasingly becoming ubiquitous to enable new IoT applications in our daily life, including smart homes, smart grid, smart cities, intelligent transportation, smart healthcare, etc. [1]. Within this progress, designing IoT security solutions is a challenging task. Indeed, IoT devices and users need to be authenticated and authorized to access IoT services. Furthermore, the IoT, with its intrinsic characteristics, including heterogeneity, constrained resources, and large-scale network infrastructure, has given birth to different security requirements and challenges, such as scalability, interoperability, and dynamicity. All these security challenges can be considered as a significant barrier to the deployment of a secure IoT system. Therefore, to build a secure IoT system, we conducted a survey on IoT solutions over the past years, and the selection of the research works in the literature is based on the criteria mentioned below:

- i. The surveyed research should be designed for IoT environments, such as a wireless network, sensors network, or other connected IoT objects.
- ii. The surveyed IoT solutions should mainly study one of the following indispensable security features: the authentication and the authorization.
- iii. These studies should consider one of the advanced features of the IoT environment, such as scalability, heterogeneity, dynamic changes, group communication, and limited resources.

As previously mentioned, the IoT network aims to connect everything, including people, devices, organizations, applications, services data, etc., leading to a massive amount of extensive data that should be secured. Thus, any disclosure of the exchanged and transmitted data impacts the IoT system's security and functionality, which leads to many serious risks [8].

Since the security issue is primordial for the IoT environment, especially the authentication¹ and authorization² mechanisms, we classified these IoT solutions into two main categories to handle these security features while addressing the previous IoT unconventional characteristics, specifically scalability, heterogeneity, dynamic changes, group communication, and limited resources. In this context, we analyze the exiting authentication solutions in the literature that tried to adapt the security solutions proposed for wireless sensor networks (WSNs) to the context of IoT. Throughout our analysis of these selected solutions, we can highlight that some of the authentication approaches relying on a centralized architecture, make their implementation in IoT applications much more complicated because of the high number of IoT objects. Hence, they cannot handle the peculiar characteristics of a distributed IoT environment. Besides, we reviewed some distributed approaches that are considered beneficial to handle the scalability issues but causing an important overhead. At this level, we can observe that is essential to build a secure and efficient IoT system addressing the security requirements, explicitly, confidentiality, integrity, availability, and privacy. For that reason, we surveyed the authentication and the access control in the literature [9] [10] [51] [52] [98] [104], and it turns out challenges about how to manage authentication and the access control permissions for a large number of IoT devices delivering many IoT services. Furthermore, it points out the importance of eliminating the dependence on a connected third party to protect IoT devices and IoT users' security and privacy and ensure IoT availability over the various attacks.

Throughout this chapter, we first provide a comprehensive presentation of the IoT environment and highlight its deployment challenges. In particular, we specify the security issues and the requirements of unconventional IoT characteristics. Then, we investigate the recent studies and survey the authentication and authorization solutions. Finally, we point out specific approaches to build a secure IoT system.

2.2 Internet of Things (IoT)

The Internet of Things (IoT) is the future of the Internet, enabling a fully connected "smart" world to provide various services to Information Technology (IT). More specifically, the IoT concept is based on interconnecting "things" and devices that take the form of wearables, sensors, actuators, mobiles, computers, meters, or even vehicles, which communicate through the Internet, as shown in Figure.2.1. These inter-networked "things" interact and cooperate to achieve a common goal by sensing, transmitting, and processing valuable data [10], which define the emerging homes applications and the buildings automation, smart cities and infrastructure, smart industries, and smart-everything.

Besides, the IoT network is a dynamic system connecting digital devices based on interoperable communication and characterized by self-configuration capabilities such as identities, physical attributes, and virtual personalities. Therefore, the IoT paradigm transforms the physical objects from being conventional to smarter ones by exploiting communication technologies' advancement, which expands the communication from human-human to human-

¹ The authentication is responsible for verifying legitimate communicating parties in peer-to-peer networks.

² The authorizations framework prescribes rules to the users and IoT objects for interacting with each other and ensures the availability of the IoT system.

device or even device-device (D2D). This vision of the IoT has introduced a new dimension to information and communication technologies, where physical objects allow users' connection to the Internet from anywhere and anytime. However, these connected IoT objects' security plays a centric role and brings new challenges due to low memory, energy, and computation capacity. In the following, we first present the IoT architecture model and the IoT ecosystem's main layers. Then, we discuss IoT applications and the main challenges facing them.

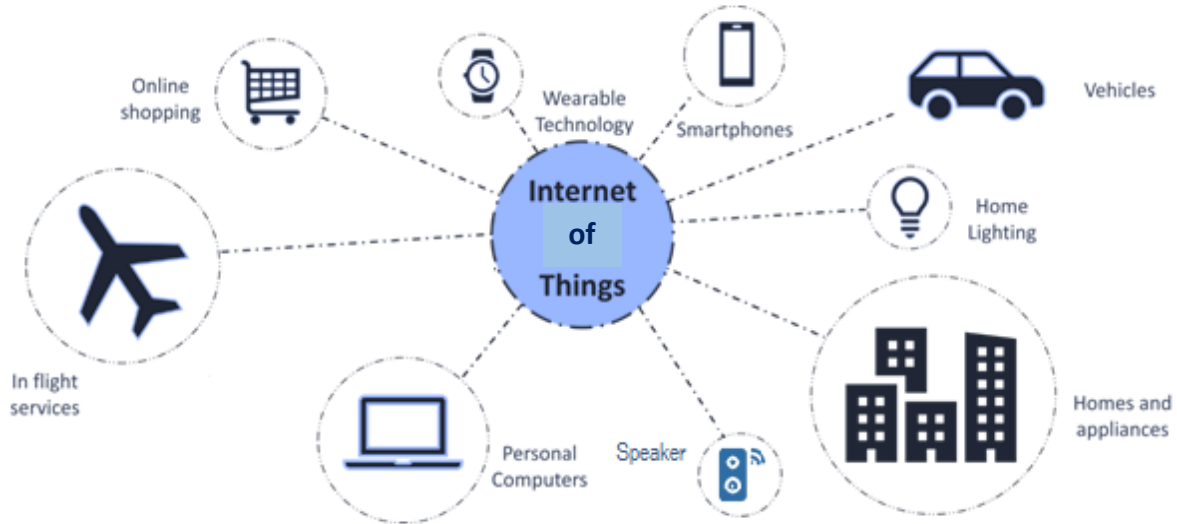


Figure.2. 1: Internet of Things

2.2.1 IoT Architecture

The high level of IoT applications explains the varieties of generic and horizontal IoT architectures proposed by many known groups and consortiums, such as M2M (Machine-to-Machine), ETSI (European Telecommunications Standards Institute), ATIS (Alliance for Telecommunications Industry Solutions), TIA (Telecommunications Industry Association). A typical IoT architecture is proposed by ETSI which is composed by three main layers, including the M2M domain layer, network layer, and application layer [9]. According to the recent literature [11], other models, including the five-layer model, have been proposed to improve the basic three-layers models and support the IoT's scalability. Consequently, the five-layer model is the most suitable model for IoT applications among the proposed models, as presented in Figure 2.2. In what follows, we briefly describe and define the different layers of IoT architecture.

2.2.1.1 Perception Layer

This layer is composed of the physical devices, such as sensors, actuators, intelligent terminals, and RFID systems required to implement the IoT environment. The features of this layer would be the sensing, actuating, and communication capabilities. In particular, it collects and gathers information about the IoT environment, such as querying location, temperature, patient health state, pressure, weight motion, vibration, acceleration, humidity, etc. These

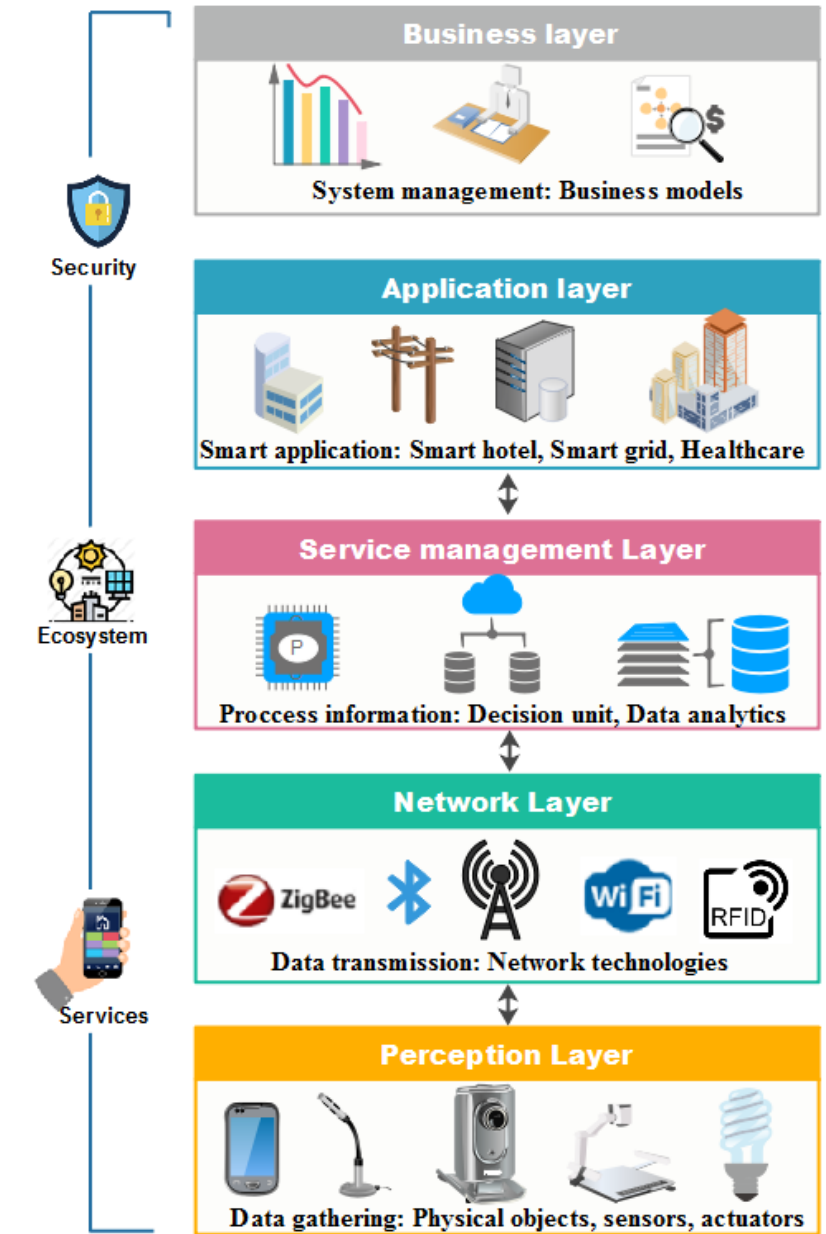


Figure.2. 2: IoT 5-layers architecture [11]

collected data are useful for performing various functionalities, such as identification and information storage, information processing using embedded edge processors, communications, control, and actuation. More specifically, these components composing the perception layer can be divided into two categories, named perception nodes and perception network [12]. The perception nodes, including the sensors, controllers, perform data acquisition and control, while the perception network, defining the communication interface of the perception nodes, transmits the collected data to the gateway. Thanks to the huge amounts of data created at this layer, the perception layer is the entry point of what we commonly call Big Data.

2.2.1.2 Network Layer

The network layer is responsible for transmitting the sensed data of the perception layer to the service management layer. Indeed, the sensed data is transmitted through various networks

such as wireless, 3G, LTE, LAN, LoRaWAN, 5G, Bluetooth RFID, and NFC. In particular, this layer includes various devices, such as switching, internet gateways, and cloud computing servers that can perform local analysis and routing messages to the service management layer. This layer is introduced as an intermediate layer to manage the tremendous number of objects through aggregating, filtering, and transmitting data and support sensitive IoT applications by adopting several communication technologies [11].

2.2.1.3 Service Management Layer

This layer comprises M2M platforms, middleware, API of M2M applications, and cloud computing technologies useful for managing the perception layer's data. In particular, the service management layer has features of information storage, analytics, and processing of the data to enable the IoT application developers to deliver high-level applications independent of any physical platform. Besides, this layer's features allow handling the received data by the vendors to provide various kinds of IoT services.

2.2.1.4 Application & Business layers

The application layer's main feature is to provide specific services based on the application type to the user through application protocols, such as HTTP, MQTT, CoAP, etc. At this layer, all required software is installed to evaluate, analyze the received data, and then afford high-quality services that meet final customers' requirements. These designed applications also answer many markets' needs in different fields such as smart building, transportation, industry, smart grids, and healthcare [13][14][15]. Furthermore, these applications should satisfy a good quality of service and ensure an adequate reliability level to final users. At this level, a business layer is defined on the top of the application layer to manage the entire IoT system, especially the business and profit models, in a user-friendly way with privacy. More specifically, this business layer is responsible for complex data processing, such as restructuring, cleaning, and combining to develop more effective business models, predict customer behaviors, and show high-level metrics, graphs, and flowcharts. This processing data process may be in the context of performing big data analytics to transform data and information into actions to support decision-making processes.

2.2.2 IoT Applications

The IoT revolution has emerged with a remarkable potential to cover a wide range of applications in various domains. These domains deal with almost every area of our daily lives, such as smart homes, smart buildings, intelligent transportation, smart healthcare, smart grid, and other industrial applications. More specifically, the IoT paradigm combines some features (sensing, communication, networking, identification, and computing) to provide ubiquitous IoT services for users anytime and anywhere. In this context, the latest 2020 economic analysis of

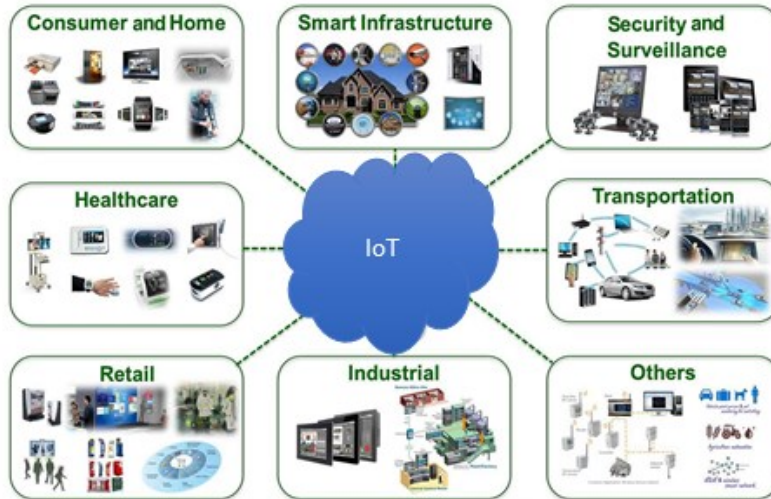


Figure.2. 3: IoT main applications

IoT-based services have recorded considerable growth [16]. Figure 2.3 shows the various IoT applications where some of them are briefly describe in the following subsections.

2.2.2.1 Smart Home

The smart home is one of the most known IoT applications as it is considered as a promising solution to enhance personal lifestyle. The smart home deploys various sensors and actuators to control / monitor home appliances remotely (e.g., microwave, lights, heating, ventilation, and air-conditioning – HVAC systems) and perform security surveillance. Moreover, it enables owners to configure time schedules to control costs and be more energy-efficient (e.g., green homes).

2.2.2.2 Smart Grid

One of the most attractive IoT applications that has a considerable industrial value is the Smart Grid. In particular, this technology plays an essential role in economic development as in modern cities we use IT technologies to optimize electricity production and improve the energy consumption of houses and buildings. This technology is a data communications network integrated with the power grid, called the advanced metering infrastructure (AMI), installed between the electricity production centers and the end customers to collect, analyze, monitor, and coordinate energy production and consumption customers' needs. The smart grid's primary goal is to improve final customers' quality of experience, increase efficiency, and optimize electricity production. To better understand in detail how IoT can improve electricity production in smart grids, the reader is referred to [17][18].

2.2.2.3 Transportation Systems

The future generation of transportation is mainly presented by the intelligent transportation system (ITS). In fact, due to the development of the embedded systems and communication technologies, this system aims to link people, roads, and intelligent vehicles & infrastructures. This intelligent system employs four main components, namely: the vehicle on-board unit (OBU), the station subsystem that represents the roadside unit (RSU), the ITS monitoring

center, and the security subsystem [19]. Connected vehicles use four types of communications: Vehicle to Vehicle (V2V), Vehicle to Infrastructure (V2I), Vehicle to Cloud (V2C) and Vehicle to Pedestrian (V2P). A new type of communication has recently emerged, called Vehicle to Grid (V2G), which has a primary goal to ensure electrical Vehicles charging based on the smart grid's energy electricity distribution [17].

2.2.2.4 Healthcare

Smart Healthcare has emerged as a prominent IoT application due to the technological advancement in biomedical sensing, signal processing, and wireless communication. In fact, healthcare IoT applications are based on embedding sensors and actuators in patients' bodies to monitor their physiological statuses. In particular, IoT-based healthcare equipped with the embedded sensors aims to collect information directly from the patient's body area, analyze and transmit information to healthcare providers. This latter guarantees real-time monitoring of the patient state and make the right decision at the right time. Healthcare-based IoT applications currently have gained significant interest as they hugely impact society mainly due to the aging population and the cost related to medical treatment. In this context, adopting new IoT based technologies to monitor the patients in real-time is indispensable [21] [125].

2.2.2.5 Smart Cities

Smart cities are considered one of the emerging paradigm applications in IoT. Indeed, the smart city aims to enhance public resource usage, improve information sharing and coordination, and increase service quality to citizens [22]. In this perspective, a smart city environment is composed of smart devices deployed all over the roads, buildings, smart cars, etc., which can better manage the traffic, adapt to the weather, lighting follows the sun's position. Furthermore, it can avoid domestic incidents with alarms and thus enhance the comfort and security of citizens.

2.2.2.6 Manufacturing and Industrial IoT

Automation in manufacturing using IoT has emerged as a prominent role in the industry. In fact, it is considered a promising solution to enhance productivity and efficiently monitor and control the production chain. The Industrial IoT (IIoT) deploys new technologies such as Machine-to-Machine (M2M) communication, Wireless Sensor Networks (WSN), automation technologies as well as Big Data to produce an intelligent industrial ecosystem [23]. More specifically, it ensures an accurate, fast, and reliable production process based on four elements [24]: transportation, processing, sensing, and communication to provide better control of final products.

2.2.3 IoT Challenges

The IoT has appeared as a significant industry that provides many new opportunities and benefits to end-users and manufacturing. Indeed, it accentuates a considerable positive impact while enabling various applications in our daily life. However, these benefits address several complicated challenges and issues, including availability, reliability, mobility, network

performance, security & privacy, energy, consumption, and management. In fact, recent contributions demonstrate that the increasing number of connected objects causes high traffic demands enabling new traffic models. Therefore, it is essential to deal with these issues, leading to various practical and efficient IoT services [25]. We enumerate in the following the main challenges that IoT faces:

❖ **Scalability:**

Scalability is mainly about a system's ability to ensure flexibility that achieves and responds to the growth required works. Its principal aim is to enable adaptability to the changing environment and technology, leading to seamless connectivity and supporting dynamic topology changes. Therefore, the scalability's fundamental challenge is to support a massive number of heterogeneous connected objects using various hardware platforms and communications protocols and meet people's needs. Scalability is considered as an absolute necessity to provide a good functioning of the IoT environment and save the available resources [26]. Two different scalability types are defined in the context of IoT, namely vertical scalability and horizontal scalability. The vertical scalability is referred to as the ability to increase resources in terms of hardware or software by adding more processing memory and storage capability. The horizontal one is achieved by increasing the capacity by connecting multiple hardware and software to work together. To enhance the IoT applications' scalability, highly scalable cloud-based platforms, called Cloud of things [27], have been introduced as an effective architecture. Some other solutions based on fog/edge computing are used to extend cloud services, be closer to the connected objects, and improve the computing network capability. Furthermore, 5G, the new radio system, is being envisaged for massive IoT (mMTC - massive Machine Type Communication) applications that will allow the connection of very densely distributed objects, necessary for the exponential increase in the number of connected objects.

❖ **Limitation of resources and energy consumption:**

Most of the IoT devices are characterized by a limited capacity of storage and computation. Consequently, it is a critical challenge to integrate the embedded devices with the required computation process. The authors in [28] studied the challenge of improving the devices' capabilities (e.g., computation and communication) with low-cost terminal and low power consumption. However, it is mandatory to design lightweight protocols to meet the resource limitation and the customers' requirements

❖ **Reliability:**

Reliability is a critical issue in the IoT environment, especially in emergency scenarios where an appropriate time response should be provided, such as critical applications like manufacturing, transportation, and healthcare applications [11]. Indeed, as it refers to the system functionality, unreliable perception, data collection, transmission, and processing may cause long delays, loss of data, and eventually wrong decisions. Therefore, this may cause huge damages or life-threatening conditions. It is essential to design reliable systems transversely to

all the IoT architecture layers that work correctly under any circumstances and then build an efficient IoT system.

❖ **Availability:**

The availability is about maintaining the availability of services' IoT systems over time and delivering the requested services for authorized connected objects anywhere and anytime. Indeed, it is as critical as information protection to properly handle the IoT systems. Therefore, the connected objects need to be compatible with the IoT system requirements to maintain the availability and connectivity. Moreover, the communication channels of the IoT network could be vulnerable to availability issues. Thus, the IoT system should also guarantee services' continuity even in availability threats, topology changes, and consumers' mobility [29].

❖ **Management and Self-configuration:**

One of the biggest and challenging IoT tasks is managing the Fault, Configuration, Accounting, Performance, and Security aspects (FCAPS) of the complex and heterogeneous interconnected IoT environment. In this context, to provide adequate IoT services, it is mandatory to design real-time, lightweight, and secure management protocols. Specifically, the data management mechanisms should ensure several functionalities, including data aggregation, data analytics, and security aspects that meet the system requirements. Besides, the large-scale network infrastructure of connected devices must also be managed by monitoring the high traffic load and the quality-of-service requirements. Consequently, this type of management could handle the IoT environment's dynamic nature and the network elements [30].

❖ **Mobility:**

Mobility is a critical challenge in the IoT environment, where IoT services' consumers are mobile. Meanwhile, the challenge is about connecting users continuously with the requested IoT services with respecting their requirements. In this context, some existing works in the literature [31][32] managed to solve this issue by implementing efficient mobility management mechanisms to guarantee service continuity.

❖ **Interoperability:**

Interoperability is the capability of many heterogeneous systems, platforms, and devices to communicate and intercorporate together. Specifically, the IoT ecosystem comprises heterogeneous devices with different standards and technologies, which is the origin of the interoperability problem. For instance, there is still no standard for the IoT network that supports the interconnection of all heterogeneous IoT systems [11]. Therefore, interoperability in IoT systems should be achieved over the varied connected objects and the communication protocols such as IPv6, IPv4, IPv6 Routing Protocol (RPL), Constrained Application Protocol (CoAP), ZigBee, WiFi, Bluetooth, RFID, etc. Besides, there is an absolute requirement to support the heterogeneity aspects to build IoT applications and services that can be extended and integrated with other IoT systems easily [33]. In fact, the PARFAIT project [7] that defines

the context of this thesis, is designed to ensure the IoT interoperability through integrating different communication protocol including NFC, Bluetooth, and ZigBee.

❖ Security and privacy:

Security and privacy are the most critical challenges in the IoT environment. As communication in such environments is ensured through wireless channels, IoT architecture is vulnerable to various security risks, such as eavesdropping, unauthorized access, data modification, and privacy issues. Therefore, the design of adequate security countermeasures is necessary to secure the IoT network and ensure ubiquitous connectivity. Significantly, the existing cryptography algorithms and protocols are claimed unsuitable for constrained IoT devices [18] [21]. Additionally, IoT applications are characterized by their distributed nature and large-scale connected devices that impose more security and privacy challenges [34][35]. More other security and privacy challenges are related to definite IoT applications, including transportation systems, industrial automation systems, smart cities, and healthcare systems [36]. At this level, the challenges are carried out independently and prudently to meet each application's requirements.

In Table 2.1, we highlight a summary of the leading security challenges related to some IoT applications. Specifically, we present the severity of each issue in the different IoT applications. In the next section, we detail security issues in the IoT environment.

Table 2. 1: Main security issues vs. Applications

Applications vs. Challenges	Smart Home	Intelligent Transportation	Industrial Automation	Smart Healthcare	Smart Grid	Smart City
Resource constraints	High	Not applicable	Low	High	Low	Medium
Mobility	Low	High	Low	Medium	Not applicable	High
Scalability	High	High	Medium	Medium	High	High
Availability	High	High	High	High	High	High
Interoperability	Medium	Medium	Medium	Medium	Medium	High
Management and configuration	Medium	Medium	Medium	Low	Medium	High

2.3 Security in the Internet of Things

Regarding the high presence of the IoT in the industry and our daily lives and the previously mentioned IoT challenges, we must sort out the security requirement to design a secure IoT system. In particular, securing IoT systems is based on several fundamental and specific

security requirements, from the CIA of information security (confidentiality, integrity, and availability), to the five pillars of information assurance (confidentiality, integrity, availability, authenticity, and non-repudiation). Many researchers have discussed the security considerations related to IoT systems. For instance, authors in [37] figure out the IoT challenges like scalability, heterogeneity, and mobility and review the importance of the security and privacy considerations, including CIA and trust. Besides, other authors [38] classified the required security services for the different IoT applications regarding their importance, such as in the smart grid, availability is the most critical service, while for healthcare, authentication is a more serious service. Consequently, the IoT paradigm imposes many concerns over data security due to economic espionage, infection of sensitive computer systems, identity theft, etc. At this level, secure IoT infrastructures should provide reusable security services such as confidentiality, integrity, authentication, authorization, availability, and privacy. In the following, we describe the properties of the mentioned security services:

<p style="text-align: center;">Confidentiality</p> <p>It prevents unauthorized persons, entities, or processes from retrieving sensitive data [39]. For that reason, confidentiality should be addressed on two crucial security mechanisms, including the authentication and the authorization processes. Furthermore, confidentiality should also be operated through the different layers of an IoT architecture [39]. Particularly, it considers protecting data in IoT devices and in IoT applications concerning specific users from disclosure and tampering. Furthermore, data stored at the third-party service providers require confidentiality service that avoids malicious attackers to steal sensitive information. Otherwise, these centralized service providers are straightforward to many intruders.</p>	<p style="text-align: center;">Integrity</p> <p>It is mainly about ensuring the truth, honesty, and reliability of the data. Indeed, it is the assurance that the transferred data is not modified by a third-party accidentally or intentionally [40]. Therefore, as the number of connected devices and IoT consumers are becoming very high, providing reusable security services, such as integrity, becomes a core issue regarding IoT security. In fact, integrity in IoT devices guarantees that these devices are trusted and not hijacked by malicious attackers [41]. Besides, ensuring integrity through the network layer evades signaling data and then avoids denial of service attacks [42]. Likewise, at the application layer, integrity concerns the users' data protection.</p>
<p style="text-align: center;">Non-repudiation</p> <p>Non-repudiation guarantees the sender of the message in IoT systems. Therefore, the sender cannot deny being the author of a transmitted message [41]. The non-repudiation aims to protect against false denial of involvement in a communication. Attackers can manipulate an IoT system by forging the identifying credentials that threaten the origin of service data providers and the user data. Hence, a non-repudiation service is an effective security service that should be implemented and built on IoT to provide genuine high confidence in the transmitted data.</p>	<p style="text-align: center;">Availability</p> <p>It implicates that all IoT services and devices of the IoT system are accessible for authentic users and resistant to several malicious attacks. Indeed, availability highlights the IoT security systems at runtime, where systems can deliver services to others. Otherwise, the availability of services has no meaning. Due to the highly distributed nature of the IoT environment, availability could ensure the interconnectivity and accessibility of IoT systems' services. In contrast, systems with low availability could incur many security concerns such as attacks on reliability. In fact, malicious intruders can control IoT systems by gaining control of IoT systems, such as capture attacks and impersonation attacks [43]. In particular, maintaining the availability in the devices layer aims to prevent physical attacks and DoS attacks.</p>

	Besides, availability guarantees the accessibility of the networks, services, and applications.
<p style="text-align: center;">Privacy</p> <p>It concerns principally the users and particularly the application layer. Indeed, a privacy mechanism gives users the ability to control their personal data and determine the amount of information to reveal to others [42]. Moreover, it ensures the non-traceability of the user's behaviors and performed actions in the system. Therefore, privacy, defined as individuals, groups, and institutions' rights, is considered a severe security issue. For example, the RFID tag tracking attack and the eavesdropping attack are all about the individuals. At this level, intruders may misuse the hidden RFID to retrieve sensitive data like credit card information [25]. It is evident that privacy-preserving goals to protect users' sensitive information like identity, location, mobility trace, etc., [38, 17].</p>	

At this level, we confirm that designing authentication and authorization mechanisms are indispensable to meet the cited security services requirements. Indeed, **authentication** is the process of verifying the genuine and originality of the sender and validating whether a given identity fits the pretended IoT entity [8]. In particular, authentication is necessary to prevent illegal access and tampering related to IoT devices, while at the network layer, authentication is about protecting signaling data to avoid the DoS attacks. Similarly, the authentication operates over the service layer to provide the key management and access control policies. Finally, authentication identifies, authenticates, and authorizes users in the IoT environment at the application layer. Furthermore, **authorization** is about granting the required access permissions to the authenticated user identity [9]. The authorization is achieved after the successful authentication of the trusted identity user. At this level, the IoT system can give the user the corresponding right to get data or service from the IoT environment. Consequently, establishing efficient authentication and identity mechanisms and protocols are needed for authorization protocols. These protocols operate over the service and application layers and are imposed at the device layer.

To design the previously mentioned security services and respond the limited resources IoT requirement, several cryptographic mechanisms are used through the literature [38] and Table 2.2 shows some of these mechanisms.

Table 2. 2:Security services

SECURITY SERVICES	SECURITY MECHANISMS	CRYPTOGRAPHY EXAMPLES
CONFIDENTIALITY	Message encryption / message signature	❖ Symmetric cryptographic mechanisms (AES, CBC, etc) ; asymmetric mechanisms (RSA, DSA, IBE, ABE, etc).
INTEGRITY	Hash functions, message signature	❖ Hash functions (SHA, MD5, etc) ; Message Authentication Codes (HMAC)

AUTHENTICITY	Chain of hash, Message Authentication Code	❖ HMAC, CBC-MAC, ECDSA
NON-REPUDIATION	Message signature	❖ ECDSA, HMAC
AVAILABILITY	Pseudo-random frequency hopping, Access control, Intrusion prevention systems, firewalls	❖ Signature-Based Intrusion Detection, Statistical anomaly-based intrusion detection
PRIVACY	Pseudo-anonymity, unlinkability, k-anonymity, Zero Knowledge Proof (ZKP)	❖ EPID, Pedersen Commitment

In the following subsection, we firstly explore the security challenges in IoT and point out the critical security requirements. Then, we survey the existing authentication and authorization solutions in the context of IoT.

2.3.1 IoT Security Challenges and Security High Requirements

In addition to the previously mentioned security services in the IoT environment, we still have to sort other security requirements to build a secure IoT infrastructure. In fact, as the number of connected objects in the IoT is continuously increasing, the current state-of-the-art network security solutions cannot address some security IoT challenges [8]. The challenges of IoT environment presented previously, lead to many critical security challenges in such environment. We highlight in what follows these security challenges:

- ❖ **Heterogeneity:** It refers to the diversity in security requirements and resource availability. In particular, the potential number of communication standards and information system technologies having heterogeneous security configurations requires service management, which will impact IoT systems' security. Consequently, due to the IoT network components heterogeneity, the reuse of the current network protocols is inadequate in the IoT environment, and it is necessary to provide security standards that work with different IoT platforms and protocols [45][38].
- ❖ **Support for scalability issues:** The distributed nature of the IoT environment imposes scalability security challenges for the IoT network. In fact, the large scale of connected IoT devices and users define a high volume of communication traffic, including one-to-many traffic patterns such as broadcasting or publish-subscribe protocols such as MQTT [45] [49]. Hence, the management of the security of the presented device and traffic communication introduces several challenges. For instance, applying efficient updates and security patches over the distributed environment characterized by high heterogeneity is challenging. This environment with great scalability should build effective key management protocols to secure the communication through the wireless network [46]. Otherwise, attackers can exploit IoT devices' interconnection to disclose private information and reveal criminal activities such as a man-in-the-middle attack, etc.

- ❖ **Vulnerability related to communication systems:** It refers to the increasing risks caused by the communication in the IoT environment where adversaries have physical and wireless access to IoT devices. Indeed, the remote access mechanisms and the sensitive exchanged data over the wireless channel raise the probability of attacks. Thus, IoT systems are susceptible to some of these attacks considering IP spoofing, injection, DoS/DDoS in which an adversary may exploit network protocols with massive traffic [47]. Further, the attacker may violate the communication based on traffic analysis, eavesdropping, and passive monitoring, implying an efficient security requirement for M2M communication.
- ❖ **Dynamic changes in IoT systems and environments:** The characteristic of dynamic changes in the IoT environment is mainly a fundamental property of the IoT. The dynamic changes are particularly related to the IoT devices' behavior over time, such as started and standby, sleeping and waking up, leaving and joining networks [48]. However, since the number of connected devices is continuously increasing, it causes a very dynamic IoT environment due to the continually changing status and thus emerging of many threats. For instance, a publish-subscribe based IoT system characterized with a high changing network topology and unbounded network size is an attractive area for many threats and attacks [40]. Consequently, a secure IoT infrastructure needs to resist to these dynamic changing environments and afford effective security services (authentication and authorization).
- ❖ **Frequent authorization and authentication:** Regarding the dynamic changes in the IoT environment resulting from the changing states, such as connected / disconnected and the context of devices including speed and location, specific authentication and authorization mechanisms are needed. Furthermore, dynamically varying situations may also include the IoT users resulting from changing interest over time, which may change their authorization access to the IoT devices. Therefore, frequent demands of authorization require a continuous management for access control, which can avoid and limit the dynamic feature of IoT environment.
- ❖ **Automated distributed mutual authentication:** Due to the IoT network's scalability approaches, including the publish-subscribe protocols [49], it is excessive for users to remember passwords for a large number of devices. Thus, the IoT devices must be able to authenticate themselves without user intervention to keep the practical functionality of the IoT system.
- ❖ **Dynamic registration of IoT entities:** Distinct from traditional Internet communication, the IoT includes devices with shorter life cycles and users with various interest over time (subscription to IoT services for a shorter time). Meanwhile, devices may be added and removed, and users may subscribe and unsubscribe from authentication/authorization systems dynamically. Therefore, managing the adding and removing entities in IoT should be operated strongly to build a secure IoT architecture.
- ❖ **Consideration for resource constraints:** Designing robust security measures requires developing strong cryptographic protocols, which are based on many computing operations. Nevertheless, some IoT devices suffer from the limited resources of computation and storage, and thus an excessive energy consumption can harm their

availability. At this level, the authentication and authorization security services should be lightweight to save energy consumption with respect to the security requirements.

- ❖ **Locality:** authentication and authorization services should be maintained and not impacted by the internet connections and remote server operations. For that, the IoT security measure must be enhanced with edge services. Designing the previously mentioned security services in the edge network of the IoT environment, could guarantee an improved authentication and authorization services for IoT users.

To sum up, the IoT environment with high scalability, heterogeneity, and dynamic changes constitute new security challenges and requirements. Adding to essential security services the confidentiality, integrity, availability, non-repudiation, and privacy it is necessary to address the security challenges related to a large-scale distributed IoT environment. Table 2.3 presents a classification of previous research works that cope with the different challenges:

Table 2. 3:Security IoT challenges

IoT security challenges	Research area
Heterogeneity	IoT platforms and architectures [73] [98] [99]
	Device management [68] [69]
	Network management [66] [96]
	Data management [94] [95]
Scalability	Large-scale issues [71] [72] [97]
	Low power communications [63] [64] [65]
	Availability and reliability of IoT applications and services [56] [57] [77] [81] [85]–
	Continuous connectivity [78] [79] [80]
	Infrastructure reliability [82] [83] [84]–
Vulnerability related to communication systems	QoS and QoE evaluation [80-82]–
	Traffic models and loads [61] [62]
	Application layer protocols [89] [90]–
	Network layer protocols [66] [86-72]–
	Link layer protocols [85-71] [87-73]
	Security issues [97-99]–
	Privacy issues [94-96]

Dynamic changes	–Mobility management of smart devices [58] [59] [60]
------------------------	---

2.3.2 Authentication in IoT

As discussed previously, authentication is the fundamental security in the IoT environment. Furthermore, authentication should be operated over the different IoT architecture layers. Therefore, various authentication techniques in the literature concerning IoT applications have been proposed. We summarized, as shown in Figure.2.4, these techniques in a taxonomy of IoT authentication schemes categorized into several criteria selected in the literature [44] [45].

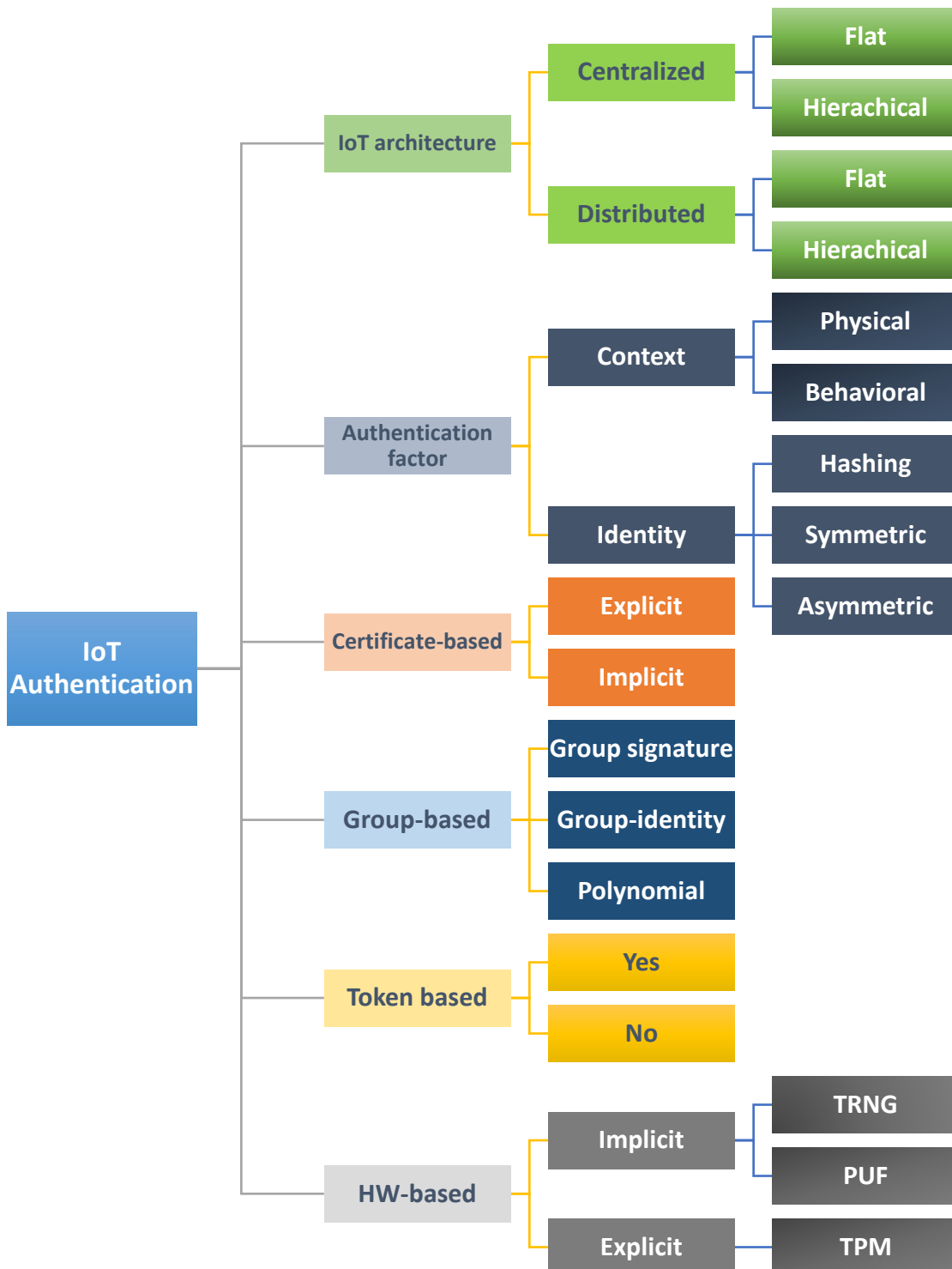


Figure.2. 4: Taxonomy of IoT authentication schemes

Related to the discussed metrics of IoT characteristics and the previously mentioned security requirements (section 2.2), we mainly evaluate in the following the authentication in IoT according to the two authentication architectures: (i) centralized and (ii) distributed. We review IoT systems for each architecture in compliance with classified categories presented in the taxonomy.

2.3.2.1 Centralized IoT Authentication Architecture

The centralized authentication architecture is based on a centralized server or a trusted third party to manage and disseminate the credentials useful for the authentication process. The centralized architecture can also be hierarchical, where it uses multi-level architecture to handle the authentication process or flat without using the hierarchical feature to deal with the authentication procedure. Authentication is basically based on verifying legitimate users (identities) and IoT devices. Therefore, when an IoT system wants to check the communicating component's identity ID, it first must trust the issuer of this identity, like a national government. In incoming subsections, we review and discuss research works based on a centralized authority, which are classified as follows: (i) Multi-factor authentication (MFA), (ii) Certificate-based authentication (CBA), (iii) Token-based authentication, (iv) Group-based authentication, and (v) Hardware authentication.

❖ Multi-factor authentication

Multi-factor authentication is mainly achieved through two attributes: identity and context. The **identity** defines one party to be authenticated with another party of communication [50]. Furthermore, the identity-based authentication schemes are designed with one or a combination of hash, symmetric, or asymmetric cryptographic algorithms. Otherwise, the **context** can be physical, which is defined by the biometric information based on physical characteristics of an individual, e.g., fingerprints, hand geometry, retinal scans, etc. Besides, the context can also be behavioral, which is explained by an individual's biometric behavioral features, e.g., keystroke dynamics (the model and time of the person's rhythm during typing), gait analysis (the process used to measure the way we walk or run), voice ID (voice-print of the voice authentication), etc., [52].

The multi-factor technique is widely used in the literature. Authors in [53] proposed a two-factor authentication scheme to authenticate the user. The two-factor used are the smartphone with Near Field Communication (NFC) feature and fingerprint of the user. Their scheme uses a database library to verify the embedded personal data in the NFC tag with the fingerprint, and then authenticate the user and give him/her access to the internal library network. In [54], the authors provided a two-factor authentication protocol called E-SAP (Efficient-Strong Authentication Protocol) for hierarchical wireless sensor networks for healthcare applications. The proposed protocol, using smart card and password as two-factor, involves hash and XOR operations to ensure lightweight feature and make it highly suitable for resource-constrained devices. Besides, it guarantees mutual authentication between sensors, confidentiality, the ability to change passwords, and resilience against several attacks using symmetric cryptography. Furthermore, authors in [55] combine a unique contextual attribute fingerprint, a physical biometric, to succeed in IoT object authentication. This scheme guarantees that each

object is identified through a unique fingerprint referring to numerous characteristics, including location, physical state, or transmitter state. In fact, authors have based on the transfer learning technique to authenticate devices, which is useful to separate the regular changes related to the environmental effects from the malicious changes caused by intruders. The proposed authentication methodology accomplishes improved performance results than conventional authentication techniques. Besides, authors in [56] discussed a behavioral-based authentication mechanism to authenticate the user. Their approach is based on the network traffic patterns generated during the user access to the IoT application, using a small amount of information extracted from end-user devices, such as smartphones. This scheme ensures a high degree of accuracy concerning the security requirement. Moreover, the authors of [57] used a new factor called the device capability to present a two-factor device authentication scheme. This factor is a mathematical challenge or even a cryptographic-based puzzle solved by the device. Combining this factor with a digital signature helps achieve mutual authentication between an IoT device and the server. Indeed, the device sends a request to communicate with the server, this latter responds with a nonce encrypted with its private key and the timestamp to avoid replay attacks. At this level, once the device receives the responses, it solves the nonce with the functional operation to finalize the mutual authentication with the server.

To achieve stronger authentication with a remote user, using a password and another factor is still vulnerable to security attacks. Therefore, in [58], the authors proposed a novel authentication protocol using three-factor: the user smart card, personal biometrics, and a password to authenticate a remote user as authentication attributes. The scheme ensures various security properties like mutual authentication and sensing node anonymity. However, it costs in terms of computation and communication overhead as it uses the fuzzy fingerprint function. Therefore, authors in [59] designed a lightweight remote user authentication for IoT communication using elliptic curve cryptography. The scheme is a three-factor remote user authentication based on ECC that ensures mutual authentication between the user and the gateway and between the gateway and the sensor node. The scheme's performance analysis proves its efficiency and effectiveness regarding the communication and computation overhead as it applies only cryptographic hash functions along with the symmetric encryption/decryption. Also, it considers several security issues such as data confidentiality, integrity, and availability.

❖ **Certificate-based authentication**

A certificate-based authentication technique involves a third trusted party, known as the Certificate Authority (CA), responsible for registering and generating certificates to the various entities. Also, a Registration Authority (RA) is defined to ensure the validity and the correctness of the registration process. The digital certificates issued by the CA are verified and signed by the CA. Thus, every entity in the network can verify the certificate through the CA's signature. This certificate has mainly three elements, including the identification data, a public key, and a digital signature to identify the user. Besides, the certificate's use might be implicitly or explicitly, where explicit certificates are managed and signed by a trusted third party (a CA), while the implicit is a variant of a public key certificate [60].

The most widely used certificate standard is X.509 that identifies the format of the public key certificates. In [61], the authors used the explicit X.509 certificate in the IoT by combining

the identity and certificate to reduce the storage overhead introduced when using multiple certificates. In addition, an improved use of the explicit certificate X.509 for the IoT was designed in [62]. Indeed, this scheme eliminates the fields not used and required from the certificate and compresses the useful fields, which improves the system's efficiency. The exploitation of implicit certificates in the IoT was presented in [63] and [64]. The authors propose a two-way authentication protocol in these works, namely the certificate registration and the authentication, using the Datagram Transport Layer Security (DTLS). However, it is noticed that their schemes are not compliant with the DTLS standard. Hence, the authors solved this problem in [65] by proposing the PAuthKey (Pervasive Authentication Protocol and Key establishment) protocol, which relies on the link layer's security IEEE 802.15.4.

❖ **Token-based authentication**

In contrary to the certification-based authentication that uses a set of asymmetric keys, the token-based authentication are essentially based on symmetric keys. A token-based authentication concerns the creation of a piece of data by a server called an identification token. This identification token is used to authenticate the user or the IoT device in the IoT. The widely known servers responsible for creating the token are OAuth2 [66] [67] or open ID [68]. Otherwise, a non-token-based authentication implicates the regular use of credentials, such as username and password, when there is a need to exchange data.

The Kerberos authentication system [69], a widely used approach based on a centralized trusted third party, uses temporary tokens called tickets to authenticate users and servers to get access from services. The authors of [70] implemented a prototype on an Android smartphone and an MSP430 based MCU of an authentication token. This token permits a fast authentication procedure without the need for additional user action. The authors of [71] suggest an authentication framework for the IoT that exploits the security model of OAuth 1.0a. Their scheme ensures the self-securing tokens that provide an independent security stack from all the network using signatures on the token. This work uses the basic functionalities of Public Key Infrastructure PKI to enhance the trust between the devices. Therefore, simplifying the exchange of tokens and enhancing the level of security for IoT devices.

❖ **Hardware-based authentication**

The design of authentication protocol might require using the physical characteristics of the hardware or the hardware itself. We find two types: implicit hardware-based, built with the physical characteristics embedded on the hardware to enhance the security, such as Physical Unclonable Function (PUF) or True Random Number Generator (TRNG), and explicit hardware-based, established by the use of a Trusted Platform Module (TPM) [52]. The current trend of hardware security is using the PUF regarding its advantages over software security approaches. Indeed, a combination of software solutions (lower cost) and hardware solutions (more secure) should be considered. The authors of [72] designed a new hardware-based authentication approach, using a hardware fingerprint to authenticate IoT devices with their Physical Unclonable Functions (PUF). In fact, the PUF exploits the random physical factors to create a unique identifier for each IoT object. The authentication is achieved by applying machine learning-based to avoid modeling-based attacks on PUF and hence developing a

software model on the PUF. In [73], authors implement PUF-based algorithms for IoT devices using elliptic curves for enrollment, authentication, decryption, and digital signature. Otherwise, due to the variations of the physical IoT environment, it affects the usage of the PUF when using error correction codes. Thus, the authors combined the PUF with ECC to encrypt generated key and handle the machine-learning attacks. In [74], the authors proposed a lightweight authentication protocol for RFID tags based on PUF. The protocol is achieved through three transactions: tag recognition, verification, and update. The first transaction consists of recognizing the tag reader, while the second is the mutual authenticity verification between the tag and the reader. The third transaction concerns the update process, where each one should keep the last recent used key. To provide anonymous authentication for RFID systems, the authors of [75] presented a PUF-based authentication scheme for classic RFID tags. Indeed, the scheme suggests an improved authentication protocol for a noisy PUF environment. However, this scheme does not consider updating the server with the new Challenge-Response Pair (CRP) once the exiting pool becomes empty, which vulnerable the system.

❖ **Group-based authentication**

Due to the large number of IoT devices requiring access to the network, the authenticating server is overloaded. Therefore, it is essential to design a new authentication type to enhance the IoT system's effectiveness concerning the security requirement. For that reason, a group-based authentication has been introduced to meet the system requirement and the IoT devices' requirements. According to that, authors in [76] proposed a group based lightweight authentication and key agreement scheme called GLARM to attain mutual authentication and secure key agreement for resource-constrained devices. This scheme consists of two essential phases: the identification phase and the second one, a group authentication and key-agreement phase. This work uses a combination of message authentication code of a group of devices to achieve the authentication of a group of devices. The performance results also prove this scheme's efficiency in terms of the system's communication and response time. In [77] [78], the authors provided a threshold authentication protocol to support secure and privacy-preserving communications in VANETs. This work uses a group signature scheme that accomplishes the threshold authentication, anonymity, and traceability during vehicles' communication. Furthermore, to allow remote users access to the internet services, authors in [79] introduced a new technique to afford secure roaming for anonymous users through the group signature method. This scheme ensures mutual authentication and privacy-preserving features. Indeed, it also offers devices to move between the access points without the need to re-authenticate. This fact is achieved by transmitting the roaming members' information to the Base Station (BS) after a first authentication. A group manager is responsible for collecting and aggregating all group members' information to send them back to the BS. This work removes the complication of certificate management in signature cryptography. Moreover, a multicasting key establishment scheme was provided in [80] to enable sensor nodes to join a multicasting group. This scheme uses an ECC secret key and Elliptic Curve Digital Signature Algorithm (ECDSA) to verify that a sensor node belongs to the multicasting group. The advantage of using ECDSA is mainly about avoiding the DoS attack, the man-in-the-middle attack, and the replay attack.

This work guarantees high efficiency and effective performance compared to other benchmarking approaches.

To sum up the centralized authentication architecture is based on a centralized trusted third party. This trusted third party is responsible for managing and disseminating the credentials to succeed the authentication process. Nevertheless, the number of connected devices and users consuming the IoT services is increasing progressively, which causes a bottleneck and congestion problems. Thus, distributed IoT authentication architectures are proposed to handle with these problems, which is the subject of the next section.

2.3.2.2 Distributed IoT Authentication Architecture

A distributed authentication architecture in IoT is defined within distributed trust communicating parties, where the participants coordinate autonomously to build further trust [52]. A distributed architecture could be hierarchical, where the authentication procedure is achieved through using multi-level architecture or flat, where no hierarchical architecture is required to deal with the authentication procedure. There is no central trusted third party in distributed authentication architecture schemes that can evade the problem related to a centralized authority, including congestion signaling leading to a single point of failure [52]. In particular, IoT systems that rely on a centralized third party cannot handle the unconventional security requirements, particularly scalability and dynamic changes in IoT. However, the distributed trusted systems still suffer from the non-repudiation identity problem since anyone can establish a trusted identity provider. Consequently, it is necessary to trust all service providers, which might lead to interoperability issues. Therefore, trustless distributed identity providers in peer-to-peer networks are beneficial to handle with all previously mentioned security requirements. In the following, we discuss the distributed trusted authentication IoT architecture and the trustless distribution authentication IoT architecture based on Blockchain technology.

2.3.2.2.1 Distributed Trusted Authentication IoT Infrastructure

A distributed trust authentication architecture requires that every trusted authority needs to verify the legitimacy of communicating members. In this context, the authors in [81] suggested an authentication protocol referred to as distributed aggregate privacy-preserving authentication (DAPPA). This work is built to ensure the vehicle system's authentication using multiple trusted authorities and the one-time identity-based aggregate signature techniques. Indeed, this scheme allows each vehicle to verify many messages once time and aggregate the related signatures into one message. At this level, the data collator and the vehicle save storage space in their memory needed for the authentication procedure. Regarding the previously mentioned scheme's benefits, the authors of [82] also introduced an authentication protocol using identity-based aggregate signatures to secure communication for vehicular ad-hoc networks.

A distributed trust lightweight authentication protocol was proposed in [83] to ensure a fast authentication and authorization. This work uses the token technique to ensure an energy-efficient distributed lightweight authentication and encryption system for IoT. The token generation relies on the devices' trustworthiness, where the receiver generates a token for each

sender. Then, token expiration time might be concluded based on each sender's trust value, while the sleep period of the receiver radio is determined based on its remaining energy. Furthermore, this scheme applies Cipher Block Chaining-Message Authentication Code to encrypt exchanged messages. This proposed distributed scheme achieves mutual authentication between users and IoT devices and ensure higher resilience against node capture attack. However, IoT objects using the proposed encryption and token generation strategy cannot store trust values in limited storage memory.

To manage the storage of the trust value among sensor-enabled mobile devices in the IoT environment, the authors of [84] provided a trust management mechanism. This scheme introduces a security manager to initiate a request to authenticate the devices that cannot hold the security. The security manager is responsible for establishing communication between two nodes that want to exchange information or services from each other. At this level, the authentication of the node is ensured by verifying the request sent to the security manager. This work provides a confidentiality security service by adopting a public key for the two communicating nodes during every communication among nodes. Furthermore, this approach provides users with confidentiality, authentication, and integrity based on the encryption used. However, the performance is uncertain and theoretical evaluation is insufficient to prove the effectiveness of the proposed model.

2.3.2.2.2 Distributed Authentication Architecture Based on Blockchain

The distributed trust infrastructure insists that every trusted entity is responsible for evaluating and maintaining the trust among the communicating parties. In general, these schemes' security is more resilient than centralized schemes, and it handles the unconventional security requirement such as the scalability issue. Otherwise, the trusted distributed schemes are more vulnerable to collusion attacks as it is harder to keep track of the whole system. Furthermore, the overhead of an individual entity is higher than centralized approaches. Therefore, the Blockchain technology is introduced to achieve a distributed trustless authentication concerning security requirements.

2.3.2.2.2.1 Blockchain Overview

Blockchain is a recent effective technology of secure computing without relying on the centralized authority in an open system. Furthermore, according to the data management viewpoint, a blockchain is a distributed database, where transaction records are held and organized into a chain of blocks [85]. Besides, referring to the security perspective, the blockchain is a peer-to-peer network secured by using intelligent cryptography with crowd computing [86]. Consequently, as a secure ledger, the blockchain contains all the transaction records made by all the participating entities to constitute the expanding chain of blocks. Basically, the nodes constituting the blockchain network do not trust any other node while trusting the whole blockchain network. These nodes carry on a pair of cryptographic keys used to generate transactions for blocks. A block maintains information about transaction records, the hash value of the entire block itself, and the hash value of its preceding block, which serves as a cryptographic linkage to the previous block in the blockchain. Such a block's commitment

in the network is achieved through a consensus procedure enforced by the network (a classification of the consensus is given in the appendix 1). The consensus procedure controls the admission of new blocks into the blockchain, the read protocol for secure verification of the blockchain, and the consistency of the data content of transaction records included in each copy of the blockchain maintained on each node [86]. As a result, the transaction is immutable and cannot be altered and tampered with by hindsight. Therefore, a blockchain is a secure and distributed ledger that archives all transactions between any two parties of an open networked system effectively, persistently, and in a verifiable manner (the transaction procedure is presented in the appendix 1)

Blockchain offers appropriate features that would enhance security in the IoT environment. Indeed, new emerging IoT applications are taking advantage of the security transaction messaging. In particular, these features include tamper resistance, distributed ledgers, cryptography secured records, and resilience to a single-point failure [91]. In addition, blockchain consists of an efficient way to automate business and create smart contracts among smart devices without referring to central entities. A smart contract is a kind of digital rules forming the terms of contract [92]. Concretely, a smart contract consists of a computer program that is automatically executed by smart objects and defines a set of rules and conditions based on the terms of the contract. Blockchain could help to ensure the smooth running of the contracts in a distributed way. The benefits that blockchain technology can add to the security domain in IoT are [86]:

- Blockchain security aims to make data tampering infeasible by storing data copies at as many possible locations.
- The blockchain system is characterized by decentralization, and the distributed nature of the ledger provides availability and integrity.
- Blockchain is supposed to run on untrusted distributed devices without a central authority.

Some blockchain-based solutions have recently been proposed to solve security and privacy issues in IoT in the literature, especially authentication, which is the subject of the next subsection.

2.3.2.2.2.2 Authentication in IoT based on Blockchain

According to the existing IoT authentication solutions, the distributed model could improve the IoT systems' scalability and ensure high-level security and privacy for users compared with centralized architecture, where users and devices have to trust a third party. Nevertheless, these distributed systems suffer from the non-repudiation identity problem since anyone can involve a trusted identity provider. Therefore, the blockchain technology emerges as a prominent perspective to design IoT security solutions in distributed trustless environments. Taking advantage of this feature, many researchers proposed distributed authentication solutions for IoT based on blockchain.

In [93], the authors provided a distributed lightweight anonymous authentication protocol for vehicular fog services based on blockchain. Their scheme is a consortium blockchain that adopts the Practical Byzantine Fault Tolerance (PBFT) consensus algorithm to validate new

blocks. This work also introduces service manager nodes responsible for ensuring cross-data authentication and the blocks' validation. Furthermore, the authors guarantee the anonymity of the client by generating a pseudonym with every authentication. The blockchain's use offers vehicle clients the possibility to choose to non-reauthenticate with the system when changing the location. However, in the designed scheme, the blockchain is not deployed for keeping authentication keys but for storing authentication results, while the keys are generated in a corporation with a fully trusted authority. In addition, mutual authentication between vehicles and services managers is not achieved, which is an essential feature in the authentication protocol. Hence, the authors of [94] extended this work to resolve these problems. Indeed, they used elliptic curve cryptography (ECC) to provide mutual authentication between vehicles and service manager. However, the key generation still depends on a fully trusted authority. This scheme is more efficient than [93] in terms of computation and communication overhead and safer in terms of security.

The authors of [95] proposed an efficient distributed authentication and access control management for the IoT. This scheme uses a set of validators to apply a distributed consensus protocol and agree with an IoT device's admission using predefined rules. The blockchain nodes maintain the full IoT device's information and the corresponding certificates gathered from authorities. Furthermore, blockchain provides the integrity and validity of IoT devices' information, leading to easy and secure access from anywhere. In particular, the IoT device no longer needs to send its certificate to the system to be authenticated. Otherwise, the authentication is ensured by checking the public key of the IoT devices used to sign its request in the valid status stored in the blockchain. Their scheme has considerably reduced both the communication and computation overheads associated with the use of certificates.

It is noticed that changing some parts of the blockchain network costs much more than building a new network. As a result, the cost of upgrading the IoT system is very high. For this reason, the authors of [96] have presented an authentication scheme for the IoT using blockchain. Their protocol allows users to access and manage IoT device information with respect to the privacy-preserving. In addition, this work can establish a secure authentication in IoT applications. However, this scheme is vulnerable to various attacks such as secret disclosure, traceability, and replay attacks, impacting privacy and trust. The authors of [97] provided an enhanced version of this protocol in terms of security and cost.

Moreover, to ensure user integrity, a lightweight authentication and authorization framework for the Blockchain-enabled IoT network was proposed [98]. The proposed protocol consists of two services named, applications and networks. The IoT applications are cloud-based services, like public mobility assistance, offering seamless mobility for users to interact. This characteristic ensures a reasonable availability of the system. Otherwise, the network service concerns the sensed data transferred in the network by a user who needs to forward into the cloud via the predefined path. These data are stored continuously to be available for all entities in the network, such as gateway cloud services. Every entity in the network is defined and attached with a public-key certificate to authenticate the entire service information and get data access. Experimental results analysis show that this framework is robust and highly secure, and reliable compared to others.

To sum up, authentication is the process of confirming and ensuring the identity of objects. In the IoT context, each object should have the ability to identify and authenticate all other objects in the system. Once authenticated, an IoT object needs to get permission to access another IoT entity or have something [38]. In the following section, we explain the access control mechanisms in the IoT environment.

2.3.3 Access Control in IoT

Access control or authorization is a fundamental element to address IoT security, and mainly it concerns regulating who can access what kind of resources or services. Indeed, an effective access control system satisfies the main security requirements of availability by assuring data access by legitimate users when requested, integrity by preventing resources from being modified without authorization resources, and confidentiality by avoiding unauthorized data exposure. Furthermore, to address a successful authorization, the following phases are required: first is about defining a security policy by setting rules, the second concerns selecting an access model to encapsulate the defined rules, then applying the model with the access policy [9]. Various access control models are designed through the literature in IoT to handle the growing security requirements, which are discussed in the next subsections. In the following, we discuss the traditional access control solutions adopted for the IoT network and then present the access control solutions based on the group key management technique.

2.3.3.1 Traditional Access Control Solutions

Role-based Access Control (RBAC) is an access control approach and framework applied to control and restrict user access privilege to resources based on roles. The RBAC model comprises four different components: the core RBAC, the hierarchical RBAC, the static separation of duty relations, and the dynamic separation of duty relations. Each component assigns various functionalities to the RBAC. In fact, this model alleviates the effort of managing access rules by assigning roles to permissions instead of granting access rights directly to users [9]. RBAC approach was introduced to the access rights to the smart things that are managed via the Web [100]. The integration with the Web has the purpose of performing a mapping between RBAC entities: Users, Permissions, Objects, Authorization policies, Session, and the different components of the Web of Thing. However, RBAC is unsuitable for distributed networks such as IoT, as this kind of access model is not flexible and scalable enough. Indeed, the user should have access to credentials and profiles on every device he/she owns, which is an issue of scalability. At this level, we could confirm that RBAC cannot handle millions or even billions of devices, where each one has a specific role to access and many users to administrate, which makes it unsuitable for a large IoT environment.

Another access control solution, named Attribute-Based Access Control (ABAC), is more flexible and scalable comparing to RBAC. ABAC model access is established according to various attributes presented by a subject. This subject is identified through the attributes associated with some characteristics [101] [102]. Besides, access policy rules specify conditions over a set of attributes under which access is authorized or denied. Indeed, when a subject initiates an access request to accomplish operations on objects, this request could be denied or granted according to the attributes defining the object and the subject. Therefore, the ABAC

model facilitates the assignment of rules and the definition of an access control list. In fact, instead of defining the access permission for each system entity, the attributes authorities are responsible for managing and distributing the set of attributes to proper users. Consequently, access management is effectively simplified, considering the number of attributes less than the system's number. Many works have been suggested in the literature using the ABAC model in IoT. Recently, the authors of [102] have proposed an efficient authentication and access control scheme for the perception layer of the IoT. This scheme adopted the ABAC-based authorization method for access control policy based on ECC. This work designed an efficient mutual authentication based on secure key establishment protocol. Moreover, the access to the data in this approach is based on user attribute certificates that ensure fine-grained access control. Nevertheless, this model involves complex management, where each entity should update the attributes to maintain a continuous authorization before, during, and after the access execution permission, which is not suitable to be applied to constrained devices.

An advanced access control model named the Usage Control (UCON) proposed by [103] introduces numerous novelties compared to traditional access control such as RBAC and ABAC. In particular, it is composed of eight components, including authorizations, obligations, conditions, continuity, and mutability. As the traditional models, UCON uses the notion of subjects and objects associated with their attributes. Specifically, the subject can be an entity in a system and is represented by several properties and capabilities related to its attributes, while the object is associated with object attributes. The subjects hold rights on objects, which leads to grant access or usage of an object. Both subject and object attributes can be mutable at this level, which means that the value can be modified only by administrative action and not by its user's activity. Moreover, UCON handles the changing of access attributes while the access is in progress, which avoids dissatisfaction with the security policy. Also, it solves the problem of continuous authorization before, during, and after the access execution permission. Even though the UCON model's claimed novelties, UCON still a conceptual approach, and only theoretic experiments have been conducted. Thus, practical feasibility and the construction of this model in IoT should be carried out.

Capability-Based Access Control (CapBAC) was introduced to address an appropriate authorization model for the IoT environment requirements. The concept of CapBAC is based on using a cryptographic token, ticket, or key that permits to grant access rights and privileges. Many CapBAC approaches were presented in the literature, citing the [105], where the authors used the Access Control Matrix (ACM), and [106] that use Access Control List (ACL) to build capability-based access control models. Actually, CapBAC might be classified as a centralized and distributed model adopted in many large-scale projects [107] and is widely used in the IoT field. Indeed, the centralized model provides interoperability, reduces computation complexity, and enhances memory efficiency. However, this mechanism is also susceptible to a single point of failure as the access decision's delegation depends on a central entity. The distributed approach assigns access control logic to smart devices [104]. However, the IoT environment is characterized by resource-constrained devices that are easily compromised. Consequently, CapBAC is unsuitable to address a secure access control mechanism in untrustworthy IoT environments.

We summarize these traditional access control solutions in the Table 2.4 as follow:

Table 2. 4: Access Control Solutions Comparison

Factors	RBAC	ABAC	UCON	CapBAC
Access Control to Information	Through roles	Through attributes	Through object attributes	Through ACL
Access Control Based on	Classification of roles	Evaluation of attributes	Evaluation of subjects and objects	Classification of roles
Flexibility for Accessing Information	High	High	Very high	High
Access Revocation Complexity	Easy	Easy	Very easy	Very easy
Support for Multilevel Database System	Yes	Yes	Yes	Yes

With the advent of the IoT environment, these access control models designed for centralized systems become obsolete due to the rapid growth of roles and policies. In practice, ensuring protection for IoT systems is a great challenge due to the IoT environment's dynamic nature. Indeed, IoT devices should maintain the connection to the Internet because of the access control configuration and satisfy particular needs. Consequently, it becomes easier to compromise IoT devices that are issue to cyber-security risks and attacks with severe impacts. Besides, more and more factors and parameters should also be considered when designing access control solutions to meet the requirement of scalable decentralized IoT systems, and day-to-day access control decisions are becoming a group key management (GKM) responsibility.

2.3.3.2 Group Key Management Solutions in IoT

Regarding the continuous growth of connected objects in IoT, group-based applications have emerged the communication in the IoT environment. In these group communications, numerous members are participating in exchanging and sharing information. Thus, securing group communication among members should be taken into consideration, including authentication and authorization. For that reason, implementing a system to control the assignment of permission system must be built. The Group Key Management (GKM) has been emerged as a prominent solution to achieve the access control and assignment of permissions [108]. The GKM is a peer-to-peer access control mechanism for IoT applications. Indeed, GKM provides its access control based on signed permission certificates. Furthermore, since the group members in a heterogeneous network like IoT are characterized with a high dynamicity, where members can join and leave the group, managing a secure group communication is difficult. Therefore, the group key must be changed whenever a member leaves or joins the group to ensure forward and backward secrecy. Figure.2.5 shows the taxonomy of Group Key

Management Protocols, which are classified into three categories: centralized, decentralized, and distributed to be discussed in what follows:

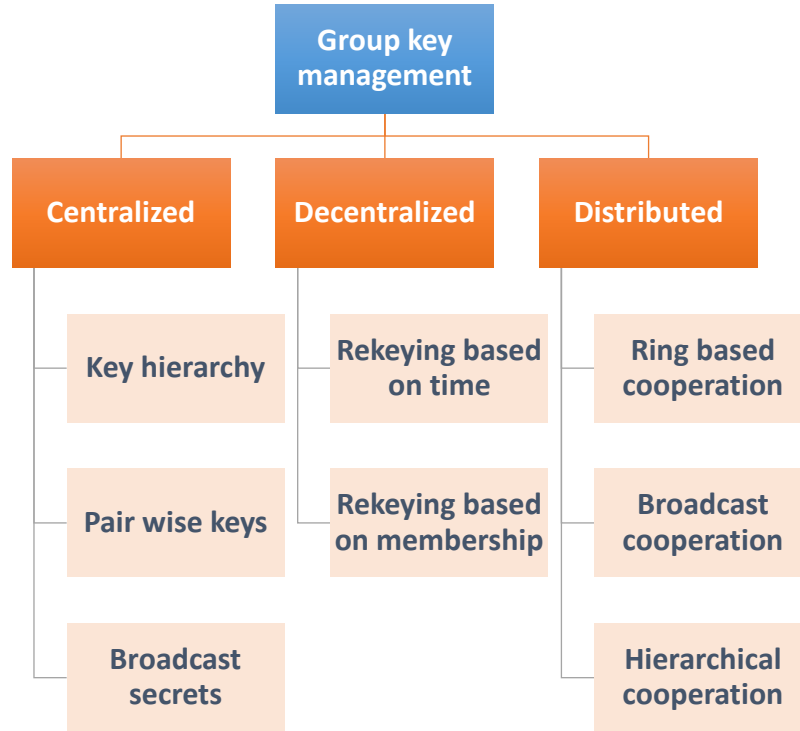


Figure.2. 5: Group Key Management Taxonomy

❖ Centralized GKM:

The centralized group key management operates with a single entity for controlling the group communication. The group key is requested from a central server. This central server handles the request by creating and disseminating the key to the appropriate group members. Various encryption mechanisms are used in the centralized key management to setup a secure group communication. Indeed, it adopts the symmetric keys, asymmetric keys, and the secrets to manage key distribution.

Various are the works that have provided centralized group key management mechanisms, the two-fundamental centralized GKM approaches are the Logical Key Hierarchy (LKH) [109] and the One-way Function Tree (OFT) [110]. Both methods design a hierarchical key tree based on symmetric keys, including traffic keys and encryption keys. The traffic keys are designed to encrypt the data among the group, while the encryption keys are used to encapsulate and distribute the updated group key and traffic keys. In contrast to LKH, all the OFT implementations suffer from collusion attacks and increase devices' computational overhead for obtaining group keys. Hence, OFT is far from ideal in an IoT environment, where the communicating devices may have limited computational power. Indeed, in the LKH, upon a join and leave events, the key distribution center engenders $O\log(n)$ complexity to reach the group key to all group members, making this protocol more suitable for small groups.

In order to reduce the impact of rekeying operations, the authors of [111] have introduced an interval-based centralized protocol. This scheme suggested a mechanism that can predict the time of a member leaving the group. In fact, when a member first joins the group, the key

distribution center transmits the needed rekeying materials according to the period for the member's intention to be part of the group. Once this period expires, the member leaves the group without any triggering of rekeying events. Nevertheless, this approach has numerous drawbacks as predicting leaving members' time is not practical for highly dynamic networks. Moreover, remaining for a long time in the group may risk increasing the storage in IoT constrained devices. Hence, this protocol cannot meet the requirement of dynamic IoT environments with a high number of unpredictable leaving events.

All previously mentioned schemes are designed for single multicast groups. Authors of [112] accommodate various services' groups to ensure many multicast groups. Their scheme addressed rekeying in the wireless mobile environment, based on a centralized architecture and an LKH mechanism to manage multiple communications. Besides, taking advantage of the construction proposed by [112], authors in [113] established a two-tier centralized system, where groups run the LKH method to handle updates of keys efficiently. This scheme addresses the requirement of dynamic nature in the IoT environment. However, communication within user groups is based on symmetric keys, increasing the centralized center's rekeying operations costs.

The centralized GKM presents several problems, including the latencies caused by the central server's workload. In fact, the procedure of creating key groups, defining keys for access considering the permissions, and disseminating keys to group members takes time and effort. Due to the growing number of groups on the IoT network, the number of group members grows. Consequently, the workload at that central server quickly reaches capacity, which can lead to a single point of failure.

❖ **Distributed GKM:**

A distributed key management mechanism has no explicit key distribution center KDC. Furthermore, the group key is generated either in a collaborating manner between group members or by one member. Hence, all members might perform the access control decisions and then contribute with information to create a shared group key. Besides, each group member should maintain and keep track of the other members to make robust and secure communication among the group. Various cryptographic mechanisms are adopted to securely achieve distributed key management, classified into three categories, including ring-based cooperation, broadcast cooperation, and hierarchical cooperation.

Some typical distributed key management schemes known in the literature include Conference Key Agreement [114], Distributed Logical Key Hierarchy [115], Distributed One-way Function Tree [116], Diffie-Hellman Logical Key Hierarchy [117], and Distributed Flat Table [118]. Recently researchers and references paid more attention to collaborative group key agreement. In fact, the authors of [119] provided a completely distributed approach for group key management based on distributed hash tables. In this approach, key management is not controlled by any central authority. Indeed, anyone can create groups and principles by collaborating with members. This protocol is characterized to set with various applications. Furthermore, it enhanced the security and privacy level after removing the central authority. However, the member group should keep a hash table of all members to ensure the security requirements. Hence, this approach is very costly in terms of computation and storage for each

group member, which is not suitable for the devices' dynamic nature and the constrained resources character in the IoT environment.

The authors of [120] provided a distributed key management scheme to decrease the communication overhead by adopting a Distributed Batch-based Group Key. Their work is based on polynomial to set up and generate the group key for collaborative groups in the IoT environment. It also studies the heterogeneity of the devices with multiple capabilities under IoT enabled sensing networks. Nevertheless, this scheme is limited to manage the communication in one group, and it does not consider multiple communications among different groups. Also, a large group's dynamicity with the join and leave events makes the system more vulnerable to attacks, while it is crucial to protect the backward and forward secrecy.

However, for large groups collecting a contribution from every member, the distributed key management approach is time processing-consuming and capacity power-consuming [120]. Likewise, the scalability issue imposed by the IoT environment is not fulfilled.

❖ **Decentralized GKM:**

The decentralized group key management mechanisms split the network of a large group into several smaller subgroups. Each subgroup is associated with the group manager responsible for creating and distributing keys among the group members. This group manager tries to reduce the problem of concentrating the work on a single server, which can avoid the single point failure issue. Besides, the decentralized key management mechanisms meet the unconventional security requirement, including scalability and reliability in a dynamic IoT environment. Indeed, they offer beneficial solutions that secure multicast communication by restricting the impact caused by the membership change in one group. Also, in the decentralized solutions, the group member should not keep track of the other members, which reduces the overhead. The decentralized key management techniques are classified depending on rekeying operations, which concerns updating the group keys regarding some conditions: rekeying basing on time and rekeying based on membership.

Protocols like Scalable Multicast Key Distribution (SMKD) [121], Intra-domain Group Key Management Protocol (IGKMP), Hydra fall under the membership-driven category [116], Kronos, MARKS [112], and Dual-Encryption Protocol (DEP) [120], are examples of decentralized key management solutions. Recently, a Decentralized Batch-based Group Key (DBGK) scheme was suggested in [122]. This scheme involves several sub-groups managed by the area key management server, while the general keying server manages the whole group. In this work, the group key is composed of long-term and short-term keys. Similarly, security credentials are shared with member nodes in the group, ensuring the availability of resources. Also, it achieves and enhances the efficiency of the system, including storage, computation, and residual energy.

Likewise, in [123], the authors proposed an enhanced decentralized key management using a distribution list of the session key and key update slot for each subgroup. This list is centrally managed by a node called the area key distributor. The proposed protocol alleviates the 1-affect-

n phenomenon and transmission overhead of the core network, but it does not ensure forward secrecy. Hence, the authors of [124] extended the proposed scheme [123] to another protocol called area based multiple GKM that securely provides services when users migrate to different wireless networks, ensuring forward secrecy. Nonetheless, its high overhead, due to revocation events, makes it unsuitable for dynamic IoT environments.

Although the benefits that accord the decentralized schemes compared to the other GKM model, some challenges should be considered while building the decentralized GKM. The first one is about ensuring efficient communication between different group key management schemes to distribute keys among member subgroups securely. The second is about establishing a trust communication between the third parties involved in a decentralized manner. Then, to ensure the authentication of members participating in the session group even if they belong to the same or different network.

2.4 Summary & Discussion

To design a secure IoT system in a peer-to-peer network, we studied and reviewed the existing IoT solutions in the literature in compliance with the established security requirements of the developing IoT environment. Throughout our analysis of the selected research works, we notice that authentication and authorization are the IoT environment's principal security requirements. Indeed, we observe that the traditional security solutions, which are mainly based on cryptographic techniques, were improved for IoT applications. These solutions are generally efficient in terms of storage, communication, and computation. However, they cannot handle and fulfill the new IoT environment security requirements, including scalability, heterogeneity, interoperability, dynamic changes, etc. Although we highlighted some beneficial approaches to handle some of IoT's unconventional security requirements, some of the existing approaches are still closely associated with the previous major security paradigms. In fact, distributed authentication trusted frameworks are useful to handle the scalability issue and eliminate the load and trust on a third party, but the distributed trusted server brings more attacks' attention. Therefore, the blockchain technology-based on trustless distributed nodes might be beneficial to deal very well with scalability and heterogeneity issues. However, blockchain technology is energy and time consuming due to the consensus mechanism to validate transactions. For that reason, it is not suitable to implement blockchain for constrained resource devices. However, it is essential to take advantage of the trustless secure infrastructure of the blockchain, which can significantly preserve the security of a distributed IoT environment and enhance users' security.

Furthermore, we also surveyed and outlined the exiting access control and key management solutions, which are the sources and origin for designing authorization solutions. We illustrated various access control models, including Role-based Access Control (RAC), Attribute-based Access Control (ABAC), Capability-based Access Control (CapBAC), and Relationship-based Access Control (RBAC). However, all of them are not suitable for a large dynamic environment and limited resources IoT environments. In fact, they need all to rely on a connected third party to provide the access permissions continuously to the demanding objects or users. Otherwise, key management techniques eliminate the dependency on an online third party. Considering IoT characteristics such as scalability, heterogeneity, dynamicity, and security, we provided the

shortages of traditional key management solutions in the context of a multi-services IoT environment. Therefore, we are experiencing impressive challenges to secure and protect large dynamic IoT environment due to the significant increase in the attack surface.

2.5 Conclusion

Throughout this chapter, we surveyed a comprehensive overview of the IoT by presenting the IoT architecture and the different challenges related to its continuous progressing. This IoT revolution has emerged with a remarkable potential to cover a wide range of applications in various domains, such as smart homes, smart industry, smart healthcare, smart cities, and intelligent transportation. The tremendous number of connected objects transmitting data and supporting sensitive IoT applications makes IoT environment vulnerable to many attacks. We investigated reviewing security solutions proposed for IoT, and we identified the security inherent challenges and limitations. For that reason, we enumerated the basic security service, including confidentiality, integrity, availability, non-repudiation, and privacy. To meet all these security services, designing authentication and authorization are fundamental for any IoT application. However, in addition to these security services, IoT environments have reached a remarkable development; therefore, they are facing many new security challenges and issues that need to be resolved for effective deployment. Thus, we listed these security requirements in this chapter, namely scalability, heterogeneity, limited resources of objects, interoperability, and dynamicity related to high changes of connected objects and users of the IoT environment. For that, we discussed the existing IoT solutions to handle the mentioned issues, such as blockchain technology, which builds a trustless distributed infrastructure. Moreover, we presented the group key management technique used to ensure a robust authorization and handle the group communication issue in IoT environment. Despite the presented security solutions that take care of the context in which IoT applications involve, there are still a lot of open issues to be addressed, such as scalability and dynamism issues, mainly because IoT is becoming an Internet of Everything where humans, data, processes, and objects are developing together in a highly dynamic and complex system. Therefore, throughout this thesis, we investigated to design a secure IoT solution to fit these issues, including the scalability, dynamic changes, and limited resources IoT environment through achieving the two primary security features, namely authentication, and authorization. In our first contribution, presented in the next chapter of this manuscript, we design a new lightweight mutual authentication based on the token concept. In this work, we added a new security layer for the authentication protocol to meet the IoT applications' security requirements (such as the reservation Smart hotel system) by combining the user credential with the token to identify legitimate users for a predefined period.

Token-based Lightweight Authentication for IoT environment

3.1. Introduction

As presented in the previous chapter, the rapid growth of the Internet of things has given rise to many different applications related to environmental sensing and industrial areas (e.g., smart city, smart hotel, smart office) [35]. This huge number of connected objects brought more security challenges to IoT environments concerning data protection, access control, and authentication between the user and smart devices. In particular, authentication is becoming more challenging with the new IoT platforms [126], where the user needs to be authenticated for a predefined fixed interval of time with a list of smart devices. Nevertheless, most IoT devices are resource-constrained devices, where the computation capacity and energy consumption are limited. Indeed, the authentication process should be adequately adapted to deal with these challenges and save power consumption to increase IoT devices battery lifetime.

Throughout the literature, different user authentication solutions were investigated in IoT environments. Researchers in [127] have introduced a continuous user authentication in IoT based on a secret shared scheme to prove the user's legitimacy for a predefined interval of time. However, their solution is based on a password mechanism only and hence considered as a heavy solution and vulnerable to many security attacks. In particular, as IoT devices communicate over insecure communication channels, the probability of an illegal user (attacker) that can break the security and gain access to the smart device increases during communication. Hence, the security mechanism should adopt a firm policy, such as multi-factor authentication and encryption [128] [58]. The authors in [58] have proposed a three-factor user authentication in the IoT environment to enhance security during communication. However, compromising one secret key in their scheme, an attacker may deduce any previous session key, which represents a severe threat. For this purpose, it is essential to ensure the perfect forward secrecy, which represents a fundamental security property for session key-based authentication.

To overcome the important issues mentioned above, we have proposed in this chapter a new efficient and secure user authentication protocol named *Token-Based Lightweight User*

Authentication (TBLUA) to reach a robust security and ensure the perfect forward secrecy for such IoT environment. In this context, we have introduced a software *token-based authentication* as an efficient solution to create a strong binding between the users and the smart devices. Indeed, we generated an additional security layer of authentication by adopting a software token technique that offers access to a specific resource for a predefined fixed interval of time. Furthermore, we used only lightweight computation operations such as XOR and hash functions as cryptography techniques to authenticate the user with IoT devices. To this end, we guarantee a remarkable decrease in computation time and saving energy of IoT devices during the authentication process, while preventing the most widespread security attacks and ensuring the known security properties, especially perfect forward secrecy. We evaluated the robustness of our solution in terms of security using AVISPA as a formal verification tool. Results have shown that the proposed *TBLUA* is secure under various kinds of attacks. We also evaluated its performances, and results have shown its efficiency in terms of computation and communication. Finally, we conducted a proof of concept that describe the smart hotel use case in the context of the PARFAIT project.

The rest of the chapter is structured as follows. We briefly present a survey of various existing IoT authentication schemes proposed in the literature. Then, we give an overview of the cryptography background used in this chapter. After, we present a general description of the network and the threat models. Later, we give a detailed description of the proposed scheme *TBLUA* for user authentication in IoT environments. After that, informal security analysis and formal security evaluation using AVISPA tools are presented. To prove our approach's effectiveness, we achieve a performance comparison with the existing relevant schemes through providing a simulation analysis, results, and discussions. Finally, we present a description of the smart hotel use case, studying the vulnerabilities of such environment and giving the accomplished simulation.

3.2. Related Works

IoT environments are exposed to their potential users in general and IoT devices communicate through the Internet. Therefore, adversaries can easily access those devices which makes IoT environments vulnerable to various security threats. Consequently, authentication becomes a fundamental mechanism for the user to be first authorized to the Gateway (GW) as well as the smart device (IoT device) before granting access to the real-time data. In this section, we study existing authentication solutions in IoT presented in the literature. Indeed, to achieve user authentication, Wong et al [129] proposed a lightweight hash-based user authentication scheme, but Das [130] found out that is vulnerable to replay attack and stolen-verifier attack. Subsequently, Das [130] presented a two-factor authenticated key establishment scheme for WSNs, which claimed to provide strong authentication and resist to various kinds of attacks. However, many articles [9-13] pointed out that Das's scheme [130] is still vulnerable to privileged insider attacks and parallel session attacks. Although the abovementioned schemes [9-13] have much better performance than Das' scheme [130], they still have various defeats such as smart card loss attacks and forgery attacks. In 2012, Das et al. [135] presented a better scheme than the previous two-factor authentication to solve these weaknesses. Unfortunately,

the security of this new scheme was not satisfactory due to its vulnerability to some attacks such as privileged insider attack and stolen smart card attack [15-17]. Turkanović *et al.* [136] designed a lightweight user authentication protocol for wireless sensor networks (WSN) tailored for an IoT environment. Their protocol is based on symmetric key encryption, hash and XOR computations that tends to save both computation and communication resources. However, it has also several security flaws, as it does not protect privileged insider, offline password guessing, user impersonation attacks and untraceability [128].

Chang and Le [137] recently designed smartcard-based user authentication protocols P1 and P2 with the help of user password: P1 is greatly lightweight since it is based only on bitwise XOR and hash functions; P2 is not lightweight as it applies ECC along with bitwise XOR and hash functions. Unfortunately, Das *et al.* [138] found out that both P1 and P2 are insecure against offline password guessing and session specific temporary information attacks. In addition, P1 is also insecure against session key breach attack. Most recently in 2016, Gope and Hwang [139] designed a practical authentication scheme, which ensure mutual authentication, user anonymity and perfect forward secrecy. Nevertheless, the protocol causes the desynchronization attack in the communication between the gateway and the smart device because the hash chain value is updated after each successful session. In 2017, Wazid *et al.* [58] proposed a three-factor authentication scheme that ensures various kinds of imperative security properties like, mutual authentication, sensing node capture, impersonation, and privileged insider attacks. Unfortunately, it requires more communication and computation costs compared to other schemes. Furthermore, it does not ensure perfect forward secrecy, which is an indispensable security property for authenticated schemes.

Table 3. 1: Evaluation of IoT Authentication Schemes

Scheme	Environment	Authentication technique	Strength(+)/Weakness(-)
[129]	Wireless Sensor Network (WSN) environment	Single Factor authentication uses: Hashing/ XOR functions	<ul style="list-style-type: none"> + Resilience to the insider attack. + Low computation and communication overhead. - Vulnerable to replay attack and stolen-verifier attack. - Perfect forward secrecy not considered.
[130]	Wireless Sensor Network environment	Two-Factor authentication uses: Hashing/XOR functions	<ul style="list-style-type: none"> + Low computation and communication overhead. + Resist against the replay attack, and denial of service attack. - Vulnerable to privileged insider attacks and parallel session attacks. - Cannot ensure mutual authentication and session key verification.
[132]	Wireless Sensor Network environment	Single Factor authentication uses:	<ul style="list-style-type: none"> + Low computation and communication overhead.

		Hashing/XOR and Symmetric key Encryption functions	<ul style="list-style-type: none"> + Anonymous authentication for remote users. - Vulnerable to the insider attack problem. - Perfect forward secrecy not considered.
[133]	Wireless Sensor Network environment	Two-Factor authentication uses: Hashing/XOR	<ul style="list-style-type: none"> + Protection against Gateway node bypassing attack. + Mutual authentication between GW and sensor nodes. - Vulnerable to smart card loss attacks and forgery attacks. - No backward and no forward secrecy are considered.
[134]	Internet of Things (IoT) Environment	Three-Factor authentication uses: Elliptic Curve Cryptosystem	<ul style="list-style-type: none"> + Ensuring user anonymity and forward secrecy. + Resist to impersonation, replay and dictionary attack. - Backward secrecy not considered. - High communication and computation cost.
[136]	Internet of Things (IoT) Environment	Single Factor authentication uses: Hashing/XOR and Symmetric key Encryption functions	<ul style="list-style-type: none"> + Saving both computation and communication. + Resilience against Denial of Service attack. - Vulnerable to insider attack, stolen smart card and offline password guessing attacks. - Vulnerable to session key disclosure.
[137]	Wireless Sensor Network environment	Single Factor authentication uses: ECC along with bitwise XOR and hash functions	<ul style="list-style-type: none"> + Mutual authentication is achieved. + Ensuring the perfect forward secrecy + Resilience against DoS attack. - High communication and computation overhead. - Vulnerable session key breach attack.
[139]	WSN environment	Single Factor authentication uses: Hashing/XOR and Symmetric key Encryption functions	<ul style="list-style-type: none"> + Anonymous authentication. + Low complexity. + Ensure perfect forward secrecy. - Vulnerable to session key disclosure - Vulnerable to the desynchronization attack in the communication between the GW and the smart device
[128]	Internet of Things (IoT) Environment	Three-Factor authentication uses: ECC along with bitwise XOR and hash functions	<ul style="list-style-type: none"> + Anonymity and untraceability in the authentication. + Resilience to several security attack such as DoS, replay attack, and man-in-the-middle attack. - Heavy computational cost. - No backward and no forward secrecy are considered.

[58]	Internet of Things (IoT) Environment	Three-Factor authentication uses: Hashing/XOR and Symmetric key Encryption functions	<ul style="list-style-type: none"> + Ensuring mutual authentication. + Resilience sensing node capture, impersonation and privileged insider attacks. - It requires more communication and computation. - Perfect forward secrecy is not provided.
------	--------------------------------------	--	--

To the best of our knowledge and as it is presented in Table 3.1, most of the authentication schemes have several security limitations especially in providing the perfect forward secrecy³ feature, which is a basic and important security property for authentication in IoT environment.

Some schemes [139] attempt to achieve this issue using the one-time hash chain technique. However, this latter causes desynchronization attack. Moreover, most schemes present high communication and computation costs in order to provide several security services. Besides, with the growth of IoT environment and applications (smart hotel, smart office, etc.), new security challenges emerged, where authentication is necessary for different predefined periods. Motivated by the above fact, we construct a new efficient authentication scheme for IoT environment based on token technique to insure a secure and lightweight mutual authentication between the user, the gateway, and the smart device. Token is used to enhance the authentication scheme and to offer access to a specific resource for a predefined period. Thus, using this mechanism, each communication will be valid only for a fixed period, which reduces risks of stolen identity.

3.3. Background

In this section, we provide a brief description about the one-way hash function mechanism and the symmetric key cryptography mechanism, which serve as techniques to design our solution.

3.3.1. One-way Hash Function

A cryptographic one-way hash function is a powerful cryptography technique that accepts a variable length block of data as input and outputs a fixed-size bit string, known as the hash value or message digest. The hash function is used to provide data integrity to check whether an adversary has modified the message in transit from the source to the destination. Furthermore, the hash technique is a lightweight mechanism as its execution time is very low compared to other cryptography mechanisms.

Considering the one-way hash function $h: \{0, 1\}^* \rightarrow \{0, 1\}^l$ takes an arbitrary length input $x \in \{0, 1\}^*$, and produces a fixed length (say, l -bits) output $h(x) \in \{0, 1\}^l$ hash value. The hash function has the following properties:

- h can be applied to a data block of all sizes.

³ Forward secrecy: when an adversary compromises the secret key of one session, then he/she can learn any previous session key, which is a serious threat.

- For any given input x , the message digest $h(x)$ is easy to operate, enabling easy implementation in software and hardware.
- The output length of the message digest $h(x)$ is fixed.
- Deriving the input x from the given message digest $y = h(x)$ and the given hash function $h(.)$ is computationally infeasible. This property is called the one-way property.
- For any given input x , finding any other input $y \neq x$ so that $h(y) = h(x)$ is computationally infeasible. This property is known as weak-collision resistant property.
- Finding a pair of inputs (x, y) , with $x \neq y$ so that $h(x) = h(y)$ is computationally infeasible. This property is referred to as strong-collision resistant property.

3.3.2. Symmetric Key Cryptography

The symmetric key encryption mechanism uses a single key for encryption/decryption. Consider the model of symmetric encryption shown in Figure 3.1. Before the secure communication takes place, both the sender, denoted S , and the receiver, denoted R , share the same secret key k . Hence, S can encrypt a plaintext with the key k using the encryption function when he/she wants to communicate securely with R . Indeed, S produces a ciphertext using the symmetric encryption function and sends it to R over a public channel (insecure channel). At this level, R can recover the original plaintext by decrypting the ciphertext using the same secret key k .

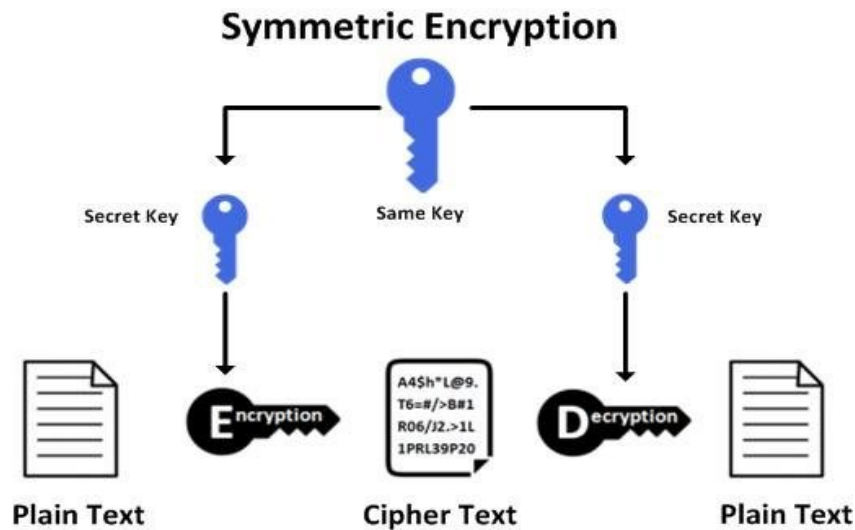


Figure.3. 1: Symmetric key cryptography

Since the channel is public, an adversary A can eavesdrop, modify, or delete the messages from the channel. In this model, A can try to derive the secret key and the plaintext with the help of the eavesdropped ciphertext. This kind of attack is known as ciphertext-only attack.

3.4. System Model and Security Requirements

In this section, we present the proposed system model that ensures a token-based lightweight authentication for IoT environment and the related threat model.

3.4.1. System Model

The system model, depicted in Figure 3.2, is composed of the following components:

- The Reservation Server (RS) responsible of generating reservation tokens for users and distributing them to the registration authority.
- The Registration Authority (RA) is a trust server responsible for registering all smart devices and gateway securely.
- The Gateway (GW) node, which is more powerful than smart devices, is used as the trusted third-party entity to help establishing the mutual authentication and key agreement [58].
- The End user, who wants to access data from smart devices for a predefined interval of time, registers himself/herself at the trusted RS.
- The Smart devices representing the IoT devices that collect and publish data to the legitimate user during the prefixed interval time.

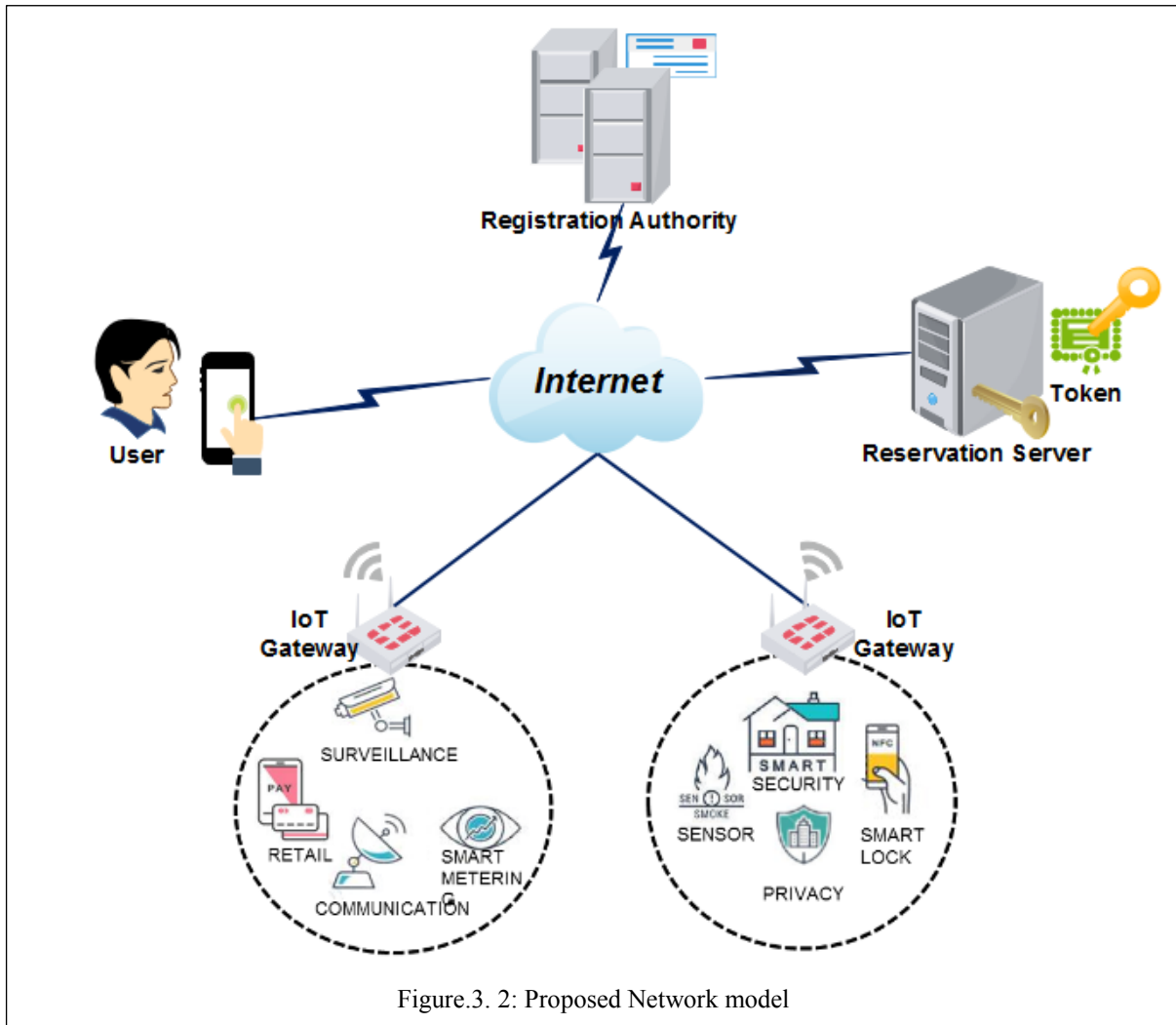
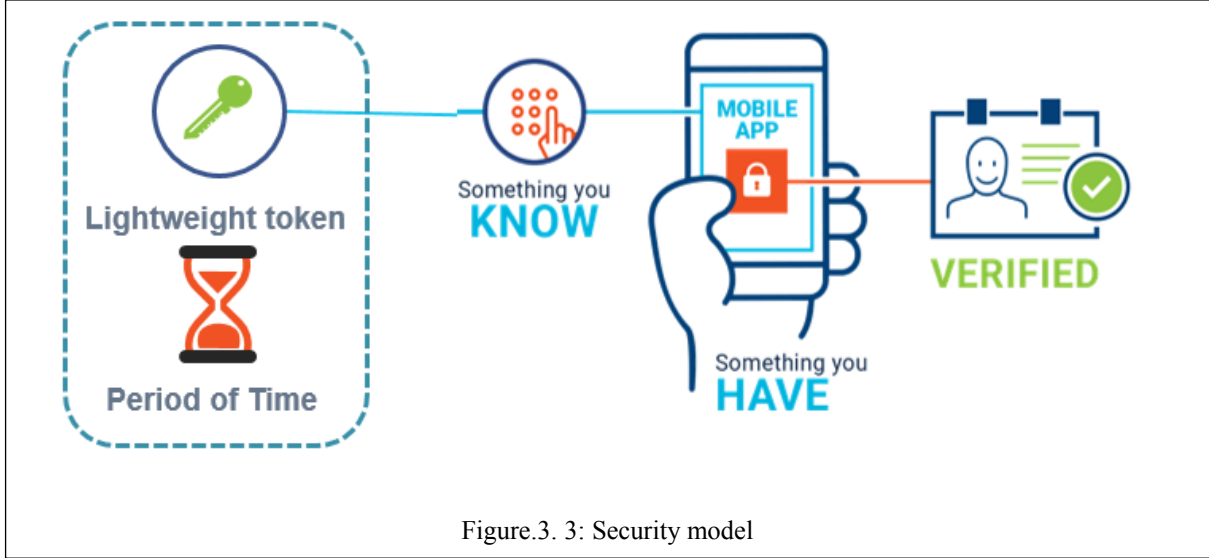


Figure.3. 2: Proposed Network model

Moreover, we assume that all the heterogeneous devices (i.e., GW, users with their smart phones and smart IoT devices) are synchronized with their clocks and agree (mutually) on a maximum transmission delay (ΔT) to protect replay attacks in the proposed scheme [140].

The main idea of our solution is to generate an additional security layer of authentication by adopting a software token technique that offers access to a specific resource for a predefined fixed interval of time. As mentioned in Figure 3.3, the authentication process is guaranteed through the login, password, and a lightweight token defined for a period.



3.4.2. Security and Threat Model

We have used the Dolev-Yao threat model [131], in which two communicating parties interact over insecure channel. According to this model, the endpoint entities such as user U_i and smart device SD_j are not considered as trustworthy. An adversary A can eavesdrop the exchanged messages, and thus modify or delete the messages during transmission. Furthermore, smart devices are not tamper-resistant and thus, some smart devices can be physically compromised by A . Therefore, A can extract sensitive information stored in those nodes using the well-known power analysis attacks [143]. Nevertheless, we assume that the GW in the proposed scheme is a trusted node and is not compromised under any circumstances; otherwise, the whole network is compromised [58]. Furthermore, RA and RS are also fully trusted and cannot be compromised by an adversary. Finally, the user's smart phone SP can be lost/stolen by A and the stored sensitive information, such as the token of identification, can also be extracted from its memory using the power analysis attacks [143].

3.5. Proposed Token Based Lightweight Authentication for IoT Environment (*TBLUA*)

After introducing the system model and the threat model, we describe in more detail the proposed authentication and key agreement protocol (*TBLUA*) that secures data transmission after a successful reservation. In Table 3.2, we define the most important notations used in this chapter.

Table 3. 2: Symbols and their descriptions

Symbols	Descriptions
RS	Reservation Server
RA	Registration Authority
U_i	User i
GW	Gateway node
SD_j	Smart device node j
PW_i	Password of U_i
ID_i	Identity of U_i
SP	User's Smart Phone
ID_{SD_j}	Identity of SD_j
K	Secret key of GW.
K_{UG}	Shared key between User U and GW.
K_{SG}	Shared key between SD and GW.
TID_i	Temporary identity generated by GW for U_i
R_1	Random nonce created by U_i
R_2	Random nonce created by GW.
R_3	Random nonce created by SD_j
$EK(\cdot)/DK(\cdot)$	Symmetric encrypt/decrypt using key K
NS	Sequence number
T	Current timestamp
ΔT	Maximum transmission delay
$h(\cdot)$	Cryptographic one-way hash function
\parallel	Concatenation operation
\oplus	Bitwise XOR operation

To design *TBLUA*, we develop the following phases, which we detail in the subsequent subsections:

- 1) Offline smart device and GW registration phase,
- 2) User reservation or registration phase,
- 3) Token distribution between GW and smart devices phase,
- 4) Login, authentication, and key establishment phase,
- 5) Password change phase.

3.5.1. Offline Smart Device and GW Registration phase

During this phase, the registration authority server (RA) is responsible for registering smart devices and gateway nodes. More specifically, the RA manages the request for the initial enrolment from IoT devices, the gateway, and the user by generating the necessary keys and identities for them.

In fact, RA selects a unique identity ID_{SD_j} for each deployed smart device SD_j , then generates a unique random 160-bits secret shared key, K_{SG} , between the GW and SD_j , where $1 \leq j \leq n$ (n is the number of smart devices) and produces the initial sequence numbers for the smart device,

NSD_j , and the gateway, NSG_j respectively, $NSD_j = NSG_j = 0$. Subsequently, the RA stores $\{ID_{SD_j}, NSD_j, K_{SG}\}$ into the smart device SD_j memory, and $\{ID_{SD_j}, NSG_j, K_{SG}\}$ into the GW memory. We adopt the concept of sequence number to counter the desynchronization attack and promote the authentication process.

The RA further randomly generates a unique GW's identity ID_{GW} , and a unique random 1024-bit gateway secret key K . Then, depending on the localization and the IoT service, the RA defines the different groups of SD_j composing the system, which is identified by $G_i = \{SD_j; 1 < j < N, N \text{ is the number of } SD_j \text{ in } G_i\}$. Each group is associated and controlled by a gateway. After that, RA computes the corresponding smart device secret information $S_j = h(ID_{SD_j} || G_i || K)$ for each SD_j and finally RA updates the SD_j node information's table entry with $\langle ID_{SD_j}, S_j, NSG_j, K_{SG}, G_i \rangle$ in the GW memory. At the end of this phase, the system is designed with many groups of IoT devices controlled with a gateway.

3.5.2. User Registration Phase

In this phase, the user U_i needs to register himself with the RA and get the necessary information for authentication to access securely to the services of a particular smart device SD_j . As depicted in the Figure.3.4, many steps are carried out by the mentioned components: the user U_i , the reservation server RS, and the registration authority RA as follow:

- **Step1:** The user picks an identity ID_i , a password PW_i , computes the masked password: $MPW_i = h(ID_i \oplus PW_i)$ and sends a request message containing $\langle ID_i, MPW_i \rangle$ to the reservation server securely using either the TLS (Transport Layer Security) protocol or in an offline mode.
- **Step2:** The reservation server RS reserves a group of smart devices G_i from the existing groups defined in the system to the user. Then, RS generates a reservation token for this user basing on the user and the gateway's identities, the selected devices group, and the time of reservation $Token_u = EK(ID_i, ID_{GW}, G_i, Te)$, where Te is the expiration time of the token. Then, RS sends the $Token_u$ to the RA through secured channels.
- **Step3:** After receiving the $Token_u$, RA generates a unique random 128-bits number n and computes a shred secret key with the user $K_{UG} = h(ID_i || n) \oplus ID_{GW}$. Subsequently, RA also generates a random number R_i to hide the masque password and selects a different temporary identity TID_i for the user U_i in each session that ensure the user's anonymity and untraceability. At this level, RA computes the necessary information for the user to be used during the authentication phase as follow:
 - ✓ $Reg_i = h(ID_i || R_i || MPW_i || K_{UG}) \rightarrow$ Ensure a mutual authentication between the user and the gateway.
 - ✓ $A_i = R_i \oplus MPW_i$,
 - ✓ $TK_{U_i} = Token_u \oplus h(ID_i \oplus R_i \oplus MPW_i \oplus K_{UG})$
 - ✓ $D_i = R_i \oplus h(TID_i || K_{GW})$.

Finally, RA stores the couple $\langle TID_i, D_i \rangle$ into the GW memory for further use and forwards to the user U_i the registration information $\langle TID_i, Reg_i, A_i, TK_{U_i}, K_{UG} \rangle$ through a secured channel.

- After receiving $\langle TID_i, Reg_i, A_i, TK_{U_i}, K_{UG} \rangle$, the smart phone of the user updates the shared secret key $K_{UG}^* = K_{UG} \oplus h(h(ID_i) \oplus h(PW_i))$ to ensure the perfect forward secrecy, and then stores in its memory $\langle TID_i, Reg_i, A_i, TK_{U_i}, K_{UG}^* \rangle$.

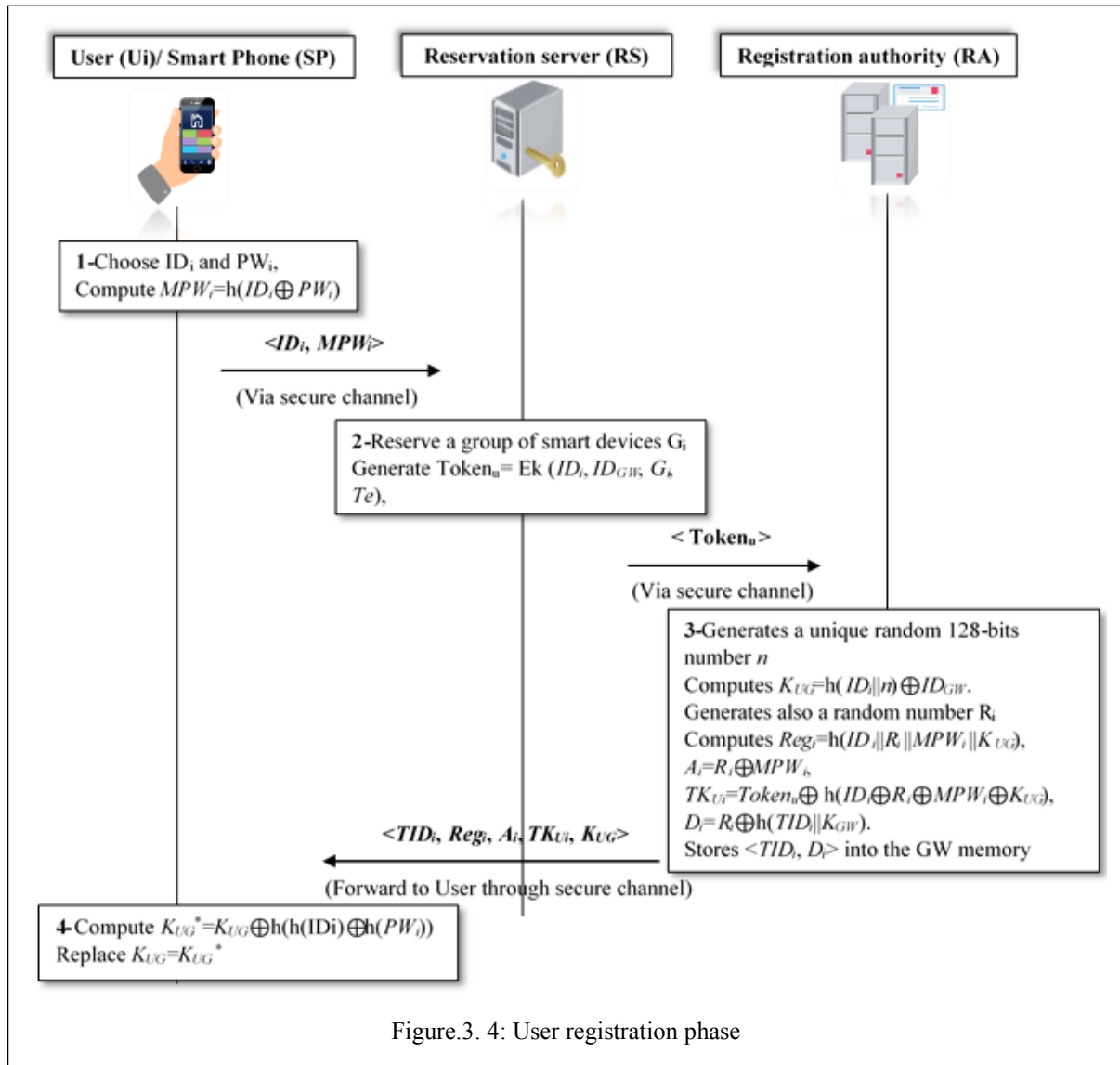


Figure.3. 4: User registration phase

3.5.3. Token Distribution Between GW and Smart Device Phase

After a successful reservation phase, the GW distributes the generated Token of the user U_i to the selected group of smart devices. This phase is presented in the Figure.3.5, and detailed as follows:

- **Step1:** $GW \rightarrow SD_j: \{D_1, D_2, t_1\}$.

The RA sends the $Token_u$ securely to the GW. Then, the GW decrypts first the token using its secret key K ; decrypts $DK(Token_u)_K = (ID_i, G_i, Te, ID_{GW})$ and retrieves all identities' smart devices ID_{SD_j} of the group G_i . Then, for each smart device SD_j , the GW generates a random number r_j and Timestamp t_1 , and forms a request to authenticate the SD_j by computing:

$$\checkmark D_1 = h(K_{SG_j} || r_j || ID_{SD_j} || t_1),$$

$$\checkmark D_2 = r_j \oplus h(K_{SG_j}).$$

Finally, the GW forwards the authentication request $\langle D_1, D_2, t_1 \rangle$ to the selected smart device SD_j .

- **Step2:** $SD_j \rightarrow GW: \{D_3, D_4, t_2\}$.

After receiving the request from GW, SD_j checks the timestamp $|t_1^* - t_1| < \Delta T$: if it matches the smart device SD_j computes the challenge $r_j^* = D_2 \oplus h(K_{SG_j})$, and the device's information $D_1^* = h(K_{SG_j} || r_j^* || ID_{SD_j} || t_1)$. If equation $D_1^* = D_1$ holds, SD_j authenticates its gateway and generates a random number s_j and timestamp t_2 , computes the following messages D_3 and D_4 to authenticate in return the gateway:

$$\checkmark D_3 = h(ID_{SD_j} || K_{SG} || r_j^* || s_j || t_2),$$

$$\checkmark D_4 = s_j \oplus h(K_{SG})$$

Then, the smart device SD_j sends its response $\langle D_3, D_4, t_2 \rangle$ to the GW. Otherwise, if $D_1^* \neq D_1$, this phase breaks immediately.

- **Step3:** $GW \rightarrow SD_j: \{F, Tx, t_3\}$.

The GW checks first the timestamp $|t_2^* - t_2| < \Delta T$: if it holds, the GW verifies the received message through computing:

$$\checkmark s_j^* = D_4 \oplus h(K_{SG_j}),$$

$$\checkmark D_3^* = h(ID_{SD_j} || K_{SG_j} || r_i || s_j^* || t_2),$$

Subsequently, the GW compares D_3^* and D_3 . If $D_3^* = D_3$ holds the device is authenticated, thus, the GW continues this phase and generates timestamp t_3 . Afterward, the GW computes a factor F to identify SD_j with the corresponding token $F = h(Token_u || K || ID_{SD_j})$, calculates the expiration time $Tx = Te \oplus h(K_{SG_j})$ and updates the shared secret key with the smart device, $K_{SG_{new}} = h(ID_{SD_j} || K_{SG_j})$. Finally, the GW sends the user token under the factor F , the time expiration Tx , and the timestamp parameter $\langle F, Tx, t_3 \rangle$ to the corresponding SD_j . Otherwise ($D_3^* \neq D_3$) the GW breakdowns the communication.

- **Step 4:** the smart device SD_j , after receiving $\langle F, Tx, t_3 \rangle$, updates the shared secret key with the GW. $K_{SG_{new}} = h(ID_{SD_j} || K_{SG_j})$ and stores the factor F and the Tx parameter.

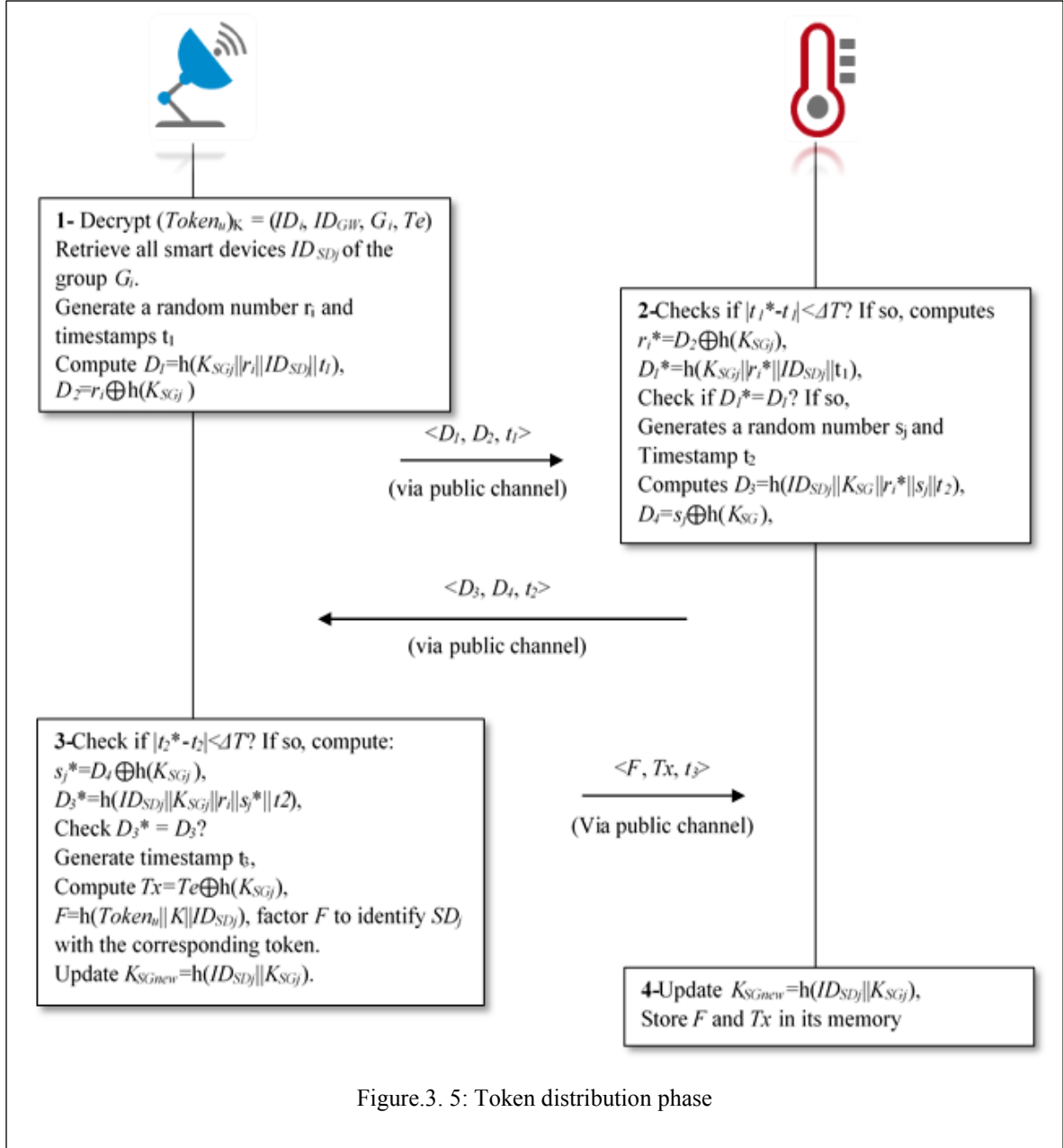


Figure.3. 5: Token distribution phase

3.5.4. Login, Authentication, and Key Agreement Phase

Once the registration process is completed, a user U_i is now ready to login into the system proceeding as follows (step 1 in Figure.3.6):

- **Step1:** $SP \rightarrow GW: \{TID_i, CID_i, CID_{SDj}, M_1, R_0, T_1\}$

U_i enters his/her ID_i and PW_i into his smart phone SP . Then, SP computes the MPW and Reg_i to verify the legitimacy of the user as follows:

- ✓ $MPW_i^* = h(ID_i \oplus PW_i)$,
- ✓ $R_i^* = A_i \oplus MPW_i^*$,
- ✓ $K_{UG} = K_{UG}^* \oplus h(h(ID_i^*) \oplus h(PW_i^*))$,
- ✓ $Reg_i^* = h(ID_i || R_i^* || MPW_i^* || K_{UG})$,

Besides, SP aborts the login request if $equation\ Reg_i^* = Reg_i$ does not hold. Otherwise, SP proceeds for further operations. At that moment, the user enters the identity ID_{SDj} of the smart device with which he/she wants to communicate; after that, the SP produces a random nonce R_l and a current timestamp T_l . Further, SP forms a login message basing on the registration information, with his identity, registration token, and the smart device's identity. In fact, SP calculates CID_i and CID_{SDj} to hide the real identities and ensure the anonymity. Moreover, it computes the message M_l to authenticate with the gateway:

- ✓ $Token_u^* = TK_{U_i} \oplus h(ID_i^* || R_i^* || MPW_i^* || K_{UG}),$
- ✓ $CID_i = ID_i \oplus h(TID_i || K_{UG} || R_i^* || T_1),$
- ✓ $R_0 = h(K_{UG} || R_i^*) \oplus R_1,$
- ✓ $CID_{SDj} = ID_{SDj} \oplus h(Token_u^* || K_{UG} || R_i^* || T_1),$
- ✓ $M_1 = h(ID_i || R_1 || Token_u^* || K_{UG} || T_1)$

Finally, the SP sends $\langle TID_i, CID_i, CID_{SDj}, M_1, R_0, T_1 \rangle$ to GW through a public channel. At this level, an authentication process begins among the user, the gateway, and the smart devices, which leads to establishing a session key between the user and the smart device. The authentication and key agreement are executed through the steps expressed below and shown in Figure.3.6:

• **Step2:** $GW \rightarrow SD_j: \{M_2, M_3, M_4, NSG_j, T_2\}$

The GW checks the legitimacy of the user once receiving the login request. Thus, the GW checks if $|T_l^* - T_l| < \Delta T$ to resist the replay attack: if so, the GW searches the temporary identity TID_i in its memory to retrieve the corresponding D_i from the couple $\langle TID_i, D_i \rangle$. At this level, GW verifies the user identity through computing the necessary information to ensure the user's legitimacy using the message M_l :

- ✓ $R_i^* = D_i \oplus h(TID_i || K),$
- ✓ $ID_i^* = CID_i \oplus h(TID_i || K_{UG} || R_i^* || T_1),$
- ✓ $R_1^* = R_0 \oplus h(R_i^* || K_{UG}),$
- ✓ $M_1^* = h(ID_i^* || R_1^* || Token_u || K_{UG} || T_1).$

Now, if equation $M_1^* = M_1$ holds, the GW assumes that the message sent by U_i is authentic; otherwise, it discontinues the protocol's operations. After that, the GW verifies the time expiration of the token, hence the GW decrypts $(Token_u)_K = (ID_i, ID_{GW}, G_i, Te)$, generates a current timestamp T_2 . If equation $Te > T_2$ does not hold, the GW dismisses this phase. Otherwise, the token has not expired and the GW verifies the SD_j 's identity chosen by the user by computing:

- ✓ $ID_{SDj} = CID_{SDj} \oplus h(Token_u || K_{UG} || R_i^* || T_1),$
- ✓ $S_j^* = h(ID_{SDj}^* || G_i || K)$

Besides, the GW checks whether $S_j^* = S_j$ holds, to ensure the validity of the requested devices, otherwise, this phase is corrupted.

After verifying the legitimacy of the user U_i and the time expiration of the token, the GW generates a random nonce R_2 to prepare the user request to the corresponding smart device. For that, the GW calculates M_2 including the user information that ensure the authentication

between the user and the smart device. In addition, GW should authenticate the smart device through computing M_3 and M_4 , and then updates the sequence number NSG_j to maintain the synchronization with the smart device:

- ✓ $M_2 = h(h(ID_i || ID_{SD_j}^* || R_1^* || R_2) || h(Token_u || K || ID_{SD_j}) || K_{SG_j} || R_2 || NSG_j || T_2),$
- ✓ $M_3 = h(ID_i || ID_{SD_j} || R_1^* || R_2) \oplus K_{SG_j},$
- ✓ $M_4 = R_2 \oplus h(K_{SG_j})$
- ✓ $NSG_j = NSG_j + 1.$

Finally, the GW forwards $\langle M_2, M_3, M_4, NSG_j, T_2 \rangle$ to SD_j through the public network.

• **Step3:** $SD_j \rightarrow GW: \{M_6, M_7, T_3\}$

Once receiving the request from the GW, the SD_j checks first if $|T_2^* - T_2| < \Delta T$, and verifies whether $1 \leq NSG_j - NSD_j \leq N$, where N is a threshold, which is set according to the specific applications requirements. If they do not hold, SD_j terminates the session. Otherwise, SD_j retrieves the nonce R_2 from the M_4 and prepares the session key through M_5 :

- ✓ $R_2^* = M_4 \oplus h(K_{SG_j}),$
- ✓ $M_5 = M_3 \oplus K_{SG_j},$
- ✓ $M_2^* = h(M_5 || h(Token_u || K || ID_{SD_j}) || K_{SG_j} || R_2^* || NSG_j - 1 || T_2).$

Afterward, if equation $M_2^* = M_2$ holds the user is authentic, the SD_j generates a random number R_3 and the current timestamp T_3 , and computes $Te = Tx \oplus h(K_{SG_j})$ to check if the token is expired. If $Te > T_3$ does not hold, this phase ends. Else, the SD_j computes the session key and calculates the response to the GW:

- ✓ $SK = h(M_5 || R_2^* || R_3 || T_3),$
- ✓ $M_6 = h(SK || R_3 || K_{SG_j} || NSG_j || T_3),$
- ✓ $M_7 = R_3 \oplus h(R_2).$

Finally, the SD_j derives a new shared secret key with the GW, $K_{SG_j} = h(K_{SG_j} || ID_{SD_j})$, updates its shared secret key $K_{SG_j}^* \leftarrow K_{SG_j}$ to ensure the perfect forward secrecy, updates the sequence number $NSD_j \leftarrow NSG_j$ to maintain the synchronization and forwards $\langle M_6, M_7, T_3 \rangle$ to the GW through the public network.

• **Step4:** $GW \rightarrow U_i: \{M_7, M_8, M_9, M_{10}, T_4\}$

After receiving $\langle M_6, M_7, T_3 \rangle$, the GW checks if $|T_3^* - T_3| < \Delta T$; if it holds, the GW retrieves the challenge of the smart device R_3 using M_7 , the session key to compute M_6 :

- ✓ $R_3^* = M_7 \oplus h(R_2),$
- ✓ $SK^* = h(h(ID_i || ID_{SD_j} || R_1^* || R_2) || R_2 || R_3^* || T_3),$
- ✓ $M_6^* = h(SK^* || R_3^* || K_{SG_j} || NSG_j || T_3),$

Besides, the GW checks whether $M_6^* = M_6$ holds to ensure the legitimacy of the smart device. If it is incorrect, the GW abandons the connection. Otherwise, the GW generates a timestamps T_4 and a new temporary identity $TID_i^* \neq TID_i$. Then, the GW calculates a response to the user U_i including the session key:

- ✓ $M_8 = R_2 \oplus h(ID_i || R_1),$
- ✓ $M_9 = h(ID_i || SK^* || R_3^* || K_{UG}),$
- ✓ $M_{10} = TID_i^* \oplus h(R_2 \oplus R_3^*),$

After, the GW updates its memory by a new-shared secret key with the user $K_{UGnew} = h(K_{UG} || ID_i)$ and with the smart device $K_{SGnewj} = h(K_{SGj} || ID_{SDj})$, which ensure the perfect forward secrecy. Finally, the GW sends the response message $\langle M_7, M_8, M_9, M_{10}, T_4 \rangle$ to the user U_i through the public channel.

- **Step5:** After receiving $\langle M_7, M_8, M_9, M_{10}, T_4 \rangle$ U_i checks the timeliness of T_4 : if equation $|T_4^* - T_4| < \Delta T$ holds, the SPs of the user retrieves the smart device and the GW challenges R_2 and R_3 from M_8, M_7 and the temporary identity using M_{10} to compute the session key and M_9 :

- ✓ $R_2^* = M_8 \oplus h(ID_i || R_1),$
- ✓ $R_3^* = M_7 \oplus h(R_2^*),$
- ✓ $TID_i^* = M_{10} \oplus h(R_2^* \oplus R_3^*),$
- ✓ $SK^* = h(h(ID_i || ID_{SDj}) || R_1 || R_2^*) || R_2^* || R_3^* || T_3),$
- ✓ $M_9^* = h(ID_i || SK^* || R_3^* || K_{UG}),$

Then, the SP's user checks whether M_9^* matches with the received M_9 . If it is correct, U_i considers that all the received message $\langle M_7, M_8, M_9, M_{10}, T_4 \rangle$ is valid and then transmits a confirmation message to the GW, which confirm a session key is established successfully. The SP now updates the old TID_i with the new TID_i^* and also updates the shared secret key with the GW. $K_{UGnew} = h(K_{UG} || ID_i)$ to preserve the perfect forward secrecy. Similarly, the GW computes a new value $D_i^* = R_i \oplus h(TID_i^* || K)$ and replaces $\langle TID_i, D_i \rangle$ with $\langle TID_i^*, D_i^* \rangle$. Otherwise, if the M_9^* is different from M_9 , the authentication fails.

3.5.5. Password Change Phase

During this phase, we offer the user the possibility to update periodically his password without the help of the GW. The required updates are explained in the following:

- **Step1:** U_i inputs ID_i and PW_i into his smart phone SPs. Then, SP calculates:
 - ✓ $MPW_i^* = h(ID_i \oplus PW_i),$
 - ✓ $R_i^* = A_i \oplus MPW_i^*,$
 - ✓ $K_{UG} = K_{UG}^* \oplus h(h(ID_i^*) \oplus h(PW_i^*)),$
 - ✓ $Reg_i^* = h(ID_i || R_i^* || MPW_i^* || K_{UG}),$

Afterward, the SP verifies whether the condition $Reg_i^* = Reg_i$ holds. If it is not valid, SP ends the password change phase; otherwise, the SP retrieves the corresponding Token $Token_u^* = TK_{U_i} \oplus h(ID_i \oplus R_i \oplus MPW_i \oplus K_{UG})$ and proceeds for further computations.

- **Step2:** After verifying the legitimacy of U_i , the SP requests the user U_i to enter a new password.

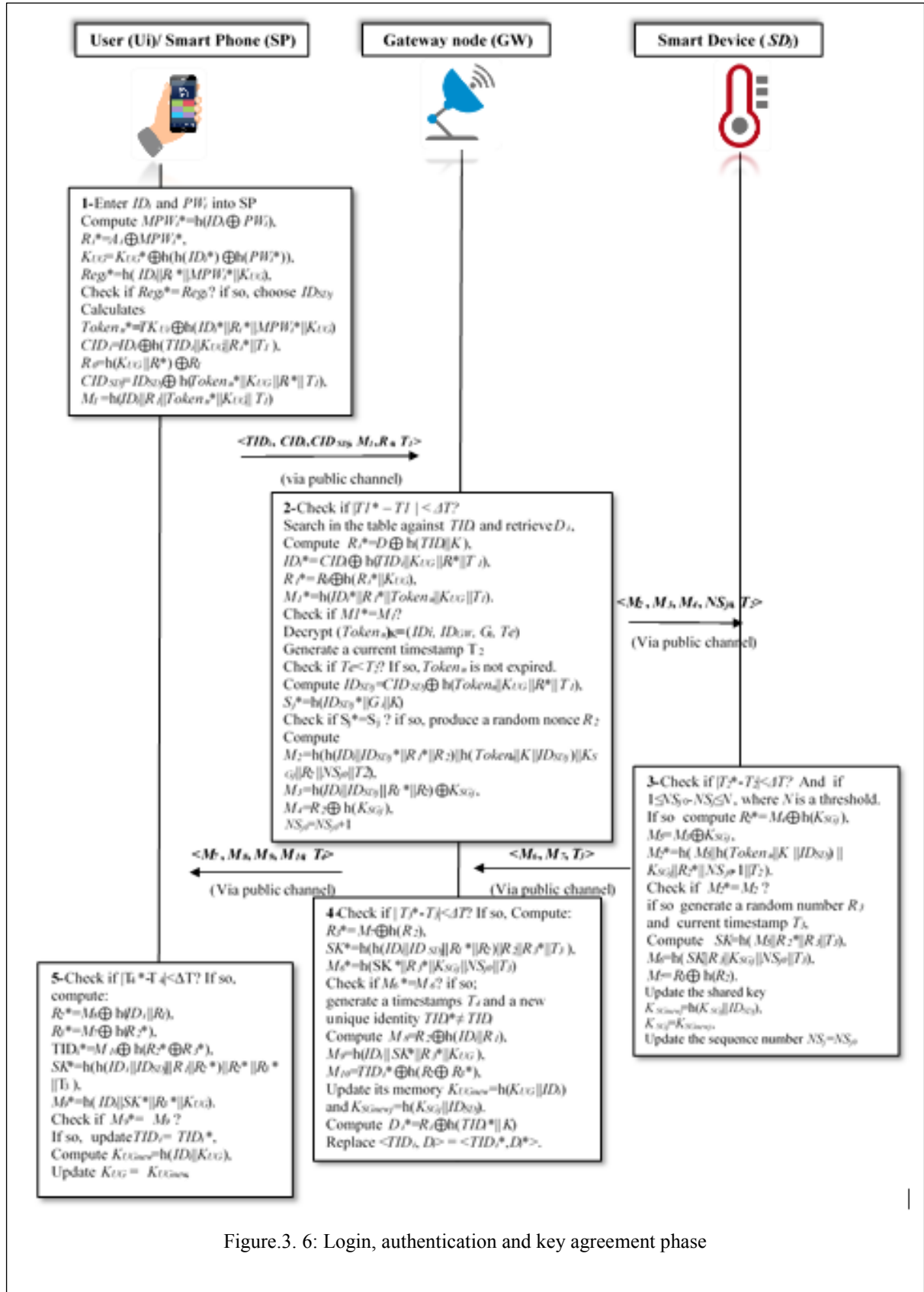


Figure.3. 6: Login, authentication and key agreement phase

- **Step3:** The user U_i inputs a new password PW_{i-new} , then the SP calculates:
 ✓ $MPW_{i-new} = h(ID_i \oplus PW_{i-new})$,

- ✓ $Reg_{i-new} = h(ID_i || R_i^* || MPW_{i-new} || K_{UG}),$
- ✓ $A_{i-new} = R_i^* \oplus MPW_{i-new},$
- ✓ $TK_{U_{i-new}} = Token_u^* \oplus h(ID_i \oplus R_i \oplus MPW_{i-new} \oplus K_{UG}),$
- ✓ $K_{UG-new}^* = K_{UG} \oplus h(h(ID_i) \oplus h(PW_{i-new})).$

Finally, the SP's user drops the existing user information related to the old password $\langle Reg_i, A_i, TK_{U_i}, K_{UG}^* \rangle$ and stores $\langle Reg_{i-new}, A_{i-new}, TK_{U_{i-new}}, K_{UG-new}^* \rangle$ into its memory. Thus, the U_i can easily change the password without involvement of the GW.

3.6. Security Evaluation

In this section, we evaluate the security of the proposed scheme *TBLUA*, and show that it ensures many security properties and withstands most popular security attacks. For that, we analyze *TBLUA* security and provide formal verification using the AVISPA tool [145].

3.6.1. Security Analysis

In this section, we present a discussion about the main security analysis and the proof of properties that our proposed *TBLUA* protocol ensures.

(i) Anonymity of the User and the Smart Device

An outsider person may try to guess the identities ID_i and ID_{SDj} . For that, we suppose that an adversary A traps all exchanged messages between user, GW and SD_j . After the login phase, the user sends the login message, and we assume that A traps $\langle TID_i, CID_i, CID_{SDj}, M_1, R_0, T_1 \rangle$. He cannot compute ID_i and ID_{SDj} from CID_i , M_1 and CID_{SDj} without knowing R_i , $\langle Token_u, R_1, K_{UG} \rangle$ and $\langle Token_u, R_i, K_{UG} \rangle$, respectively. During the authentication and key agreement phase, the GW hides ID_i and ID_{SDj} to compute the message $\langle M_2, M_3, M_4, T_3 \rangle$. Consequently, the extraction of ID_i and ID_{SDj} using these information is computationally infeasible owing to property of hash function $h(\cdot)$. Furthermore, the identity ID_i of U_i and ID_{SDj} of smart device are not directly involved in $\langle M_6, M_7 \rangle$, A cannot derive ID_i and ID_{SDj} when A traps $\langle M_6, M_7 \rangle$ during the protocol run. In addition, the GW hides ID_i and ID_{SDj} in $\langle M_7, M_8, M_9, M_{10}, T_4 \rangle$, where the identities ID_i and ID_{SDj} are protected by $h(\cdot)$, thus the extraction is computationally infeasible and the probability of guessing is negligible. We can claim that A is unable to break the anonymity of the proposed protocol using public messages.

(ii) Mutual Authentication Property

In client-server communication model, mutual authentication is an extremely essential security aspect of any authentication protocol. In Step 2 of the login and authentication phase (step2 in Figure.3.5), the GW first ensures the legitimacy of the user U_i before starting further computation. In Step3, the SD_j checks if the condition $M_2^* = M_2$ to authenticate the user U_i and the GW. This latter checks the legitimacy of the smart device through verifying the condition $M_6^* = M_6$ in Step4. Finally, after receiving $\langle M_7, M_8, M_9, M_{10}, T_4 \rangle$ the U_i authenticates GW

and SD_j in Step 5 by checking the received M_9 . Hence, the authentication is ensured mutually between all the participants during the authentication phase in all the exchanged messages.

(iii) Perfect Forward Secrecy

The perfect forward secrecy (PFS) feature prevents the leakage of any prior session key even if the secret key is revealed. In the proposed scheme, let suppose the adversary has obtained the long-term shared secret keys between both the (user, gateway) and the (smart device, gateway) that are K_{UG} and K_{SG} , respectively. After each transaction, both the user and the GW update the keys K_{UG} ; $K_{UG}^* = h(K_{UG} || ID_i)$, and the smart device and the GW update the shared secret keys K_{SG} ; $K_{SG}^* = h(K_{SG} || ID_{SD_j})$ by one-way hash function. Under this assumption, if the adversary compromises a smart device or the smart phone, he can manage only K_{SG}^* and K_{UG}^* . Since the hash function is one way, the adversary cannot obtain K_{UG} and K_{SG} from K_{UG}^* and K_{SG}^* . Therefore, as the identities of the user and the smart device are hidden the adversary cannot obtain the future shared keys. Thus, the perfect forward secrecy is retained in our proposed protocol *TBLUA*.

(iv) Mobile Device Stolen Attack

In this attack, an adversary A attempts to extract confidential information and then tries to misuse this information. We assume that A has got the mobile device of a legal user U_i and extracted all the information $\langle TID_i, Reg_i, A_i, TK_{U_i}, K_{UG}^* \rangle$ with the help of the power analysis attack [143]. Note that Reg_i is protected by one-way hash function $h(\cdot)$. Therefore, A is not capable to extort any information from Reg_i owing to the one-way property of $h(\cdot)$. The probability of guessing ID_i and PW_i using Reg_i is negligible. In addition, A is unable to compute MPW_i without knowing R_i . The confidential information in the mobile device is $Token_u$, which is used to compute TK_{U_i} , but the adversary A cannot compute $Token_u$ without knowing $\langle ID_i, R_i, MPW_i, K_{UG_i} \rangle$. In addition, computing R_i is not feasible without knowing the secret key of the GW , which makes the proposed protocol withstand this attack.

(v) User Impersonation Attack

In an impersonation attack, an adversary A makes efforts to impersonate the identity of a legitimate user U_i . In our scheme, to forge a user, the adversary A must generate a valid value $\langle TID_i, CID_i, CID_{SD_j}, M_1, R'_0, T'_1 \rangle$ by incorporating a new random number and timestamp to be accepted by the GW. However, to compute CID_i , CID_{SD_j} , and M_1 , he needs the correct identities ID_i , ID_{SD_j} , the secret shared key K_{UG} , and $Token_u$. Besides, the identities and the token are not transmitted in the public channel clearly, and the shared secret key K_{UG} is updated after each transaction. Therefore, our proposed scheme can resist the user impersonation attack.

(vi) Offline Password Guessing Attack

Offline password guessing attack is the most damaging threat, where an adversary chooses a password from a dictionary to guess the sensitive information of the user. We suppose that A has got the mobile device of U_i and extracted the stored information in its memory

$\langle TID_i, Reg_i, A_i, TK_{U_i}, K_{UG}^* \rangle$. In the proposed protocol, PW_i is used to compute the masque password $MPW_i = h(ID_i \oplus PW_i)$. Otherwise, using the mobile device information, A may guess PW_i from $(Reg_i = h(ID_i || R_i || MPW_i || K_{UG}))$, and $A_i = R_i \oplus MPW_i$; however, the probability to get the true PW_i is negligible in polynomial time. Therefore, A cannot guess the password PW_i , and our scheme resist the offline password guessing attack.

(vii) Desynchronization Attack

Our scheme employs different shared keys between the user, GW and smart devices and one-time hash function techniques to provide PFS. Furthermore, an additional synchronization method is essential to maintain the consistency of several one-time values between the GW and the smart device. Hence, we use the sequence number to resist the desynchronization attack. If an adversary A blocks the message flow between the smart device and the GW, the synchronization will be lost, and the hash chain values of the two participants will not match each other. For that, we use a sequence number NSG_j and NSD_j to record the updated number of hash chain, where NSG_j is the sequence number of the GW side. Thus, after the GW sends the message flow, the value of hash chain NSG_j in GW side must be updated. Besides, the SD_j receives the message $\langle M_2, M_3, M_4, NSG_j \rangle$ and synchronizes the one-time hash chain value through performing the operation below $NSG_j - NSD_j$ time hash functions.

(viii) Physical Node Capture Attack

Capturing node attack enables an adversary to extract sensitive information stored in those captured smart devices to compromise a secure communication among non-compromised smart devices in the network. Let's assume that SD_j is a compromised smart device, where an adversary A can extract the secret key K_{SGj} and even the session key SK , established between the compromised node SD_j and a legitimate user U_i . In the proposed authentication protocol, we use distinct shared secret keys between the gateway and each smart device SD_i . Thus, SD_i establishes a distinct session key with U_i , which is different from all other session keys in the network. Therefore, all non-compromised smart devices can still communicate with the legitimate user U_i with higher secrecy. Hence, our proposed scheme withstands smart device capture attacks.

(ix) Node Impersonation Attack

An impersonation attack means that a malicious attacker may try to masquerade as a valid smart device SD_j . If an attacker wants to impersonate as a smart device SD_j node, he/she will need to forge the message $\langle M_6, M_7, T_3 \rangle$, sent to the GW. For that, the adversary needs to know both identities ID_i and ID_{SDj} to calculate the session key embedded in M_6 and the corresponding shared secret key K_{SG} of SD_j . However, it is computationally hard for adversary A to get the true value as its information is protected by the one-way hash function. Hence, we can declare that our protocol resists the node impersonation attack.

(x) Token Impersonation Attack

In this attack the adversary A tries to create a new token and duplicates an existing token, where $Token_u = Ek(ID_i, ID_{GW}, G_i, Te)$. If adversary A intercepts the messages exchanged between user, GW and SD_j node, he cannot guess both identities ID_i and ID_{GW} , which are protected with the one-way hash function $h(\cdot)$. In addition, A cannot create a token without prior knowledge of the GW's long-term secret key K , which confirm that our scheme is resistant to this attack.

(xi) Token Modification Attack

An attacker may generate a fake token or modify the token contents (such as the expiration time or the list of the identity of SD_j) of an existing token. Thus, a malicious client may modify the assertion to gain access to information that they should not be able to view. It can be observed that the token is protected with a symmetric cipher function using the secret key of GW. Hence, A cannot either modify or delete any information of the token. Therefore, our scheme resists against this attack.

(xii) Token Replay Attack

Under this attack, an attacker attempts to use an expired token that has already been used with a user in the past to get access in the future. However, A cannot reuse a token, as each token is characterized by an expiration time parameter, and the use of this token is limited for a prefixed time.

3.6.2. Formal Verification Using AVISPA Tool

We described our protocol using the AVISPA's High-Level Protocol Specification Language (hlpsl) [145]. The AVISPA tool allows the designers of security protocols to detect potential attacks and verify if their protocols meet the attended security services. In our protocol, we defined three roles in the HLPSL language, named: the gateway (GW), the smart device (SD), and the user (U) roles, which correspond to the different agents of our system. Specifically, we modeled a channel (dy) in our specifications based on Dolev-Yao intruder model, which means that all the exchanged messages between all the agents (GW, SD , U) are intercepted by the intruder. This last can analyze, modify the intercepted messages, or eventually decrypts them if he knows the required keys. In our protocol, we examine some security properties, which are specified in the goal section of HLPSL specifications, through verifying the following properties:

- ❖ GW authenticates U on $h(ID_i || R_1 || Token_u^* || K_{UG} || T_1)$: U generates a nonce value R_1 and sends the challenge $R_0 = h(K_{UG} || R_i^*) \oplus R_1$. If GW is able to construct $h(ID_i || R_1 || Token_u^* || K_{UG} || T_1)$ from the challenge R_0 , GW authenticates U .
- ❖ SD authenticates GW on $h(M_5 || h(Token_u || K || ID_{SD_j}) || K_{SG_j} || R_2^* || NS_{j0} - 1 || T_2)$: GW generates a nonce value R_2 and sends the challenge $M_4 = R_2 \oplus h(K_{SG_j})$. If the SD is able to construct $h(M_5 || h(Token_u || K || ID_{SD_j}) || K_{SG_j} || R_2^* || NS_{j0} - 1 || T_2)$ from the challenge M_4 , SD authenticates GW.

- ❖ GW authenticates SD on $h(SK||R_3||K_{SG_j}||NS_{j0}||T_3)$: SD generates a nonce value R_3 and sends the challenge $M_7 = R_3 \oplus h(R_2)$. If GW is able to construct $h(SK||R_3||K_{SG_j}||NS_{j0}||T_3)$, from the challenge M_7 , GW authenticates SD.
- ❖ U authenticates SD on $h(ID_i||SK^*||R_3^*||K_{UG})$: SD generates a nonce value R_3 and sends the challenge $M_7 = R_3 \oplus h(R_2)$. If U is able to construct $h(ID_i||SK^*||R_3^*||K_{UG})$, from the challenge M_7 , U authenticates SD.

We performed the test through AVISPA tools using the On-the-fly Model-Checker (OFMC) that executes protocol falsification and bounded verification by exploring the transition system. Besides, the backend OFMC ensures an automatic verification of security properties. We can see clearly in figure 3.7 the obtained results on OFMC that prove the security of our *TBLUA* protocol.

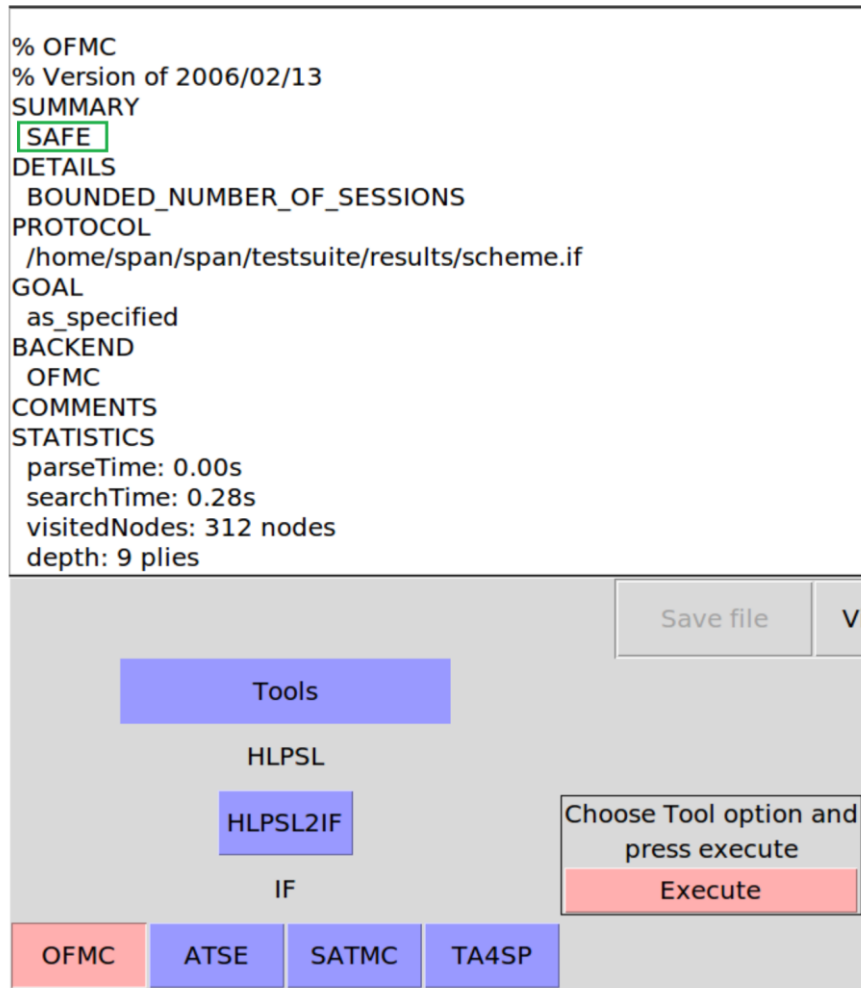


Figure.3. 7: Results reported by the OFMC backend

3.7. Performance Analysis

In this section, we analyze the performance of our scheme *TBLUA* regarding the functionality features, communication, and computation costs. In fact, we compare our proposed scheme with four prior related works [58][136][137][142]. Furthermore, we only concentrate on comparing communication and computation costs during login and authentication phases since the registration phase, token distribution phase, and password update phase are not used frequently.

3.7.1. Functionality Comparison

In this subsection, we present a comparison of the functionality features between the existing schemes and the proposed scheme in Table 3.3. We consider many security features such as mutual authentication, user anonymity, offline password guessing, and perfect forward secrecy, etc. In addition, we present many possible attacks for such an environment, especially we consider token attacks, where an attacker may abuse the token to compromise a communication. As shown in table 3.3, our proposed scheme can resist against various kinds of known attacks and fulfill the desirable security features particularly the perfect forward secrecy. Thus, the proposed scheme is more secure compared to other schemes.

Table 3. 3: Functionality Features Comparison

Properties	[16]	[18]	[23]	[5]	<i>TBLUA</i>
Mutual Authentication	-	+	+	+	+
Key agreement	+	+	+	+	+
Intractability	-	-	+	+	+
User anonymity	-	-	+	+	+
SD. anonymity	-	-	+	+	+
Off line PW guessing	-	-	+	+	+
User impersonation	-	-	+	+	+
GW impersonation	-	-	+	+	+
SD impersonation	-	-	+	+	+
Privileged-insider	-	-	+	+	+
Perfect Forward Secrecy	-	+	-	-	+
Replay attack	-	-	+	+	+
Man-in-the-middle	-	-	+	+	+
Stolen verifier	+	-	+	+	+
De-synchronization	-	-	-	-	+
GW bypassing	-	-	+	+	+
Node capture	-	-	+	+	+
Token impersonation	N/A	N/A	N/A	N/A	+
Token replay	N/A	N/A	N/A	N/A	+
Token modification	N/A	N/A	N/A	N/A	+

Note: N/A: Not Aplicable, (+): resists, (-): does not resist

3.7.2. Computation Costs Comparison

In the computation comparison, we study the evaluation of our proposed scheme in terms of the number of executions of cryptographic operations such as encryption, decryption, and hash functions with respect to the number of performed actions. We consider the notations $T_h=0.5\text{ms}$ to be the time for one hashing operation (usage of SHA-256 hash function). Furthermore, $T_{Enc}=T_{Dec}=8.7\text{ms}$ be respectively the time for one encryption/decryption using symmetric cryptography operations, and $T_{ECC}=T_{FE}=63.075\text{ms}$ represent the time for one elliptic curve cryptography and one fuzzy extraction operation [128][58] respectively. Besides, we omit XOR operation due to its negligible computational cost. In Table.3.4, we provide computation cost separately for user, gateway node, and smart device during the login and authentication phase.

Table 3. 4: Computation Costs Comparison

Scheme	User	GW	Smart Device
[136]	$7T_h = 3.5\text{ms}$	$5T_h = 2.5\text{ms}$	$7T_h = 3.5\text{ms}$
[137]	$2T_{ECC}+7T_h = 129,6\text{ms}$	$9T_h=4,5\text{ms}$	$2T_{ECC}+5T_h = 128,6\text{ms}$
[142]	$7T_h+T_{Dec}+T_{Enc} = 20\text{ms}$	$11T_h+2*T_{Dec}+2* T_{Enc}= 40.3\text{ms}$	$4T_h+T_{Dec}+T_{Enc}= 19.4\text{ms}$
[58]	$T_{FE}+13T_h+T_{Dec}+ T_{Enc}= 87.0\text{ms}$	$5T_h+2T_{Dec}+2T_{Enc}= 37.3\text{ms}$	$4T_h+T_{Dec}+T_{Enc}= 19.4\text{ms}$
TBLUA	$16T_h = 8\text{ms}$	$19T_h+T_{Dec} = 18.2\text{ms}$	$7T_h = 3.5\text{ms}$

For the smart device and the user, we notice that our protocol minimizes the number of encryption/decryption operations against increasing the number of hash computations. However, the cost of a hash function is negligible compared to the high cost of encryption/decryption operations. Indeed, our scheme *TBLUA* offers a lower computation cost compared to the other schemes, which proves that a *TBLUA* is a lightweight solution. Figure.3.8 plots the computation costs of our scheme compared to existing methods [58][136][137][142] presented in Table.3.4. Actually, the computation cost on the smart device presents less value, which is explained by using the hash function operation to achieve the authentication process. Besides, the cost on the user's smartphone and the gateway is less than the cost in [58][137][142] schemes but higher than in [136] scheme. In fact, as shown in the table.3.3, we can see that [136] fulfill only key agreement security feature, which explains that their solution is not secure. Finally, we confirm that *TBLUA* presents a better computation cost with respect to the security issues in such an environment.

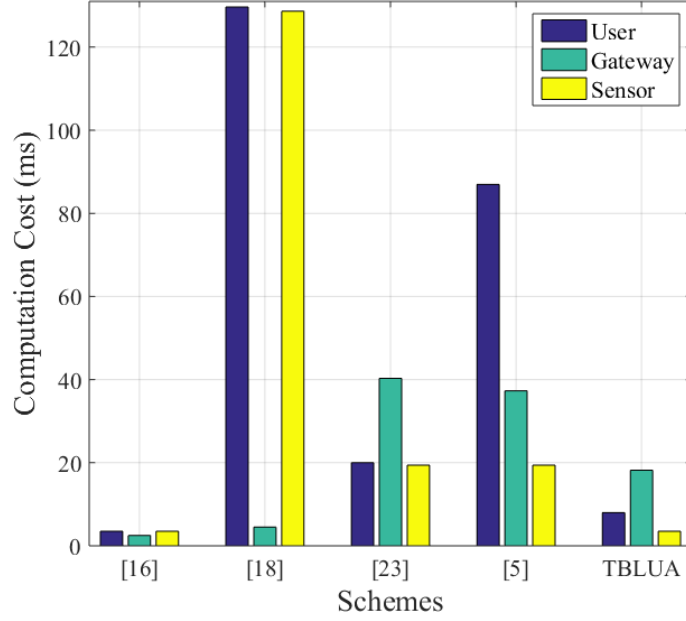


Figure.3. 8: Computation costs

3.7.3. Communication Costs Comparison

In the communication comparison, we evaluate our scheme in terms of the number of messages exchanged during the login and authentication phase execution. We assume 160 bits as the length of the user identity and 32 bits as the length of the smart device's identity. Furthermore, we consider the size of the challenge equal to 128 bits, while the timestamp is equal to 32 bits, and the size of the sequence number is set to 64 bits. In addition, the length of each hash value is set to 160 bits (i.e., if SHA-1 hashing algorithm is applied [144]), and the size of symmetric encryption/decryption block size is set to 128 bits (i.e., if we use AES-128 algorithm [141]). For elliptic curve cryptography (ECC) operations, we assume each ECC value's length is equal to 160-bit.

During the authentication phase, the user in our scheme needs only to transmit the request of authentication below: $\{TID_i, CID_i, CID_{SD_j}, M_1, R_0, T_1\}$. In fact, the communication cost can be calculated as follow:

$$user_{comm} = size(TID_i + CID_i + CID_{SD_j} + M_1 + R_0 + T_1)$$

$$user_{comm} = 160 + 160 + 32 + 160 + 128 + 32 = 672 \text{ bits} = 84 \text{ Bytes}$$

In addition, the smart device should only response to the authentication request to the gateway $\{M_6, M_7, T_3\}$, where $M_6 = h(SK || R_3 || K_{SG_j} || NS_{j0} || T_3)$, and $M_7 = R_3 \oplus h(R_2)$. Hence, we can conclude the communication's cost of the smart device is:

$$smart_device_{comm} = size(M_6 + M_7 + T_3)$$

$$smart_device_{comm} = 160 + 160 + 32 = 352 \text{ bits} = 44 \text{ Bytes}$$

Finally, the gateway, which is the relay between the user and the smart device, transmits these messages during the authentication phase: $GW \rightarrow SD_j: \{M_2, M_3, M_4, NS_{j0}, T_2\}$, and $GW \rightarrow U_i: \{M_7, M_8, M_9, M_{10}, T_4\}$. Thus, the GW communication cost is calculated as follow:

$$GW_{comm} = size (M_2 + M_3 + M_4 + NS_{j0} + T_2 + M_7 + M_8 + M_9 + M_{10} + T_4)$$

$$GW_{comm} = 7 \times 160 + 2 \times 32 + 64 = 1248 \text{ bits} = 156 \text{ Bytes}$$

The Figure.3.9. plots the communication cost comparison with benchmarking schemes. Indeed, we can see that our scheme presents an enhancement in the communication cost compared to [136] [137] [58]. Otherwise, *TBLUA* presents a desirable communication costs compared to [142] scheme. In fact, as presented in the Table.3.3, [142] scheme does not ensure the perfect forward secrecy and cannot resist the desynchronization attack. Whereas, *TBLUA* achieves the perfect forward secrecy through updating the shared secret keys after each transaction and transmitting a sequence number that resists to desynchronization attack, which explain the communication cost of *TBLUA* comparing to the [142] scheme. We conclude that our proposed scheme offers a tradeoff between the security and the communication costs.

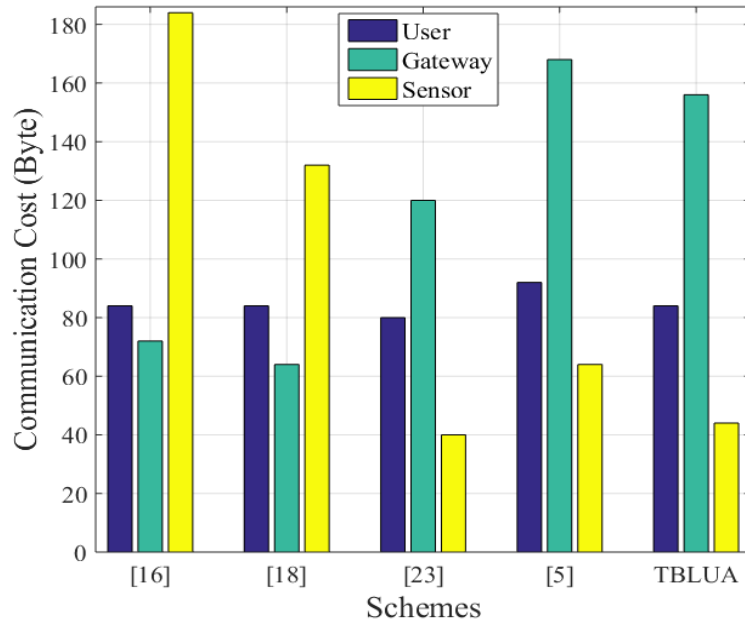


Figure.3. 9: Communication costs

3.8. Proof of Concept Within Smart Hotel Use Case

The smart hotel is an intelligent system with a range of IoT objects working together to make the guests' stay more comfortable, lower the energy consumption, and help the staff and management with their tasks. The high density of guests with a regular change of reservation status might be problematic in such an intelligent environment. Thus, securing communication in such a smart system is primordial. In what follow, we present the architecture of the smart hotel use case used in the context of PARFAIT project [7]. Then, we study the possible vulnerabilities and the risk in the corresponding use case. Finally, we describe the designed prototype and the different performed simulation

3.8.1. Smart Hotel Use Case

To secure a smart hotel, we design and create an architecture of an intelligent control system that can be used in the hotel environment in the context of the PARFAIT project, such as shown

in Figure.3.10. Our architecture is composed of a reservation smart hotel server responsible for generating the token of authentication for guests based on the period of accommodation and the availability of rooms in the hotel, and for the employers basing on their tasks. The reservation server communicates with the registration authority server to sign the users' initial enrollment requests with the smart hotel. These servers may be locally or maintained in the cloud. At this level, the registration authority, after receiving requests, registers the user under the smart hotel system and issues certificated tokens to the corresponding users.

The smart hotel comprises an extensive range of IoT objects that communicates via the Internet through the ZigBee gateway. This latter is maintained to connect the IoT objects to the external world network. In particular, the smart lock receives the token of identification through the gateway node to access the legitimate user during the accommodation. All the necessary information about the guest is loaded into the system from the registration phase previously filled. Consequently, the room environment, such as temperature or lighting, is prepared before the guest's arrival based on his/her preferences. At the guest's arrival, he/she could unlock the doors using the sensors on his/her smartphone, such as NFC. We can confirm that the smart hotel is also beneficial for the hotel staff as the information about the guest's preferences can save time if they have special requests.

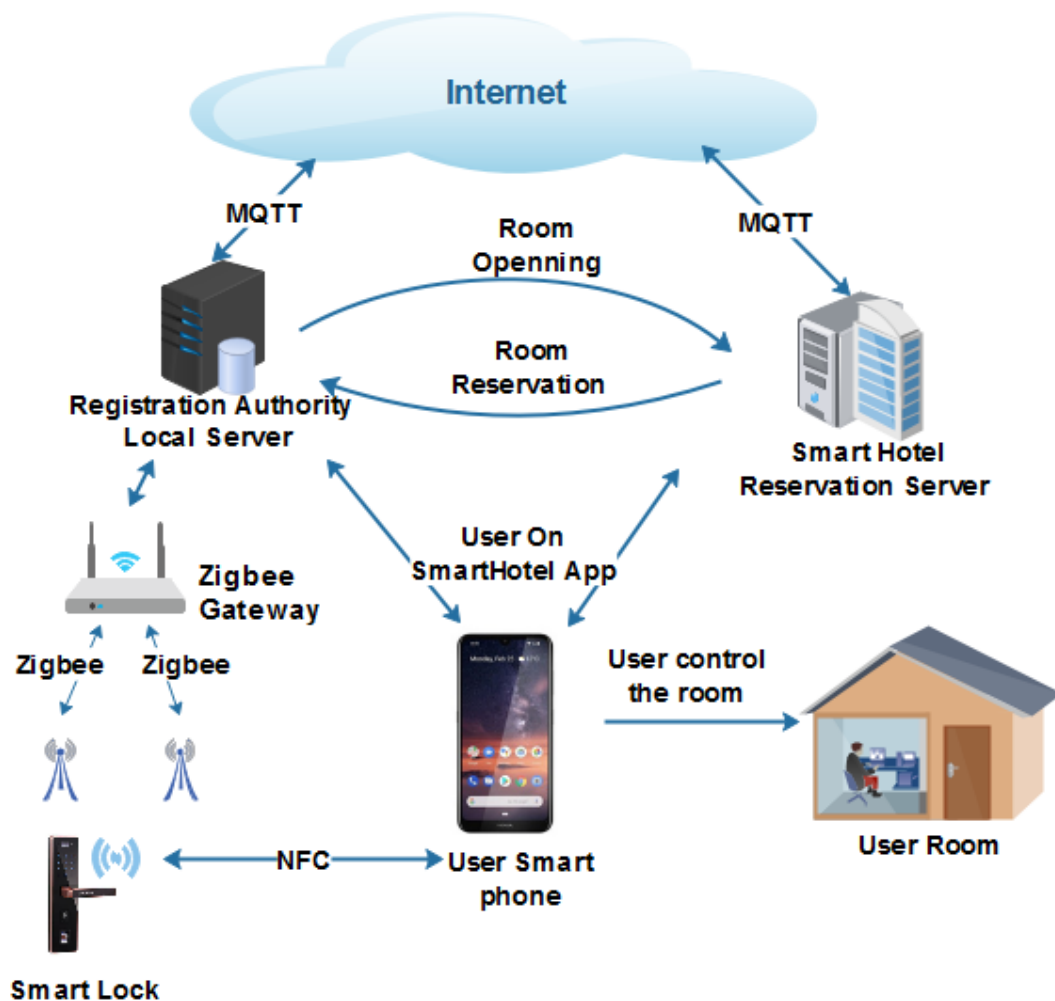


Figure.3. 10: Smart Hotel architecture

3.8.2. Risk Management and Vulnerability

Smart hotel based on IoT network contains several critical components required for the proper functioning of guest and hotel staff, prone to several vulnerabilities and threats. These vulnerabilities allow a malicious entity to attack the IoT devices and threaten security goals. It is essential to understand the vulnerabilities and possible attacks at the different communication stack layers in this context. Table 3.5 shows the principal vulnerabilities that can be exploited in the smart hotel environment reporting different attacks that menace the smart hotel's main different components.

Table 3. 5: Vulnerability analysis of a smart hotel

Vulnerability exploited	Attack \ threat	Consequence(s)
Communication network		
Vulnerabilities in the initial handshake between a user's smartphone and the smart lock.	Man-in-the-middle attack	The adversary can eavesdrop on the communication and inject fake messages.
Weak link-layer authentication and lack of anti-replay protection	Denial of Service (DoS) attack	Smart hotel unavailability and connection interruption
Lack of network access control.	Eavesdropping attack	The intercepted messages may contain sensitive data information related to user accommodation.
Weak authentication and anti-replay protection.	Spoofing attack	The adversary can transmit fabricated data to a smart lock from a fake source.
Authentication packets shortage protection	Data tampering	The adversary can delete or replace part or all of eavesdropped information.
Weak anti-replay protection.	Replay attack	The adversary could reuse the guest request to get access to the room
Sensitive traffic to identify the session token is not protected.	Hijacking attack	The adversary can use the token to make a request as a user
Weak authentication process: manipulation of unknown data.	Desynchronization attack	Interrupting communication between guest and smart lock to cause a sequence of retransmissions
Mobile application		
Insufficient knowledge and lack of awareness.	Phishing attacks	An adversary can gain access to guest's accommodation information using phishing mail techniques such as fraudulent notification that contains malware.
SLack of encryption, and unprotected communication network.	Sniffing attack	An adversary could get guest personal data such as his/her name, identity, and password.

Misconfigure the smart hotel's application on guest's smart phone.	Software attacks (malware, viruses, worms, etc.)	Software configuration updates and changes in guest monitoring devices can lead to a system malfunctioning.
Limited resources	Data flooding attack	Exhausted memory resources and the guest could not make a request to access the room.
Weak authentication and access control mechanisms.	Impersonation attack	The adversary masquerading as a legitimate guest can control and access the smart lock.
Server		
Insufficient knowledge and lack of awareness.	Social engineering attack	An adversary can gain access to guest's accommodation information using social engineering techniques that contains malware.
Weak application and network layer security	Denial of Service (DoS) attack	Guest cannot access to their accommodation information and smart hotel reservation server cannot perform any operation.
Manufacturing fault. - Unprotected interfaces. - Weak application and network layer security.	Hardware failure	The guest cannot access to their reservation information, and all the smart hotel services
Smart lock		
Insufficient cryptography of authentication factor	Brute force attack	An adversary could get the guest password and then get access and control to the guest room
Limited resources	Data flooding attack	Exhausted memory resources and the lock could not respond to the guest request

3.8.3. Design of a Smart Lock Prototype

This section illustrates the implementation of the smart lock prototype designed to authenticate users in the context of the PARFAIT project [7]. In fact, our testbed, as shown in Figure 3.11, contains (A) the smartphone, (B) the server, and (C) the smart lock; the smart lock comprises two communicating interfaces: an NFC module to communicate with the smartphone and a ZigBee module to communicate with the server. Besides, the server is composed of two main parts; the first one is connected to the Internet responsible for interconnecting smartphones to our system, while the second part is joined through the ZigBee to the smart lock.

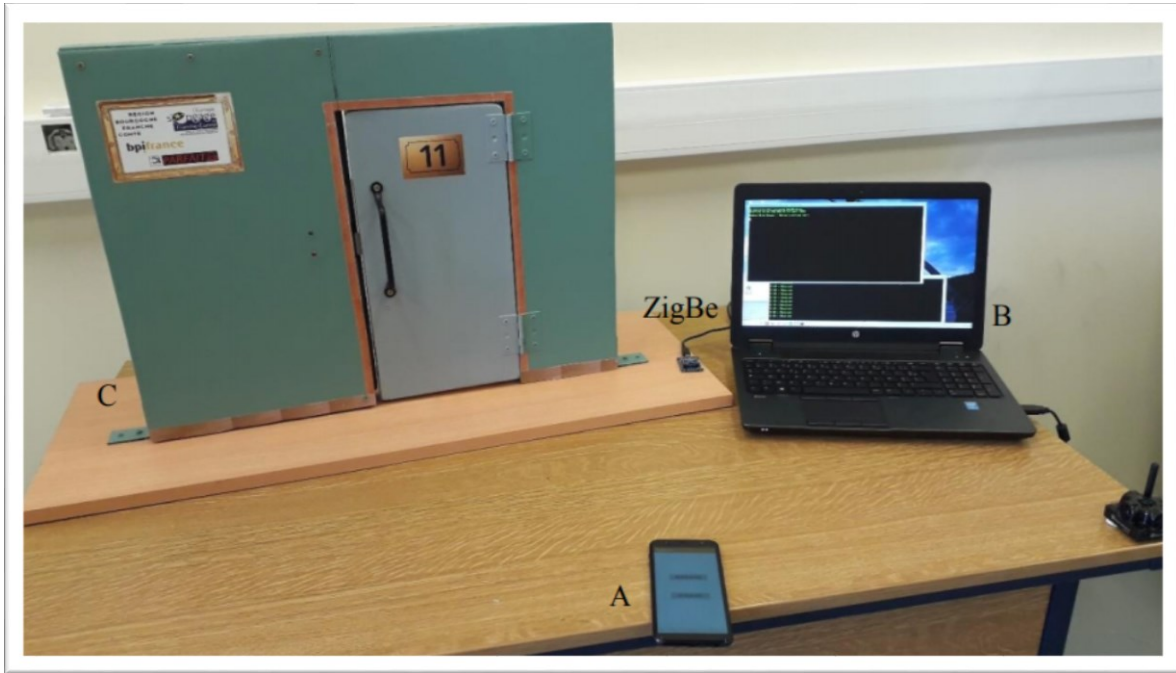


Figure.3. 11: Smart Hotel prototype

The smart hotel use case involves three main phases detailed hereafter, including the (i) reservation phase, (ii) token generation and distribution, and (iii) user access phase.

- **The reservation phase:**

The user needs to connect to the smart hotel application by creating an account with his/her name and password. On the reservation application page, the user fills in the necessary information (*name_identity*, *start_date*, *depart_date*, *room_number*) to make a room reservation. All filled information are organized to prepare a request to the server.

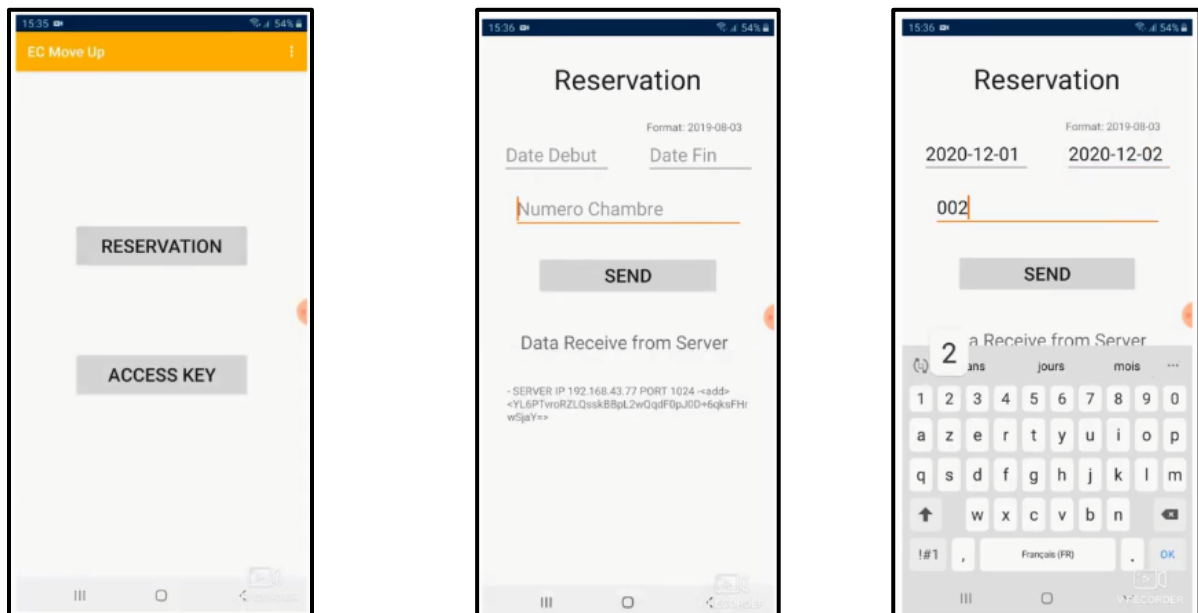
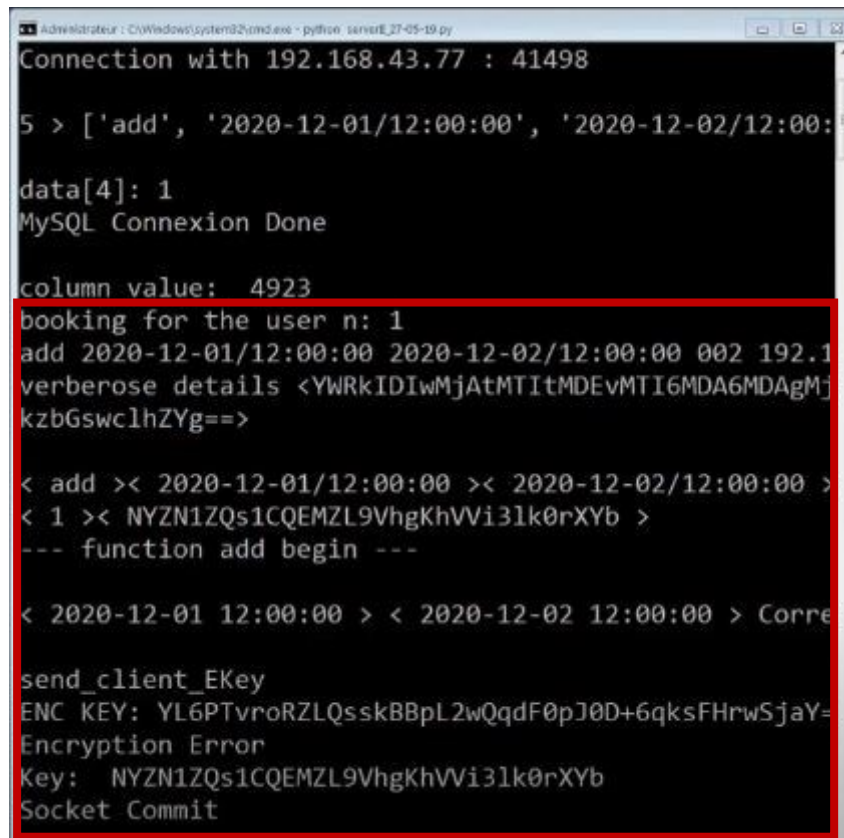


Figure.3. 12 : Reservation phase

- **The token generation and distribution phase:**

Once the server receives the smartphone's reservation request, it starts to process token generation according to the request's validity. Indeed, the server verifies the dates of accommodations and their availability in the smart hotel. At this level, if the dates are valid, the server proceeds to generate a software token of identification based on AES (Advanced Encryption System), using the user identity, the smart lock identity, and the period of the accommodation, as shown in Figure 3.13:

$$Token = Encry(ID_i, ID_{SL}, Te), Te \text{ is the period of the accommodation}$$



```

C:\Windows\system32\cmd.exe - python serverE_27-05-19.py
Connection with 192.168.43.77 : 41498

5 > ['add', '2020-12-01/12:00:00', '2020-12-02/12:00:00', '1', '192.168.43.77']
data[4]: 1
MySQL Connexion Done
column value: 4923
booking for the user n: 1
add 2020-12-01/12:00:00 2020-12-02/12:00:00 002 192.168.43.77
verbose details <YWRkIDIwMjAtMTItMDEvMTI6MDA6MDAgMjYzZGswclhZYg==>

< add >< 2020-12-01/12:00:00 >< 2020-12-02/12:00:00 >
< 1 >< NYZN1ZQs1CQEMZL9VhgKhVVi3lk0rXYb >
--- function add begin ---

< 2020-12-01 12:00:00 > < 2020-12-02 12:00:00 > Corre

send_client_EKey
ENC KEY: YL6PTvroRZLQsskBBpL2wQqdf0pJ0D+6qksFHRwSjaY=
Encryption Error
Key: NYZN1ZQs1CQEMZL9VhgKhVVi3lk0rXYb
Socket Commit
  
```

Figure.3. 13: Token generation

After successfully generating the token, the server stores the user credentials in its database MySQL and transmits the generated token securely to the user through the Internet with a PIN access code. Besides, the server transfers the token to the corresponding smart lock through the ZigBee interface. This data transmission involves different random challenges to ensure the smart lock's legitimacy, as mentioned in section 3.5.3.

- **The User Access phase:**

Throughout this phase, a lightweight authentication is achieved between the user and the smart lock, as shown in section 3.5.4, and two scenarios are possible: authorized access or a denied access.

- **Authorized access:** through using the smart hotel application, a legitimate user fills the access page with the PIN code and then, after activating the NFC shield, approaches the smartphone to the smart lock. At this moment, an authentication request is sent to the smart lock with the valid token. After receiving the request, the smart lock verifies the received access token's legitimacy and expiration time. Once the data validation is completed successfully, a session key is established between the user and the smart lock for secure access, and the smart lock is opened as figured in Figure. 3.14:

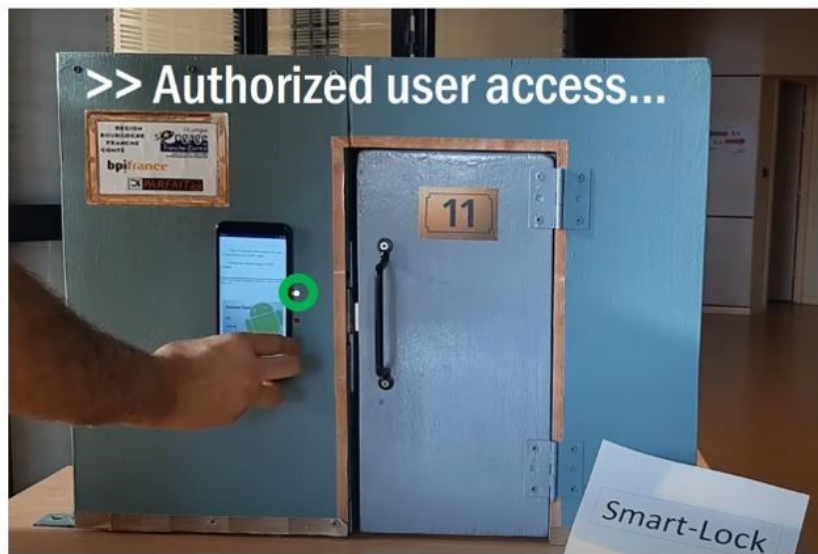


Figure.3. 14: Authorized access

- **Denied access:** through this scenario, we verified that only a user with a valid token, including the time expiration, the user's identity, could access the smart lock. In fact, we tried the access using a false token with an expired period, and then the request was rejected. Besides, with a non-expired token but a wrong user's identity, the smart lock stills locked and refuse the request, as mentioned in Figure. 3.15.

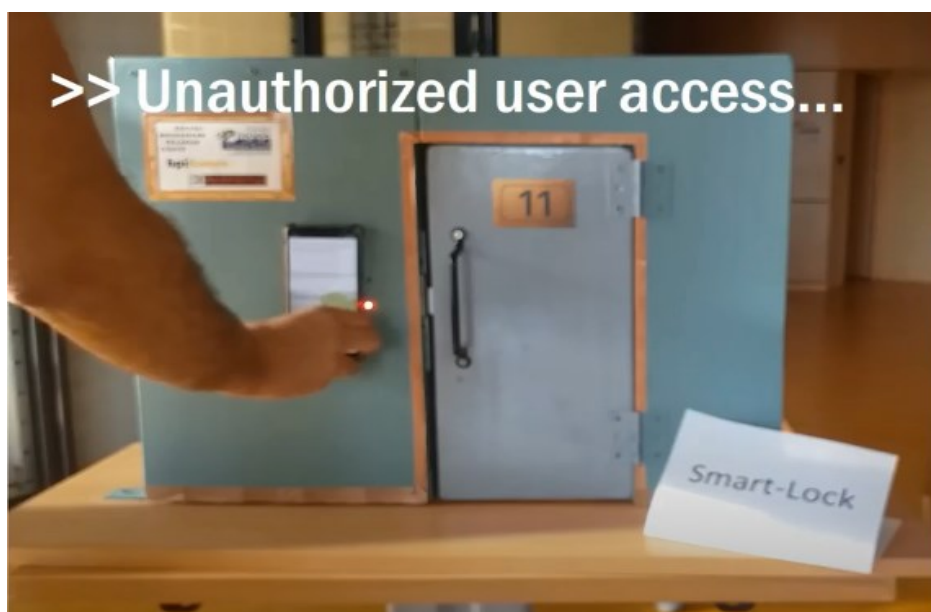


Figure.3. 15: Unauthorized access

All information of the user access phase is sent to the local server promptly through the ZigBee wireless communication. Besides, in the context of the PARFAIT project, our local server communicates this data securely to the Ericsson MQTT broker. A successful access to the smart lock sent to Ericsson MQTT is shown in Figure 3.16 (a) and the unsuccessful access is shown in Figure 3.16(b).

```
MQTT: Connected with result code {'session present': 1}
>> MQTT server online...

>> NO + Received

>> OK Acces Allowed - 845f4050afc4139b9ac2 - 2020-07-29 15:36:47

DB Commit

MySQL connection is closed

Sending to MQTT Server...
>> 2020-07-29 15:36:48 : Acces Allowed - room: 001 ( 845f4050afc4139b9ac2 )
```

(a)

```
>> OK Acces Denied - 85b619db0b0ddb23cb3b - 2020-07-29 15:37:06

DB Commit

MySQL connection is closed

Sending to MQTT Server...
>> 2020-07-29 15:37:06 : Acces Denied - room: 001 ( 85b619db0b0ddb23cb3b )
```

(b)

Figure.3. 16: (a) Successful access and (b) Unsuccessful access

To sum up, the authentication procedure using a smartphone that communicates via the NFC works correctly. The user needs to download our smart hotel's application, and then he/she could ensure the reservation by choosing the date of reservations and the room number. Besides, a PIN code will be sent to the user after a successful reservation. Finally, using the delivered code PIN, the user could get access to the corresponding room during the reservation dates securely, and all access information is sent promptly to the local server, as well as to the Ericsson MQTT server.

3.9. Conclusion

In this chapter, we have proposed a lightweight authentication protocol based on token concept to ensure authentication for a period of time and response to the needs of modern IoT applications (smart hotel, smart office, etc.). Therefore, the proposed protocol, *TBLUA*, could be adopted in any system reservation to ensure mutual authentication between the communicating parties (User, GW, smart device). In particular, the concept is based on adding a new security layer through adopting the software token, which enhances the security level. In

fact, the user could gain access securely to any smart device for a predefined interval of time with respect to the efficiency capability. Furthermore, we demonstrated the tradeoff between the effectiveness and efficiency of the proposed *TBLUA*. Indeed, the security analysis, using the AVISPA toolkit, showed that our protocol is robust against various attacks and provides relatively more security features such as anonymity and perfect forward secrecy. Moreover, we have demonstrated through performance analysis that *TBLUA* has a low computation and communication overhead compared to benchmarking schemes. Finally, a proof of concept is conducted, where we designed the smart hotel use case that ensure a successful authentication through the smart lock prototype.

In the next chapter, we address the problem of allowing many users access to the group of IoT devices. We propose a novel decentralized group key management for access control in an IoT environment that aims to control and manage users' access according to their subscriptions.

DGKM-AC: Decentralized Group Key Management for Access Control in IoT

4.1. Introduction

As shown in the previous chapter, user authentication is the primary security issue in the IoT environment as the IoT network is progressively permeating every aspect of our daily lives and is widely used in various kinds of applications (e.g., smart city/hotel/office) [154]. Recently, a new productive ground for developing a new type of group-based applications has been presented due to the increasing number of connected objects in the wireless networks. Mostly, we cite, for example, the wireless sensor networks [146], mobile ad-hoc networks [147], and IoT environment [80]. In fact, in the IoT environment, which is characterized by a large scale of connected devices, IoT objects (data subscribers) could request to communicate with the same IoT devices (data publishers). Thus, grouping communication might alleviate the IoT network, but it produces new security challenges. To safeguard IoT data from tampering and unauthorized access, designing an appropriate access control for group-based applications is the most critical and necessary security issue than ever. To ensure this, group key management (GKM) is one promising approach, which would be used to provide access control to data streams for legitimate users only [80]. GKM consists of creating a group key shared between a devices' group and its current subscribers, such that the device can encrypt its data, and only the subscribers can decrypt it. This encryption-based access control mechanism is suitable for large IoT environments characterized by a dynamic structure, where IoT users change their interest frequently over time. Indeed, GKM needs to disseminate permission keys when new members join, or old ones leave the system. Therefore, it does not require a continuously connected server to manage the access control.

Besides, the limited capabilities of IoT devices and the frequent and dynamic changes of the network had given birth to new challenges in the IoT domain. Given that, whenever a new user or IoT node is granted to join or leave the network, a new shared key should be distributed to

authorized users in the network, which can cause a severe problem with rekeying overhead. Several generic solutions, based on the centralized structures [80] and symmetric pre-shared key framework [146], have been introduced throughout the literature. However, these schemes are not entirely applicable to all IoT environments, as IoT networks are composed mainly of resource-constrained devices. Additionally, the current GKM schemes for access control in IoT networks are not suitable for a scalable and dynamic IoT network with frequent changes.

In fact, many users can subscribe to numerous services offered by different IoT devices and change their interests frequently over time. Thus, maintaining an efficient GKM in a dynamic IoT environment remains a challenging issue due to the rekeying process that affects all members in the same group for joining/leaving events. Therefore, all members should update their shared group access keys. Hence, an efficient group key mechanism should be introduced to reduce the rekeying dependence of members in the same group, and thus reducing the overhead.

To solve the rekeying dependence, minimize resulting overhead and achieve scalable access management in a dynamic IoT environment, this chapter introduces a new *Decentralized Lightweight Group Key Management Architecture for Access Control* named *DLGKM-AC*. We consider the smart hotel use-case mentioned in the previous chapter, in the context of the European project PARFAIT [7]. In this scenario, key cards and smartphones might be interchangeably used to give guests access permissions in different rooms. They can also be used to control the usage of various facilities according to room classes' and purchased services. When a guest checks out and the room becomes vacant, devices should stop sending the room's information, like temperature, cleaning status, hotel services status, etc., and deny access to the room by blocking the smart lock for this guest.

The main idea of *DLGKM-AC* is to create an efficient and flexible mechanism to secure the distribution of content to eligible subscribers. In particular, *DLGKM-AC* is a hierarchical scheme comprising a central *Key Distribution Center* (KDC) and several *Sub Key Distribution Centers* (SKDCs) to manage permission keys' dissemination.

The remainder of this chapter is structured as follows: First, we briefly present the related work in the literature. Then, we discuss some background necessary to understand the proposed scheme before presenting the overall system architecture, attacker model, and system requirements. Subsequently, we detail the proposed *DLGKM-AC* scheme before analyzing its security and its resistance to various attacks. Finally, we study its performance in terms of storage, communication, and computation overheads.

4.2. Related Works

GKM is essential for group communication to secure group data. More specifically, GKM guarantees access control in the group communication, where the group members share a group key to define the access permissions. In particular, the dynamic nature of group communications in IoT environments makes safeguarding data from unauthorized access a significant challenge. Indeed, it leads to a rekeying process, which causes significant network

resource consumption. Hence, it is crucial to reduce the overhead when updating shared keys among subscribers and publishers.

Table 4.1 summarizes and classifies existing GKM solutions based on different attributes and criteria as follows: (i) **Environment** of its application, such as wired Internet [146], wireless sensor networks (WSN) [80][148][112], ad hoc networks [147], wireless body area networks (WBAN) [149] and IoT environment [113][122]. (ii) **Network model** of the GKM access control that could be centralized, decentralized, or distributed. (iii) Used **cryptography types**. Then, we enumerate for each scheme its weaknesses and strengths. We deal with the essential security services **backward secrecy and forward secrecy**, where shared keys need to be updated whenever a new member joins or an existing one leaves its group. The **key Independence** permits to alleviate the rekeying process where updating group keys do not affect all the keys in the group. The **vulnerability to collusion attack** (the collaboration of adversaries to compromise a communication) for which rekeying is important to maintain security. Moreover, the rekeying process may cause a lot of key management overhead and leads to a **single point of failure**, especially in a **scalable** environment that supports **multiple group services** and is composed of **dynamic publishers** and dynamic subscribers. Hence, ensuring **subscribers' Independence** makes subscribers of one group independent from the entire group in the rekeying process of the group key after a join/leave event in this group.

In [150], the authors surveyed numerous key distribution schemes over wireless networks and classified them into centralized, decentralized, and distributed schemes. Centralized schemes use only one server known as the key distribution server (KDC) to create and distribute encryption keys. Distributed schemes do not have a specific KDC; they instead generate group keys either in a collaborative manner between the group members or by one member. Moreover, each member must keep track of the other members to make robust communication. Besides, membership change events (join/leave) cause high processing and communication overheads [157], leading to a congestion problem in a dynamic IoT environment. In contrast, decentralized schemes divide the system into several subgroups, thus, reducing the load on the KDC and offering a solution to scalability issues. Furthermore, a subgroup manager is responsible for keeping track of the group's members, reducing computation and storage overhead on members.

The distribution of encryption keys in the different cited GKM architectures is further ensured by using two main cryptographic types (symmetric and asymmetric). Two fundamental and efficient GKM schemes were proposed: The Logical Key Hierarchy (LKH) [109] and the One-way Function Tree (OFT) [110] based on symmetric keys (traffic key and encryption key) to distribute the updated encryption keys. In contrast to LKH, all the OFT implementations suffer from collusion attacks and increase devices' computational overhead for obtaining group keys. Hence, OFT is far from being ideal in an IoT environment, where the communicating devices may have limited computational power. Additionally, [152] [153] schemes provided fine-grained access control Attribute-Based Encryption (ABE) to manage keys' update. However, ABE is a cumbersome mechanism that relies on asymmetric cryptography, unsuitable for running on resource-constrained IoT devices [154]. Besides, asymmetric encryption mechanisms are also used in key management schemes [155] [156]. Specifically, Porambage

et al. [80] proposed a group key establishment protocol for multicast communication using the Elliptic Curve Cryptographic (ECC) operations. Even though their solution is suitable for resource-constrained devices, it does not efficiently manage the rekeying process.

Furthermore, all previously mentioned schemes are designed for single multicast groups, but users may subscribe to multiple services. To ensure many multicast groups, Park et al. [112] accommodate various services groups. Their scheme addressed rekeying in the wireless mobile environment based on a centralized architecture and an LKH mechanism to manage multiple communications. Likewise, Mapoka et al. [123] proposed using a distribution list of the session key and key update slot for each subgroup. This list is centrally managed by a node called the area key distributor. The proposed protocol alleviates the 1-affect-n phenomenon and transmission overhead of the core network, but it does not ensure forward secrecy. Hence, Zhong et al. [124] proposed another protocol called area-based multiple GKM that securely provides services when users migrate to different wireless networks, ensuring forward secrecy. Nonetheless, its high overhead, due to revocation events, makes it unsuitable for dynamic IoT environments.

Besides, for addressing the IoT environment's rekeying process, Tsai et al. [151] proposed a lightweight symmetric key establishment based on the Kronecker product. However, their protocol does not consider the key update when users or devices join or leave the system, which lacks forward and backward secrecy. Furthermore, Abdmeziem et al. [122] proposed a decentralized batch-based group key that includes several subgroups managed by key servers. This scheme considered long-term and short-term keys per group, which are common to all nodes. Nevertheless, it does not ensure communication between multiple groups, and it requires large storage and computation resources. Thus it was enhanced to decrease the communication overhead by adopting a Distributed Batch-based Group Key [120]. It is based on polynomial cryptography to set up the key for collaborative groups in the IoT environment. However, these schemes are limited to managing communications in one group and do not consider communications between different groups and services. Kung et al. [113] took advantage of the Chinese Remainder Theorem (CRT) based construction proposed by Park et al. [112] to accommodate multiple device groups. They established a two-tier centralized system, KDC, where each group (devices or users) runs LKH to handle updates of keys efficiently. However, communication within a user group is based on the symmetric group key, which leads to the dependence between members. Therefore, after each event (triggered by a join/leave user operation), the rekeying process induces all the members in the entire group to update their group key, increasing the computation overhead.

In summary, and as mentioned in Table 4.1, existing GKM solutions do not support members' Independence in the same group, where each member needs to update its key after every join/leave event. Specifically, they focus only on the symmetric group key per subgroup communication. Consequently, the rekeying performance is decreased when the number of subscribers is high and varies frequently. Moreover, lesser attention is paid to achieve efficient and scalable GKM for access control among a dynamic IoT environment, where many users (subscribers) can subscribe to different IoT services and frequently change their interests over time. Hence, throughout this chapter, we propose a flexible access management protocol that

is based on the GKM mechanism. Therefore, we suggest a new decentralized GKM to secure group communication, which offers the scalable feature in a dynamic IoT environment, alleviates the rekeying overhead caused by the member changes, and reduces the load on the KDC.

Table 4. 1: Comparison of existing GKM Schemes

Schemes	Environment	Network model	Cryptography type	Strength(+) /weakness(-)
[149]	Wireless Body Area Network	Centralized key distribution architecture	Symmetric cryptography & polynomial cryptography	<ul style="list-style-type: none"> + Backward secrecy + Forward secrecy - Does not support multiple group services. - Not scalable.
[112] [148]	Wireless Sensor Network	Centralized key distribution architecture	Symmetric & Asymmetric cryptography & LKH mechanism	<ul style="list-style-type: none"> + Support multiple group services + key Independence - Does not achieve the forward secrecy. - Subscribers are dependent to each other.
[152] [153] [154]	Cloud Computing	Centralized key distribution architecture	Attribute Based Encryption ABE & Asymmetric cryptography	<ul style="list-style-type: none"> + Backward secrecy + Forward secrecy - Vulnerable to collusion attack - Does not ensure key Independence
[113]	Internet of Thing environment	Two tier-Centralized key distribution architecture	Symmetric & Asymmetric cryptography & LKH mechanism	<ul style="list-style-type: none"> + Support publishers' dynamism + Support multiple group services. - Does not support subscribers' Independence. - Vulnerable to collusion attack
[123] [124]	Wireless Sensor Network	Decentralized key distribution architecture	Symmetric & Asymmetric cryptography	<ul style="list-style-type: none"> + High scalability + Alleviate 1 affect n phenomenon - Does not support publisher dynamism. - High overhead due to dynamic environment
[147]	IPv6 Ad Hoc Networks	Decentralized key distribution architecture	Symmetric cryptography	<ul style="list-style-type: none"> + Resist collusion attack + Support multiple group services. - Key Independence. - Moderate scalability.
[122]	Internet of Thing environment	Decentralized key distribution architecture	Symmetric cryptography,	<ul style="list-style-type: none"> + Alleviate 1 affect n phenomenon key Independence. - Large storage and computation resources. - Limited scalability.
[157] [120]	Internet of Thing environment	Distributed key distribution architecture	Asymmetric cryptography & polynomial cryptography	<ul style="list-style-type: none"> + Backward secrecy + Forward secrecy - does not support multiple group services. - Limited scalability.

4.3. Background

In this section, we briefly present the background and the main mechanisms used in our approach. We first define the Group Key Management (GKM), then we present the techniques used in our scheme for GKM. We detail the Master Key Encryption (MKE) based Generalized Chinese Remainder Theorem (GCRT) that is used for managing multiple user groups (UGs) and various users. Then, we describe the Logical Key Hierarchy (LKH) and One-Time Pad (OTP) schemes used for efficient key management of different device groups (DGs).

4.3.1. Group Key Management (GKM)

Group key management (GKM) is a cryptographic technique used to ensure access control for group communications. It secures one-to-many or many-to-many group communication by encrypting the group's data using a traffic encryption key (TEK). In particular, GKM supports establishing and preserving these keys, where only members of a group (called subscribers) could decrypt the data. Besides, due to the dynamic group of members, who change their membership frequently for various kinds of service demands, GKM accomplishes a rekeying process to maintain security among the group members. The rekeying procedure should be ensured with regard to the backward and forward secrecy. Indeed, a new joining member should be prevented from computing the joined group's old group key and learning the previous exchanged data to meet the backward secrecy requirement. A member who leaves a group should be prohibited from calculating the future group key and knowing the upcoming exchanged data to respond to the forward secrecy requirement. Therefore, for a large-scale environment such as the IoT with a highly dynamic group of members, designing an efficient GKM is essential to maintain and enforce access control.

4.3.2. Master Key Encryption (MKE)

The concept of MKE is defined as a key management scheme based on the Group Chinese Remainder theorem (GCRT). MKE permits multiple decryption keys to decrypt the same message encrypted by an encryption key [112]. The main idea of the master key encryption scheme is to generate one master key and several slave keys, where the master key encrypts a message that can be decrypted by each legitimate slave key. The MKE scheme can alleviate the rekeying cost resulting from symmetric cryptography. Hence, Park et al. [112] have proposed a general MKE algorithm to minimize the rekeying cost of the group key using a master key based on the following theorem:

❖ **Theorem 1:** Let $\{p_1, p_2, \dots, p_N, q_1, q_2, \dots, q_N\}$ a set of safe prime numbers and $\{e_1, e_2, \dots, e_N\}$ a set of corresponding public keys. If all public keys satisfy the following condition, $e_1 \equiv e_2 \equiv \dots \equiv e_N \pmod{4}$, then there exists a unique master key, e_M modulo $4x_1y_1x_2y_2 \dots x_Ny_N$, where $x_i = (p_i - 1)/2$ and $y_i = (q_i - 1)/2$, $1 \leq i \leq N$.

Theorem Proof: to explain the computation of slave keys and the master key, we consider there are N public/private slave key pairs (e_i, d_i) , $i \leq N$ with (p_i, q_i) being the i^{th} safe prime number pair, and one master key pair (e_M, d_M) . For simplicity, we now consider the modulus of the

prime pairs $\emptyset(p_i q_i) = (p_i - 1)(q_i - 1)$ are mutually prime to each other. For a plaintext P and a ciphertext C , the master key should satisfy:

$$P^{e_M} \equiv P^{e_i} \bmod (p_i q_i) \quad (4.1)$$

$$C^{d_M} \equiv C^{d_i} \bmod (p_i q_i), 1 \leq i \leq N \quad (4.2)$$

According to Euler's theorem, the necessary condition for the equation above is:

$$e_M \equiv e_i \bmod (\emptyset(p_i q_i)), \quad d_M \equiv d_i \bmod (\emptyset(p_i q_i))$$

The set of safe prime numbers, presenting the slave key, satisfies the following condition: $e_1 \equiv e_2 \equiv \dots \equiv e_N \bmod(4)$. Then, there exists a unique master key, $e_M \bmod(4x_1 y_1 x_2 y_2 \dots x_N y_N)$, where $x_i = (p_i - 1)/2$ and $y_i = (q_i - 1)/2$, $1 \leq i \leq N$, solution of a system congruence that can be calculated by the GCRT as follows:

$$e_M = \sum_{i=1}^N e_i M[i] N[i], \text{ Where } M[i] = (\prod_{j=1}^N x_j y_j) / x_i y_i \text{ and}$$

$$N[i] \text{ is an integer such that } M[i] N[i] \equiv 1 \bmod(4x_i y_i).$$

Based on *Theorem 1*, [112] proposes a general MKE algorithm, which generates and modifies the master key and the key pairs, respectively. Our proposed scheme takes advantage of this algorithm and proposes an optimized algorithm for membership *renewal* and *revocation*. This algorithm is described in the *DLGKM-AC* scheme section 4.5.3.2.

4.3.3. Logical Key Hierarchy (LKH)

The LKH mechanism is used to handle the rekeying issue efficiently in a secure group communication. This method minimizes communication costs by reducing the number of transmissions in rekeying as well as storage requirements. In fact, LKH is presented through a

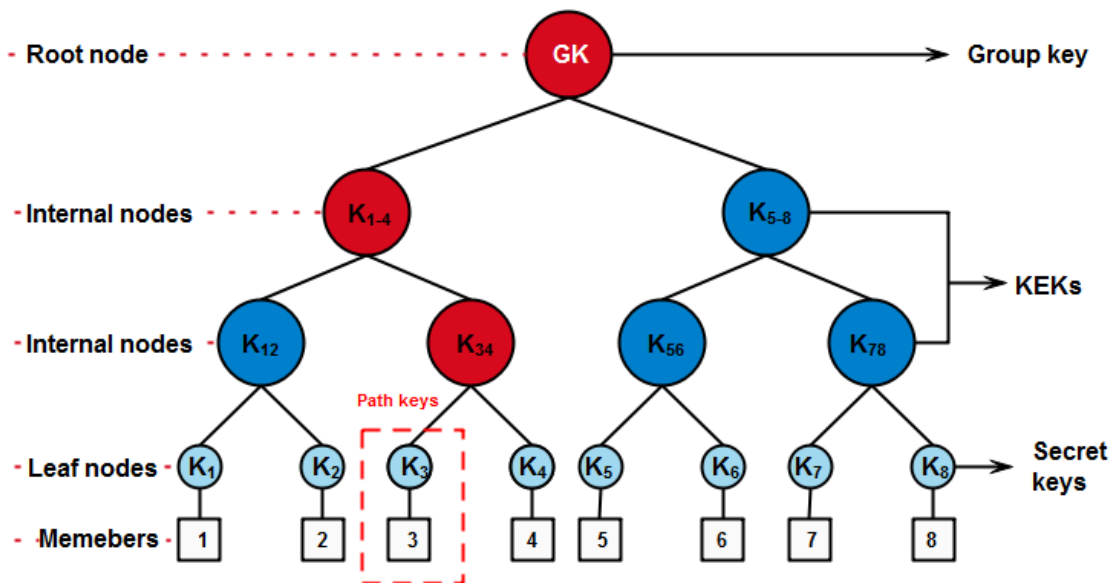


Figure.4. 1: The logical key hierarchy tree structure

binary tree (as shown in Figure.4.1) structure to manage keys' distribution. Indeed, the tree structure is composed of the root node that holds the group key (GK), the internal nodes that hold the Key Encryption Keys (KEK), and the leaf nodes. Each leaf node shares a secret key with the root and maintains the subtree rooted with the corresponding internal nodes. In particular, the constructed subtree composes a path key (PKt) with the internal KEKs that are used later to update the GK efficiently. The binary LKH tree structure guarantees an enhancement in the storage and communication overhead compared to other tree structures (section 4.2). Indeed, in a complete tree with n devices on leaf nodes, each leaf node stores $\log(2n+1)$ keys [109], while the rekeying procedure requests multicasting $O(\log(n))$ KEKs for these n devices instead of multicasting $O(n)$ KEKs.

4.3.4. One-Time Pad (OTP) key

The One-Time Pad key encryption mechanism is a strong encryption technique that cannot be damaged. In particular, while the proof of OTP security does not depend on any hardness assumptions, OTP is considered a perfectly secure mechanism. The OTP encryption is achieved by using a pre-shared key with the same size or longer to the exchanged messages. Fix an integer $l > 0$, a message space M , space of key K , and ciphertext space C can all be a set of l -bit strings. The key generation is ensured by selecting an l -bit key randomly and is never reused, which makes OTP entirely secure [33]:

- Encryption works as follows: given a key $k \in \{0,1\}^l$, and a message $m \in \{0,1\}^l$, output $c = E_K(m) = m \oplus k$, where \oplus is the “exclusive OR” operator.
- Decryption works as follows: given a key $k \in \{0,1\}^l$, and a ciphertext $c \in \{0,1\}^l$, output $m = D_K(c) = c \oplus k$.

4.4. DLGKM-AC General Overview

This section briefly describes the proposed decentralized group key management scheme, where we introduce the system architecture, the attacker model, and the system requirements.

4.4.1. System Architecture

We propose a new decentralized architecture for access control in group communication based on group key management. Indeed, the proposed *DLGKM-AC* illustrates a typical three-tier scheme composed of three essential layers, as shown in Figure.4.2. Two layers, named *publisher* and *subscriber* layers, defining groups of IoT devices (*DGs*) and users (*UGs*). In addition, the middle layer defines the decentralized controller, which is responsible for key management between and within groups. All these layers are described hereafter:

- **Publisher layer:** is the layer composed of constrained IoT devices with limited computational power, memory, or energy availability, such as smart door locks or IP cameras, collecting and sending data to subscribers. In our architecture *DLGKM-AC*, we use many groups of IoT devices, which are grouped related to their functionalities,

localization, and security requirements. Therefore, a new joining IoT device is assigned to precisely one of the device groups (DG).

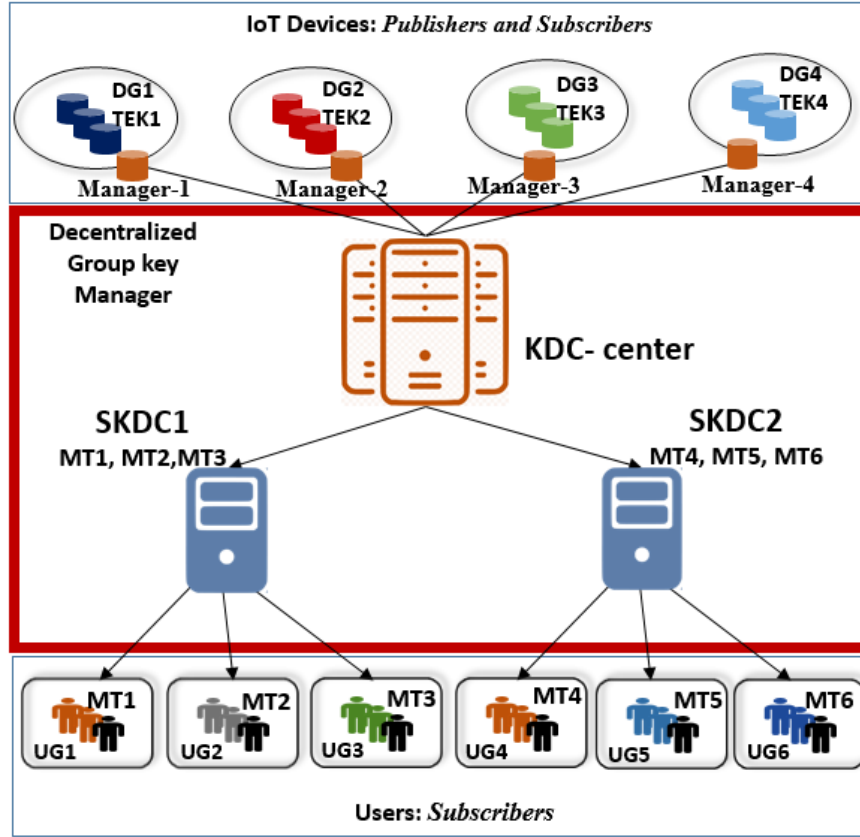


Figure.4. 2: Proposed system model

- **Subscriber layer:** is the layer composed of users communicating with their smartphones to retrieve data from the publisher layer. A user can be a device owner with legitimate, full, and permanent control or an IoT device with only limited access. Consequently, users and IoT devices subscribe to different *DGs* related to their wishes and desires. Then, a group of users *UG* is created based on the user's interest and reservation period.
- **Decentralized group key manager layer:** is the layer responsible for maintaining the purpose of security by generating the system parameters and managing group members by providing required encryption keys used to control data access. In particular, this layer presents a decentralized architecture of servers, and it is composed of a KDC, a KDC backup, and several SKDCs. The number of SKDC is not fixed and depends on the IoT application needs. More specifically, the number of SKDC is influenced by the characteristics of SKDC like storage, computation capacities, and the number of registered users.

The intended approach considers a dynamic reservation system in an IoT environment, where both the number of users and IoT devices might frequently change over time. Indeed, a user may join or leave at any time. Likewise, an IoT device can be introduced in or removed

from the system at any time. Thus, it is crucial to managing the distribution of encryption keys to secure both group communication and data transmission from possible threats that will be defined in the next subsection.

4.4.2. Threat Model

Consider the adversaries' capabilities in a data life cycle in the proposed system model, which is defined as compromising the GKM access control scheme based on the active insider and active outsider adversary models. Indeed, attacker A may be either an outsider who has no access to any IoT device or an insider who attempts to increase the access possibility. In fact, an insider attacker A such as a revoked user who no longer has access to future communication and yet still tries to retrieve information on access policies to extend access scope. Moreover, an outsider attacker A is an attacker that aims to extract sensitive information, such as the encryption key, to break the current encryption scheme and get access to data without proper permissions. Further, A may cooperate with other members in the system to derive keys that he/she cannot obtain individually, which is known as a collision attack. Besides, the attacker may also be a compromised device, where he/she may masquerade as a legitimate communication partner before initiating communication with other participants in the network to gain access to data. However, he/she cannot compromise or break the cryptographic primitives.

4.4.3. System Requirements

A practical GKM scheme should respond to several requirements related to security and efficiency [4] [150]. These requirements are explained in what follows:

➤ Security requirements

In a dynamic IoT environment, the security of the transmitted data is a primordial issue. In particular, the forward secrecy security property, which is based on avoiding any leaving member of any group from getting the future group key and decoding any exchanged messages after her/his departure. Hence, this security feature also has an objective to prevent the collusion attack. Furthermore, the backward secrecy security property that prevents any new member joining an existing group from decrypting the group communication established before joining. Forward/backward secrecy are accomplished through an efficient key updating process, where all keys should be completely independent of each other to safeguard the key independence security service.

➤ Efficiency requirements

The efficient functioning of key management protocols is justified by a minimum overhead cost of different metrics. First, it should reduce the number of keys stored on both users and IoT devices, which results in low storage overhead. Second, it needs to decrease the required computation power from users, IoT devices, and servers, increasing efficiency by reducing the system response time. Further, it should minimize the number of exchanged messages on the system, which raises the overall system's flexibility and thus achieves a low communication cost.

In addition, the group key management schemes should ensure the scalability capability to handle variable group sizes and high membership changes. Indeed, this may guarantee that the delay will not increase dramatically when the network size becomes large. Besides, key management schemes suffer from the 1-affects-n phenomenon, where a failure of a single server leads to the collapse of the whole system. Hence, it is essential to avoid this phenomenon and assure availability in a large and scalable system.

4.5. DLGKM-AC Detailed Description

This section describes the proposed decentralized group key management for access control (DGKM-AC) in IoT. Indeed, we start with an overview of its functionalities before detailing its different phases, namely, system initialization, registration of IoT devices and users under the system, and keys' update that explains how to handle members' joining and revocation events.

4.5.1. Overview

Our scheme is designed for a dynamic IoT environment, where users and IoT devices are frequently dynamic and continuously change their subscriptions to IoT services. This makes ensuring data confidentiality in the group communication as backward/forward secrecy a challenging task. However, using symmetric keys by both publishers and subscribers can provide a heavy solution. Indeed, the access control policies in our scheme are disseminated using different and separate GKM mechanisms for users and IoT services. In the following, we describe in detail the different architecture layers mentioned in Figure 4.2 and explain the interactions of those entities.

Typically, in the current GKM schemes, users should maintain and update group information to get the updated group key after the user's leaves and join events. For that, we introduce Master Token Encryption (MTE) to manage the communication within a group of users. The main idea of MTE is adopted from the master encryption key (MEK) based Chinese Remainder Theorem (CRT) [112], described in section 4.3.2. The MTE mechanism generates Master Token (*MT*) and several valid slave tokens (*STs*) for a predefined period, which means that all related slave tokens decrypt all messages encrypted by the master token during its lifetime. Hence, each user in the group should only maintain a slave token to get the updated information in the group, which improves the effectiveness of the communication within the user group. More specifically, users in the proposed DLGKM-AC could subscribe to many device groups (*DGs*) to get data. For this purpose, each user gets all traffic keys *TEKs* of *DGs* to which it is subscribed. In fact, after each user joins/leaves the system, it is essential to update these *TEKs*, which is ensured by MTE to reduce the rekeying cost.

In addition, to manage the group communication within IoT devices groups, we use two different mechanisms: the LKH structure and OTP, described in the section 4.3.3 and 4.3.4. Since multiple users may subscribe to the same IoT device group, it would be more efficient if all these devices and all their subscribed users share a group key for encryption. Traffic Encryption Key (*TEK*) is a traffic key used to encrypt data published by a device group to its subscribers. This traffic key should be efficiently updated when a new user joins or an old one leaves to ensure forward and backward secrecy. Moreover, to use the OTP mechanism, we

define a manager to control each device group DG. In particular, the manager is responsible for outlining the permissions of access control for devices in the DG. Typically, we consider managers as lightweight devices in our architecture, and all IoT devices registered in the system are under the control of their group's manager to which they belong. Once registered under the control of the manager, this latter receives the authorization for the new IoT device based on the group permission from the KDC. Furthermore, the manager maintains the keys of the access list management and ensures the required updating of keys to maintain the security of the system.

KDC is the central server that relates publishers to the rest of the system, and it manages the keys' update process within DGs. Further, KDC has a backup server that maintains the last updated version of keys in the system, which is sent to the backup periodically after the rekeying process. Besides, SKDCs manage the group communication within UGs, where users frequently join and leave the system. Hence, the decentralized aspect of the controller, where SKDCs are used, allows reducing the load on the KDC. Multiple user groups are under the control of one SKDC depending on users' localization, which solves the problem of single-point of failure (SKDC failure) and ensures the scalability of the system. Besides, we assume that the decentralized KDC can establish a one-time secure channel with users and devices, which can be used to authenticate and configure a newly joined user/device (e.g., by installing a shared secret key) before sharing with them the encryption keys.

We can summarize the different encryption keys in our scheme into two main categories: (i) Traffic Encryption Key (TEK) and (ii) Key Encryption Key (KEK). The traffic keys are used to encrypt/decrypt data, while the key-encryption keys are used to encrypt/decrypt traffic keys to distribute them securely, as mentioned in Table.4.2.

Table 4. 2: Keys' description

Traffic Encryption Keys (encrypt data)	<ul style="list-style-type: none"> • TEK: encrypts data of DG • DK: encrypts data of one device
Key Encryption Key (encrypt traffic key)	<ul style="list-style-type: none"> • KEK & GK encrypt and distribute <i>TEK</i> within a DG • MT encrypts updated <i>TEK</i> keys to users in SKDC • ST decrypts updated <i>TEK</i> keys in UG

After presenting, in general, the global functioning of our scheme, we detail in what follows its effective working. Table.4.3 summarizes the main used notations in the following sections, and we start with explaining the initialization phase in the next subsection, which is primordial to the system setup.

4.5.2. Initialization of the System

During this phase, the group key manager performs the initialization and the setup of the system parameters, which will be used in the eventual registration and rekeying phases. We note that the KDC and at least one SKDC run the initialization phase, which is presented in what follows:

Table 4. 3: Summary of symbols and their description

Symbol	Description
TEK	Traffic Encryption Key
KEK	Key Encryption Key
M	Total number of Device Groups
N	Total number of Slave Keys under SKDC
(e_M, d_M)	Master Key
(e_i, d_i)	Slave Key i
MT, ST	Master Token, Slave Token
DG_y	Device Group y
DK_j	Shared secret key between device j and KDC
UG_x	User Group x
UK_i	Shared secret key between user i and SKDC
GK_y	Group Key for device group y
PK_t	Path Key
$h(.), [.]_K$	Hash Function, Encryption function using encryption key K .

- ❖ KDC generates a master key and several slave keys by running the master key generation **MkeyGen** Algorithm 1 based on the GCRT scheme (presented in section 4.3.2) [112]. In fact, **MkeyGen** Algorithm 1 takes as input N safe prime numbers and computes a set of corresponding pair keys (e_i, d_i) based on the *theorem1*, where all these pairs' keys are slave keys that verify the equation with the unique solution of (e_M, d_M) . In our scheme, KDC uses the master key encryption cryptography to communicate with the SKDCs under its control. Indeed, after each new SKDC is added to the system, KDC has to run the **MkeyGen** algorithm to generate a slave key for the new SKDC and update its master key. In particular, KDC creates groups of devices DGs and groups of users UGs (depending on their subscription to devices). Finally, KDC assigns UGs to the corresponding SKDC. Furthermore, KDC establishes a secure channel with devices and users through sharing secret keys. In Figure.4.3, we illustrate the system architecture with 4 DGs and 6 UGs.
- ❖ In our system, the SKDC manages many groups of users; Thus, SKDC needs to run the same algorithm master key generation **MkeyGen**, where we consider N to be the maximum number of slave keys provided by SKDC. Indeed, the SKDC generates a master key (e_M, d_M) and a set of N public-private key pairs, named slave keys, $SK = \{(e_i, d_i); 1 \leq i \leq N\}$ through **MkeyGen**. Furthermore, SKDC defines a function f which maps a key pair from a set of slave keys to $\{0, 1\}$ as follows:

$f: S \rightarrow \{0, 1\}$, where:

$$f: \begin{cases} f((e_i, d_i)) = 1, & ((e_i, d_i) \text{ is assigned to a user}) \\ f((e_i, d_i)) = 0, & ((e_i, d_i) \text{ is not assigned to any user}) \end{cases} \quad (4.3)$$

After that, SKDC initializes all key pairs using (4) as follows: $1 \leq i \leq N, f((e_i, d_i)) = 0$.

Algorithm 1 Master Key Generation MKeyGen

Inputs: A set of safe prime numbers $p_1, p_2, \dots, p_N, q_1, q_2, \dots, q_N$.

Output: One master key e_M and N slave public-private key pairs $S = \{(e_i, d_i); 1 \leq i \leq N\}$

1: $S = \{\};$

2: **For** $i = 1$ **to** N

$\varphi_i = (p_i - 1) \times (q_i - 1);$

$x_i = (p_i - 1)/2;$

$y_i = (q_i - 1)/2;$

$e_i = 4 \times \text{Random} + 1;$

$d_i = e_i^{2(x_i - 1)(y_i - 1) - 1} \bmod 4x_i y_i;$

$S = S + \{(e_i, d_i)\};$

End For

3: $product = 1;$

4: **For** $i = 1$ **to** N

$product = product \times (x_i y_i);$

End For

5: **For** $i = 1$ **to** N

$M[i] = n/(x_i y_i);$

$N[i] = M[i](x_i - 1)(y_i - 1) - 1 \bmod (x_i y_i);$

End For

6: $e_M, d_M = (0, 0);$

7: **For** $i = 1$ **to** N

$e_M = (e_M + (e_i \times M[i] \times N[i]));$

$d_M \equiv (d_M + (d_i \times M[i] \times N[i]))$

End For

4.5.3. Registration Phase

During this phase, the user and the IoT devices are registered under our system in order to get the corresponding access keys. In particular, the groups of IoT devices and users are established through this phase. We detail this phase for both the user and IoT devices in the next two subsections.

4.5.3.1. Device Groups Registration

In order to manage the communication with a group of devices, the KDC creates many groups of IoT devices, DG , based on similar devices' attributes (location, functionalities...), according to the two different mechanisms:

- A binary LKH tree for the universe of devices in each devices group, for the publisher IoT devices.
- A modified structure of One-Time Pad (OTP) in each devices group, for the subscribers IoT devices.

i. LKH Mechanism:

To define the groups of devices, the KDC sets a binary LKH tree for the universe of devices in each DG , which will be used to distribute updated keys to devices. In the tree, each intermediate node holds a KEK . A set of $KEKs$ on the path nodes from a leaf to the root are called Path Keys (PK_t), as described in section 4.3.3. The LKH tree is constructed by KDC as follows:

- KDC generates key encryption keys ($KEKs$) for the intermediate node in each DG .
- Devices in DG are assigned to the leaf nodes of the tree, and random keys DK_j are generated and assigned securely to each leaf node.
- The root node holds group key GK to communicate with devices and TEK to encrypt data of DG .
- Each device D_j in DG receives the path keys PK_t from the root node to the parent node of the tree, using the shared secret key DK_j .
- Then, the path of keys is used as $KEKs$ to encrypt the group key by the KDC in each rekeying process and to distribute updated encryption keys to leaf nodes.

ii. OTP Mechanism:

In practice, IoT devices may also subscribe to other IoT devices to get access to data. In fact, to manage the communication between groups of devices, we present a modified structure of One-Time Pad (OTP). Firstly, KDC defines a manager for each group device, as described in section 4.3.4; We define M as the set of managers M_x , GK as the set of group keys GK_x of each DG , and TEK as the set of traffic encryption keys TEK_x used to encrypt data controlled by each manger M_x . Furthermore, ID is the set of devices identity ID_j of each device under the control of M_x manger, and DK is the set of secret shared device key between each device and manager. Based on the OTP, KDC defines:

- ❖ An array A for each DG , used to distribute and broadcast information useful for updating group key GK . A is presented where $A[i] = \sum_{j=1, j \neq i}^n DK_j \oplus ID_i$ corresponds to the device with identity ID_i .
- ❖ A group key GK_x for each DG_x is generated; $GK_x = \sum_{j=1}^n \oplus DK_j$.

Finally, once all group devices and their managers are successfully registered, KDC sends for each manager the list of permission access keys, which are defined with the traffic keys as follows: $TEK_{x \rightarrow Resource} = \{TEK_{x \rightarrow y} ; y \in Resource\}$, where $TEK_{x \rightarrow y}$ refers to the subscription of DG_x to DG_y . After the registration phase, each device in DG_x has $(ID_i, DK_i, A[i], GK_x)$, and controlled by the manager $(GK, A, ID, TEK_x, TEK_{x \rightarrow R})$.

4.5.3.2. User Groups Registration

In this phase, multiple user groups UG_K are constructed, and we assume that each UG_K accommodates r_k users with the same interest for a period T . Each user U_i in UG_K is

authenticated before joining the system and shares a secret key UK_i with SKDC. The SKDC assigns a user group ID based on the following definition:

Definition: Let $U = \{U_1, U_2, \dots, U_n\}, n \leq N$ be the universe of users controlled by one SKDC. Each user in a network can subscribe to one or more services of device groups DGs among a total of M (DG) denoted by $\{DG_1, DG_2, \dots, DG_M\}$. Let $UG \subset U$ be the set of users who subscribe to the same set of DGs during the same time T . Let $\{UG_1, UG_2, \dots, UG_u\}$ be the set of user groups UGs. Here, each UG_k possesses an ID_{UG_k} defined as described in equation 4.4:

$$ID_{UG_k} = \left\{ \begin{array}{l} A_{j,b} \mid 1 \leq j \leq M \mid b \in [0,1], \text{ where} \\ A_{j,0} = 0, UG_k \text{ is not subscribed to the } DG_j \\ A_{j,1} = 1, UG_k \text{ is subscribed to the } DG_j \end{array} \right\}, (4.4)$$

Where j defines the device group DG_j , and b outlines the user group subscription to this corresponding DG_j .

At that point, the SKDC needs to generate the necessary keys to communicate with the group of users. Indeed, we proposed a new master key encryption algorithm, which reduces the communication and computational complexities as it supports efficient key updating named

Algorithm 2 Master Token Generation MTokenGen

Inputs: Number of user r , Time T , e_M , S
Output: MT_K Master Token of UG_K and list S_K

- 1: $e_{M_K} = e_M$;
- 2: $Comp = 0$;
- 3: $S_K = \{ \}$;
// Select a list of slave keys for UG_K , $S_K = \{e_i^{1K}, e_i^{2K}, \dots, e_i^{rK}\}$.
// $e_i^{1K} = e_i$ assigned to user in UG_K
- 4: **While** ($Comp < r$) **do**
Select a random (e_i, d_i) from $S = \{(e_i, d_i) \mid 1 \leq i \leq N\}$
- 5: **If** $f((e_i, d_i)) == 0$
Then
 $S_K = S_K + \{(e_i, d_i)\}$;
 $f((e_i, d_i)) = 1$;
 $comp ++$;
End if
End while
- 6: **For** $i = 1$ **to** N
If $e_i \notin S_K = \{e_i^{1K}, e_i^{2K}, \dots, e_i^{rK}\}$
Then
 $e_{M_K} = e_{M_K} - e_i \text{ M}[i] \text{ N}[i]$;
End if
- End For**
- 7: $MT_K = (e_{M_K} + T)$

MTokenGen. Therefore, **MTokenGen** (*algorithm 2*) is designed to generate a master token MT_K , which has the role of group key and a set S_K of slave tokens STs for the corresponding users' group UG_K . Then, each user member U_i in UG_K receives a ST through a secure unicast. Finally, the SKDC adds users' group information $(ID_{UG_K}, MT_K, S_K, r_K, T)$ to the list of active users' groups.

At this level, we confirm that the *DLGKM-AC* is successfully initialized, the users and IoT devices are effectively registered under our system. Hence, we can see, as mentioned in Figure 4.3, the different disseminated keys at each level of our architecture, including the devices' groups, KDC, SKDC, and users' groups. Besides, the subscribers to different IoT services change their interest over time. Therefore, maintaining security in a dynamic IoT environment involves an effective rekeying procedure, which is the next subsection's main subject.

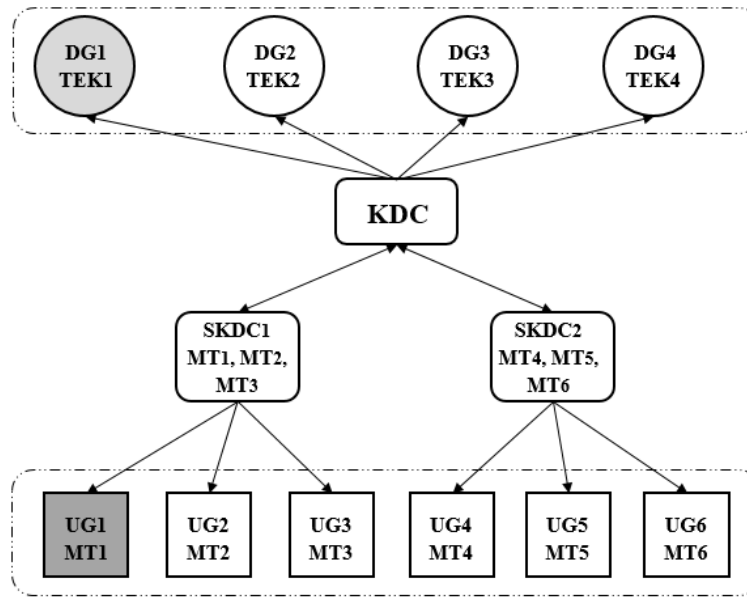


Figure.4. 3: Key distribution in our system model

4.5.4. Key Update Scenarios

Users and IoT devices join and leave the communication session over time. In fact, the dynamic feature of membership affects the security of the system. Hence, the keys should be changed after each user/device “join” and “leave” event to ensure backward and forward secrecy, which are detailed in the following.

4.5.4.1. User joins/leaves Events Scenarios

The key updating scheme for users is illustrated according to two events: the join user event and the leave user event. In order to describe the keys' update process of DLGKM-AC, and for simplicity, we consider the case shown in Figure 4.4 of a user who joins/leaves the user group $UG1$, where users are subscribed to $DG1$, $DG2$, and $DG4$.

i. When a User Joins a Group

Consider a user U_{join} , registered to SKDC after being authenticated, who is joining an existing group UG_K . Hence, SKDC conducts the join key update *Algorithm 3 “JoKeyUpdate”* to update the group key (MT_K, e_M) and generates a new slave token ST for the new user U_{join} . To this end, the SKDC searches for a slave token not affected to any user by mapping the f function $f((e_i, d_i)) = 0$, as explained in *Algorithm 3*. Furthermore, our scheme ensures that the existing users in the joined UG_K still can decrypt the recently sent messages, encrypted with the new MT_K , using their previous ST s.

Algorithm 3 JoKeyUpdate

Inputs: UG_K information (ID, MT_K, S_K, r_K, T) and (e_M, S) .

Output: updated MT'_K, S'_K, r'_K, e'_M and S' .

A new user joins the UG_K

1: Find e_i from $S = \{(e_i, d_i) \mid 1 \leq i \leq N\}$ where $f((e_i, d_i)) = 0$

// $e_i^{\{join\}K} = e_i$, is added to S_K

2: $S'_K = \{e_i^{1K}, e_i^{2K}, \dots, e_i^{r_K}\} + \{e_i^{\{join\}K}\}$

3: $r'_K = r_K + 1$;

4: $e_{M_K} = (MT_K - T)$;

5: $e'_{M_K} = e_{M_K} + e_i^{\{join\}K} M[i] N[i]$;

6: $MT'_K = (e'_{M_K} + T)$;

After that, SKDC runs the join key distribute *Algorithm 4* named “**JoKeyDistribute**” to disseminate the necessary rekeying messages occurring in the system when U_{join} joins the group UG_K . We explain the necessary steps below:

- SKDC shares secret key UK_{join} with the new joined user U_{join} securely, using a unicast message.
- Then, SKDC notifies the KDC about the joining event, and all existing users subscribed to the same device groups through a multicast message to update the corresponding traffic key TEK_j .
- Consequently, old users update TEK_j by using a hash function to minimize the system's communication overhead. Hence, the new user cannot access previous exchanged data.

Algorithm 4 JoKeyDistribute

Inputs: $TEKs, DKs, MT$

Output: new and updated keys $ST, U_i, MT', TEKs', DKs'$

1: $SKDC \xrightarrow{\text{unicast}} \text{User } i$: establish a shared secret key with user i U_i

2: $SKDC \xrightarrow{\text{multicast}} \text{All}$: Notify KDC , old users of the joined group and other user groups which subscribed to the same DG to update $TEK' = h(TEK)$.

3: $KDC \xrightarrow{\text{multicast}} \text{Devices}$: update their key $DK' = h(DK)$

4: $SKDC$: update MT of this group joined

5: $SKDC \xrightarrow{\text{unicast}} \text{User}$: $[ST_i, DKs, TEK]_{UKi}$

- Finally, SKDC sends the required updated keys to the new user U_{join} through a unicast message, including his slave token ST generated through *Algorithm 3*.

An illustrative example of a user joining a user group

Suppose a new user $U4$ that wants to get access to $DG1$, $DG2$, and $DG4$, as shown in Figure.4.4. Hence, $U4$ needs to get the corresponding traffic keys TEK_1 , TEK_2 , TEK_4 . For that, after being authenticated and authorized, $U4$ requests to join $UG1$. Thus, SKDC creates a shared secret key $UK4$ with $U4$; Then, it multicasts a notification based on the identities of user groups subscribed to the same devices groups to update the TEK_1 , TEK_2 , TEK_4 , so that the new user cannot access to previous exchanged data.

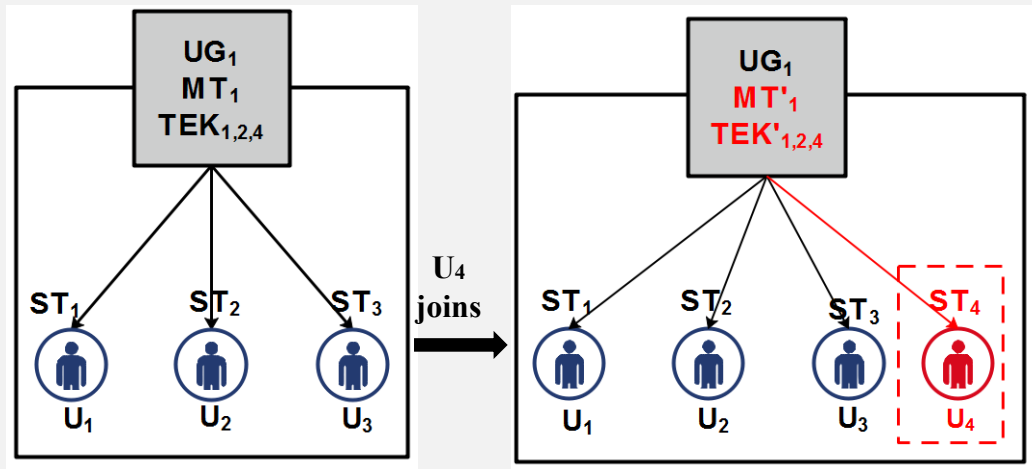


Figure.4. 4 : Structure inside UG1 when U4 joins

Besides, SKDC updates the group key $MT1'$ of $UG1$ as mentioned in **JoKeyUpdate** algorithm to protect previous communications between the existing users and SKDC from intruders, and generates a new ST for $U4$, while existing users of $UG1$ still be able to decrypt data of new group key $MT1'$. Moreover, devices of $DG1$, $DG2$, and $DG3$ update $TEK'_1=h(TEK_1)$, $TEK'_2=h(TEK_2)$, $TEK'_4=h(TEK_4)$. Finally, SKDC sends, in unicast, to the new user $U4$ the slave token ST and the updated TEK'_1 , TEK'_2 , TEK'_4 keys.

ii. When a user leaves a group

In this phase, we assume that a user U_{leave} wants to leave a user group UG_K , thus he is not allowed to obtain the exchanged messages after his revocation. To secure a user's leaving event, we detail a description designed to afford the forward secrecy. At this level, SKDC conducts the leave key update *Algorithm 5* named "**LeKeUpdate**" to update the master token of the group (MT_K , e_M), and users group's information. Actually, the updating of the master token MT_K is ensured by deleting the ST of the leaving user, while the remaining users still could get access and decrypt data of the new updated MT_K using their previous slave tokens.

Algorithm 5 LeKeyUpdate

Inputs: UG_K information (ID, MT_K, S_K, r_K, T) and system information (e_M, S).

Output: updated MT'_K, S'_K and r'_K .

The i^{th} user leaves the UG_K

- 1: $f(e_i^{\{leave\}K}) = 0$
 - 2: $e_i^{\{leave\}K}$ is revoked from S_K
 - 3: $S'_K = \{e_i^{1K}, e_i^{2K}, \dots, e_i^{rK}\} \setminus \{e_i^{\{leave\}K}\}$
 - 4: $r'_K = r_K - 1$;
 - 5: $e'_i = e_i^{\{leave\}K} = 4 \times \text{Random} + 1$;
 - 6: $e_{M_K} = (MT_K - T)$;
 - 7: $e'_{M_K} = e_{M_K} - e_i^{\{leave\}K} M[i] N[i]$;
 - 8: $e'_M = e_M - e_i M[i] N[i] + e'_i M[i] N[i]$;
 - 9: $MT'_K = (e'_{M_K} + T)$;
-

At this level, SKDC runs the leave key distribute *Algorithm 6* named “**LeKeyDistribute**” to disseminate the necessary rekeying message in the whole network after user U_{leave} leaves the UG_K . The algorithm executes some steps as described below:

- Firstly, the user U_{leave} announces his willing to leave the system to SKDC, which verifies the request and unicasts a message to KDC to signal a leave event.
- Then, KDC updates all TEK_j to which U_{leave} was subscribed according to the group identity ID_{UG_K} by generating new TEK_j based on the updating method ($TEK_j | \text{random processes of KDC}$), and broadcasts the new $TEKs$ to SKDCs.
- At that point, SKDC enforces an access control level for the user group using its identity ID_{UG} ; $[TEK_j^{new}, \forall j | A_{j,b} = 1 \text{ of } UG_K]$. Thus, according to ID_{UG} , SKDC encrypts the updated TEK_j^{new} using the corresponding master token MT of each user group UG and encrypts the results with the master key of SKDC. Consequently, the message is broadcasted to all corresponding users. Notice that only users with a valid ST can decrypt the new TEK_j^{new} .

Algorithm 6 LeKeyDistribute

Inputs: $TEKs, DKs, MT$

Output: new generated keys $MT', TEKs', DKs'$

- 1: SKDC updates MT of the group UG has been left
 - 2: SKDC $\xrightarrow{\text{unicast}}$ KDC: notify that UG has been left
 - 3: KDC $\xrightarrow{\text{multicast}}$ SKDCs: $(TEK' | DK')_{MK}$
 - 4: KDC $\xrightarrow{\text{multicast}}$ Devices: $(TEK')_{GK}$
 - 5: SKDC $\xrightarrow{\text{multicast}}$ user groups UG : $((TEK')_{MT})_{MK}$
-

An illustrative example of a user leaving a user group:

Suppose the user U_3 leaves the group UG_1 , as shown in Figure.4.5. Meanwhile, she/he unsubscribes from DG_1 , DG_2 , and DG_4 , which leads to losing the access privilege to those DGs . Since the data of DG_1 , DG_2 , and DG_4 should not be visible to this user anymore, traffic keys TEK_1 , TEK_2 , TEK_4 are updated to meet the requirements of the forward secrecy [5].

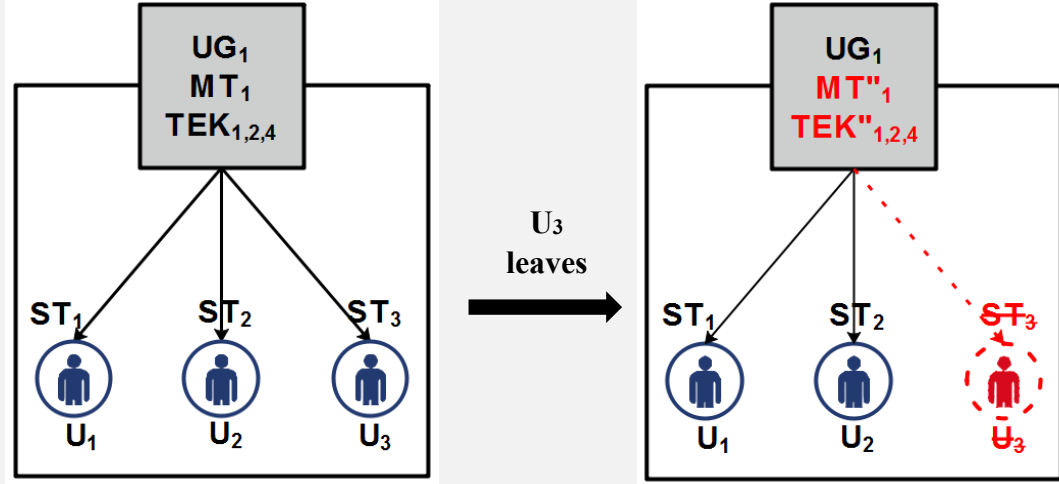


Figure.4. 5: Structure inside UG_1 when U_3 leaves

For that, SKDC first updates the master token MTI'' of the UG_1 , as shown in Figure.4.5, while all users of this left group still get access with their previous STs . Then, KDC generates a new $TEKs$ and broadcasts it via an encrypted message $(TEK''_i, DK_j)_{update}$ methods, $i=1,2,4)_{MK}$ to SKDCs. After that, SKDC transmits updated traffic keys TEK''_1 , TEK''_2 , TEK''_4 encrypted with the new master token MTI'' securely to all members of the UG , based on the user group identity. The remaining users still could decrypt, with their previous STs , the message to handle the updated information. Finally, devices in groups DG_1 , DG_2 , and DG_4 get the new $TEKs$ keys encrypted with the conforming $KEKs$ and GK keys, sent in multicast by the KDC, which prevent a leaving user from obtaining additional data.

4.5.4.2. IoT Devices Joins/Leaves Events

To describe the update key process of DLGKM-AC during IoT device join/leave events, we consider the two cryptography mechanisms named LKH and OTP, described at section 4.3.3 and 4.3.4. In what follows, we detail the rekeying mechanisms for IoT devices groups after each join and leave events.

i. LKH Cryptography Mechanism

The binary LKH tree is used for grouping the universe of devices that are only publisher of data in our IoT system. Otherwise, these publishers are added and deleted from the IoT system over time, damaging the security. Therefore, a rekeying process after each joins and leaves event is primordial, as explained in follows:

a. When an IoT Device Joins a Group

During this event, we consider an IoT device D_{join} is joining a DG_y . Therefore, the KDC runs the distribute join key update *Algorithm 7* named “**DeJoKeUpdate**”, to disseminate the rekeying keys. Thus, few steps are conducted:

- KDC shares a secret key with D_{join} that joins the device group DG_y .
- Then, KDC updates the necessary part of the LKH tree in which the device resides.
- After, KDC multicasts to the existing devices in the DG_y a notification to upgrade their group key GK .
- Finally, KDC send in unicast to D_{join} the path key PK_t and the traffic key TEK of the DG_y .

Algorithm 7 DeJoKeUpdate

Inputs: $KEKs, GK$

Output: new and updated keys $DK, D_j, KEKs', GK'$

1: KDC \rightarrow device D_j : establish a shared secret key with the new device (D_j)

2: KDC $\xrightarrow{\text{multicast}}$ old devices in DG: update group key $GK' = h(GK)$

3: KDC $\xrightarrow{\text{unicast}}$ devices: update KEK' encrypted either by secret keys or shared KEK

b. When an IoT device leaves a group

When a device D_{leave} leaves a DG , the KDC rearranges the LKH tree structure in the group and runs the distribute leave key update *Algorithm 8* named “**DeLeKeUpdate**” to multicast an updated group key GK' to the remaining devices. This group key is encrypted with the corresponding $KEKs$, which defines the LKH tree of the leaved DG_y . Then, KDC broadcasts a message to announce that D_{leave} is no longer a valid device.

Algorithm 8 DeLeKeUpdate

Inputs: $KEKs, GK$

Output: new keys $KEKs', GK'$

1: KDC $\xrightarrow{\text{broadcast}}$ All: “leaving device j is no longer available.”

2: KDC $\xrightarrow{\text{multicast}}$ DG: update GK' and KEK's.

An illustrative example of an IoT device joining and leaving a device group:

In this example, we consider the device group $DG1$ composed initially of three devices $D1$, $D2$, and $D3$. Firstly, we suppose a new device $D4$ joining the system, which is assigned to the device group $DG1$ as shown in Figure.4.6. KDC notifies the devices in $DG1$ to update $GK'1 = h(GK1)$ and creates a shared secret key $D4$ with device 4 to send necessary information $TEK'1$, $GK'1$, $KEK2$ encrypted with the secret key $D4$ through a unicast communication. Finally, KDC sends $KEK2$ to $D3$.

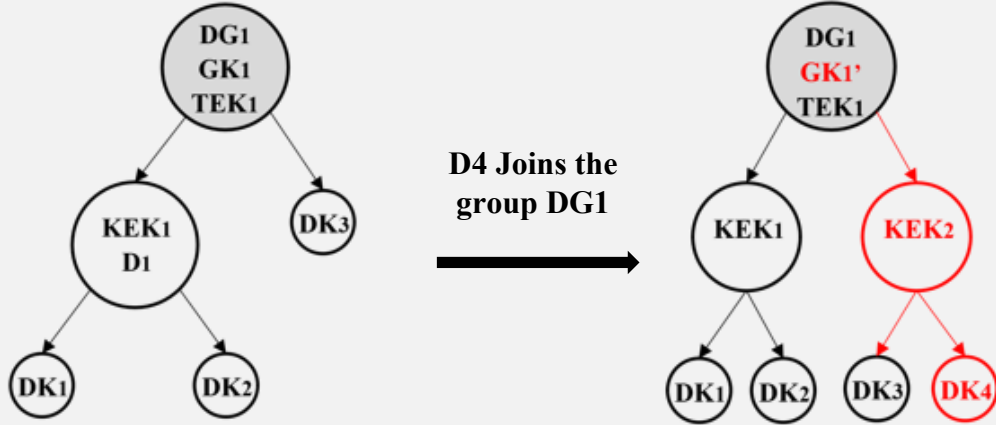


Figure.4. 6: Examples of LKH structure updates for device join.

At this stage, we suppose device $D2$ leaves the group $DG1$, as shown in Figure.4.7, thus, KDC makes a new device group key ($GK1''$ |update method) and multicasts it to the remaining devices in the group, namely $D1$, $D3$, and $D4$.

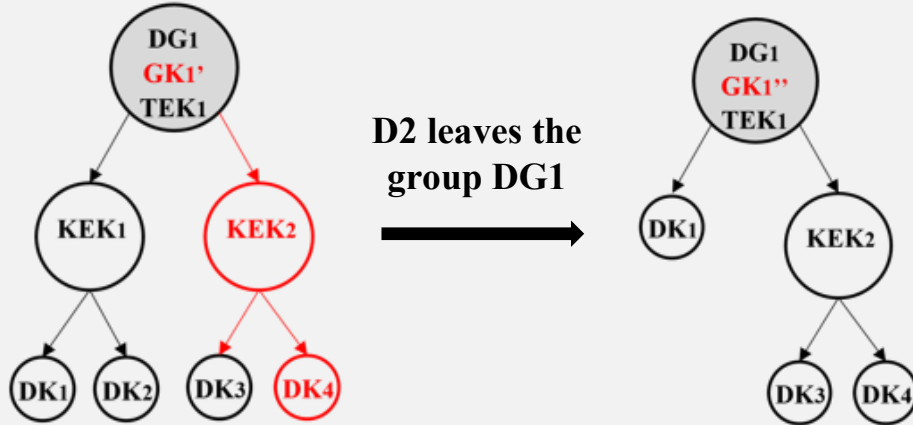


Figure.4. 7: Examples of LKH structure updates for device leave.

ii. OTP Cryptography Mechanism

In practice, an IoT device does not only collect data but also may read data from other devices. For that, an IoT device could also subscribe to other devices and get access to data. Since the subscribers change their interest very frequently, we introduce a modified OTP key distribution structure suitable for very constrained resource devices to ensure the desired

security level and the efficiency of our scheme. We present in this subsection a full description of the rekeying operations of DLGKM-AC after an IoT device joins/leaves a group as below:

a. When an IoT Device Joins a Group

A new joining IoT device to an existing IoT device group should not learn anything about exchanged group communication before joining. Therefore, to guarantee backward secrecy, it is crucial to handling a rekeying process and update the group key. Indeed, we propose an improved OTP protocol to update the group key among the IoT device group securely. The different steps of the OTP based protocol are defined and presented in Figure 4.8 as follows:

- The manager of the device group establishes a secret key DK_j with the new device D_j and generates a random long-term key s chosen from a large set of bits and computes the related information u of the new IoT device $u = DK_j \oplus s$, necessary to update the old group key GK_{old} .
- Then, the manager encrypts u with the current group key and broadcasts it securely to the existing IoT devices in the group under its control.
- All existing IoT devices can decrypt this data u encrypted with the GK_{old} , except the new one as it does not hold the GK_{old} . Then the existing IoT devices could update the group key through computing $GK_{new} = GK_{old} \oplus u$.
- After, the manager updates the old group key GK_{old} and finally generates an identity ID_j , computes the $A[j] = \sum_{i \neq j}^n DK_i \oplus ID_j$ for further updating group key, and sends to the newly joined device.

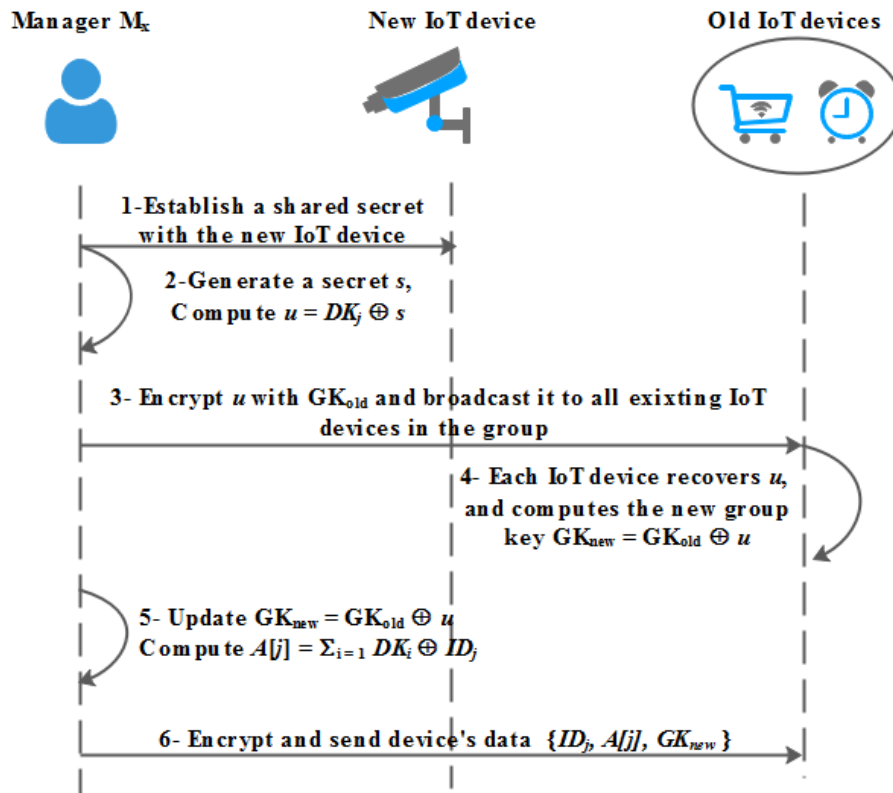


Figure.4. 8: Rekeying procedure based on OTP when a device joins a group

b. When an IoT Device Leaves a Group

When an IoT device wants to leave a group to which it belongs, it should not be able to learn any more about future keys after its departure. This is achieved as follows: Let D_j the node that leaves the group DG_x . The manager M_x of DG_x securely and randomly generates the key value s' and computes the corresponding one-time-pad value $u' = s' \oplus DK_j$. Then it updates $A[i] = s'' \oplus ID_i$ for each IoT device D_i still in DG_x , while $A[j]$ of the left device is set to Null. After that, the manager broadcasts updated group information A , and each legitimate IoT device in the group recovers s'' and derives the new GK_{new} . The protocol steps are figured in Figure.4.9.

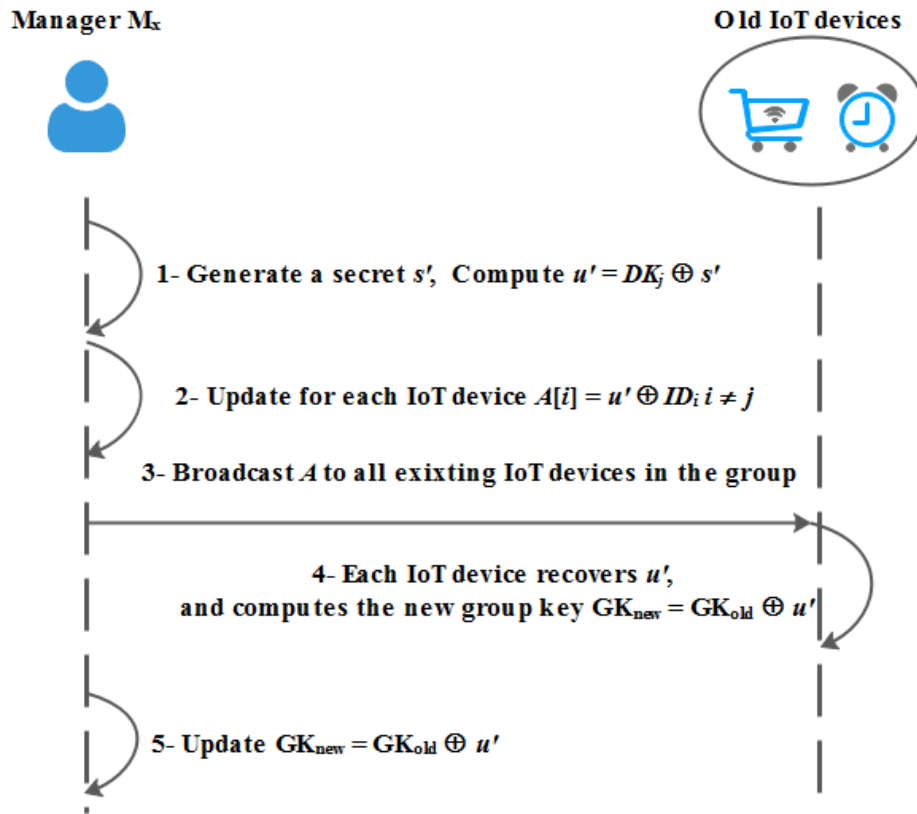


Figure.4. 9: Rekeying procedure when a device leaves a group

In the proposed scheme, users are not affected by IoT device movement (join/leave) in a group, which is explained by the use of a manager for each device group. The manager maintains the traffic key TEK to encrypt data of the group to subscribers during DG life. Otherwise, only when a DG_x is deleted, KDC broadcasts to the subscribers that the corresponding traffic key TEK_x is no longer useful.

4.6. Security Analysis

In this section, we prove the effectiveness of the proposed scheme DGKM-AC in terms of forward, backward secrecy and resistance to the collusion attack.

4.6.1. Forward Secrecy

In the proposed DGKM-AC scheme, we provide the forward security property to both users and IoT devices. We detail and analyze through proofing the two theorems as follow:

❖ **Theorem 1:** *The proposed group key management scheme between SKDC and users provides forward security against an adversary. In other words, the revoked user cannot get access to the ongoing communication.*

Proof: Consider the case that the key pair (e_j, d_j) should be revoked when user U_j leaves the group UG_K . The SKDC updates its master key e_M and the corresponding master token e_{M_K} . At this level, the master token of the left user group UG_K satisfies equations 4.1 and 4.2:

$$P^{e'_{M_K}} \equiv P^{e_i} \bmod (p_i q_i) \quad (4.1)$$

$$C^{d'_{M_K}} \equiv C^{d_i} \bmod (p_i q_i); \forall i \in [1, r_K], i \neq j \quad (4.2)$$

Besides, the data source specifically the plaintext P is encrypted using the master key encryption: $P^{e'_{M_K}} \bmod (\prod_{i \neq j=1}^N \phi(p_i q_i)) = C^*$. After receiving the new ciphertext C^* , each user in the group can decrypt it with its individual private key $C^{*d_i} \bmod (p_i q_i) = P, \forall i \neq j$. Although the left user from UG_K knows the old keys (e_j, d_j) , he/she cannot obtain the correct plaintext from the ciphertext C^* through the old keys and get a false plaintext different from the sent plaintext $C^{*d_j} \bmod (p_j q_j) = P^* \neq P$.

❖ **Theorem 2:** *The proposed group key management scheme between KDC and IoT devices provides forward security against an adversary. In other words, the revoked IoT device cannot get access to the current communication.*

Proof: At this level, we suppose A_1 be an adversary who colludes with the left IoT device D_j in the device group DG_K . In particular, the adversary A_1 may obtain all information stored in left IoT device $(DK_j, GK_K, TEK, KEKs)$ and wants to derive the current group key, GK'_K . After the IoT device is revoked, KDC is responsible for updating the LKH tree of DG_K , similarly updating the path key from the revoked leaf node to the root node $\{KEK_i \in PK_t \text{ of } D_j\}$, which are used to encrypt and broadcast the new group key GK'_K to the remaining devices. However, A_1 cannot decrypt the rekeying messages and get GK'_K , as all key encryption keys are updated in the LKH tree.

Furthermore, based on the OTP mechanism, after an IoT device leaves a group DG_K , the manager updates the OTP value and the array. In fact, it computes the new $A[i] = s'' \oplus ID_i$ for each IoT device D_i still in DG_K , while $A[j]$ of the left device is set to Null. Hence, an adversary A_1 , who wants to collude data with the left IoT device, using the value $A[j]$ cannot compute the new GK'_K . Thus, a left node cannot compromise the whole network and learn about future communications, which proves that our protocol provides forward secrecy in DG .

4.6.2. Backward Secrecy

In this section, we analyze the backward security property to both users and IoT devices through proofing the two theorems below:

❖ **Theorem 3:** *The proposed group key management scheme between SKDC and users provides backward security against an adversary. In other words, the newly joined user cannot get access to previous communications.*

Proof: Suppose a new user U_j is joining a group UG_K with the key pair (e_j, d_j) . The previous data source P is encrypted as follows using the master token $P^{e_{M_K}} \bmod (\prod_{i \neq j=1}^N \phi(p_i q_i)) = C$, where $\forall i \in [1, r_K]$ and $i \neq j$, the master token satisfies equation (4.1 and 4.2):

$$P^{e_{M_K}} \equiv P^{e_i} \bmod (p_i q_i), \text{ and } C^{d_{M_K}} \equiv C^{d_i} \bmod (p_i q_i);$$

After the new user joining the system, the SKDC updates its master key e_M and the corresponding master token e_{M_K} of UG_K , where $\forall i \in [1, r_K]$ and $i \neq j$, e'_{M_K} satisfies:

$$P^{e'_{M_K}} \equiv P^{e_i} \bmod (p_i q_i), C^{d'_{M_K}} \equiv C^{d_i} \bmod (p_i q_i);$$

At this level, the user joining the group UG_K with the keys pair (e_j, d_j) , cannot obtain the correct previous plaintext from the ciphertext C through the new keys as:

$$C^{d_{M_K}} \bmod (\prod_{i \neq j=1}^N \phi(p_i q_i)) = P, \forall i \in [1, r_K] \text{ and } i \neq j$$

$$C^{d_j} \bmod (p_j q_j) \equiv C^{d'_{M_K}} \bmod (\prod_{i=1}^N \phi(p_i q_i)) = P^* \neq P, \forall i \in [1, r_K] \text{ and } i \neq j$$

We conclude that our scheme offers the backward secrecy security property when a new user joins the system.

❖ **Theorem 4:** *The proposed group key management scheme between KDC and IoT device provides backward security against an adversary. In other words, the joined IoT device cannot get access to the previous communication.*

Proof: Suppose a new IoT device D_j joining the device group DG_K and has the new keys $(DK_j, GK'_K, TEK, KEKs')$. After that, the KDC updates the LKH tree of DG_K , similarly updating the path key from the joined leaf node to the root node $\{KEK_i \in PK_t \text{ of } D_j\}$, which are used to encrypt and broadcast the new group key GK'_K to the existing devices. Meanwhile, knowing the secret key, the new GK'_K , and path keys, the newly joined device cannot derive anything about the previous group keys.

Furthermore, using the OTP mechanism, the manager computes a new OTP value u and sends it for the existing IoT devices encrypted with the previous group key GK_{old} . At this point, each existing IoT device could decrypts the OTP value u and computes the new group key as follows: $GK_{new} = GK_{old} \oplus u$, while the new one gets only the new information of the group key. Thus, knowing the new group key GK_{new} , ID_j and the corresponding $A[j] =$

$\sum_{i=1}^n DK_i \oplus ID_j$, the new joined IoT device could not compute GK_{old} as the OTP value u is encrypted with GK_{old} .

We hence prove that our scheme offers the backward secrecy security property when a new IoT device is joining the system for both LKH and OTP mechanisms.

4.6.3. Collusion Attack Analysis Using Random Oracle Model

In this section, we analyze the resistance of our solution to the collusion attack, and we prove that using the Random Oracle Model (ROM) standard [157].

❖ **Theorem 5:** *The proposed GKM is secure against collusion attack.*

Proof: Let G^{cr} be the adversarial game for collusion resistance. This game is played between two adversaries: one acts as the challenger C^{cr} who interacts with the adversary A^{cr} trying to win C^{cr} . It is worth noting that C^{cr} can simulate all the oracles O^{join} , O^{leave} , $O^{ciphertext}$ and $O^{decrypt}$ functions and output signed messages as a real signer. G^{cr} consists of the following phases:

Setup: C^{cr} runs the **MTokenGen** algorithm for a random choice of ID by A^{cr} . Rekeying operation is simulated after that, and the timeline is started ($t=0$).

Queries: It can query the oracle O^{join} , O^{leave} , $O^{ciphertext}$ and $O^{decrypt}$ to control group dynamicity.

Challenge: A^{cr} issues one challenge query to C^{cr} at time $t_{challenge}$ (which is the choice of the A^{cr}). Before responding to the challenge, C^{cr} retrieves the set challenge $S_{challenge}$ from the list L_s , and forms the list of leaving members L_g , for all $ID \notin S_{challenge}$. Then, for each identity $ID \notin S_{challenge}$, C^{cr} issues the query $O^{extract}(ID)$ to obtain S_{ID} . Besides, C^{cr} encrypts $(TEK, ST, S_{challenge})$ to get $(A_{i,b}', TEK')$, where $A_{i,b}'$ defines the authorized receivers of TEK challenged with C^{cr} . After, C^{cr} chooses a bit $b \in \{0, 1\}$ at random and sets K_b to TEK' and K_{b-1} to a random TEK from the key space. Finally, it challenges with $(A_{i,b}', K_0, K_1)$.

Guess: A^{cr} outputs a bit $b' \in \{0, 1\}$ as its guess. C^{cr} passes on b' as its guess to A^{cr} .

The adversary advantage in winning the game is defined as $Adv_{GKM}^{cr} = \left| pr[b' = b] - \frac{1}{2} \right|$; Hence, we can see that the advantage that A^{cr} breaks the collision resistance of GKM is the same that C^{cr} breaks chosen-ciphertext attack (CCA), meanwhile, breaks the encrypted messages. Thus, if there exists no adversary who can break CCA security with non-negligible advantage, then there cannot be any adversary A^{cr} , who can break the collision resistance of GKM with non-negligible probability.

4.7. Performance Analysis and Evaluation

In this section, we analyze the proposed DLGKM-AC scheme's performance in terms of storage overhead, computation overhead, and communication overhead. Then, we compare the results with existing methods in the literature. We also discuss the time complexity to renew the master token and revoke the slave token of the proposed MTE algorithm for communication with users in the same group.

4.7.1. Performance Analysis

In this subsection, we present the performance analysis of the proposed DLGKM-AC for IoT environment. We ensure the analysis in terms of different metrics such as storage, computation, and communication overhead. In order to guarantee generality, we assume that IoT devices are equally distributed in each device group, and the LKH structures are all balanced binary trees.

4.7.1.1. Storage Overhead

The storage overhead is an expensive metric of any access control scheme in the IoT environment as it is based on the memory capacity required to store the keys. In the proposed scheme, the storage overhead is formulated at each user in UG_x and each device in DG_y .

A user belonging to the user group UG_x has a slave token ST , an Asymmetric Key (AK), many traffic keys $TEKs$, Symmetric Keys (SK) equal to the number of DGs for which UG_x is subscribed, and his secret key shared with SKDC. Hence, we can calculate the storage of keys for each user in UG_x using equation 4.5:

$$SO_{U \in UG_x} = AK + (\sum_{i=1}^M A_{i,b} + 1)SK \quad (4.5)$$

In addition, the analysis of a single d -degree key tree accommodating n member requires the tree depth denoted by $f_d(n)$. It is known that $f_d(n)$ is either L_0 or $L_0 + 1$, where $L_0 = \log_d(n)$. The authors of [112] made useful inequality (4.6) in order to analyze the storage overhead for key trees:

$$E[f_d(n)] \leq E[\log_d(n)] + 1 \leq \log_d E[n] + 1, \quad (4.6)$$

where the expectation, $E[\cdot]$, is taken over the distribution of n devices and the length of the branches on the key trees.

A device belonging to DG_y , containing n devices, has a traffic key TEK , a group device key GK , and as many symmetric keys, including the KEKs and the individual key, as the length of the branch. Since we consider that devices are distributed in binary trees, we can calculate the number of keys for each device in DG_y using equation 4.7:

$$SO_{D \in DG_y} = (\log_2 n + 3) \times SK \quad (4.7)$$

With regard to storage using the OTP mechanism, each node needs only to store its private ID_i , DK_i pair, the corresponding array entry $A[i]$ and the group key GK_x . These are all n -bit sequences, which are easy to save even on devices with limited storage capacity.

4.7.1.2. Computation Overhead

The computation overhead can be measured as the total time consumption for encryption and decryption cost and processing requirements. We can measure the cost of computation on the server, user as well as on the device sides, after each member (user/ IoT device) joining or leaving events. We explain the different necessary computation operations as follows:

➤ **When a user joins a subgroup UG_x :** The SKDC assigns a slave token to the new joining user U_{join} and updates the master token of UG_x and its master key. The new user needs one symmetric decryption to gain the slave token ST , new TEK , and all DKs of the devices in the device groups to which he is subscribed. An existing user needs to do one hash function to update TEK . Finally, the devices need to perform one hash function to update their TEK and another hash function to derive their new device keys DK .

➤ **When a user leaves a subgroup UG_x :** The SKDC needs to update the master token of UG_x and its master key in order to send TEK securely to users. The remaining users need to perform one asymmetric decryption and one symmetric decryption to gain the updated information. Devices, to which user groups are subscribed, need to do one symmetric decryption to obtain the update information TEK and DK .

➤ **When a device joins a device group DG_y :** The existing devices of the left device group, based on the LKH structure, require to do one hash function to update the device group key GK_y . Moreover, the LKH tree structure will change, and some devices need to decrypt $O(\log(n))$ KEK updated messages. Besides, the new device needs only to decrypt one message sent from KDC to obtain $KEKs$. In contrast, when using the OTP mechanism, the existing devices require to do XOR operation to get the new group key, and update the array. Finally, users subscribed to the group joined by the new device require to decrypt the message sent by KDC to gain the new device key.

➤ **When a device leaves a device group DG_y :** The remaining devices execute one symmetric decryption to gain the new group key in the LKH structure, while they need only XOR operations to get the new group in OTP mechanism. Moreover, users subscribed to the leaving groups, do not need to perform extra computation.

4.7.1.3. Communication Overhead

The encrypted data and keys should be transmitted to the users and/or IoT devices after each join/leave event. In this context, the communication overhead is mainly associated to the number of transmitted messages during the dissemination keys process. We analyze the communication overhead of the new DLGKM-AC for the IoT environment, as shown in Table.4.4:

Table 4. 5: Communication analysis.

Events		Communication cost
User leave's event		SKDC broadcasts the new TEK and DK to subgroups KDC sends $\log(n)$ messages to devices
User join's event		SKDC unicasts a message to the new user SKDC notifies all users to update TEK
Device join's event	LKH	KDC unicasts a message to the new device

		KDC broadcasts the subscribers with the new DK
	<u>OTP</u>	Manager unicasts access keys information to new device Manager broadcasts OTP value to existing devices in group
Device leave's event	<u>LKH</u>	KDC notifies the subscribers that the leaving device is no longer available. KDC multicasts $\log(n)$ messages for the remaining devices to update group key.
	<u>OTP</u>	Manager notifies the subscribers that the leaving device is no longer available. Manager broadcasts the new array A to remaining devices in group

4.7.2. Performance Evaluation

In this section, we present the experimental results of DLGKM-AC scheme developed on MATLAB. We evaluate DLGKM-AC performances in terms of storage, computation, and communication costs caused by rekeying process. The rekeying transmission overhead corresponds to the additional signaling load after each join/leave event. For that, we compare the new proposed DLGKM-AC scheme with two other key management solutions designed for access control between subscribers and publishers; a centralized scheme that supports groups of publishers (GroupIT [113]) and a decentralized scheme that does not support groups of publishers (SMGKM [123]).

4.7.2.1. Storage Costs

The storage cost of the proposed scheme is formulated at both sides, user and IoT devices. In order to achieve a comparable security strength, we assume the symmetric encryption/decryption key length to AES-256 bits, the ECC-512 decryption key length to 512 bits. We consider computing the storage cost at the user side through two different scenarios:

❖ **Scenario 1:** *in this scenario, we vary the number of publishers DGs to which users are subscribed while fixing the number of users per user group to 20 users for example.*

Through Figure.4.10, we notice that, unlike existing solutions such as GroupIT [113] and SMGKM [123], our scheme is less affected by the increase of DGs number to which users are subscribed. Indeed, the user in our scheme *DLGKM-AC* requires less memory storage, even if the number of IoT devices to which the user is subscribed is high. In fact, comparing to the GroupIT [113] scheme, we proposed a decentralized architecture in which keys are less stored on IoT devices, while SKDCs take the responsibility of keeping traffic keys. Besides, compared to SMGKM [123], our *DLGKM-AC* ensure the grouping of IoT devices, where users need only to store the traffic key of all the group, not all the traffic keys of each IoT device.

- ❖ **Scenario 2:** in this scenario, we vary the number of users in each UG and consider the number of DGs set to 4 and the number of devices set to 20 IoT devices per group DG.

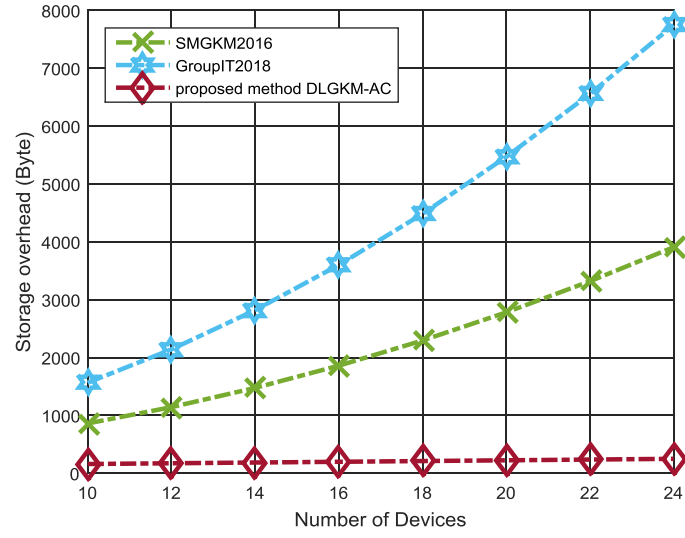


Figure.4. 10: Users' storage overhead while varying the number of devices

Figure.4.11 shows that, in GroupIT [113] and SMGKM [123] schemes, when the number of users per group increases, the storage on users rises too, explaining that the rekeying in user group affects all users. In particular, the larger the number of users in each UG, the more these schemes incur users' storage overhead. In our scheme, we ensure the dissemination of keys in the user group through the proposed MTE mechanism, which is not sensitive to the number of users in each UG. Hence, the total number of users in the user group UG does not affect the storage on each user and can reduce the storage overhead per-user more efficiently.

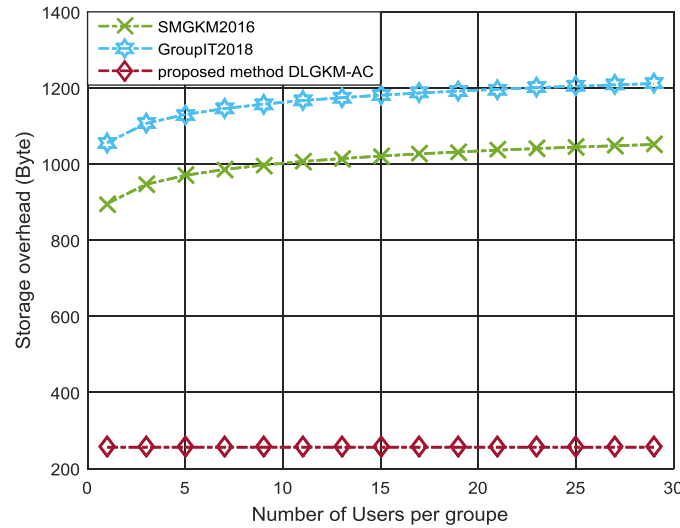


Figure.4. 11: Users' storage overhead while varying the number of users

At this level, we study the storage on devices when varying the number of devices. As mentioned in the analysis section, our proposed scheme is not affected either by the number of users or by the number of devices in other different *DGs* because devices are considered as data publishers in the LKH structure, while they are only affected by the number of devices of their group. Figure.4.12 shows that SMGKM and our scheme have mainly the same storage. Otherwise, GroupIT does not hold the notion of grouping the devices (publishers), and the storage on devices (publisher) is not affected by the number of devices in the same group.

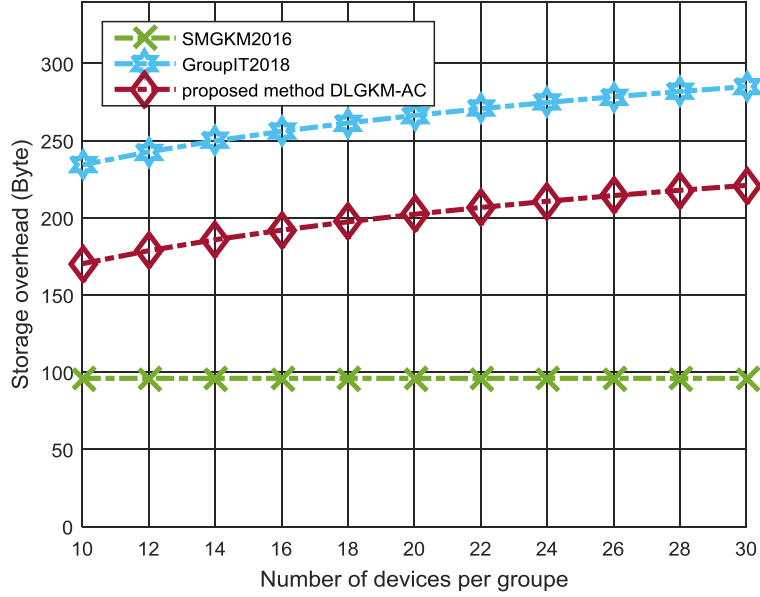


Figure.4. 12: Devices storage overhead

4.7.2.2. Computation Cost

We simulate the cryptographic operations with Miracle Library [158], which is a cryptographic library designed for use in constrained environments in terms of computational power [120]. All simulations are implemented on a computer with the following features: an Intel i5-4200 CPU@ 2.5 GH with a physical memory of 8 GB; and Ubuntu 12.04 OS over VMware workstation 15. We provide the time cost for different cryptographic operations. As a result, we define $T_h = 2,445\mu s$ be the time for one hashing operation using SHA-256 function on a 64-byte block. Then, $T_{Enc} = T_{Dec} = 2,7\mu s$ be respectively the time for one encryption/decryption operation using symmetric cryptography AES-256 encryption on a 64-bytes, and $T_{ECC} = 365,63\mu s$ represents the time for one elliptic curve cryptographic operation.

Since our protocol is designed for a dynamic IoT environment, the computational cost is measured based on leave and join operations of both users and devices. We detail the computation on users as well as on IoT devices and servers in the following subsections. We start by the computational costs triggered by user leave/join events before comparing the rekeying's cost triggered by device join and leave operations.

4.7.2.2.1. When a User Leaves a Group

Consider a user U leaving the user group UG_K . At this moment, the left user is not allowed to obtain the rekeying message no more to ensure the forward secrecy. In particular, we compare the computation cost through different cases on the remaining users and on the SKDC as follows:

✚ Computation cost on the remaining users' side:

We consider two cases:

- ❖ **Case 1:** In the first case, we vary the number of publishers DGs to which users are subscribed, while fixing 20 users per UG.

Figure.4.13 depicts that our scheme is not hugely impacted with the number of DGs to which users are subscribed compared to the state of the art solutions. More specifically, when a user leaves a user group, the remaining users need to update the traffic key of the data. In fact, in our proposed scheme each SKDC is responsible to update this key and disseminate it to users through MTE mechanism. At this level, these remaining users need only to decrypt one message to get the new traffic key. Thus, we can ensure that a decentralized architecture reduces the computation overhead resulting after a leave event such an IoT environment.

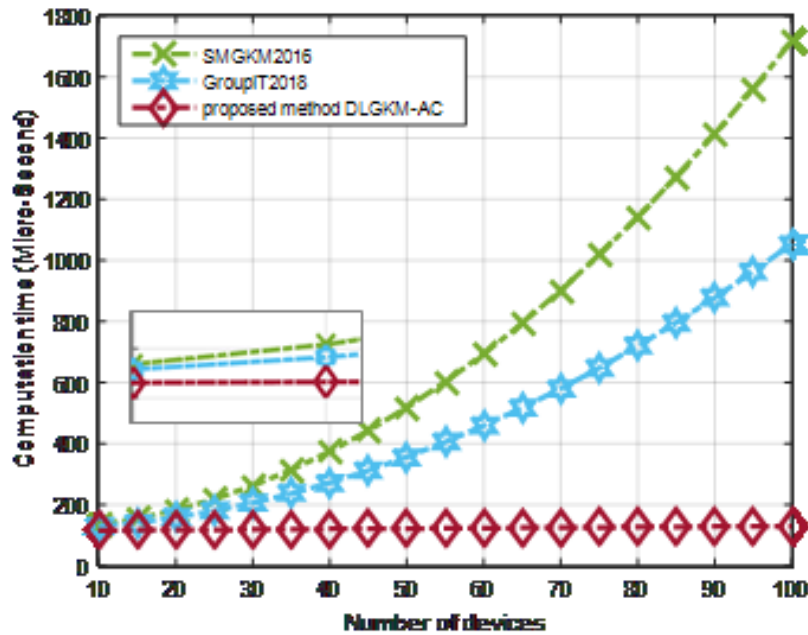


Figure.4. 13: Remaining user computation overhead varying devices' number (user leave)

- ❖ **Case 2:** In the second case, we modify the number of users in each UG and consider the number of DGs is fixed to four and the number of devices per group DG is fixed to 20.

Similarly, in Figure.4.14, we plot the computation cost on remaining users after a user leaving the group with varying the number of users. In this figure we note that the more the number of users in each UG is large, the more the computation overhead is high for GroupIT and SMGKM schemes. This explains that each user depends on all other users in the same group, while users in the proposed scheme are not affected with the number of users. The proposed MTE algorithm, managing communication within user groups, guarantees that users in the same group can get the updated group key with only one decryption. This explains why our proposed system has low computational cost while having a high number of users in UG.

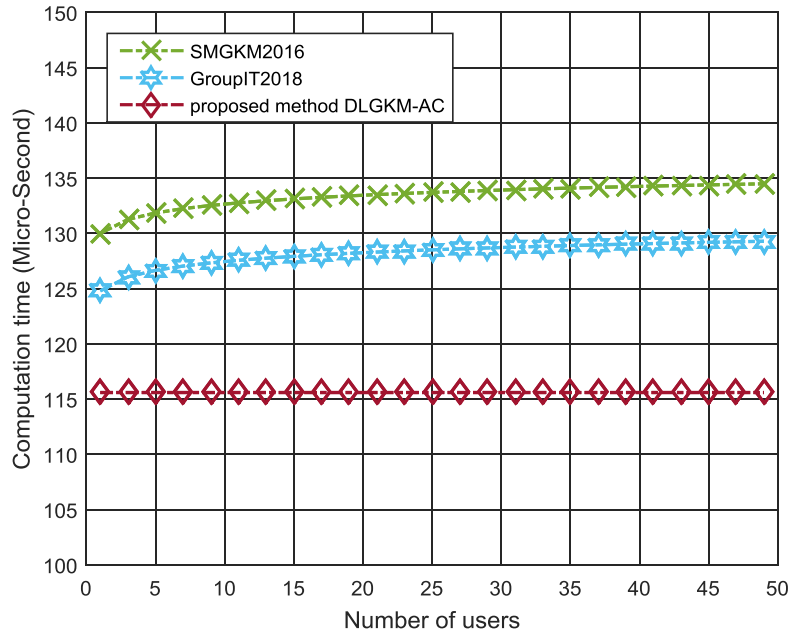


Figure.4. 14: Remaining user computation overhead varying users' number (user leave)

✚ Computation on the server-side:

At this level, we consider the group key updating time of SKDC to prove the efficiency of our master token encryption MTE for updating keys compared to traditional master key encryption mechanisms. Figure.4.15 shows that, compared to the traditional master key encryption MKE algorithm, our solution consumes less time for the key updating when a user is revoked. In fact, the traditional MKE needs to repeat all the steps of *Algorithm 1* when updating the master key. Otherwise, the MTE algorithm proposes only two operations to get the newly updated master token.

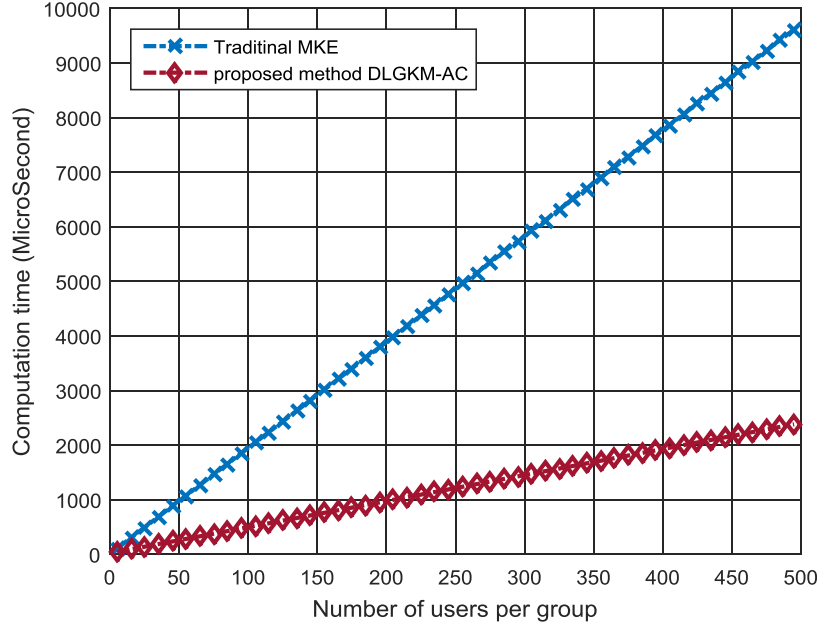


Figure.4. 15: Server time update on the user-leaving event

4.7.2.2.2. When a User Joins a Group

Assume a user U is joining a user group UG_K . As the new user U should not be allowed to access previous communications, the rekeying operation is triggered. Hence, we compare the updating overhead when a user joins a group as follows:

✚ Computation cost on old users' side:

We also consider two cases:

- ❖ **Case 1:** In the first case, we vary the number of publishers DGs to which users of UG are subscribed while fixing 20 users per UG .

In Figure.4.16, we present a comparison of the computation cost for the existing users in the joined user group UG while varying the number of devices to which UG is subscribed. The results show that the centralized architecture requires more rekeying operations compared to the decentralized architecture. Besides, Figure.4.16 shows that the computation cost of *DLGKM-AC* varies very slowly with the number of the publisher compared to the literature GroupIT and SMGKM. In fact, when a user joins a group, the existing users need only to update the traffic keys TEK of the corresponding publishers. In contrast, the other schemes need to update the keys of the user group. This outcome is mainly explained by using subgroup controllers SKDCs to manage the key updating process for each user group and thus reducing computation for end-users.

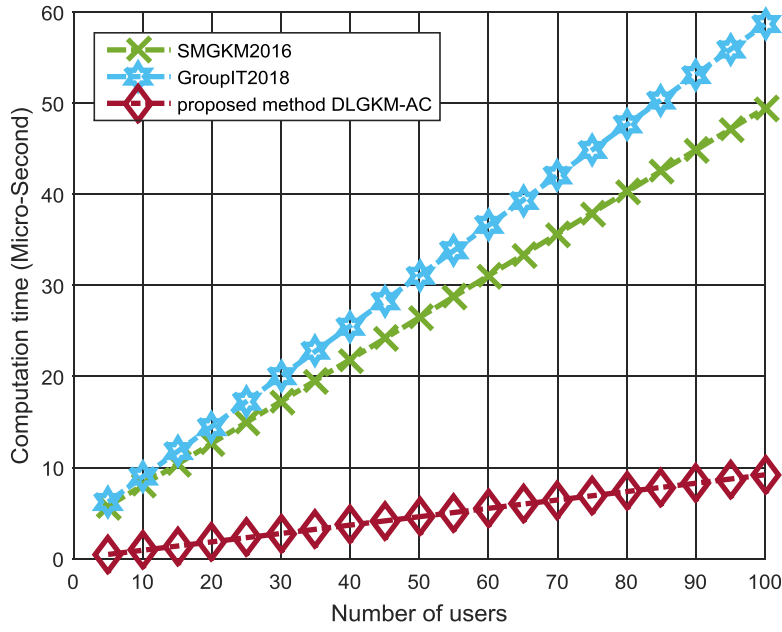


Figure.4. 16: Old user computation overhead varying the devices' number (user join)

- ❖ **Case 2:** In the second case, we vary the number of users in each user group UG and consider four devices groups DGs and 20 devices per group DG .

Similarly, the Figure.4.17 depicts the comparison of the computation cost on the existing users, but when varying the number of users per group. Indeed, we notice that the number of users in the same group does not affect the existing user in the group in our proposed scheme. However, the shape of the other schemes, GroupIT and SMGKM, is increasing with the number of users. It is evident that the new joining user affects all members of the user group, which is explained through using an LKH structure in

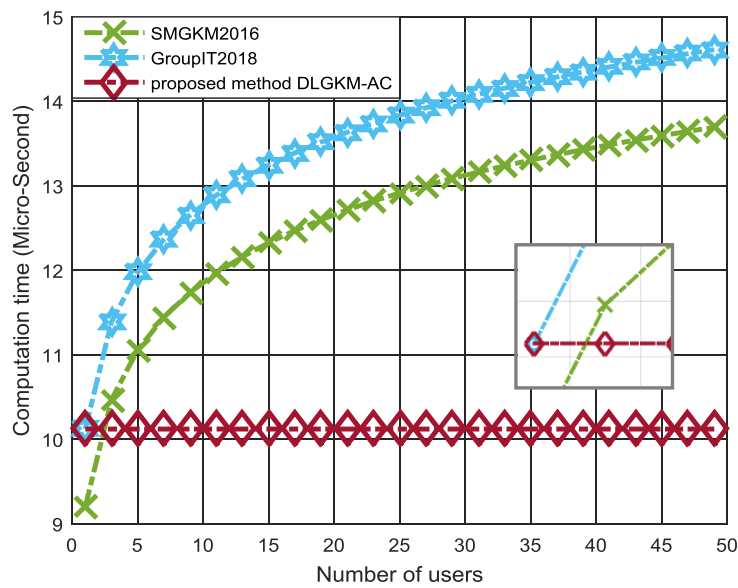


Figure.4. 17: Old user computation overhead varying the users' number (join)

GroupIT and SMGKM while using MTE as key management for user groups in DLGKM-AC.

✚ Computation cost on new users' side:

At this level, we analyze the new user's computation cost in Figure.4.18. Actually, when a new user joins a group of users, the computation cost of our DLGKM-AC and SMGKM is almost negligible compared to GroupIT. These results are explained by the fact that the new user needs only to decrypt received messages to get necessary information. While in GroupIT, a new user needs to compute the device keys to which he/she is subscribed.

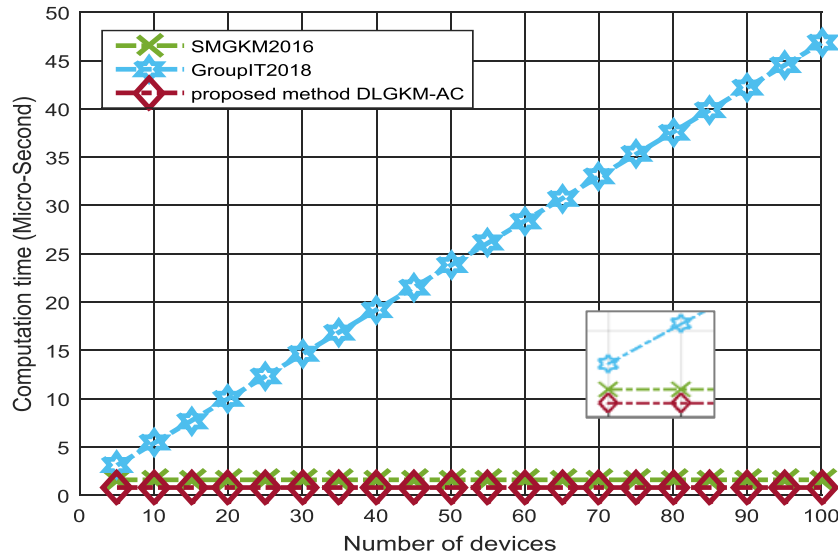


Figure.4. 18: New user computation overhead varying devices

✚ Computation cost on the server-side:

In order to prove the efficiency of the proposed master token algorithm, we plot in Figure.4.19 the average time to update keys when there is a user joining operation. More specifically, the time needed to execute the **JoKeyUpdate** algorithm when varying the number

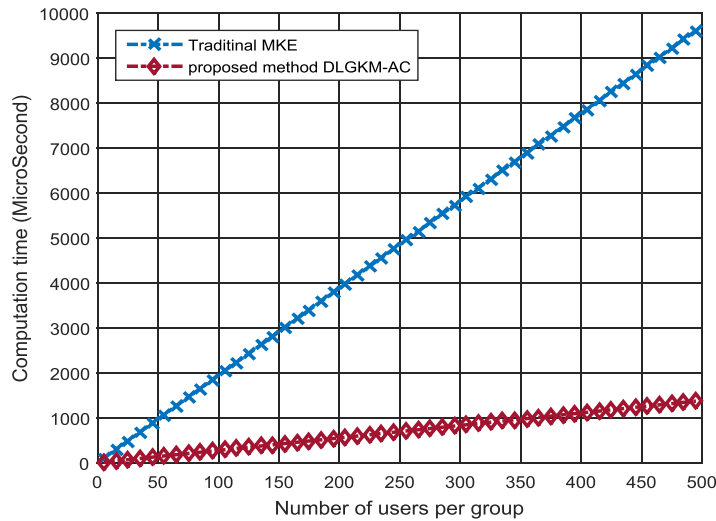


Figure.4. 19: Server time update on the joining event

of users per group. Unlike the traditional master key encryption MKE, the execution time of our scheme increases slowly with the increase of the number of users per group.

4.7.2.2.3. When an IoT Device Joins a Group

Figure.4.20 plots the computation cost triggered by a device joining a device group when varying the number of IoT devices in the group. In fact, we measure the overhead on both the existing devices and the new device sides. We notice that the computation cost of the existing devices in our scheme is less impacted than GroupIT by the number of IoT devices. In contrast, the new device in *DG* has the same cost to get the updated keys. Moreover, the SMGKM is not impacted by varying devices in the group, as they do not consider grouping devices.

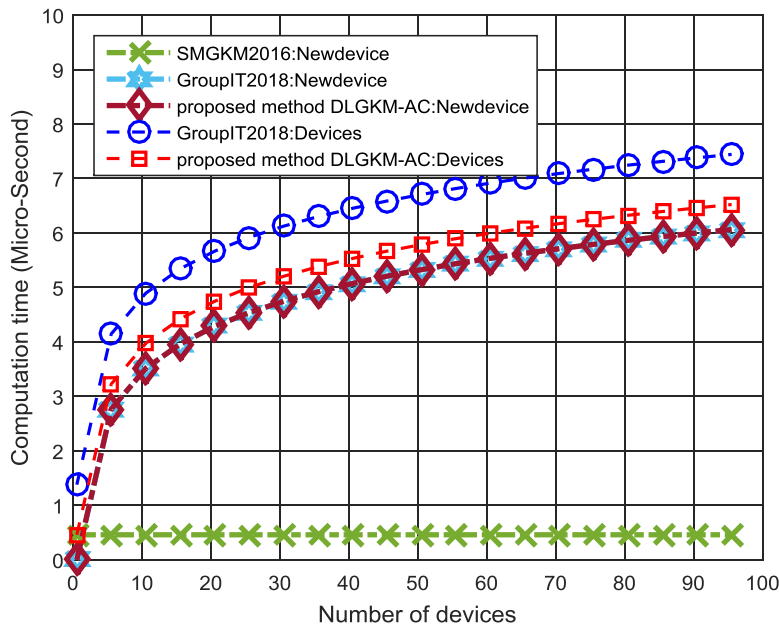


Figure.4. 20: Computation cost: device join

4.7.2.2.4. When an IoT Device Leaves a Group

Figure.4.21 shows the computation cost when a device leaves a DG, varying the number of IoT devices per group. The cost is measured on both the group's remaining devices and the users subscribed to the DG. In our scheme, the users' computation cost is not affected by the leaving device's operation, while, in SMGKM, it increases with the number of devices. The advantage of grouping devices explains this result. Moreover, the remaining devices in our scheme have less computation cost compared to GroupIT, which is explained through using a decentralized scheme, where KDC reduces the load on devices.

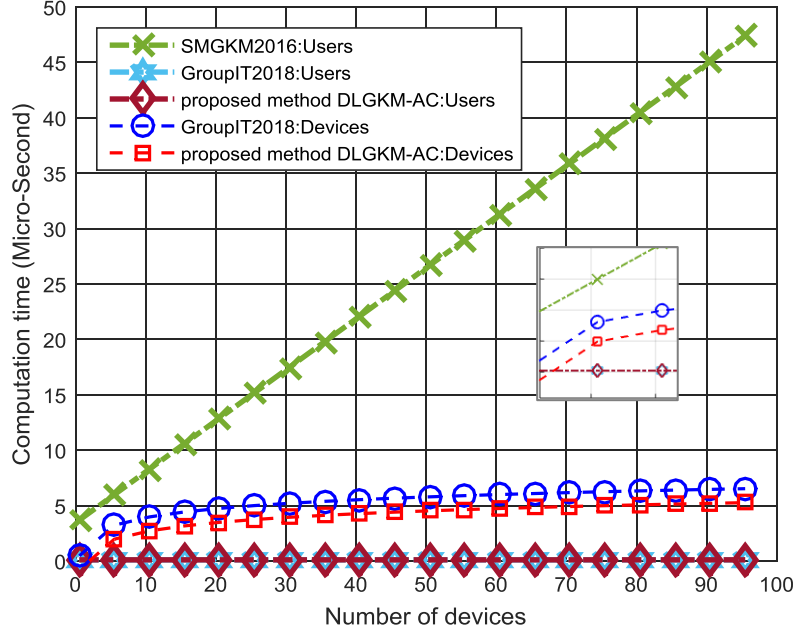


Figure.4. 21: Computation overhead: device leave

4.7.2.3. Communication Cost

The communication cost of DLGKM-AC for the IoT environment is evaluated based on the number of updating keys messages transmitted during user joins/leaves events. Figure.4.22 (a) and (b) plot the communication cost after the user joins and leaves events when varying the number of IoT devices to which the user is subscribed. It is evident from Figure.4.22 that GroupIT and SMGKM schemes are affected by the number of devices, and it causes many rekeying messages when a user joins/leaves a user group. Therefore, our scheme incurs much less communication overhead, which is explained by grouping the devices and introducing MKE for grouping users. In particular, regarding the use of the master token encryption methodology for communication with groups of users, our scheme DLGKM-AC decreases the

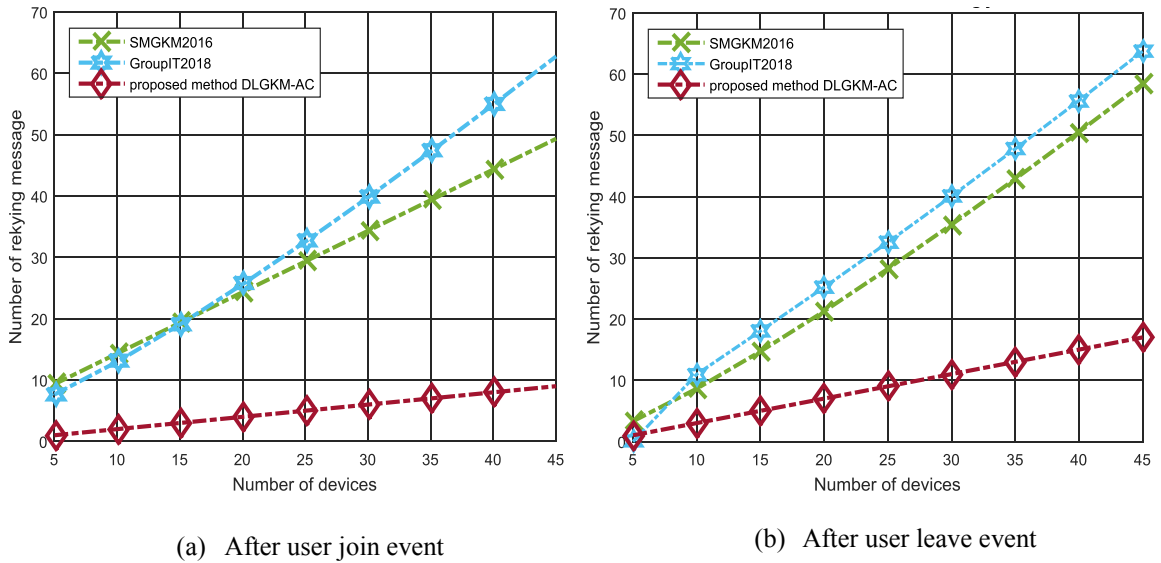


Figure.4. 22: Communication costs

unnecessary rekeying operations. Indeed, the proposed DLGKM-AC maintains the lowest communication cost.

4.8. Conclusion

The main objective of this chapter is to define a scalable, generic, and lightweight group key management (GKM) for access control in the IoT environment. For that, we introduced a new access management architecture DLGKM-AC, which alleviates the problem of managing numerous constrained IoT objects. The proposed solution is fully decentralized, which is based on different and separate GKM for users and IoT devices. Besides, a new master token encryption algorithm has been introduced to ensure members' Independence in highly dynamic group communication. Subsequently, we presented an optimized notion of the logical key hierarchy and one-time pad (OTP) to enable a secure group communication within IoT devices. This fusion makes our solution lightweight as it offers the best performance on the user and IoT device side compared to the realized benchmarking studies.

Therefore, DLGKM-AC solution can be perfectly adapted to IoT applications, where devices typically have constrained computational power. Additionally, we handle smoothly the mobility, where both the backward and forward secrecy are ensured with a few keys' updates. Moreover, our solution alleviates the 1-affects-n issue, which is explained when users can always get access to data even if one SKDC is affected. Furthermore, extensive security analyses covering a wide range of desired security properties have also been provided. Additionally, performance analyses show that our proposed scheme offers better performances by reducing storage, communication, and computation overheads. Finally, adopting a decentralized architecture with different GKM makes our scheme more suitable for a dynamic IoT environment, where subscribers change their interest over time frequently. However, even we ensure flexible access control and smooth changing updates, subscribers always need to be authenticated with all IoT devices before joining the system. In the next chapter, we propose a distributed group authentication for subscribers, which offers subscribers a flexible authentication with the ability to choose to non-re-authenticate in the system.

DiGABlock: Distributed Group Authentication based on Blockchain Technology

5.1. Introduction

As thoroughly discussed in the previous chapter, new types of group-based applications have been presented due to the increasing diffusion of the IoT networks. Specifically, many new IoT applications and services are introduced, such as smart hotels, smart grids, and industrial automation based on group communication. However, this may lead to a set of new challenges and concerns. Although an access control for group-based applications scheme is achieved to safeguard IoT data from tampering and unauthorized access, efficient authentication is required for the group-based applications. More specifically, these environments pose a challenge for defining a global standard authentication protocol in IoT networks [159]. Due to the diverse heterogeneous architectures and environments that support IoT devices, there exist numerous authentication mechanisms [52], such as key-based authentication (e.g., public/private key) [160] and knowledge-based authentication (e.g., password) [161][162]. All these user authentication schemes are a one-to-one type of authentication. Hence, users who subscribe to multiple IoT devices will have to store as many authentication data as the number of IoT devices. Therefore, the increase of IoT network communication explains the engagement of IoT network in group-based communication [163].

In this context, a few group-based authentication schemes have been studied. Some of them ensure the group authentication of participants that belong to the same group [164]. Others [165] achieve authentication of one user with a group of IoT devices. In a subscribe-publish IoT system, users may subscribe to many IoT services, where an IoT service is a group of IoT devices. Thus, it is difficult for the user to be authenticated quickly due to an unbounded number of devices and the centralized Trust Third-Party (TTP) authority server [160]. Therefore, the IoT users will suffer from authentication signaling congestion and high network access latency. Besides, this will increase the communication delay and the response time of the authority server. Furthermore, the existing mechanisms manage their databases by a single manager,

making them potentially vulnerable to collusion attacks from malicious nodes who want to infiltrate the system. Consequently, a secure and efficient group authentication that authenticates users with multiple groups of IoT devices minimizes the interaction with the authority server and meets the scalability issue is required.

Numerous distributed authentication mechanisms have been proposed in the literature [166] [174] [175] [176] [177] to respond to the scalability issue required in the IoT environment. In fact, these distributed schemes, based on distributed trust, increase the computation overhead and require multiple interactions among the system with the trusted authorities, causing high communication delays. Furthermore, they cannot resolve the non-repudiation identity problem since it is easy for anyone to set up the so-called trusted identity provider. Recently, few researchers have introduced Blockchain technology for authentication mechanisms [167], which emerged as a prominent solution for IoT security in trustless environments. Blockchain is a distributed ledger composed of many nodes used to protect data information against damaging attacks and alleviate the signaling congestion on the TTP, thus improving the system efficiency. Hence, this technology is especially suitable for delay-sensitive and large distributed IoT applications. All these mentioned features are motivating to explore blockchain technology and design a new authentication scheme. Therefore, we propose a new distributed authentication mechanism named Distributed Group Authentication system based on Blockchain technology (DiGABlock) to build a secure and efficient authentication system in the IoT environment composed of many IoT devices' groups defining many IoT services.

The main idea of DiGABlock is to design a distributed group authentication protocol to allow the users to authenticate within many groups of IoT devices in a distributed manner efficiently and simultaneously. Hence, users who subscribe to numerous groups of IoT devices could perform a full authentication process only once with a group of IoT devices. After that, the user needs only to send requests to be authenticated and access the remaining groups of IoT devices. In particular, DiGABlock avoids the redundant actions of exchanging authentication data and protects users from identity vulnerability. In fact, thanks to the Blockchain technology, DiGABlock is qualified with another significant advantage related to a distributed group authentication mechanism because it resists the distributed denial of service attack (DDoS). Furthermore, DiGABlock is designed to enhance IoT network resources' availability and guarantees users' activity tractability and efficient authentication. Thus, DiGABlock improves the system response time by minimizing the communication overhead caused by the redundant authentication process. Likewise, it reduces the overhead computation time as well as the energy consumption during the users' authentication.

The remainder of this chapter is organized as follows: we briefly describe the related work to group-based authentication and distributed based authentication. Then, we discuss the necessary background related to our scheme. After that, we present the overall system architecture, the attacker model, and the different system requirements. Then, we detail the proposed solution DiGABlock. Finally, we summarize its security and performance analysis in terms of communication, computation overheads and energy consumption.

5.2. Related Works

To secure communication in IoT environments, authentication between two communicating parties is an essential security requirement. Meanwhile, users and devices in IoT must be authenticated for privileged access to IoT services. In the literature, many solutions for authentication and key agreement in IoT environment have been proposed. The diversity of solutions is mainly due to the diverse and heterogeneous underlying architectures and environments that support IoT devices. Authors in [52] surveyed the different existing authentication techniques in the IoT environment. Table 5.1 summarizes a comparison between these authentication schemes according to various criteria. Thus, some authentication schemes are based on: (i) a centralized server authority, while others use a distributed mechanism based on Blockchain technology. These schemes designed different (ii) types of encryption and (iii) key generation mechanisms to ensure secure authentication, which is mainly (iv) an end-to-end authentication or group authentication. Further, we present some of their security features as they almost guarantee mutual authentication and resist the well-known attacks, and we enumerate their weaknesses related to the efficiency and scalability issues.

Authentication is a fundamental security issue in IoT to authenticate the communication between two parties. Indeed, many schemes have proposed a lightweight and secure authentication protocol for one-to-one scenarios in the literature [58]. However, since the continuous growth of the number of connected IoT devices leads to many group applications, one-to-one scenario authentication is costly for these applications and causes new security challenges. Therefore, designing secure authentication for group applications should be addressed to enhance efficiency and flexibility.

Li et al. [170] proposed a group-based authentication protocol based on an aggregate signature scheme, which enables the group leader to aggregate several signatures from distinct group members to a single signature. In addition, the authors of [77] and [78] proposed a threshold authentication protocol to support secure and privacy-preserving communications in VANETs. The protocol uses a group signature scheme for achieving threshold authentication, anonymity, and traceability during vehicles' communication. However, the aggregation signature method is too costly. Lai et al. [76] proposed a group-based lightweight authentication scheme for resource-constrained machine-to-machine communication (GLARM), which is based on defining each member in the group with a code of authentication, and then aggregating the message authentication codes of all members in the group. However, this protocol presents a single point of failure since it needs a group leader to send and respond to messages with the server: if the group leader is unavailable, then the authentication process fails. Lein Harn [171] proposed an improved group authentication scheme (GAS). This scheme exploits Shamir's secret sharing scheme [172] to issue a private token to each group member who participates in the group authentication without the leader. However, an attacker can launch several trials before the secret is recovered to get both the system secrets and the group members' secret tokens. Chien [173] improves *GAS* by publishing simple public data to the group members. Their scheme creates and publishes tokens through the elliptic curve cryptography and bilinear pairing.

All the discussed works are limited to authenticating one group at once. In particular, these works authenticate only one group of participants through a centralized authentication architecture. Otherwise, as more as the IoT applications are extensive, users would get access to many IoT services, which require a new group authentication mechanism. In addition, the use of one trusted authority requires multiple interactions with users for authentication. Hence, the redundant data exchange during the authentication process may lead to exploiting the authentication mechanism and make the system vulnerable to attacks. Besides, these presented works shortage flexibility and scalability. Therefore, with the prevalence of digital cryptocurrency, Blockchain is introduced as a promising solution to provide a scalable and more trusted authentication services with low interactions.

Several Blockchain-based distributed authentication mechanisms have been proposed. Zehui et al. [174] analyzed the advantages of Blockchain in future IoT systems. The authors of [175] proposed an authentication scheme for IoT systems based on Blockchain called Bubbles-of-Trust. This scheme uses the public Blockchain implemented with Ethereum to validate the communication between different devices. Wang et al. [176] introduced a Blockchain-based cross-domain authentication model named BlockCAM to ensure the safety and the efficiency of accessing resources in different domains. Yao et al [177] proposed an improved cross-domain authentication that achieves the non-interactivity feature. Although these schemes ensure data security, they are costly ones. The authors of [178] addressed this problem by designing an efficient Blockchain-based distributed authentication system using the ECC cryptography mechanism.

Nevertheless, the existing solutions for IoT authentication based on Blockchain have achieved numerous security requirements like anonymity, resisting DDoS attacks [176], and increasing authentication efficiency. However, these schemes do not achieve efficient authentication for group communications, where a user needs to authenticate with multiple IoT services composed of many groups of IoT devices. Hence, in the following, we propose a distributed group authentication mechanism based on Blockchain technology DiGABlock. We also use Blockchain edge nodes to define an IoT service, which can offer an edge group authentication and minimize the interactivity caused by the authentication process.

Table 5. 1 : Comparison of existing Authentication Schemes

Schemes	Environment	Key generation mechanism	Authentication type	Strength(+) /weakness(-)
[58]	IoT Environment	Hash & XOR operations and Symmetric cryptography	Centralized authentication mechanism One-to-one & Single authentication	+ Secure against many attacks + Mutual authentication - Limited scalability - High communication overhead for a large IoT environment

[164][165]	IoT Environment	Hash function Symmetric and ECC cryptography	Centralized Group authentication	<ul style="list-style-type: none"> + Group IoT device authentication + Resilience against DoS attack - Limited scalability - Does not support multiple group authentication
[77][19][76]	VANETs	Aggregation signature & Asymmetric cryptography	Centralized Group authentication	<ul style="list-style-type: none"> + Group signature for threshold authentication + Data anonymity & traceability - High computation overhead - Limited scalability
[171][173]	IoT Environment	Shamir ' shared secret & Asymmetric cryptography	Centralized Group authentication	<ul style="list-style-type: none"> + Multiple IoT users authentication + Resist insider collusion attack - Does not support multiple group authentication - High communication overhead - Limited scalability
[176][177]	VANETs	Hash function & Asymmetric cryptography & ECC signature	Distributed Single authentication	<ul style="list-style-type: none"> + Cross domain distributed authentication + Minimize the interactivity feature - High communication overhead - Heavy computational cost
[178]	IoT Environment	Hash function & Asymmetric cryptography & ECC signature	Distributed Single authentication	<ul style="list-style-type: none"> + Distributed mutual authentication + Ensure the data integrity - High communication and computation overhead for group application IoT environment - Vulnerable to insider collusion attack

5.3. Background

In this section, we briefly present the background and the main mechanisms used in our approach. We first describe the Elliptic Curve Cryptography (ECC) asymmetric cryptography technique. Then, we present the Shamir Shared secret scheme *SS* that is used for sharing a secret. Finally, we briefly present the Blockchain technology and the different consensus types.

5.3.1. Elliptic Curve Cryptography (ECC)

ECC, based on the algebraic structure of elliptic curves over finite fields, is an approach used for public-key cryptography. ECC ensures security depending on the ability to compute a point multiplication with a random point, as well as the inability to figure out a multiplicand given the original curve and product points. ECC guarantees the same level of security afforded by an RSA-based system with a larger key [179].

An elliptic curve E is a plane curve over a prime finite field E_p , where all points of the curve E and the infinity point O (obtained when a point of E is multiplied by 0) form a cyclic group G , which is often defined by equation 5.1:

$$y \bmod p = x^3 + ax + b \bmod p \quad (5.1)$$

In a cyclic group, if two E points are added or an E point is multiplied by an integer, the result is another E point from the same cyclic group. In particular, consider two cyclic groups G_1 and G_2 with the same prime order, q . G_1 is an additive cyclic group and G_2 is a multiplicative cyclic group. We define the pairing map: $e : G_1 \times G_1 \rightarrow G_2$ with basic properties of bilinear map for the security proofs as follow:

- Non-degeneracy: for every $P \in G_1$ there exist Q such that $e(P, Q) \neq 1$.
- Bi-linearity: $e(P + R, Q) = e(P, Q) \cdot e(R, Q)$ and $e(aP, bQ) = e(P, Q)^{ab}$, $\forall a, b \in \mathbb{Z}_q^*, \forall P, Q, R \in G_1$.
- Computability: It is efficient to compute $e(P + R, Q)$; $\forall P, Q \in G_1$.

Through this chapter, we mainly use the two next properties of ECC:

- The first property, called Elliptic Curve Diffie-Hellman (ECDH), is an anonymous key agreement protocol that allows two parties that have elliptic curve public-private key pairs to establish a shared secret over an insecure channel [179]. Let G be an additive cyclic group consisting of points on the elliptic curve, and its order is prime integer q . Let P be a generator of G . Given, $P, xP, yP \in G$; ($x, y \in \mathbb{Z}_q^*$) calculating the product of xyP is a hard problem.
- The second property is called Elliptic Curve Discrete Logarithm Problem (ECDLP): Let G be an additive cyclic group consisting of points on the elliptic curve, and its order is prime integer q . P is a generator of G . It is noted that knowing $xP \in G$ and P , calculating x is hard.

5.3.2. Review on Shamir's Secret Sharing Scheme

In cryptography, secret sharing refers to a method for distributing a secret amongst a group of participants by giving each one of them a part of that secret. These parts are called shares. The distributed secret can be reconstructed if a subset of shares is combined. Otherwise, individual shares are of no use on their own. Since the collection of at least k different points can reconstruct a polynomial of degree $(k-1)$, Adi Shamir [172] proposed a scheme for cryptographic systems based on the secret sharing enabling the reconstruction of a parameter from a set of secret shares. Shamir defined a (k, n) threshold scheme, where a secret D is divided into n pieces D_1, D_2, \dots, D_n , and can be recovered by only k pieces ($k < n$) taken randomly from the n pieces. The Shamir's scheme defines a polynomial function $f(x)$ with degree $(k - 1)$:

$$f(x) = a_0 + a_1x + \dots + a_{k-1}x^{k-1} \quad (5.2)$$

where a_0 is the secret D and the n pieces are defined as $D_i = f(i), i = 1, \dots, n$. Shamir guarantees to recover the secret D with a subset of $k \neq 1$ pieces, through a polynomial Lagrange interpolation [180], which can rebuild the polynomial $f(x)$ function through a set of k points $(x_1, f(x_1)), \dots, (x_k, f(x_k))$ as given in equation 5.3:

$$f(x) = \sum_{i=1}^k f(x_i) \prod_{j=1, j \neq i}^k \frac{x-x_j}{x_i-x_j} \quad (5.3)$$

5.3.3. Blockchain - Practical Byzantine Fault Tolerance Consensus Algorithm (PBFT)

The Blockchain is used as a distributed ledger that realizes a decentralized storing of data elements, where each data element is called a block. These blocks are linked in a chronological order to form a chain that is secured using cryptography [181] (each block contains a hash of the previous one). Current Blockchain systems are categorized roughly into three types [182]:

- Public Blockchain is an open network, where anyone can download the protocol and read, write or participate in the network.
- Private Blockchain allows different levels of permissions for users, so access can be restricted, and information can be encrypted to protect confidentiality.
- Consortium Blockchain is permissioned that provides an additional level of security over typical Blockchain systems, as they require an access control layer.

In this contribution, we use the consortium Blockchain to establish a distributed group authentication system. The consortium Blockchain is semi-decentralized since only some nodes would be selected to participate in the consensus and validate the block. In fact, in the context of the blockchain network, the consensus is a distributed process where several nodes cooperate to validate a block of transactions. Indeed, numerous consensus algorithms have been designed for distributed systems [183][184]. Examples are shown in Table 5.2; Proof of Stake (PoS),

Table 5. 2: Comparison of consensus algorithms [27]

Algorithm	PoS	DPoS	Casper	PoET	PBFT
Decentralized	complete	complete	complete	semi	semi
Tokens	yes	yes	yes	no	no
Evil number	51%	51%	51%	51%	33%
Performance	relatively high	high	relatively high	high	high
Technical maturity	mature	mature	not applied	not applied	mature

delegated PoS (dPoS), Casper, Proof of Elapsed Time (PoET) and Practical Byzantine Fault Tolerance (PBFT) [177]. These consensus algorithms have mainly three phases:

- Verifying identity,
- Selecting primary peers,
- Synchronizing data in the Blockchain.

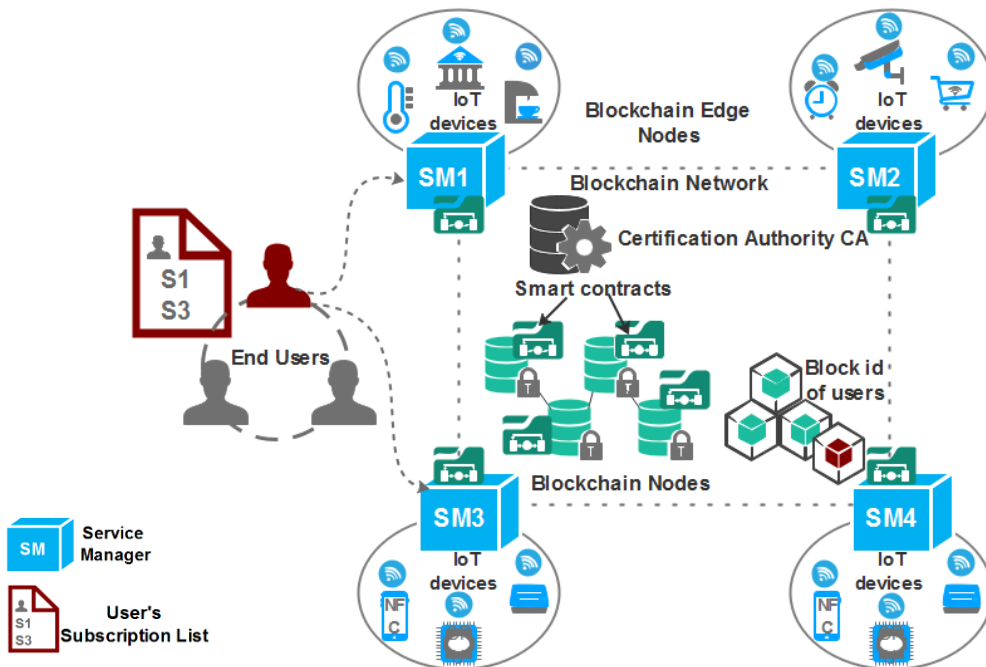
Only the PBFT algorithm assumes fewer malicious nodes compared to other algorithms, less than a third of total nodes. A selected leader orders the transactions and ensures the consensus with the blockchain node (peers) to add blocks to the chain. The PBFT protocol can work in malicious environment where no more than third of total nodes of the consensus are dishonest. In fact, the goal of PBFT is that all the honest nodes, composing the consortium network, communicate with each other to help in reaching a consensus regarding the state of the system through the majority. The important advantage of PBFT is its significant performances in terms of energy consumption reduction. In this context, we use an optimized PBFT to improve authentication efficiency, where the selection of the leader peer is based on round robin than computing complex puzzles, which can alleviate the computation burden on the Blockchain network. The consensus algorithm is executed to store the user information such as the identity and subscription lists and authentication logs in the Blockchain network. Thus, it may prevent data tampering and achieve data traceability.

5.4. System Model

In this section, we introduce the overall system architecture of the proposed DiGABlock scheme, which offers secure and fast user authentication with numerous groups of IoT devices. Furthermore, we present the adaptability of the DiGABlock with a smart hotel use case. Then, we present the attacker model and enumerate the security system requirements.

5.4.1. System Architecture

The system architecture, shown in Figure 5.1, is composed of four different layers: (i) IoT devices services layer, (ii) End-users layer, (iii) Blockchain edge layer, and (iv) Blockchain network layer. In what follows, we explain the different components of our architecture:



- ***IoT devices services layer:*** outlines a large number of IoT devices that collect and publish data. These IoT devices form many groups that define different IoT services, where each service is mapped to a service manager. The IoT devices are assumed limited in their processing power, memory, and energy availability.
- ***End-users layer:*** comprises the subscribed users to the existing IoT services in the Blockchain network. In fact, the users choose a list of the desired IoT services and subscribe under the smart contract of the Blockchain network. Then, the subscribers will authenticate to the selected IoT devices through the corresponding and the nearby service manager to them.
- ***Blockchain edge layer:*** is composed of service managers (SM) nodes. Each service manager node is responsible for controlling a group of IoT devices. Besides, the SM ensures users' authentication with IoT devices and then sends authentication results to the Blockchain network. The SM proposes user authentication transactions (described in the next section) and contributes to committing a block into the blockchain network. The service managers are established in our proposed group authentication system to provide a group edge authentication service and synchronize authentication data to supervise users' activity as blockchain clients.
- ***Blockchain network layer:*** comprises the peers' nodes that store users' information in a distributed manner. In fact, they contain the distributed ledger recording the user authentication information. Each peer has a smart contract useful to verify the transactions and adding blocks into the blockchain network. Together with the service manager nodes, the peer nodes form a consortium Blockchain network through the Practical Byzantine Fault Tolerance (PBFT) for consensus establishment. Further, they are responsible for verifying, endorsing transactions, creating blocks, and committing the authentication results to the ledger. Our Blockchain network provides distributed services of storing users' authentication information over Hyperledger Fabric, a customizable consortium blockchain platform that supports smart contracts called "chaincode" [178].

5.4.2. Adaptation to the Smart Hotel Scenario

We can easily adapt DiGABlock solution to the smart hotel use case (see figure 5.2), presented in the previous chapters 3 and 4 (in the context of PARFAIT project [7]). In fact, a smart hotel which is composed of hundreds of rooms and offers various hospitality services, needs to define an efficient authentication system for different users. For that, we can define a service manager for the different services in the hotel, such as room service for each room, accommodation service, tourism service, and entertainment service. The users could be the hotel guests, the hoteliers, and the hotel staff. In this scenario, guests' smartphones should be authenticated before getting access to the room service. Also, they need authentication to access accommodation services like the restaurant, food distributors, etc. A regular change of user might be problematic as the number of requesting authentication is important. Thus, a service manager is a unit control that controls a group of IoT devices, defining one of the mentioned services and plays the role of a gateway between the user and the concerned IoT devices. These

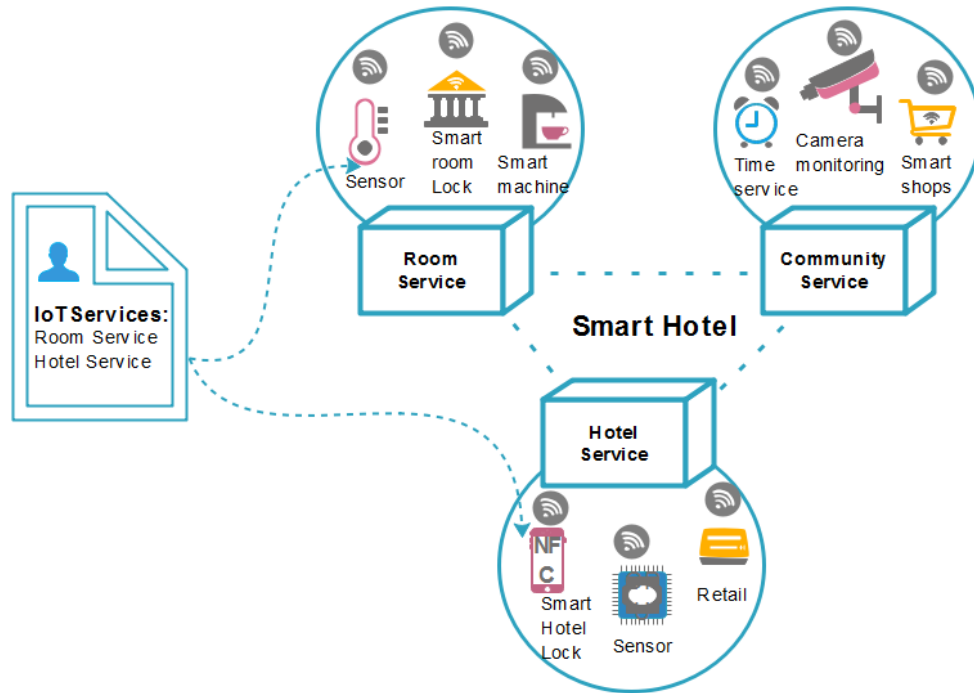


Figure.5. 2: IoT environment of a Smart Hotel

service managers present the edge part of the Blockchain network, installed to ensure an edge authentication to enhance the system response, and characterized with a high computing capacity. The Blockchain network is composed of servers that might be running on the physical hardware present inside the hotel, or they may be a cloud service provided for the hotels.

5.4.3. Threat Model

In this section, we consider adversaries targeting the authentication process, where several attacks are commonly employing the consortium Blockchain, such as:

- Denial of service attacks in which attackers aim to render the blockchain network or IoT network unavailable. Attackers may eavesdrop on data transferred from Blockchain edge nodes and try to fabricate a false signature without having access to the respective private signer key. At this level, the attackers proceed to transfer the valid authentication results signed with the false signature to the Blockchain. The alliance peers should discard this message, as it is unreliable even if the authentication results are valid.
- A forgery attack in which attackers may falsify users' identity to access the edge nodes and obtain confidential content or infect the authentication data. Further, they could also attempt to fabricate the edge nodes' identity and steal or modify the users' information, leading to the destruction of the authentication process. To produce a forgery attack, they need to eavesdrop delivered messages by the legitimate edge node and compute cryptographic keys and then tamper the delivered messages to the blockchain network or terminals with forgery signatures.
- A man-in-the-middle attack may occur during messages transmission between users and the edge nodes. Indeed, the attackers could block the delivered messages

between edge nodes and users, modify, and send them maliciously to destroy the system.

5.4.4. Security Goals

As discussed in the previous subsection, the attackers may attempt to destroy the communication of the system. Hence, it is essential to meet the security requirements of an authentication mechanism based on Blockchain technology. Besides, the proposed DiGABlock scheme should achieve the following security goals:

- Group authentication and secure key agreement, including confidentiality and integrity. Each user must be authenticated successfully with all requested IoT devices through the Blockchain network. Once successfully authenticated, a secure channel is established between users and all corresponding IoT devices. Hence, the adversary can neither decrypt nor tamper any transmitted message.
- Anonymity through hiding the users' identities in regular exchanged messages during the authentication process.
- Traceability by guaranteeing that smart contracts can trace all illegal users in case of any doubtful situation.
- Non-interactivity by allowing users to authenticate only once in the system and then get a secure access through the service manager without a full authentication process. Consequently, reducing the number of transmitted messages and enhancing the system response.
- Non-frameability that guarantees that the users' information is not abused by the single trusted entity (alliance peers and service manager) during the authentication process. The trusted entities should cooperate with the user to reveal the authentication information.

5.5. DiGABlock Description

In this section, we describe the proposed DiGABlock scheme that achieves a distributed group authentication and avoids congestion in IoT environment. Our system includes mainly six phases: namely, (i) *Blockchain setting up*, (ii) *initialization*, (iii) *user registration*, (iv) *group authentication*, (v) *consensus*, and (vi) *service delivery* phases. The setting up of the Blockchain network and the initialization phases are done only once during the system establishment, while the rest of phases are repeated through the authentication process. Figure 5.3 presents the last four phases:

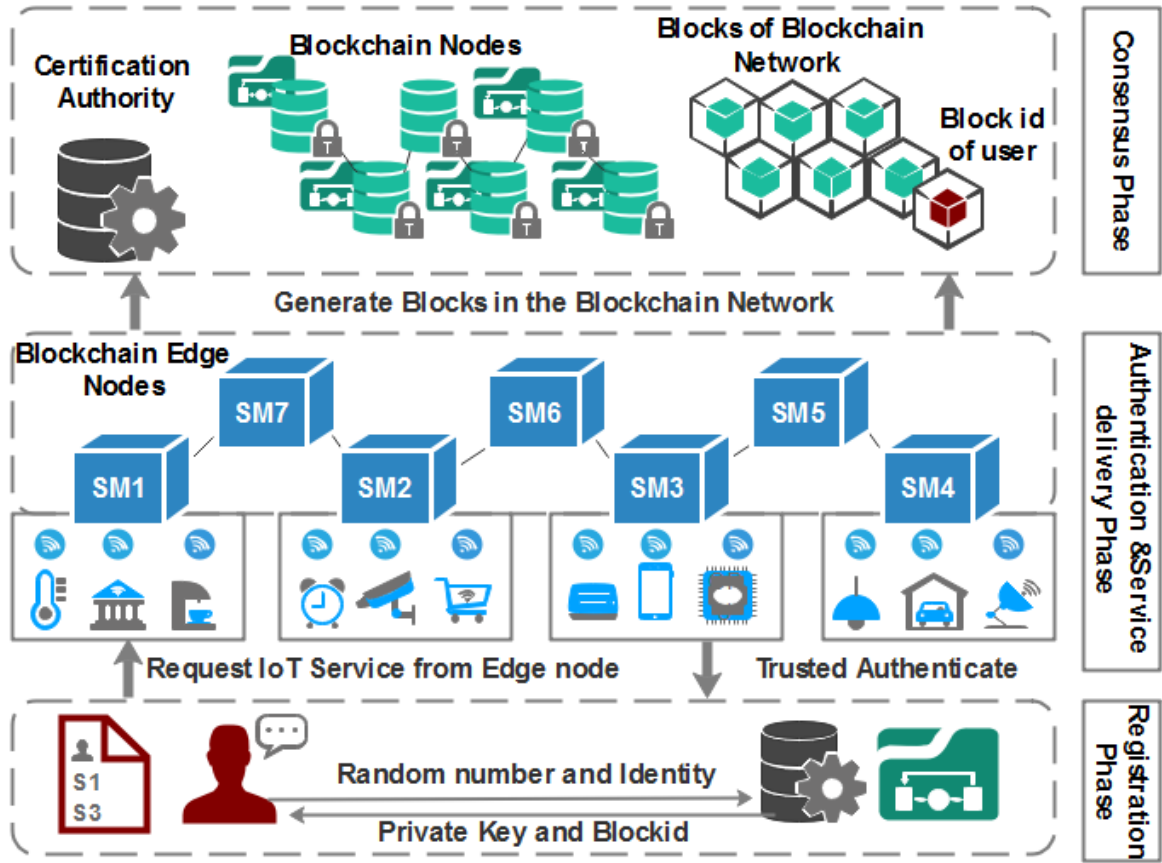


Figure.5. 3: Workflow model of the proposed scheme

The main idea of our solution is to authenticate a user with many IoT devices with less interaction with the system. Hence, the user should register himself to the system through the Certification Authority, *CA*, to get the necessary authentication credentials described in the registration phase. During the registration, the user chooses a list of IoT services to which he/she wants to subscribe. The corresponding service managers of the requested IoT services get the registrations' update from *CA*. This action allows any service manager from the subscription list to authenticate the user at the edge of the network. Therefore, the user should perform a first full authentication to one of the service managers to verify his/her identity. Mutual authentication is achieved between the user and the corresponding service manager, as well as between the user and a group of IoT devices under the control of this service manager during the first authentication. This full authentication aims to provide the user with some information that allows him/her to get access to the remaining IoT services of the subscription list without re-authenticating with the system. After verifying the user's legitimacy, the service manager generates transactions containing the user information. The service manager then sends a request to the Blockchain node to add a block for the new user and chain it in the Blockchain network. We note that we use a consortium blockchain network in our solution, which permits tracking the user in the system. Once the consensus is finished and the user's block is chained in the blockchain network, the user could access any IoT services from his/her subscription list securely through the service delivery phase.

5.5.1. Setting up the Blockchain Network

During this phase, we setup the blockchain network with the edge nodes and peers registered under the *CA* of the Hyperledger Fabric. Besides, in each edge node and blockchain node, smart contracts are deployed to maintain an updated ledger with the user authentication information. In addition, the edge node of the blockchain network named the service manager defines an IoT service and controls a group of IoT devices.

Let N be the number of all IoT devices in our system; we denote $D = \{D_1, D_2, \dots, D_N\}$ the set of IoT devices, $\chi = \{x_1, x_2, \dots, x_N\}$ the set of public information related to them, and $S = \{S_1, S_2, \dots, S_K\}$ the set of IoT services. We define each IoT device $D_j \in D$ at the smart contract with a unique identity and a group of IoT devices with an IoT service under the control of the Service Manager, SM . Let $SM = \{SM_1, SM_2, \dots, SM_K\}$ be the set of service managers used to control groups of IoT devices in our system, where $K < N$. In fact, each SM_j , where $1 \leq j \leq K$, contains a subscription list ζ_j of legitimate subscribed users to the associated IoT service S_j . Further, the subscription list is updated after each new successful user registration. We also adopt a revocation list ϑ of the revoked users in the network. Finally, we outline that the Blockchain nodes called alliance peers $AP = \{AP_1, AP_2, \dots, AP_M\}$ collaborate with the edge nodes to verify, endorse transactions, and commits blocks to the ledger containing the successful user authentication results. The notations used in this section are summarized in Table 5.3.

Table 5. 3: List of acronyms

Notations	Description
S_j	The service j
SM_j	The service manager of the S_j
ζ_j	A subscription list of the SM_j
D_i	The device i
x_1	Public information of the D_i
AP	Alliance peers
SC	Smart contract
SK_X	The secret key of X
PK_X	The public key of X
$e(.,.)$	Pairing function

5.5.2. Initialization Phase

In this phase, we initialize the system parameters that will be used in the eventual registration and authentication phases. Thus, the certification authority of the Hyperledger Fabric performs some operations to prepare the environment for the upcoming phases. In what follow, the *CA* is running this phase:

- *CA* generates two large secure prime numbers p and q , where $p > q + (N + 1)q^2$ (condition for securing group authentication from outsider attackers, discussed in section 5.5.3).

- *CA* picks an elliptic curve additive cyclic group G with order q and a generator P of G . It also selects a random $SigKey \in \mathbb{Z}_q^*$ as a private signature key and deduces the master public signature key $PSigKey = SigKey \times P$.
- *CA* declares the secure Hash functions useful in our scheme defined as follows:
 - $H_0: G \rightarrow \{0,1\}^*$,
 - $H_1: \{0,1\}^* \times G \rightarrow \mathbb{Z}_q^*$,
 - $H_2: G \times \{0,1\}^* \times G \rightarrow \mathbb{Z}_q^*$,
 - $H_3: G \times G \rightarrow \mathbb{Z}_q^*$,

Then, *CA* keeps *SigKey* secret and publishes the system parameters $\{P, F_p, F_q, PSigKey, H_0, H_1, H_2, H_3\}$.

5.5.3. User Registration Phase

Users should register under our system, as shown in Figure 5.4, to subscribe to the different required IoT services. For clarity, we assume, for example, that a user U_x wants to subscribe to the set $\{S_x, S_a, S_b\}$ of IoT services. Note that each IoT service comprises a group of IoT devices controlled by a service manager, *SM*, and we consider n is the total number of IoT devices, $\{D_1, D_2, \dots, D_n\}$ to which the user is subscribed. In fact, during the registration phase the user U_x registers himself with the certification authority *CA* of the Hyperledger Fabric by requesting the IoT services $\{S_x, S_a, S_b\}$. In particular, to secure communication between the user and edge nodes, a lightweight symmetric cryptographic protocol is designed.

Once receiving the user request, including the identity, the IoT services, and the timestamp $\langle Id_{U_x}, T_{U_x}, S_x, S_a, S_b \rangle$, the *CA* proceeds to register the user under the system for further secure communication with IoT service. Otherwise, since an IoT service is defined with a group of IoT devices in our scheme, a secret sharing protocol is designed to secure communication between the user and IoT devices. We consider, based on the Shamir scheme, n IoT devices, and the user compose the set of $n+1$ shareholders $\{D_1, D_2, \dots, D_n, U_x\}$, with respective public information $\{x_1, x_2, \dots, x_n\}$ and user Id_{U_x} , and the $(t+1)$ is the threshold of the shared secret. At this level, *CA* executes the following steps of registration by:

- Select a random polynomial $f(x)$ in F_p , where the degree is fixed to the minimum number of IoT devices per group t , such that:

$$f(x) = a_0 + a_1x + \dots + a_tx^t \text{ mod } (p),$$

$$\text{where } a_k \in F_p, \text{ for } k = 0, 1, \dots, t, a_t \neq 0, a_0 \in F_q$$

- Choose a shared secret that defines the user who wants to communicate with the n IoT devices: $Sec_U = f(0) = a_0$; $Sec_U < p$.
- Compute the shares for the requested IoT devices corresponding to the IoT services $\{S_x, S_a, S_b\}$:

$$f(x_i) \leftarrow a_0 + \sum_{k=1}^{k=t} a_k \cdot x_i^k \text{ mod}(p)$$

- Compute the user's share useful to ensure the legitimacy of the user among the group of IoT devices:

$$f(Id_{U_x}) \leftarrow a_0 + \sum_{k=1}^{k=t} a_k Id_{U_x}^k \text{ mod}(p)$$

- Generate a random number $r_{U_x} \in F_P$, to compute the user's shared secret key: $SK_{U_x} = H_1(Id_{U_x}, r_{U_x})$.
- Compute the user registration data convenient to ensure secure communication with the edge nodes:

$$l_{U_i}^* = H_3(f(Id_{U_x}) \parallel H_1(Id_{U_x}, r_{U_x})) \oplus H_1(SK_{CA} \parallel Id_{U_x})$$

At this level, the user identity, the required IoT devices information, and the hash value of the user secret $\langle Id_{U_x}, f(x_i), i \leq n \text{ and } H_0(Sec_U) \rangle$ are sent to the matching service manager SM_j for updating their subscription list ζ_j , where the user is identified with an identity $Block_{U_{id}}$.

- Send to the user over a secure channel:

$$CA \xrightarrow{\langle SK_{U_x}, f(Id_{U_x}), Block_{U_{id}}, H_1(SK_{CA} \parallel Id_{U_x}) \rangle} U_x$$

- Finally, once receiving registration information, the user stores it in its memory $\langle SK_{U_x}, f(Id_{U_x}), Block_{U_{id}}, H_1(SK_{CA} \parallel Id_{U_x}) \rangle$.

Once the user is registered successfully, he/she could ensure a group authentication with the required n IoT devices. Based on the Shamir's Secret sharing SS , Harn et. al. [171] proposed a group authentication scheme named (t, m, n) GAS t -secure m -user n -group. (t, m, n) GAS limits the number of users participating in the communication. Hence, GAS defines t the threshold of the shared secret, m the number of users participating in the authentication procedure, and n the number of members in the group. Besides, the selected secret is divided into n pieces, and then it is distributed to n users. Then, GAS guarantees that with only m users participating in the authentication to recover the Shamir's secret through Lagrange polynomial interpolation.

For that, we define a new concept of group authentication scheme based on GAS and Shamir shared secret as follows:

Definition 1: t -minSecure, d -IoT devices, m -maxSecure, n -Group Authentication (t, d, m, n) -GA

let t, d, m, n be four positive integers where $t \leq d \leq m \leq n$.

- t -minSecure defines the minimum number of IoT devices per group,
- d -IoT devices participating in the group authentication,

- $m\text{-maxSecure}$ is the maximum number of IoT devices per group,
- $n\text{-Group}$ is the number of all IoT devices to which the user is subscribed.

The $(t, d, m, n)\text{-GA}$ can resist up to $(t + 1)$ colluded group members, for a group of d IoT devices, and then, the $(t, d, m, n)\text{-GA}$ determine whether a user is authenticated to n IoT devices or not.

In $(t, d, m, n)\text{-GA}$, a user could ensure authentication with a group of IoT devices. For that, a secret Sec_U is selected, and tokens are computed for the group with $(n+1)$ members composed of n IoT devices and the user during the user's registration. These tokens are used to retrieve the shared secret Sec_U that define the legitimate user who wants to communicate

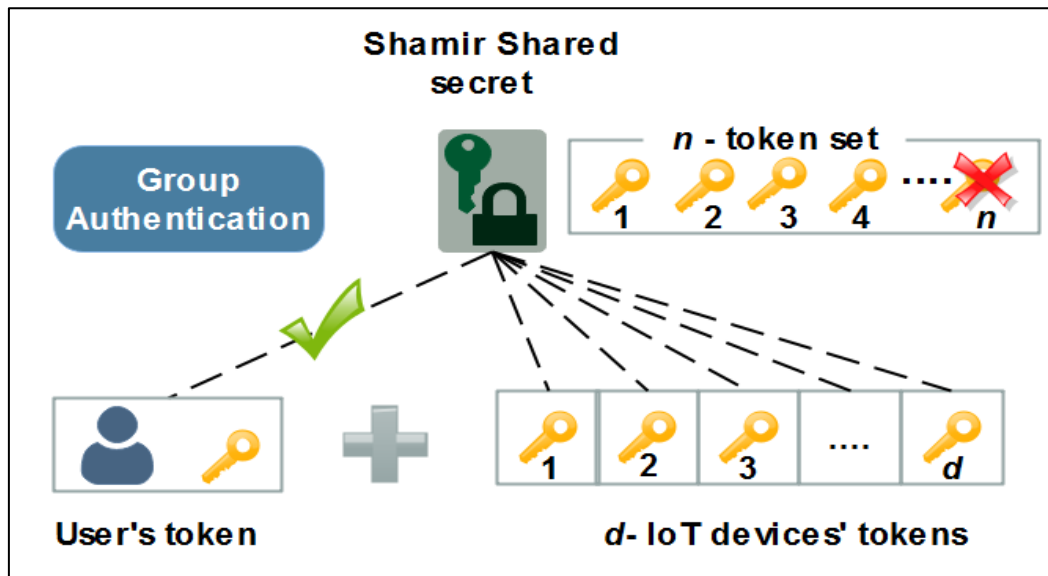


Figure.5. 4: Secret shared authenticator recovering

with the IoT group. In particular, the $(t, d, m, n)\text{-GA}$, as shown in Figure 5.4, allows the user's authentication to a group of n IoT devices, where only d IoT devices, under the control of SM, and the user participate in the group authentication. The service manager is responsible for retrieving the shared secret using IoT devices' tokens and the user's token distributed during the authentication phase.

The $(t, d, m, n)\text{-GA}$ algorithm can only detect nonmembers' existence but cannot identify them. The advantage of $(t, d, m, n)\text{-GA}$ is that the user is authenticated with all IoT devices at once, while the user is authenticated by one IoT device in conventional user authentication.

The $(t, d, m, n)\text{-GA}$ Algorithm
Input: public information of the user and IoT devices, the random polynomial $f(x)$
Output: generating tokens for user and IoT devices, Lagrange component and the shared secret.
Token generation:
Select:
<ul style="list-style-type: none"> • A random polynomial $f(x)$ in F_p, of degree t such that:

$f(x) = a_0 + a_1x + \dots + a_tx^t \bmod(p), \quad \text{where } a_k \in \mathbb{F}_p, \text{ for } k = 0, 1, \dots, t, a_t \neq 0, a_0 \in \mathbb{F}_q$ <ul style="list-style-type: none"> A secret $Sec_U = f(0) = a_0; Sec_U < p$. <p>Compute the corresponding tokens for n IoT devices:</p> <p>For each device $D_i; i = 1..n$ (of public information x_i) do</p> $f(x_i) \leftarrow a_0 + \sum_{k=1}^{k=t} a_k \cdot x_i^k \bmod(p)$ <p>End for</p> <p>Compute the user token:</p> $f(Id_{U_x}) \leftarrow a_0 + \sum_{k=1}^{k=t} a_k Id_{U_x}^k \bmod(p)$ <p>Distribute the tokens of IoT devices and user and the hash value of the secret $H_0(Sec_U)$</p>
<p><u>Group authentication:</u></p> <p>The SM computes all Lagrange component of the d requested IoT devices using the tokens:</p> $c_j = \left(f(x_j) \times \prod_{v=1, v \neq j}^d \frac{-x_v}{x_j - x_v} \frac{-Id_{U_x}}{x_j - Id_{U_x}} + r_j q \right) \bmod p, \text{ where } j = 1 \dots d$ <p>The user computes the corresponding Lagrange component using his/her token:</p> $c_U = \left(f(Id_{U_x}) \times \prod_{v=1}^d \frac{-x_v}{Id_{U_x} - x_v} + r_0 q \right) \bmod p$ <p>After receiving all the Lagrange components, the SM retrieve the shared secret:</p> $s = \left(\sum_{j=1}^{j=d} c_j + c_U \bmod p \right) \bmod q;$ <p>Then, SM verifying the validity of the retrieved shared:</p> $H_0(Sec_U) == H_0(s)$ <p>Hence, the user and all IoT devices are authenticated. Otherwise, the user is not legitimate.</p>

We verify the validity of (t, d, m, n) -GA through proving the following properties:

- **Correctness:** the shared secret is reconstructed successfully only if the user and IoT devices are acting honestly by realizing their Lagrange components. Indeed, if the user is non-legitimate, he/she has a non-valid token, and thus the released Lagrange component is illegal. The recovered secret will not match the correct secret at this stage, and the authentication is rejected.
- **Efficiency:** the communication overhead is minimal in (t, d, m, n) -GA. Indeed, the user needs to release the computed Lagrange component to the SM , while IoT devices under this SM 's control should maintain the shared keys that verify their legitimacy. Thus, the overhead cost is only deriving from the most consuming operation to compute the Lagrange component. The SM is also responsible for verifying the members of the group communication, which minimizes the computation cost related to retrieving the secret.

- **Security:** regarding the generation of the tokens are achieved by polynomials of a degree (t) , thus (t, d, m, n) -GA resists up to (t) colluded inside adversaries trying to recover the selected polynomial. For the outsider attackers trying to participate in the group authentication, they could not compute the tokens protected unconditionally with the Lagrange component. Therefore, any outsider adversary cannot also derive the user token from the Lagrange component sent to the SM during the authentication. In fact, the Lagrange component is a linear function of $k(t + 1)$ coefficients of polynomials, with each polynomial having a degree t . Thus, since $k(t + 1) > n$ (the total number of required IoT devices), an outside adversary cannot forge the valid Lagrange component when the user token is released asynchronously.

5.5.4. Distributed Group Authentication Phase

At this level, the user U_x can authenticate with all the IoT devices to which he/she is subscribed in the registration phase, as presented in the Figure 5.5. Indeed, the user requests to access one of the IoT services $\{S_x, S_a, S_b\}$, for the first time. We consider that the user U_x wants to communicate with the IoT service S_x , which is composed of d IoT devices $\{D_1, D_2, \dots, D_d\}$.

To ensure the authentication of the user U_x with the d requested IoT devices, a group authentication is guaranteed at the edge of the network through the corresponding service manager SM_x . For that, it is important to secure the communication between the user and the edge node firstly, which is achieved as follows:

- Selects a random number $r_0 \in \mathbb{Z}_p^*$,
- Generates time stamp T_{U_x} ,
- Computes the necessary authentication information using the data of the registration phase:
 - $R_0 = r_0 \times P$;
 - $R_U = R_0 \times f(Id_{U_x})$;
 - $l_{U_x} = H_0(f(Id_{U_x}) \parallel sk_{U_x}) \oplus H_1(SK_{CA} \parallel Id_{U_x})$;
 - $c_U = \left(f(Id_{U_x}) \times \prod_{v=1}^d \frac{-x_v}{Id_{U_x} - x_v} + r_0 q \right) \bmod p$; The user Lagrange component c_U to contribute to the computing of the Shamir secret.
 - $token_{U_x} = Ency(c_U, PK_{SM_x})$;
 - $Auth_{U_x} = H_2(R_0 \parallel Block_{U_{id}} \parallel T_{U_x} \parallel l_{U_x})$; User information to authenticate the user mutually.
- Sends the authentication request to the corresponding service manager SM_x :

$$U_x \xrightarrow{\langle Auth_{U_x}, r_0, token_{U_x}, T_{U_x}, Block_{U_{id}} \rangle} SM_x$$

After receiving the user authentication request, the SM_x starts the authentication process as follows:

- Selects the time stamp T_1 and verifies weather $|T_{U_x} - T_1| < \Delta T$; if it holds, it continues the authentication, else the request is declined,
- Fetches the user information identified with $Block_{U_{id}}$ in the registration user list in local database and gathers tokens $f(x_j)$ of all requested IoT devices under its control.
- Computes the authentication value of the user:

$$Auth_{U_x}^* = H_2(r_U \cdot PK_{U_x} \parallel Block_{U_{id}} \parallel T_{U_x} \parallel l_{U_x}^*)$$

and compares it with the received value from the user: $Auth_{U_x}^* == Auth_{U_x}$. If the value is correct, an authentication is achieved between the user and the SM_x . Then SM_x decrypts the user token; $decrypt(token_{U_x}, SK_{SM_x})$ to retrieve the user's Lagrange component c_U useful to ensure the group authentication with the group of IoT devices. Otherwise, SM_x tears down the connection.

- Computes all Lagrange component of the d requested IoT devices under its control using the registration information:

$$c_j = \left(f(x_j) \times \prod_{v=1, v \neq j}^d \frac{-x_v}{x_j - x_v} \frac{-Id_{U_x}}{x_j - Id_{U_x}} + r_j q \right) \text{mod } , \text{ where } j = 1 \dots d$$

- Retrieves the shared secret corresponding to the user:

$$s = \left(\sum_{j=1}^{j=d} c_j + c_U \text{mod } p \right) \text{mod } q;$$

- Computes $H_0(s)$ and compares it with the received value during the registration phase: $H_0(s) == H_0(Sec_U)?$. If it is true, we confirm that the user U_x is legitimate and can get a secure access to the all IoT devices under the SM_x control. Otherwise, the user authentication request is declined, and the service manager proceeds to update the revocation list.

At this level, to secure the further user communication with the rest of SM to which he/she is subscribed, the SM_x designs an asymmetric cryptographic algorithm based on ECC. In fact, the SM_x produces a master signature for the user useful in the service delivery phase when user requests a new IoT service and does not need to re-authenticate:

- Selects a random number $v_x \in \mathbb{Z}_q^*$, and computes the user master signature useful for further authentication:
 - $V_x = v_x \cdot P$, $h_x = H_3(Id_x \parallel V_x \parallel PK_{SM_x})$, and $\delta_{x1} = (v_x + (h_x \cdot sig) \text{mod}) q^{-1} \cdot P$
- Sends the signature to the user: $SM_x \xrightarrow{<V_x, \delta_{x1}>} U_x$ and broadcast the public information to the edge nodes.

The user verifies the legitimacy of the service manager SM_x by verifying if equation 5.4 holds:

$$e(\delta_{x1}, V_x + h_x \cdot PK_{SM_x}) == e(P, P) \quad (5.4)$$

Hence, if equation 5.4 holds, the user U_x confirms the reliability of received messages and stores the signature for further communication with IoT services.

Algorithm 1: User group authentication

Input: $f(x_i)$ of d requested IoT devices and $f(Id_{U_x})$ of the user

Output: master signature for the authenticated user, update subscription and revocation list

```

1:  User: - selects a random number  $r_0 \in \mathbb{Z}_p^*$ 
2:        - Generates time stamp  $T_{U_x}$ 
3:        - Computes:  $R_0 = r_0 \times P$ ;
4:         $R_U = R_0 \times f(Id_{U_x})$ ;
5:         $l_{U_x} = H_0(f(Id_{U_x}) \parallel sk_{U_x}) \oplus H_1(SK_{CA} \parallel Id_{U_x})$ ;
6:   $c_U = \left( f(Id_{U_x}) \times \prod_{v=1}^d \frac{-x_v}{Id_{U_x} - x_v} + r_0 q \right) \bmod p$ 
7:     $token_{U_x} = Ency(c_U, PK_{SM_x})$ ;
8:     $Auth_{U_x} = H_2(R_0 \parallel Block_{U_{id}} \parallel T_{U_x} \parallel l_{U_x})$ ;
9:  User sends to the corresponding service manager:  $U_x \xrightarrow{\langle Auth_{U_x}, r_0, token_{U_x}, T_{U_x}, Block_{U_{id}} \rangle} SM_x$ 
     $SM_x$  executes the following steps:
10:  Select the time stamp  $T_1$ 
11:  If  $|T_{U_x} - T_1| < \Delta T$  Then
    Fetch in the local database the user information identified with  $Block_{U_{id}}$ 
    Gather shared secrets  $f(x_j)$  of all devices under  $SM_x$ 
  Else
    Request declined
  End if;
12:  Compute  $Auth_{U_x}^* = H_2(r_U \cdot PK_{U_x} \parallel Block_{U_{id}} \parallel T_{U_x} \parallel l_{U_x}^*)$ 
13:  If  $Auth_{U_x}^* == Auth_{U_x}$  Then
     $decrypt(token_{U_x}, SK_{SM_x})$  and retrieve  $c_U$ 
  Else
    Tear down the connection.
  End if;
    Execute the Lagrange interpolation formula, the service manager run the following steps:
14:  Compute the Lagrange component of requested IoT devices under the control of  $SK_{SM_x}$ 
15:  For  $j = 1$  to  $d$  do
     $c_j = \left( f(x_j) \times \prod_{v=1, v \neq j}^d \frac{-x_v}{x_j - x_v} \frac{-Id_{U_x}}{x_j - Id_{U_x}} + r_j q \right) \bmod p$ 
  End For
16:  Retrieve the shared secret;
     $s = (\sum_{j=1}^d c_j + c_U \bmod p) \bmod q$ ;
17:  Compute  $H_0(s)$ ;
18:  If  $H_0(s) == H_0(SEC_U)$  Then
     $U_x$  is authenticated
     $SM_x$  selects a random number  $v_x \in \mathbb{Z}_q^*$ , computes the master signature:
     $V_x = v_x \cdot P$ 
     $h_x = H_3(Id_x \parallel V_x \parallel PK_{SM_x})$ 
     $\delta_{x1} = (v_x + (h_x \cdot sig) \bmod q) q^{-1} \cdot P$ 
     $SM_x \xrightarrow{\langle V_x, \delta_{x1} \rangle} U_x$ 
  Else
     $U_x$  is NOT authenticated, and the authentication request is denied
    Update the revocation list
    Break;
  End if.
19:  User verify the legitimacy of the  $SM_x$ 
    If  $e(\delta_{x1}, V_x + h_x \cdot PK_{SM_x}) == e(P, P)$  Then
      User  $U_x$  confirm the reliability of received messages;
      User  $U_x$  stores the signature for further communication with IoT services;
    End if

```

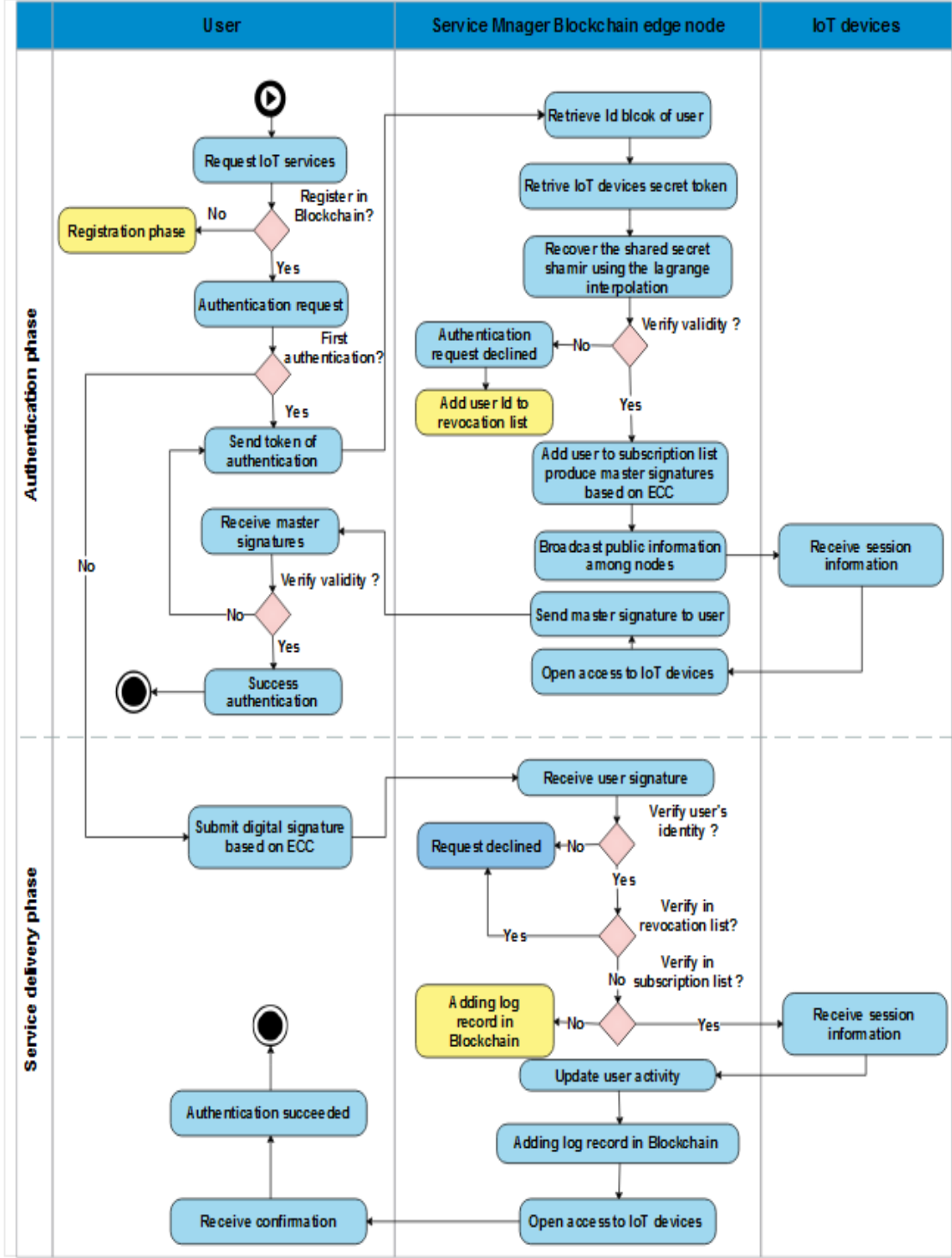


Figure.5. 5: Authentication & Service delivery phases

Once the authentication is successfully finished, the user should be added to the blockchain network. For that, the SM_x , based on the user information and the authentication results, generate transactions proposal of the new user U_x . These transactions involve the user authentication information, including the identity, the signature, the requested IoT services, the type of authentication, and the transmitted data are presented as follow:

- $TX1_{U_x} = Hash1(Id_{U_x}, \delta_{x1}, S_x, S_a, S_b, Auth, S_x)$

- $TX2_{U_x} = Hash2(Id_{U_x}, \delta_{x2}, S_x, S_a, S_b, Deliv, S_x)$
- $TX3_{U_x} = Hash3(Id_{U_x}, \delta_{x3}, S_x, S_a, S_b, Deliv, S_a)$
- $TX4_{U_x} = Hash4(Id_{U_x}, \delta_{x3}, S_x, S_a, S_b, Deliv, S_b)$

Those transactions are computed using the hash-256 function and are stored using the Merkle tree [167], a data structure tree where each non-leaf node is a hash of its respective child nodes.

Since updating the blockchain network's ledger requires the consent of the peers in the network, the SM_x initiates a consensus mechanism by sending a request for public alliance peers APs to ensure that the 'block' recording information will be 'chained' in the Blockchain network through calling the consensus algorithm 2, which is the subject of the next part.

5.5.5. Consensus Phase

In this phase, we consider an optimized PBFT consensus algorithm to form the public ledger and store authentication results and logs to promote certification efficiency. In our case, the PBFT algorithm is executed in collaboration between the edge nodes SMs and M alliance peers AP_i , $1 \leq i \leq M$. In fact, SM sends the request to the primary peer, which is selected in a round-robin manner, while alliance peers AP_i write the authentication results into the public ledger through the optimized PBFT. For each round of consensus making, and given h the height of the current block (the *block height* is an expression of the total number of individual blocks that are a part of the blockchain), an AP_i is nominated as the speaker of the consensus by using equation 5.5 where M is the number of alliance peers:

$$speaker = (h \bmod M) + 1 \quad (5.5)$$

While the other peers are congressmen. Once the block is generated, the speaker signs it and broadcasts it to all congressmen:

$$< Pre_prepare, height, AP_x, block, Sig_{AP_x}(block) >$$

Moreover, to save the time of selecting speakers, the nominated one can host the consensus process M times, as it does not influence the consensus results. The detailed procedure is explained in algorithm 2.

At this level, alliance peers receive the speaker request, verify it, and update their local states. Then, each alliance peer participating in the consensus computes the *Prepare* messages and sends them to other alliance peers and the speaker:

$$< Prepare, height, AP_i, block, Sig_{AP_i}(block) >$$

This action is finished when all peers and the speaker receive a number greater than $2f + 1$ of *Prepare* messages, where $f = (M - 1) / 3$ is the maximum number of malicious nodes in the blockchain network. Now, all peers and the speaker could update their local states and broadcast *Commit* messages among them:

$$< Commit, height, AP_i, block, Sig_{AP_i}(block) >$$

Once all peers and the speaker receive at least a number greater than $f + 1$ of *Commit* messages, they respond to the corresponding service manager SM_x with the result. Besides, all peers and the speaker update their local state, and the speaker confirms that the consensus is finished.

After the consensus phase is achieved successfully, a block recording the user information is chained in the Blockchain network. As shown in Figure 5.6, the data block structure of an authenticated user contains all associated transactions and the header block that includes the previous hash, the nonce, the timestamp, and the hash Merkle root. The transactions included within this block are hashed as part of the Merkle tree leading to the Merkle root that is stored in the block header. Besides, these transactions involve the user authentication information, including the identity, the signature, the requested IoT services, the type of authentication, and the transmitted data.

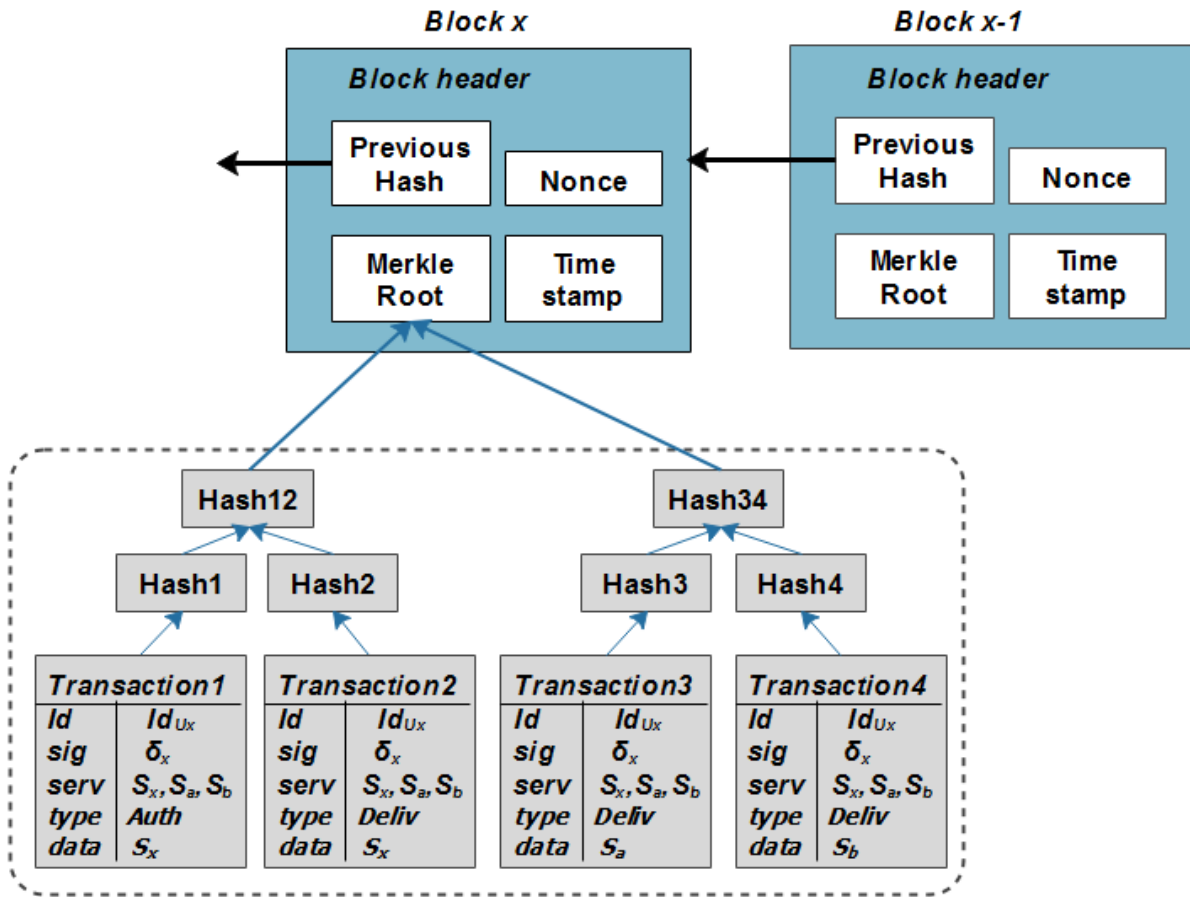


Figure.5. 6: User authentication data block

Algorithm 2: Consensus phase

Input: user authentication results log to be added,**Output:** a consensus is achieved, and user information is added to the Blockchain

```
1: / * Request
   SMx broadcasts authentication results to alliance peers APi
Repeat
2: / * Pre-prepare
   Given  $h$  the height of the current block;
   Define  $t$  the interval time of generating a block;
   Select the speaker of the consensus,  $speaker = (h \bmod M) + 1$ ;
   If interval  $t$  has expired, then
     A block is generated;
     Speaker broadcasts to all congressmen:
     < Pre_prepare, height, APx, block, SigAPx(block) >;
   End if
   If request is received & is valid then
     Alliance peers record the request;
     Alliance peers update local states;
   End if
   round  $\leftarrow$  0;
   While round  $\leq M$  do
     / * Prepare
3:   If the local state is updated, then
      $\forall i$ , APi computes the Prepare messages;
     Repeat
        $\forall i$ , APi sends Prepare messages to other alliance peers and to the
       speaker
       < Prepare, height, APi, block, SigAPi(block) >;
       Until all peers and speaker receive  $\geq 2f + 1$  Prepare messages
     End if
     / * Commit
     If all peers and speaker receive  $\geq 2f + 1$  then
4:     All peers and speaker update their local states;
     Repeat
       All APi & speaker broadcast Commit messages among them
       < Commit, height, APi, block, SigAPi(block) >;
       Until peers receive  $\geq f + 1$  Commit messages
     Else
       block is discarded;
       round  $\leftarrow$  round + 1;
     Break;
     End if
     / * Reply
     If peers & speaker receive  $\geq f + 1$  Commit messages
5:     All APi & speaker response the SMx with its result;
     All APi & speaker update local state;
     The speaker confirm that a consensus is finished;
     Break;
     Else
       block is discarded;
       round  $\leftarrow$  round + 1;
     Break;
     End if
   End while
Until consensus is finished
```

5.5.6. Service Delivery Phase

At this point and as shown in Figure 5.5, the user who is registered under our system for a list of IoT services $\{S_x, S_a, S_b\}$ and who has proceeded with the first authentication, could get permission access to a new IoT service from the subscription's list without re-authentication. We consider that the user requests another IoT service S_a from the corresponding service manager, SM_a , thus the user U_x computes a digital signature as follows:

- Given that $SK_{U_x} = h(Id_{U_x}, r_{U_x})$ the user private key; the user computes his corresponding public key $PK_{U_x} = SK_{U_x} \cdot P$.
- Computes the request: $X_u = H_2(PK_{U_x} \parallel Block_{U_{id}} \parallel V_x \parallel PSigKey \parallel S_a)$.
- Signs the request: $\delta_{x2} = (X_u \cdot (v_x + h_x \cdot SigKey \mod q) + SK_{U_x})^{-1} \cdot P$
- Submits the access requests by delivering a digital signature to the corresponding service manager SM_a of the requested IoT services S_a :

$$U_x \xrightarrow{AcRe_{U_x}=(X_u, \delta_{x2}, T'_{U_x})} SM_a$$

At this level, the service manager SM_a invokes the smart contract functions to query the user's information in the blockchain ledger, in particular it queries the transaction information related to the IoT service S_a . Once retrieved, the service manager SM_a :

- Proves that the user public key PK_{U_x} is not in the revocation list.
- Verifies the transactions' data in the user's block data(TX_{U_x}) == S_a
- Validates the received signature by verifying if equation 5.6 holds.

$$e(\delta_{x2}, X_u \cdot (V_x + h_x \cdot PSigKey) + PK_{U_x}) == e(P, P) \quad (5.6)$$

If the result is successful, the service manager SM_a confirms that the user U_x is authentic and responds to the user with the requested IoT service S_a . Otherwise, the SM_a declines the request. Algorithm 3 describes the detailed process of service delivery phase.

Algorithm 3: Service delivery phase

Input: user information sk_{U_x}, δ_{x1} , requested service S_a

Output: service access permission or denied permission

- 1: The user prepares request of the IoT service S_a from SM_a by computing :
 The public key $PK_{U_x} = SK_{U_x} \cdot P$;
 $X_u = H_2(PK_{U_x} \parallel Block_{U_{id}} \parallel V_x \parallel PSigKey \parallel m)$;
 $\delta_{x2} = (X_u \cdot (v_x + h_x \cdot SigKey \mod q) + SK_{U_x})^{-1} \cdot P$;
 - 2: The user U_x submits the access request through delivering a digital signature to S_a : $U_x \xrightarrow{AcRe_{U_x}=(X_u, \delta_{x2}, T'_{U_x})} SM_a$
 - 3: SM_a invokes the smart contract functions to get the user information registered in the blockchain network;
 - 4:
-

```

If  $U_x$  is in revocation list then
     $SM_a$  refuses to provide the service;
     $SM_a$  informs the Blockchain network;
5:   Break;
Else
6:   Read transaction data of the block related to the user  $U_x$ 
    If  $\text{data}(TX_{U_x}) == S_a$  &  $\text{type}(TX_{U_x}) == \text{Deliv}$  then
7:    $SM_a$  verifies the signature of the user;
    If  $e(\delta_{x2}, X_u \cdot (V_x + h_x \cdot \text{PSigKey}) + PK_{U_x}) == e(P, P)$  then
        User  $U_x$  is authentic;
         $SM_a$  responds the user with the requested IoT service  $S_a$ ;
8:   Break;
    Else
         $SM_a$  refuses to provide the service;
         $SM_a$  adds the user to the revocation list;
         $SM_a$  notifies the Blockchain network;
9:   End if;
    Else
         $SM_a$  refuses to provide the service;
         $SM_a$  adds the user to the revocation list;
         $SM_a$  notifies the Blockchain network;
    End if
End if

```

5.6. Security Evaluation

In this section, we evaluate the security features of DiGABlock through the correctness proof of the formal security analysis and the informal security features:

5.6.1. Correctness Proof

The correctness proof ensures that the decryption of an encrypted message returns the original plaintext. To prove the correctness of the proposed DiGABlock scheme, we need to verify that the following equations concerning the group authentication and the distribution of authentication user data are true:

- The Shamir secret: $s = Sec_U$
- The pairing function to verify the service manager legitimacy during the authentication phase:

$$e(\delta_{x1}, V_x + h_x \cdot PK_{SM_x}) = e(P, P)$$

- The pairing function to verify the user legitimacy during the service delivery phase:

$$e(\delta_{x2}, X_u \cdot (V_x + h_x \cdot \text{PSigKey}) + PK_{U_x}) = e(P, P)$$

The details of the proof correctness are as follow:

$$\begin{aligned}
\text{(i)} \quad s &= \left(\sum_{j=1}^{j=d} cj + c_U \bmod p \right) \bmod q \\
&= \left(\sum_{j=1}^{j=d} \left(f(x_j) \times \prod_{v=1, v \neq j}^d \frac{-x_v}{x_j - x_v} \frac{-Id_{U_x}}{x_j - Id_{U_x}} + r_j q \right) \bmod p + \left(f(Id_{U_x}) \times \right. \right. \\
&\quad \left. \left. \prod_{v=1}^d \frac{-x_v}{Id_{U_x} - x_v} + r_0 q \right) \bmod p \right) \bmod q \\
&= \sum_{j=1}^{j=d} \left(f(x_j) \times \prod_{v=1, v \neq j}^d \frac{-x_v}{x_j - x_v} \frac{-Id_{U_x}}{x_j - Id_{U_x}} + r_j q \right) \bmod p \bmod q \\
&\quad + \left(f(Id_{U_x}) \times \prod_{v=1}^d \frac{-x_v}{Id_{U_x} - x_v} + r_0 q \right) \bmod p \bmod q \\
&= \left(f(0) + \sum_{j=1}^{j=d} (r_j q) \right) \bmod q + (r_0 q) \bmod q \\
&= \left(f(0) + \sum_{j=1}^{j=d} (r_j q) + (r_0 q) \right) \bmod q \\
&= f(0) = Sec_U \\
\text{(ii)} \quad e(\delta_{x1}, V_x + h_x \cdot PK_{SM_x}) &= e\left((v_x + (h_x \cdot SK_{SM_x}) \bmod q)^{-1} \cdot P, V_x + h_x \cdot PK_{SM_x}\right) \\
&= e\left((v_x + (h_x \cdot SK_{SM_x}) \bmod q)^{-1} \cdot P, v_x \cdot P + h_x \cdot SK_{SM_x} \cdot P\right) \\
&= e\left((v_x + h_x \cdot SK_{SM_x})^{-1} \cdot P, v_x \cdot P + h_x \cdot SK_{SM_x} \cdot P\right) \\
&= e(P, P)^{(v_x + h_x \cdot SK_{SM_x})^{-1} \cdot (v_x + h_x \cdot SK_{SM_x})} \\
&= e(P, P). \\
\text{(iii)} \quad e(\delta_{x2}, X_u \cdot (V_x + h_x \cdot PSigKey) + PK_{U_x}) &= e\left((X_u \cdot (v_x + h_x \cdot SigKey \bmod q) + \right. \\
&\quad \left. SK_{U_x})^{-1} \cdot P, X_u \cdot (V_x + h_x \cdot PSigKey) + PK_{U_x}\right) \\
&= e\left((X_u \cdot (v_x + h_x \cdot SigKey) + SK_{U_x})^{-1} \cdot P, X_u \cdot (v_x \cdot P + h_x \cdot SigKey \cdot P) + PK_{U_x} \cdot P\right) \\
&= e\left((X_u \cdot (v_x + h_x \cdot SigKey) + SK_{U_x})^{-1} \cdot P, (X_u \cdot (v_x + h_x \cdot SigKey) + SK_{U_x}) \cdot P\right) \\
&= e(P, P)^{(X_u \cdot (v_x + h_x \cdot SigKey) + SK_{U_x})^{-1} \cdot (X_u \cdot (v_x + h_x \cdot SigKey) + SK_{U_x})} = e(P, P).
\end{aligned}$$

To sum up, our proposed scheme satisfies the correctness. Since the shares are generated by a polynomial having degree($t + 1$), this scheme can resist up to colluded (t) inside adversaries trying to recover the polynomial secret $f(0)$ selected by the CA initially. Otherwise, any outside adversary participating in the group authentication, needs to solve the discrete logarithm to derive each Lagrange component c_U , from each shared value, which is computationally infeasible.

5.6.2. Security Analysis

In this section, we conduct an informal security analysis to prove that our group authentication solution ensures the expected security requirements, and we outline its capacity to resist against the following well-known attacks: Forgery, DoS, MiM, Replay, User impersonation, and perfect forward/backward Secrecy.

5.6.2.1. Forgery Attack

Attackers may forge the identity of the Blockchain edge nodes (the service manager SM) to destroy the authentication mechanism and steal or modify the user's information. For that, an attacker selects a random number Err_x , computes $(Err_x.P)$ to eavesdrop or compute parameters delivered by the legal service manager SM using equations 5.7 and 5.8:

$$h_{Err} = H_3(Id_x \parallel Err_x.P \parallel PK_{SM_x}) \quad (5.7)$$

$$\delta_{Err} = (Err_x + (h_x.sig) \text{ mod } q^{-1}.P) \quad (5.8)$$

Otherwise, once the user receives these messages, he proceeds to compute the ECC pairing and verifies the request of A . At this level, the request of the attacker A will be rejected since A gives a false signature, and the user could detect that attack through computing the pairing ECC:

$$e(\delta_{Err}, Err_x.P + h_x.PK_{SM_x}) \neq e(P, P).$$

Hence, it is computationally infeasible to calculate correct private keys and public parameters generated by ECC, and we can confirm that our scheme prevents forgery attacks.

5.6.2.2. Denial of Service Attack:

During the authentication of the user with the service manager for the first time, a malicious user may launch a DoS attack on Blockchain edge node SM , and tries to forge the exchanged message of request authentication $\langle Auth_{U_x}, r_0, token_{U_x}, T_{U_x}, Block_{U_{id}} \rangle$. The SM can detect the forged message through checking $Block_{U_{id}}$ and comparing $Auth_{U_x}$, hence eliminate all illegal access requests. Otherwise, during the service delivery of the user with the remaining SMs to which it is subscribed, the authentication process is based on the signature verification and the result is broadcasted among SMs . If a malicious user interrupts the authentication messages distributed from SM and sings them with a fake key, alliance peers can detect the contaminated messages through the PBFT algorithm. More specifically, the consensus algorithm can be re-executed after each failing for N Time. Hence, the DoS attack may take

place only when messages delivered from all SM are contaminated, which is a difficult task regarding to the widespread of SM . Therefore, the proposed scheme resists the DoS attack.

5.6.2.3. Man-in-The-Middle Attack

Attackers may try to sniff exchanged authentication messages between the user and the service manager SM . For that, they need to guess the user's private key or the SM 's private key SK_{U_x} . Nevertheless, they cannot forge the SM 's or users' identity, as mentioned above in the DoS attack. Moreover, attackers may be able to catch the transferred data, they cannot use them against the network even they re-send it without modifying it, and then the attack does not influence the authentication system. However, all authentication messages are embedded with a timestamp, which prevents to reuse them. Thus, we conclude that this attack will fail since the probability of guessing the user's private key or the SM 's private key in a limited time ΔT is negligible.

5.6.2.4. Replay Attack

In our scheme, throughout the authentication mechanism and service delivery between the user and the service manager SM , we mark the released time of each message with a timestamp. More specifically, every received message is assigned with a time threshold ΔT . Hence, upon receiving the exchanged messages, the data consumer (user or service manager) will check the freshness of the timestamp before executing the other steps of the authentication process. In this way, the user and the service manager could detect easily if the message is reused, and then prevent the replay attack.

5.6.2.5. User Impersonation Attack

An attacker A may intend to impersonate a legitimate user through eavesdropping data flow between a user and the corresponding SM . However, A cannot get the Lagrange component of the user c_U , since c_U is embedded in the $token_{U_x}$. Moreover, A cannot decrypt the $token_{U_x}$, since the unique session key of SM cannot be adequately generated without the conforming private key of the user U_x . Furthermore, if A reuses the same $Block_{U_{id}}$ to request the authentication, the service manager SM can detect the adversary since the value of $Auth_{U_x}$ will not be matched. Moreover, only the legally authorized ones can access public ledgers such as SMs , APs . In the service delivery phase, based on ECC, it is computationally infeasible to calculate the correct key SK_{U_x} , thus even the attacker knows $Block_{id}$, A cannot sign the access request correctly $AcRe_{U_x} = (X_u, \delta_{x2}, T'_{U_x})$. Besides, if A intercepts $AcRe_{U_x} = (X_u, \delta_{x2}, T'_{U_x})$, and sends it in the next authentication procedure round, the timestamp, T'_{U_x} will prevent it from being reused.

5.6.2.6. Perfect Forward and Backward Secrecy

In our scheme, we need to provide the perfect backward secrecy, which guarantees the control of a new user who wants to get access to IoT services. Indeed, the user needs to perform a full authentication procedure with at least one SM . In fact, a random polynomial is designed for the new user, who joins the system with the desired IoT devices. Otherwise, each IoT device

D_j has a secret $f_u(x_j)$ corresponding only to the new user U_x , so that even if the new user U_x can sniff the old packets of the group; he cannot decrypt them. Moreover, the forward secrecy is guaranteed since the SM will revoke the user's binding relationship, who leave the system, with the groups of IoT devices. Furthermore, all leaved users are added to a revocation list tampered with and distributed in the Blockchain network so that the old user cannot get access to the IoT devices after his leaves.

5.7. Performance Analysis and Evaluation

This section analyzes the performance of the proposed scheme DiGABlock compared to direct authentication schemes, group authentication schemes PGA [160] & GBA [164], and distributed authentication schemes BLA [169] & BMEC [168]. We analyze the computation overhead on users at first and the response time of the service manager. Then, we study the energy consumption caused by the authentication and service delivery phases on users. Furthermore, we analyze the communication consumption and the improvement rate related to the communication cost comparing to the existing schemes.

5.7.1. Experimental Settings

To evaluate the proposed DiGABlock scheme, we measure the primitive cryptography operations by using the C/C++ Miracle Library, a cryptographic library designed for use in constrained environments in terms of computational power [158]. Furthermore, we use the HyperLedger Fabric platform [178] to measure the time of reaching PBFT consensus algorithm in the Blockchain network.

The simulations are implemented and tested on a computer with the following features: an Intel i5-4200 CPU@ 2.5 GH with a RAM of 8 GB. On the one hand, we use a virtual machine on Ubuntu 16.04 OS over VM VirtualBox, and we provide the time cost for different cryptographic operations. As a result, we define $T_{\text{Hash}} = 0,024\text{ms}$ be the time for one hashing operation using the SHA-256 function on a 64-bytes block. Then, $T_{\text{Enc}} = T_{\text{Dec}} = 0,047\text{ms}$ be respectively the time for one encryption/decryption operation using symmetric cryptography AES-256 encryption on a 64-bytes. Furthermore, the running time of one ECC point multiplication is $T_{\text{M-ECC}} = 0,365\text{ms}$, ECC basing point adding $T_{\text{A-ECC}} = 0,265\text{ms}$, the pairing operation $T_{\text{P-ECC}} = 1,05\text{ms}$, the map-to-point hash function is $T_{\text{H-ECC}} = 0,442\text{ms}$, and the Lagrange Component is $T_{\text{LC}} = 0,035\text{ms}$.

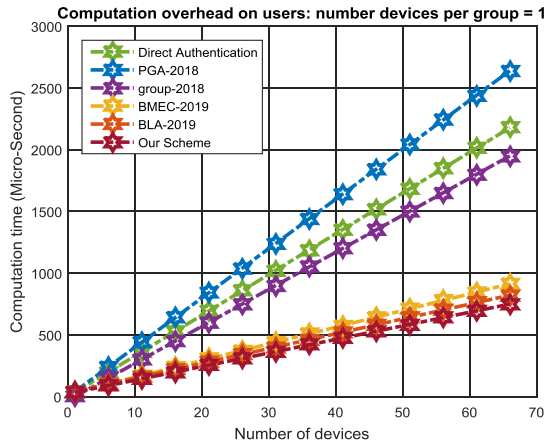
On the other hand, we use Hyperledger Fabric version 1.4 of docker 18.06 container in Ubuntu 16.04 over another virtual machine. HyperLedger Fabric uses Docker container technology to run chaincode that contains the system application logic. Several nodes are virtually hosted in a single machine, acting as alliance peers that reach the PBFT algorithm's consensus. We assume four alliance peers in our Blockchain network, and each node is 2.5 GHz. As a result, we provide the time cost for node verification $T_{\text{nv}} = 1\text{ms}$ and time for the PBFT consensus commitment $T_{\text{cons}} = 11\text{ms}$ when the number of peers is equal to four. Finally, we evaluate the proposed DiGABlock scheme with MATLAB.

5.7.2. Computation Costs

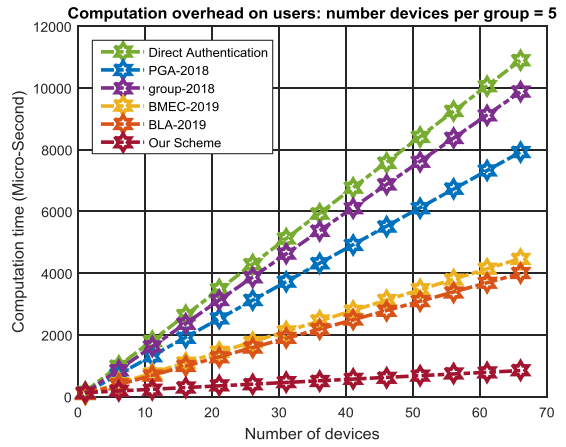
The authentication procedure provides security but also causes an increase in computation costs. Hence, it is essential to evaluate the computation overhead, which is the object of this subsection. We discuss the evaluation of our DiGABlock solution's computation cost, especially at the user level and the service manager level.

5.7.2.1. User Overhead

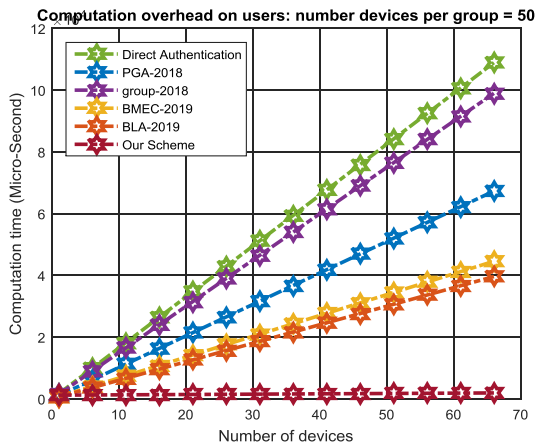
We calculate the user overhead by varying the list of subscriptions and fix the number of IoT devices per group to evaluate the impact on the computation cost. In particular, we compare the computation costs on users during the authentication process, for group-based authentication schemes PGA [165], GBA [170], distributed authentication schemes BLA [177], BMEC [178], and direct authentication scheme [58]. We consider three different cases varying the number of IoT devices per group. At first, we ponder one device per group, and we can see in Figure 5.7 (a) that the computation cost of our scheme and distributed schemes are lower than PGA [165], direct authentication [58], and GBA [170]. The reason is that we need only to authenticate once in a distributed mechanism, and hence the computation cost is not affected



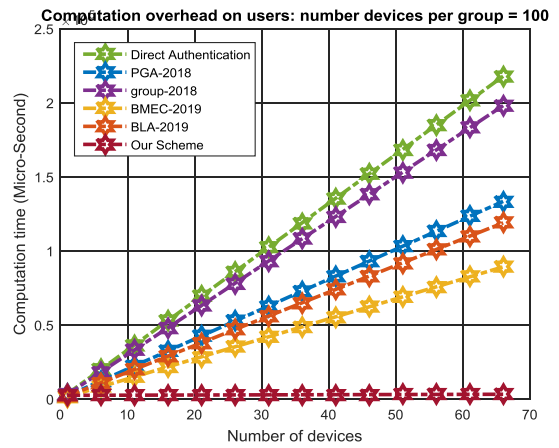
(a). Number devices per group = 1



(b). Number devices per group = 5



(c). Number devices per group = 50



(d). Number devices per group = 100

Figure.5. 7: Computation overhead on users

when the total number of IoT devices increases like in PGA [165], direct authentication [58], and GBA [170]. Furthermore, Figure 5.7 (b)-(d) show that our proposed scheme has the lowest costs than the other schemes when the number of IoT devices per group is higher than one. We can explain these results through exploiting the group-based authentication in a distributed manner, which decreases the cost of computation radically.

5.7.2.2. System Response Time

To evaluate the system response and latency, we compare the computational overhead of the service manager and the Blockchain network during the consensus phase. According to computation time, we plot the system response in three cases, where the number of IoT devices per group is equal to ($= 1, 5, 50$, and 100) and varying the number of groups of IoT devices. Figure 5.8 (a), shows that the direct authentication scheme has the lowest computational overhead when there is only one IoT device per group, which is explained by using simple authentication operations. Otherwise, as shown in Figure 5.8 (b)-(d), when the number of IoT devices increases per group, our scheme achieves lower computational overhead than other schemes PGA, GBA, BLA, BMEC. This can be explained through achieving only one group authentication instead of many group authentications with the requested IoT services. At this

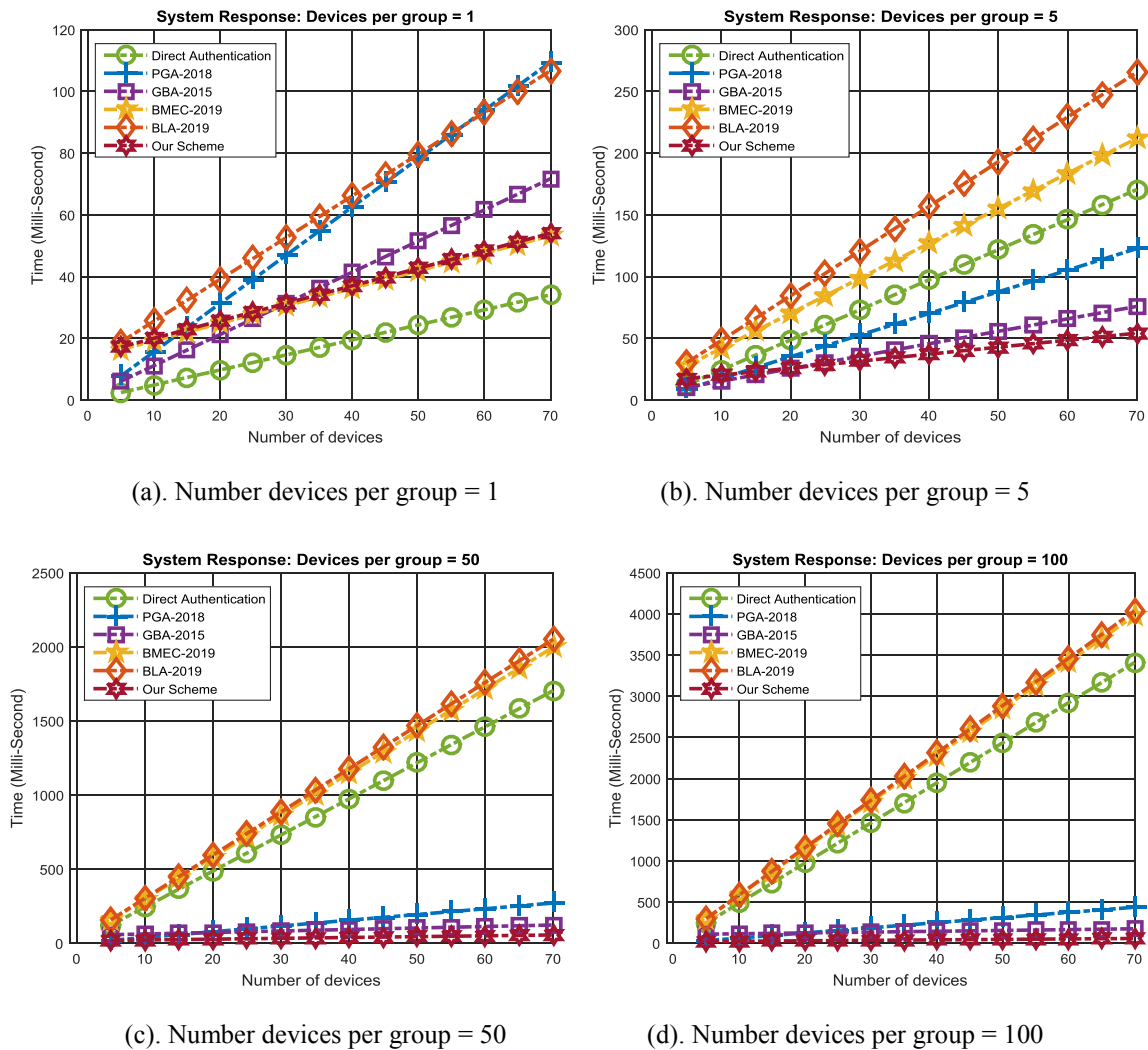


Figure.5. 8: System Response Time

level, we confirm that adopting a distributed authentication mechanism to the group authentication performs a quick authentication procedure for a large and distributed IoT environment and present an evident improvement in the system response.

5.7.3. Energy Consumption

The energy is evaluated regarding the energy dissipation during the cryptography operations. To evaluate the energy costs, we assume that the number of IoT devices' groups and the number of devices per group is variable. Table 5.4 shows the setting of parameters for evaluating energy consumption. The values adopted were carefully chosen based on the values used by [185].

Table 5. 4: Energy costs

Type	Energy
Key set-up 256 bits AES	9.92 μ J
Hashing 512 bits	48.64 μ J
Key generation -ECDH	276.7 Mj
Key exchange-ECDH	163.5 mJ
Signature creation ECDSA	134.2 mJ
Signature verifying ECDSA	196.2 mJ

It is clear from Figure 5.9 that the energy consumption increases with the number of IoT devices' groups and the total number of IoT devices for the state-of-the-art schemes PGA, direct authentication, GBA, BLA, BMEC, while our proposed is less affected by the total number of IoT devices. Hence, ensuring a group-based authentication in a distributed manner may decrease the cryptographic operations efficiently. Furthermore, when authentication is needed, our scheme DiGABlock ensures only once a complete group authentication and then delivers

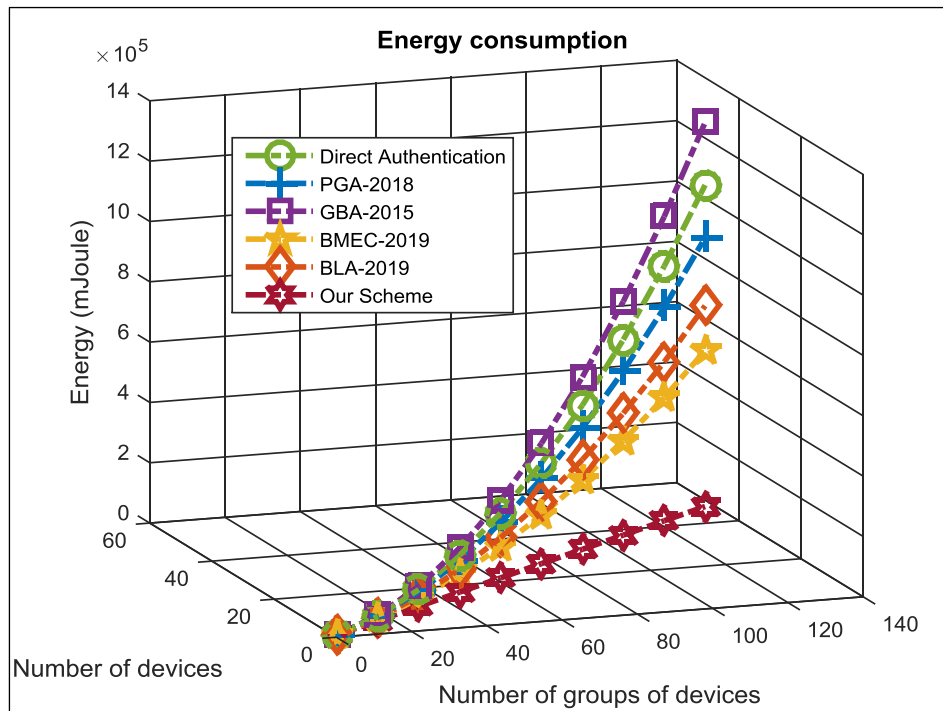


Figure.5. 9: Energy consumption

the remaining IoT groups' services. Therefore, the running cryptographic algorithm of our scheme DiGABlock can save energy consumption.

5.7.4. Communication Costs

To analyze the communication overhead, we calculate the sizes of the exchanged messages. We assume a variable number of groups of IoT devices, and we study three different cases, where the number of IoT devices per group is equal to 1, 5, 50, and 100. Table 5.5 shows the parameters used to evaluate the communication overhead.

Table 5. 5: Communication Parameters

Parameters	Bits	Parameters	Bits
Id	128	The ECDH key	192
The timestamp	17	The ECDH signature	320
Random value (nonce)	16	ECC point multiplication	163
Hash	512	ECC point addition	163
Lagrange Component	128	AES	256

The communication cost of our scheme DiGABlock and the state-of-the-art PGA [165], direct authentication [58], GBA [170], BLA [177], BMEC [178], is computed using equations 5.9 and 5.10 where N is the number of exchanged messages.

$$Com_{first} = \sum_{i=1}^N |Message_i| \quad (5.9)$$

$$Com_{remaining} = \sum_{i=1}^N |Message_i| \quad (5.10)$$

Where Com_{first} represents the communication cost of the first authentication of the user with the requested IoT service, and $Com_{remaining}$ represents the communication overhead of the authentication with the remaining IoT services, which is ensured in a distributed way in our scheme.

In fact, during the first authentication, the user is invited to ensure a full authentication process with a group of IoT devices in the proposed DiGABlock scheme. Thus, there are three exchanged messages to achieve the first authentication phase:

$$Com_{first} = \sum_{i=1}^3 |Message_i| \text{ where:}$$

- $Message_1 = |Auth_{U_i}| + |token_{U_i}| + |T_{U_i}| + |Block_{id}| = 913 \text{ bits}$
- $Message_2 = |V_x| + |\delta_{x1}| + |T_{U_i}^*| = 529 \text{ bits}$
- $Message_3 = d \times |Lc| = 128d \text{ bits}$

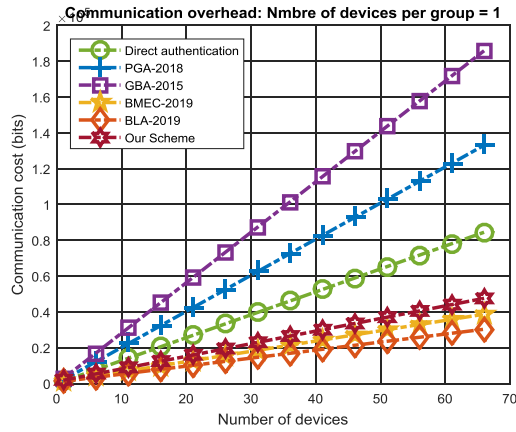
Further, after the first authentication, the user can choose not to re-authenticate by proceeding with the service delivery phase in the proposed DiGABlock scheme. Besides, only one message is exchanged to achieve the service delivery phase:

$$Com_{remaining} = Message_1 = |X_u| + |\delta_{x2}| + |T'_{U_i}| = 529 \text{ bits}$$

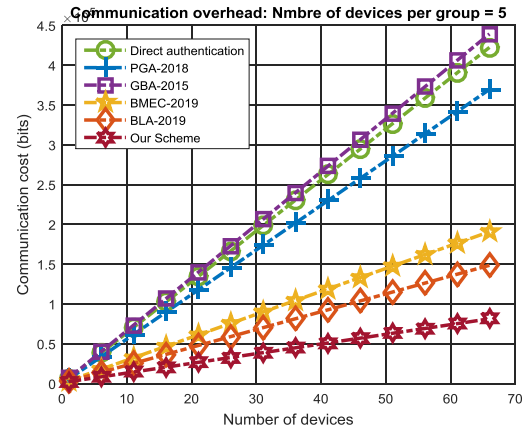
At this level, the overall communication cost of DiGABlock for m groups of IoT services and d devices per group is calculated as follow:

$$Com_{total} = Com_{first} + m \times Com_{remaining}$$

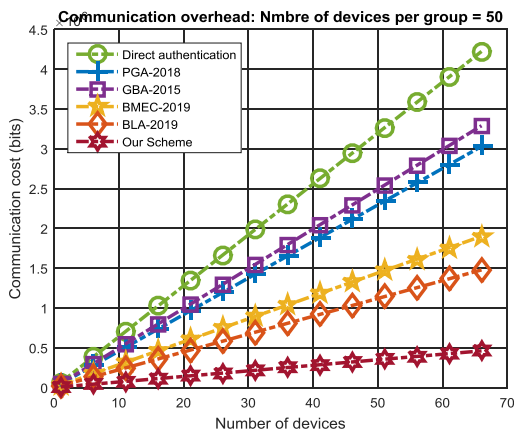
Figure 5.10 (a)-(c) plot the communication cost during the authentication procedure in each scheme, varying the number of groups, while the number of IoT devices per group is set to 1, 5, 50, and 100. We notice that both our scheme and group-based authentication schemes do not save the communication cost, when the number of IoT devices per group is limited to one, compared to distributed and direct authentication ones. Otherwise, when the number of IoT devices per group is more than one, we can see that our scheme is more effective than the others. The reason is that our scheme achieves only one group authentication, which will be distributed for all remaining IoT groups, instead of authenticating every IoT group, and distributing authentication of every IoT device. Hence, our scheme achieves the lowest communication cost when the number of IoT devices is significant.



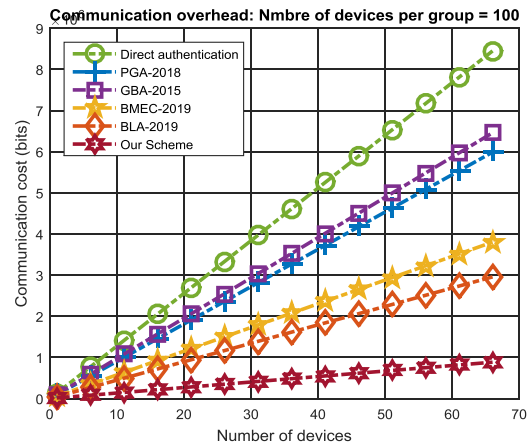
(a). Number devices per group = 1



(b). Number devices per group = 5



(c). Number devices per group = 50



(d). Number devices per group = 100

Figure.5. 10: Communication costs

5.7.5. Improvement Rate of Communication Costs

To better visualize the communication costs enhancements accomplished by our proposed DiGABlock scheme compared to PGA [165], direct authentication [58], GBA [170], BLA [177], BMEC [178], we present an improvement rate (IR), as described in [186] and given by equation 5.11.

$$IR = \frac{\text{Com_cost of the OTHER scheme} - \text{Com_cost of OUR scheme}}{\text{Com_cost of the OTHER scheme}} \quad (5.11)$$

Figure 5.11 shows the improvement rate for the communication cost of our scheme compared to the group-based authentication schemes [165][170] and the distributed authentication schemes [177][178]. We compare the IR for communication varying the number of IoT devices to which a user is subscribed. We notice an improvement of 90% in the communication cost (IR = 0.9) of DiGABlock compared to group-based authentication schemes PGA and GBA when the number of devices is more than nearly five devices. We can explain the results by using the *SM* to manage the communication with the IoT device group, which reduces the exchanged messages resulting during the authentication procedure. Besides, the DiGABlock IR compared to the distributed BLA and BMEC schemes is higher than 0.7, which means an improvement of 70 % in the communication cost. The reason is that a distributed scheme needs to distribute the authentication for each IoT device, while the proposed DiGABlock scheme assures the distribution of the authentication for a group of IoT devices. Hence, the communication cost of DiGABlock is slightly affected by increasing the network size.

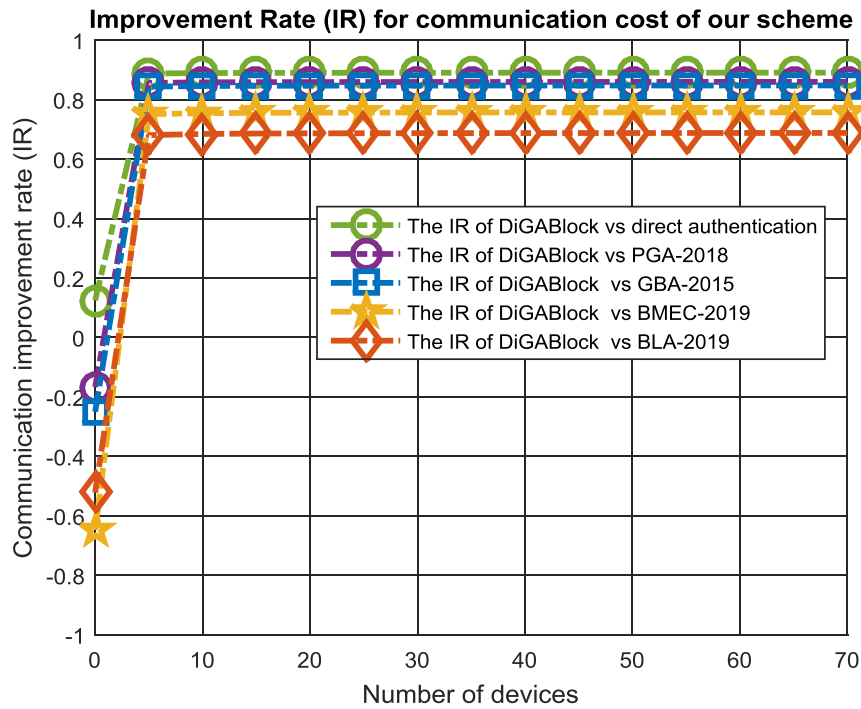


Figure.5. 11: Improvement Rate

5.8. Conclusion

Thanks to the recent technological advances, group-based applications in IoT networks are currently merging as a versatile paradigm that can be used in different IoT environments such as smart hotels, smart cities, smart grids, and smart buildings. However, the corresponding span of applications requires very often efficient authentication with high security in a limited amount of time. Nonetheless, on the one hand, due to the limited computation and energy resources available within the IoT devices, time delay and energy consumption for these constrained devices remain a significant challenge. On the other hand, the group IoT services and the dynamic nature of these IoT environments, as seen in the previous chapter, provide new authentication issues to mitigate these constrained devices and frequent subscribers' interest changing. This chapter introduces a novel distributed group authentication scheme based on Blockchain technology for IoT environment named DiGABlock. A hierarchical authentication architecture is adopted, composed of IoT devices layer, end-users layer, Blockchain edge layer, and Blockchain network layer. DiGABlock is designed to authenticate users with many IoT device groups efficiently, where users could perform only once a full authentication process with many IoT device groups. Besides, DiGABlock is non-interactive which significantly reduces the communication delay. Furthermore, our scheme achieves a trusted and efficient authentication of users and reduces authentication signaling congestion. Indeed, the evaluation of attack models proves that our scheme is attack-prevented. Additionally, we valued the proposed authentication mechanism in terms of communication, energy, and computation costs. Simulation results prove that DiGABlock offers enhanced performances.

Conclusion

6.1 Summary of the Contributions

The IoT is creating new opportunities to ameliorate the next generation of information technologies such as smart homes, smart buildings, smart grids, intelligent transportation, and smart cities. However, its extensive evolution through connecting billions of IoT devices increases the attack surfaces. In particular, securing access to the devices taking into consideration the scalability, heterogeneity, constrained resource nature, and dynamic structure of the IoT, remains a significant challenge. Thus, despite the promising growth of the IoT, numerous unconventional characteristics point out several security requirements that should be addressed to define a secure IoT system. Throughout this thesis, we focused on designing a secure IoT system that achieves the main security concepts, namely the authentication and the authorization for a large-scale IoT environment. For this issue, we are dealing with providing an effective and efficient secure IoT solution that achieves a tradeoff between the security requirements and the network performance. Considering the IoT unconventional challenges, including scalability, dynamic changes, limited resources, and heterogeneity, we proposed a secure IoT system comprising three main pillars. The lightweight authentication scheme meets the requirement of limited resources and the dynamic changes related to the one-to-one scenarios. The authentication scheme is based on a token of identification to secure access for a prefixed time interval. Then to share access between many users and devices in the IoT, a decentralized access control mechanism was introduced using a decentralized group key management to meet the scalability, heterogeneity, and dynamic changes issues. In particular, this scheme controls the access permissions for extensive scale communications based on groups. Therefore, the proposed authentication scheme for one-to-one scenarios is inefficient for such group communication, which explains the third pillar subject. Indeed, a distributed group authentication based on blockchain technology is adopted to our IoT system to meet the requirement of a large-scale heterogeneous environment. We briefly conclude our contributions in what follows:

- **Token-based Lightweight User Authentication scheme for IoT (TBLUA):** in this scheme, we designed an extra security layer in the authentication process. Indeed, using the token of identification, we provide secure authentication for a predefined period and meet the requirement of the dynamic changes in IoT. In fact, the proposed scheme ensures mutual authentication between the communicating parties such as the user and an IoT device. Throughout this scheme, we ensure a tradeoff between effectiveness and efficiency, providing relatively more security features and high-security levels such as anonymity, perfect forward secrecy, and resilience against the well-known attacks. TBLUA also achieves a low computation and communication overhead compared to benchmarking schemes.
- **Designing a Smart Hotel use case prototype:** We adopted the token-based authentication scheme in a mockup representing a reservation system in a smart hotel. Indeed, we considered a smart lock scenario, where a guest who has made a reservation for accommodation in a hotel could use his/her smartphone to enter the reserved room. For that, we reviewed the vulnerability of the smart hotel environment and analyzed the risks that could happen. Then, we designed our architecture composed of a smart lock communicating through NFC with the smartphone and through ZigBee with the server. Besides, we successfully ensured a secure reservation that generates tokens for users to open the smart lock.
- **A Decentralized Lightweight Group Key Management for Access Control for IoT environment (DLGKM-AC):** This scheme is introduced to manage the access permissions in a dynamic IoT environment. Indeed, users and IoT devices might want to access the same IoT devices. Thus, regarding the large scale of the IoT environment characterized by a dynamic nature, controlling access becomes challenging. Therefore, we adopt a hierarchical architecture using one Key Distribution Center (KDC) and several Sub KDC (SKDCs) for managing dissemination keys of access. The KDC implements many Logical Key Hierarchical (LKH) tree to manage the broadcast and update keys of groups of IoT devices publishing data, while SKDCs handle the direct communication links between devices and users. Besides, a new master token encryption algorithm has been designed to ensure members' independence in highly dynamic group communication. Thus, the frequent users' join and leave events do not impact the rekeying process in the whole system, which could alleviate the communication overhead. In DLGKM-AC, mobility is smoothly handled as we provide the backward and the forward secrecy with fewer rekeying operations. Furthermore, our protocol mitigates the 1-affects-n issue. Indeed, users can always get access to data even if one SKDC is affected. We prove that a wide range of desired security properties has also been provided throughout an extensive security analysis. Additionally, performance analyses show that the proposed scheme offers better performances by reducing storage, communication, and computation overheads. Finally, adopting a decentralized architecture increases scalability and reduces overhead for resource-constrained devices.

- **A Distributed Group Authentication scheme based on Blockchain technology for IoT environment (*DiGABlock*):** DiGABlock was introduced to ameliorate the authentication and provide a fully distributed authentication with many groups of IoT devices. Since the number of connected devices is growing, the number of offered IoT services is essential. Therefore, to ensure a high-security level for the users requesting many IoT services, DiGABlock responds to these requirements. Indeed, a hierarchical authentication architecture is adopted, consisting of IoT devices layer, end-users layer, Blockchain edge layer, and Blockchain network layer. DiGABlock is designed to authenticate users with many IoT device groups efficiently, where users could perform only once a full authentication process with an IoT device group. They can then choose to non-re-authenticate when demanding access to the other IoT device groups by performing the service delivery process. Besides, DiGABlock reduces the communication delay significantly. Furthermore, our scheme achieves a trusted and efficient authentication of users and reduces the authentication signaling congestion. Indeed, the evaluation of attack models proves that our scheme is attack-prevented. Additionally, we valued the proposed authentication mechanism in terms of communication, energy, and computation costs. Simulation results demonstrate that DiGABlock offers better performances compared to benchmarking works.

To sum up, the design of DiGABlock and DLGKM-AC for authentication and authorization in IoT is ensured without relying on third-party authorities, which respond to the IoT environment's scalability and heterogeneity characteristics. In addition, the proposed architecture not only eliminates the dependence on the third party but also enhances the flexibility of an IoT system characterized by a dynamic changing structure. Our work is open to possible extensions to enlarge the treated challenges and face new emerging ones. Therefore, we enumerate the possible enhancements, which is the aim of our future directions described in the next section.

6.2 Future Research Directions

Regardless of the presented contributions provided during this thesis to enhance IoT system security, some aspects can be additionally explored and extended for a more secure IoT system. Therefore, we devote this section to identify and study some perspectives and possible future research directions.

In the decentralized access control DLGKM-AC scheme, we need to build a threat detection system to prevent malicious attacks and provide additional security guarantees. Indeed, to build a secure smart IoT environment, preserving security and privacy is crucial. Since the vulnerabilities of such an environment create many threats that affect the normal functioning of IoT systems. Therefore, designing an intrusion detection system might be important to mitigate the exploitation of these vulnerabilities. We can adopt the KDC as a point of intrusion detection of IoT devices as it is responsible for devices' access activities while considering SKDCs as a point of intrusion detection for users. In fact, SKDCs supervise the local access related to users' activities in the system, which can help to describe each user's behavior and detect abnormal

activities. In addition, implementing an automatic detection intrusion system for access control activities guarantees IoT security at the runtime of access control sessions and during the rekeying process. These detection skills might avoid abusing the sensitive exchanged information and penetrating smart devices. It can then effectively improve the security model by detecting the known and unknown threats with an excellent level rate and low false positive alarms.

TBLUA ensures mutual authentication between a user and IoT devices with respect to the limited resource requirement of IoT devices. However, this scheme relies on a third party to guarantee the authentication, which may lead to overload and congestion of the IoT system when many users request the IoT devices. Therefore, and regarding the large-scale IoT environment, we propose a distributed group authentication scheme based on the blockchain technology DiGABlock that ensures a user group authentication in a distributed manner. However, IoT devices could also request and subscribe to other IoT devices. Hence, we intend to extend the distributed authentication protocol between IoT devices to secure exchanging information. At this level, we should pay more attention to possible distributed denial of service (DDoS) attacks that affect the IoT system's availability. Despite the blockchain mechanism being used as a potential solution to avoid the DDoS, the DDoS can overload the blockchain network by consuming the network's resources and making IoT services unreachable promptly. Therefore, a blockchain network that cannot reach the consensus to deliver IoT services might bring down the whole IoT system. Further studies need to be investigated to ameliorate the security against DDoS attacks and ensure a high IoT system availability. Besides, the traditional cryptography algorithms based on symmetric key and public-key are insufficient to maintain the security among the blockchain network. Indeed, the rise of quantum computers can breakdown the cryptography of the blockchain. Therefore, it is necessary to explore and improve the cryptography algorithm security by using the blind signature, ring signature, and aggregate signature [187]. Hence, advanced studies need to be handled at this level to keep the security and the normal functioning of blockchain through the IoT environment.

Bibliography

- [1] P. Bellavista, G. Cardone, A. Corradi and L. Foschini, "Convergence of MANET and WSN in IoT Urban Scenarios," in *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3558-3567, Oct. 2013.
- [2] IoT trends to expect in 2021, <https://krakul.eu/iot-trends-to-expect/>, visited (12/02/2021).
- [3] Pamela Cobb. German Steel Mill Meltdown: Rising Stakes in the Internet of Things. 2015. url: <https://securityintelligence.com/german-steelmill-meltdown-rising-stakes-in-the-internet-of-things/> visited (12/02/2021).
- [4] Corero. The Mirai Botnet: All About the Latest Malware DDoS Attack Type | Corero. 2016. url: <https://www.corero.com/resources/ddos-attacktypes/mirai-botnet-ddos-attack.html>, visited (12/02/2021).
- [5] Catalin Cimpanu, "BrickerBot Dev Claims Cyber-Attack That Affected Over 60,000 Indian Modems", July 2017, [Online]. Available: www.bleepingcomputer.com/news/security/brickerbot-dev-claims-cyber-attack-thataffected-over-60-000-indian-modems/, visited (12/02/2021).
- [6] Wind River. "Security in the Internet of Things: Lessons from the Past for the Connected Future," 2015.
- [7] European ITEA3 PARFAIT (Personal dAta pRotection FrAmework for IoT) project, <https://itea3-parfait.com/>
- [8] Makhdoom, M. Abolhasan, J. Lipman, R. P. Liu and W. Ni, "Anatomy of Threats to the Internet of Things," in *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1636-1675, Secondquarter 2019.
- [9] Sowmya Ravidas, Alexios Lekidis, Federica Paci, Nicola Zannone, Access control in Internet-of-Things: A survey, *Journal of Network and Computer Applications*, Volume 144, 2019, Pages 79-101, ISSN 1084-8045, <https://doi.org/10.1016/j.jnca.2019.06.017>.
- [10] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things : A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4) :2347–2376, 2015.
- [11] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang. A survey on mobile edge networks: Convergence of computing, caching and communications. *IEEE Access*, 5 :6757–6779, 2017.
- [12] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, "Security of the internet of things: perspectives and challenges," *Wireless Networks*, vol. 20, no. 8, pp. 2481–2501, 2014.
- [13] R. Khan, S. U. Khan, R. Zaheer and S. Khan, "Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges," 2012 10th International Conference on Frontiers of Information Technology, Islamabad, 2012, pp. 257-260
- [14] M. Wu, T. J. Lu, F. Y. Ling, J. Sun, and H. Y. Du, "Research on the architecture of Internet of Things," 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), Chengdu, 2010, pp. V5-484- V5-487
- [15] L. Tan and N. Wang, "Future internet: The Internet of Things," 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), Chengdu, 2010, pp. V5-376-V5-380.
- [16] Lauren Horwitz, IoT Trends 2021, online 7th January 2021, <https://www.iotworldtoday.com/2021/01/07/iot-trends-2021-a-focus-on-fundamentals-not-nice-to-haves/>
- [17] J. Liu, Y. Xiao, S. Li, W. Liang, and C. P. Chen. Cyber security and privacy issues in smart grids. *IEEE Communications Surveys & Tutorials*, 14(4) :981–997, 2012.
- [18] F. Dalipi and S. Y. Yayilgan. Security and privacy considerations for iot application on smart grids: Survey and research challenges. In 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), pages 63–68. IEEE, Aug 2016.
- [19] Y. Leng and L. Zhao. Novel design of intelligent internet-of-vehicles management system based on cloud-computing and internet-of-things. In *Proceedings of 2011 International Conference on Electronic Mechanical Engineering and Information Technology*, volume 6, pages 3190–3193. IEEE, Aug 2011
- [20] M. Gerla, E. K. Lee, G. Pau, and U. Lee. Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In 2014 IEEE World Forum on Internet of Things (WF-IoT), pages 241–246. IEEE, March 2014.
- [21] X. Yue, H. Wang, D. Jin, M. Li, and W. Jiang. Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control. *Journal of medical systems*, 40(10) :218, 2016.

- [22] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. Internet of things for smart cities. *IEEE Internet of Things Journal*, 1(1) :22–32, Feb 2014.
- [23] A.-R. Sadeghi, C. Wachsmann, and M. Waidner. Security and privacy challenges in industrial internet of things. In 2015 52nd ACM/EDAC/IEEE on Design Automation Conference (DAC), pages 1– 6. IEEE, June 2015.
- [24] C. Wang, Z. Bi and L. D. Xu, "IoT and Cloud Computing in Automation of Assembly Modeling Systems," in *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1426-1434, May 2014.
- [25] D. Uckelmann, "Performance measurement and cost benefit analysis for RFID and Internet of Things implementations in logistics," in *Quantifying the Value of RFID and the EPCglobal Architecture Framework in Logistics*. New York, NY, USA: Springer-Verlag, 2012, pp. 71–100
- [26] Satamraju KP, B M. Proof of Concept of Scalable Integration of Internet of Things and Blockchain in Healthcare. *Sensors (Basel)*. 2020;20(5):1389. Published 2020 Mar 3. doi:10.3390/s20051389
- [27] M. Roopaei, P. Rad and K. R. Choo, "Cloud of Things in Smart Agriculture: Intelligent Irrigation Monitoring by Thermal Imaging," *IEEE Cloud Computing*, vol. 4, no. 1, pp. 10-15, 2017
- [28] S. Lanzisera, A. R. Weber, A. Liao, D. Pajak and A. K. Meier, "Communicating Power Supplies: Bringing the Internet to the Ubiquitous Energy Gateways of Electronic Devices," *IEEE Internet of Things Journal*, vol. 1, no. 2, pp. 153-160, April 2014.
- [29] D. Macedo, L. A. Guedes and I. Silva, "A dependability evaluation for Internet of Things incorporating redundancy aspects," *Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control*, Miami, FL, 2014, pp. 417-422.
- [30] H. Lamaazi, N. Benamar, A. J. Jara, L. Ladid and D. El Ouadghiri, "Challenges of the Internet of Things: IPv6 and Network Management," 2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Birmingham, 2014, pp. 328-333.
- [31] S. Choi and S. Koh, "Use of Proxy Mobile IPv6 for Mobility Management in CoAP-Based Internet-of-Things Network," *IEEE Communications Letters*, vol 20, no.11, pp2284-2287, Nov 2016
- [32] F. Ganz, R. Li, P. Barnaghi and H. Harai, "A Resource Mobility Scheme for Service-Continuity in the Internet of Things," 2012 IEEE International Conference on Green Computing and Communications, Besancon, 2012, pp. 261-264.
- [33] A. Dunkels, J. Eriksson, and N. Tsiftes, "Low-power interoperability for the IPv6-based Internet of Things," 10th Scandinavian Workshop Wireless ADHOC, Stockholm, Sweden, 2011, pp. 10–11.
- [34] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in internet of things: The road ahead," *Computer Networks*, 76:146–164, 2015.
- [35] K. T. Nguyen, M. Laurent, and N. Oualha. "Survey on secure communication protocols for the internet of things," *Ad Hoc Networks*, 32:17–31, 2015
- [36] B. Miller and D. Rowe, "A survey scada of and critical infrastructure incidents," 1st Annual conference on Research in information technology, ACM, 2012, pp 51–56.
- [37] Sabrina Sicari et al. "A Secure and Quality-Aware Prototypical Architecture for the Internet of Things." *Information Systems*, vol. 58, 2016, pp. 43–55.
- [38] Djamel Eddine Kouicem, Abdelmadjid Bouabdallah, Hicham Lakhlef. Internet of things security: A top-down survey. *Computer Networks*, Elsevier, In press, 141, pp.199-221.
- [39] El Mouaatamid, O.; Lahmer, M.; Belkasmi, M. Internet of Things Security: Layered classification of attacks and possible Countermeasures. *Electron. J. Inf. Technol.* 2016, 9, 24–37.
- [40] Abomhara, M.; Koien, G.M. Cyber Security and the Internet of Things: Vulnerabilities, Threats, Intruders and Attacks. *J. Cyber Security. Mobil.* 2015, 4, 65–88
- [41] Tuhin Borgohain, Uday Kumar, and Sugata Sanyal. "Survey of Security and Privacy Issues of Internet of Things," 2015
- [42] ITU-T. ITU-T Recommendation Y.2060 Overview of the Internet of Things, Series Y: Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks. 2012.
- [43] Michael J. Covington and Rush Carskadden. "Threat Implications of the Internet of Things." *Cyber Conflict (CyCon)*, 2013 5th International Conference On. IEEE, 2013
- [44] David Evans and David M. Eysers. "Efficient Data Tagging for Managing Privacy in the Internet of Things." *Proceedings of the 2012 IEEE International Conference on Green Computing and Communications (GreenCom)*. IEEE, 2012, pp. 244–248.

- [45] Kim, H. Securing the Internet of Things via Locally Centralized, Globally Distributed Authentication and Authorization. UC Berkeley. ProQuest ID: Kim_berkeley_0028E_17305. Merritt ID: ark:/13030/m5ng9mkk.
- [46] H. Noura. Adaptation of Cryptographic Algorithms According to the Applications Requirements and Limitations: Design, Analyze and Lessons Learned. HDR dissertation, UNIVERSITY of PIERRE MARIE CURIE -Paris VI, 2016.
- [47] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan. "Internet of things (iot) security: Current status, challenges and prospective measures". 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), December 2015.
- [48] X. Zhu, "Building a secure infrastructure for IoT systems in distributed environments", these de doct., Universit e de Lyon, 2019
- [49] A. Banks and R. Gupta, MQTT version 3.1.1, OASIS Standard, <http://docs.oasisopen.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>, 2014.
- [50] El-hajj, M.; Chamoun, M.; Fadlallah, A.; Serhrouchni, A. Analysis of authentication techniques in Internet of Things (IoT). In Proceedings of the 2017 1st Cyber Security in Networking Conference (CSNet), Rio de Janeiro, Brazil, 18–20 October 2017; pp. 1–3.
- [51] El-hajj, M.; Chamoun, M.; Fadlallah, A.; Serhrouchni, A. Taxonomy of authentication techniques in Internet of Things (IoT). In Proceedings of the 2017 IEEE 15th Student Conference on Research and Development (SCORED), Putrajaya, Malaysia, 13–14 December 2017; pp. 67–71.
- [52] El-hajj M, Fadlallah A, Chamoun M, Serhrouchni A. A Survey of Internet of Things (IoT) Authentication Schemes. *Sensors*. 2019; 19(5):1141. <https://doi.org/10.3390/s19051141>
- [53] Brian, A.L.A.; Arockiam, L.; Malarchelvi, P. An IOT based secured smart library system with NFC based book tracking. *Int. J. Emerg. Technol. Comput. Sci. Electron.* 2014, 11, 18–21.
- [54] Kumar, P.; Lee, S.G.; Lee, H.J. E-SAP: Efficient-Strong Authentication Protocol for Healthcare Applications Using Wireless Medical Sensor Networks. *Sensors* 2012, 12, 1625–1647.
- [55] Y. Sharaf-Dabbagh and W. Saad, "On the authentication of devices in the internet of things," 2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), June 2016, pp. 1–3.
- [56] Ashibani, Y., Mahmoud, Q.H. Design and evaluation of a user authentication model for IoT networks based on app event patterns. *Cluster Comput* (2020). <https://doi.org/10.1007/s10586-020-03156-5>
- [57] Alizai, Z.A.; Tareen, N.F.; Jadoon, I. Improved IoT Device Authentication Scheme Using Device Capability and Digital Signatures. In Proceedings of the 2018 International Conference on Applied and Engineering Mathematics (ICAEM), Taxila, Pakistan, 4–5 September 2018; pp. 1–5.
- [58] M. Wazid, A. K. Das, V. Odelu, N. Kumar, M. Conti and M. Jo, "Design of Secure User Authenticated Key Management Protocol for Generic IoT Networks," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 269-282, Feb. 2018.
- [59] Sadhukhan, D., Ray, S., Biswas, G.P. et al. A lightweight remote user authentication scheme for IoT communication using elliptic curve cryptography. *J Supercomput* 77, 1114–1151 (2021). <https://doi.org/10.1007/s11227-020-03318-7>
- [60] M. Malik, M. Dutta and J. Granjal, "A Survey of Key Bootstrapping Protocols Based on Public Key Cryptography in the Internet of Things," in *IEEE Access*, vol. 7, pp. 27443-27464, 2019, doi: 10.1109/ACCESS.2019.2900957.
- [61] M. Schukat and P. Cortijo, "Public key infrastructures and digital certificates for the Internet of things," 2015 26th Irish Signals and Systems Conference (ISSC), Carlow, 2015, pp. 1-5.
- [62] F. Forsby, M. Furuheid, P. Papadimitratos, and S. Raza, "Lightweight X. 509 digital certificates for the Internet of Things," *Interoperability, Safety and Security in IoT*. Springer, 2017, pp. 123–133.
- [63] P. Porambage, P. Kumar, A. Gurtov, M. Ylianttila and E. Harjula, Certificate Based Keying Scheme for DTLS Secured IoT (Work in Progress), Jun. 2013.
- [64] P. Porambage, C. Schmitt, P. Kumar, A. Gurtov and M. Ylianttila, "Two-phase authentication protocol for wireless sensor networks in distributed IoT applications", *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, pp. 2728-2733, Apr. 2014.
- [65] . P. Porambage, C. Schmitt, P. Kumar, A. Gurtov and M. Ylianttila, "PAuthKey: A pervasive authentication protocol and key establishment scheme for wireless sensor networks in distributed IoT applications", *Int. J. Distrib. Sensor Netw.*, vol. 10, no. 7, 2014.

- [66] Chae, C.J.; Choi, K.N.; Choi, K.; Yae, Y.H.; Shin, Y. The Extended Authentication Protocol using E-mail Authentication in OAuth 2.0 Protocol for Secure Granting of User Access. *J. Internet Comput. Serv.* 2015, 16, 21–28
- [67] Emerson, S.; Choi, Y.K.; Hwang, D.Y.; Kim, K.S.; Kim, K.H. An OAuth based authentication mechanism for IoT networks. In *Proceedings of the 2015 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, Korea, 28–30 October 2015.
- [68] Blazquez, A.; Tsiatsis, V.; Vandikas, K. Performance evaluation of openid connect for an iot information marketplace. In *Proceedings of the 2015 IEEE 81st IEEE Vehicular Technology Conference (VTC Spring)*, Glasgow, UK, 11–14 May 2015; pp. 1–6.
- [69] C. Neuman, T. Yu, S. Hartman, and K. Raeburn, The Kerberos network authentication service (V5), RFC 4120, IETF, Jul. 2005
- [70] M. Koschuch, M. Hudler, H. Eigner and Z. Saffer, "Token-based authentication for smartphones," 2013 International Conference on Data Communication Networking (DCNET), Reykjavik, 2013, pp. 1-6.
- [71] Timothy Claeys, Franck Rousseau, Bernard Tourancheau. Securing Complex IoT Platforms with Token Based Access Control and Authenticated Key Establishment. *International Workshop on Secure Internet of Things (SIOT)*, Sep 2017, Oslo, Norway. fhal-01596135f
- [72] Mukhopadhyay, D. PUFs as Promising Tools for Security in Internet of Things. *IEEE Des. Test* 2016, 33, 103–115.
- [73] Wallrabenstein, J.R. Practical and Secure IoT Device Authentication Using Physical Unclonable Functions. In *Proceedings of the 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, Vienna, Austria, 22–24 August 2016
- [74] Xu, H.; Ding, J.; Li, P.; Zhu, F.; Wang, R. A Lightweight RFID Mutual Authentication Protocol Based on Physical Unclonable Function. *Sensors* 2018, 18, 760
- [75] Gope, P.; Lee, J.; Quek, T.Q.S. Lightweight and Practical Anonymous Authentication Protocol for RFID Systems Using Physically Unclonable Functions. *IEEE Trans. Inf. Forensics Secur.* 2018, 13, 2831–2843
- [76] Lai, C.; Lu, R.; Zheng, D.; Li, H.; Shen, X.S. GLARM: Group-based lightweight authentication scheme for resource-constrained machine to machine communications. *Comput. Netw.* 2016, 99, 66–81
- [77] Shao, J.; Lu, R.; Lin, X.; Zuo, C. New threshold anonymous authentication for VANETs. In *Proceedings of the 2015 IEEE/CIC International Conference on Communications in China (ICCC)*, Shenzhen, China, 2–4 November 2015.
- [78] Shao, J.; Lin, X.; Lu, R.; Zuo, C. A Threshold Anonymous Authentication Protocol for VANETs. *IEEE Trans. Veh. Technol.* 2016, 65, 1711–1720.
- [79] Fu, A.; Lan, S.; Huang, B.; Zhu, Z.; Zhang, Y. A Novel Group-Based Handover Authentication Scheme with Privacy Preservation for Mobile WiMAX Networks. *IEEE Commun. Lett.* 2012, 16, 1744–1747
- [80] P. Porambage, A. Braeken, C. Schmitt, A. Gurtov, M. Ylianttila and B. Stiller, "Group Key Establishment for Enabling Secure Multicast Communication in Wireless Sensor Networks Deployed for IoT Applications," *IEEE Access*, vol. 3, pp. 1503-1511, 2015.
- [81] Zhang, L.; Wu, Q.; Domingo-Ferrer, J.; Qin, B.; Hu, C. Distributed Aggregate Privacy-Preserving Authentication in VANETs. *IEEE Trans. Intell. Transp. Syst.* 2017, 18, 516–526
- [82] Zhang, L.; Hu, C.; Wu, Q.; Domingo-Ferrer, J.; Qin, B. Privacy-preserving vehicular communication authentication with hierarchical aggregation and fast response. *IEEE Trans. Comput.* 2016, 65, 2562–2574
- [83] Sudhakaran, P. and C. Malathy. "Energy efficient distributed lightweight authentication and encryption technique for IoT security." *International Journal of Communication Systems* (2019).
- [84] K. A. R. Rehiman and S. Veni, "A trust management model for sensor enabled mobile devices in IoT", *Proc. Int. Conf. I-SMAC (IoT Social Mobile Anal. Cloud) (I-SMAC)*, pp. 807-810, Feb. 2017
- [85] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller, and Steven Goldfeder. 2016. Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction.
- [86] Rui Zhang, Rui Xue, and Ling Liu. 2019. Security and Privacy on Blockchain. *ACM Comput. Surv.* 1, 1, Article 1 (January 2019), 35 pages.
- [87] S. Bouzeffrane, A. F. B. Mostefa, F. Houacine, and H. Cagnon. Cloudlets authentication in nfc-based mobile computing. In *2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, pages 267–272. IEEE, 2014.

- [88] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. url: <http://www.cryptovest.co.uk/resources/Bitcoin%20paper%20Original.pdf>
- [89] C. Cachin and M. Vukolic. Blockchain consensus protocols in the wild. arXiv preprint arXiv :1707.01873, 2017
- [90] M. Castro, B. Liskov, et al. Practical byzantine fault tolerance. In OSDI, volume 99, pages 173–186, 1999.
- [91] A. Bahga and V. K. Madiseti. Blockchain platform for industrial internet of things. Journal of Software Engineering and Applications, 9(10) :533, 2016.
- [92] K. Christidis and M. Devetsikiotis. Blockchains and smart contracts for the internet of things. IEEE Access, 4 :2292–2303, 2016.
- [93] Y. Yao, X. Chang, J. Mi, V. B. Mi, and L. Li, "BLA: Blockchain Assisted Lightweight Anonymous Authentication for Distributed Vehicular Fog Services," IEEE Internet of Things Journal, 2019, DOI: 10.1109/JIOT.2019.2892009.
- [94] Intel. 2017. Sawtooth Lake. <https://intelledger.github.io/>. (2017).
- [95] Benhadj Djilali H., Tandjaoui D. Efficient Distributed Authentication and Access Control System Management for Internet of Things Using Blockchain. In: Renault É., Boumerdassi S., Leghris C., Bouzeffrane S. (eds) Mobile, Secure, and Programmable Networking. MSPN 2019. Lecture Notes in Computer Science, vol 11557. Springer, Cham. https://doi.org/10.1007/978-3-030-22885-9_5
- [96] S.-C. Cha, J.-F. Chen, C. Su, and K.-H. Yeh, "A blockchain connected gateway for BLE-based devices in the internet of things," IEEE Access, vol. 6, pp. 24 639–724 649, 2018.
- [97] Mostafa Yavari, Masoumeh Saffkhani, Saru Kumari, Sachin Kumar, Chien-Ming Chen, "An Improved Blockchain-Based Authentication Protocol for IoT Network Management", Security and Communication Networks, vol. 2020, Article ID 8836214, 16 pages, 2020.
- [98] Muhammad Tahir & Muhammad Sardaraz & Shakoor Muhammad & Muhammad Saud Khan, 2020. "A Lightweight Authentication and Authorization Framework for Blockchain-Enabled IoT Network in Health-Informatics," Sustainability, MDPI, Open Access Journal, vol. 12(17), pages 1-1, August.
- [99] Sethi, P.; Sarangi, S.R. Internet of things: Architectures, protocols, and applications. J. Electr. Comput. Eng. 2017, 2017, 9324035
- [100] Barka E., Mathew S.S., Atif Y. Securing the web of things with role-based access control. International Conference on Codes, Cryptology, and Information Security. Springer, New York, NY, USA: 2015, pp. 14–26.
- [101] J. A. Stankovic, "Research Directions for the Internet of Things," in IEEE Internet of Things Journal, vol. 1, no. 1, pp. 3-9, Feb. 2014
- [102] J. Herranz, "Attribute-based encryption implies identity-based encryption," in IET Information Security, vol. 11, no. 6, pp. 332-337, 11 2017.
- [103] J. Park, R. Sandhu, Towards usage control models: beyond traditional access control, 7 th ACM Symp. Access Control Model. Technol. - SACMAT '02, New York, New York, USA, 2002, p. 57
- [104] J.L.Hernández-Ramos & al. "Towards a Lightweight Authentication and Authorization Framework for Smart Objects ," IEEE Journal on Selected Areas in Communications, vol. 33, no. 4, pp. 690-702, April 2015.
- [105] Aafaf Ouaddah, Hajar Mousannif, Anas Abou Elkalam, Abdellah Ait Ouahman, Access control in the Internet of Things: Big challenges and new opportunities, Computer Networks, Volume 112, 2017, Pages 237-262, ISSN 1389-1286,
- [106] Sandhu, the typed access matrix model, in: Proc. 1992 IEEE Comput. Soc. Symp. Res. Secur. Priv., IEEE Comput. Soc. Press, 1992, pp. 122–136, doi: 10.1109/RISP.1992.213266.
- [107] A. Arfaoui S. Cherkaoui, A. Kribeche, SM. Senouci, M. Hamdi, "Context-Aware Adaptive Authentication and Authorization in Internet of Things", IEEE ICC'2019, China, 20-24 May 2019.
- [108] H. Harney Group Key Management Protocol (GKMP) Architecture, Network Working Group, <https://tools.ietf.org/html/rfc2094>
- [109] H. Harney and E. Harder, "Logical key hierarchy protocol," Internet draft, Tech. Rep., 1999
- [110] Balenson, D., D. McGrew and A. Sherman. "Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization." (2000).
- [111] L. Veltri, S. Cirani, S. Busanelli, and G. Ferrari, "A novel batch-based group key management protocol applied to the internet of things," AdHoc Networks, vol. 11, no. 8, pp. 2724–2737, 2013.

- [112] M.-H. Park, Y.-H. Park, H.-Y. Jeong and S.-W. Seo, "Key management for multiple multicast groups in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 9, pp. 1712–1723, 2013
- [113] Y. Kung and H. Hsiao, "GroupIt: Lightweight Group Key Management for Dynamic IoT Environments," in *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 5155–5165, Dec. 2018.
- [114] C. Boyd, "On key agreement and conference key agreement," In *Proceedings of the Information Security and Privacy: Australasian Conference, Lecture Notes in Computer Science* vol. 1270. Springer-Verlag, New York, pp. 294–302, 1997.
- [115] O. Rodeh, K. Birman, and D. Dolev, "Optimized group rekey for group communication systems," In *Network and Distributed System Security*, San Diego, Calif., Feb. 2000.
- [116] L. Dondeti, S. Mukherjee, and A. Samal, "A distributed group key management scheme for secure many-to-many communication," *Tech. Rep. PINTL-TR-207-99*, Department of Computer Science, University of Maryland, 1999
- [117] Y. Kim, A. Perrig, and G. Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," In *Proceedings of the 7th ACM conference on Computer and Communications security*, pp. 235–244, 2000
- [118] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The VersaKey framework: Versatile group key management," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 9, pp. 1614–1631, Sept. 1999. 41. C.K. Wong, M. Gouda,
- [119] S. Tang, L. Xu, N. Liu, X. Huang, J. Ding and Z. Yang, "Provably Secure Group Key Management Approach Based upon Hyper-Sphere," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3253–3263, Dec. 2014, doi: 10.1109/TPDS.2013.2297917.
- [120] Abdmeziem M.R., Charoy F. (2018) Fault-Tolerant and Scalable Key Management Protocol for IoT-Based Collaborative Groups. In: Lin X., Ghorbani A., Ren K., Zhu S., Zhang A. (eds) *Security and Privacy in Communication Networks. SecureComm 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol 239. Springer, Cham
- [121] Rafaei, S., D. Hutchison. A Survey of Key Management for Secure Group Communication. – *ACM Computing Surveys*, Vol. 35, September 2003, pp. 309–329.
- [122] M. R. Abdmeziem, D. Tandjaoui, and I. Romdhani, "A decentralized batch-based group key management protocol for mobile internet of things (dbgk)," 2015 *IEEE International Conference on Computer and Information Technology*.
- [123] T. T. Mapoka, S. J. Shepherd and R. A. Abd-Alhameed, "A New Multiple Service Key Management Scheme for Secure Wireless Mobile Multicast," in *IEEE Transactions on Mobile Computing*, vol. 14, no. 8, pp. 1545–1559, 1 Aug. 2015.
- [124] Zhong, H., Luo, W., and Cui, J. (2017) Multiple multicast group key management for the Internet of People. *Concurrency Computat.*:doi: 10.1002/cpe.3817
- [125] A. Arfaoui, A. Kribeche, O. Merad Boudia, SM. Senouci, M. Hamdi "Context-Aware Access Control and Anonymous Authentication in Wireless Body Area Networks", *Computers & Security*, 2019.
- [126] M. A. Ferrag, L. A. Maglaras, H. Janicke, J. Jiang, and L. Shu, "Authentication Protocols for Internet of Things: A Comprehensive Survey," *Security and Communication Networks*, vol. 2017.
- [127] O. O. Bamasag and K. Youcef-Toumi. "Towards Continuous Authentication in Internet of Things Based on Secret Sharing Scheme", In *Proceedings of the WESS'15: Workshop on Embedded Systems Security (WESS'15)*. ACM, New York, NY, USA.
- [128] S. Challa, M. Wazid, A. K. Das, N. Kumar, A. G. Reddy, E. J. Yoon, and K. Y. Yoo, "Secure Signature-Based Authenticated Key Establishment Scheme for Future IoT Applications," *IEEE Access*, vol. 5, pp. 3028–3043, 2017.
- [129] K. H. M. Wong, Yuan Zheng, Jiannong Cao and Shengwei Wang, "A dynamic user authentication scheme for wireless sensor networks," *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06)*, Taichung, 2006, pp. 8 pp.-.
- [130] M. L. Das, "Two-factor user authentication in wireless sensor networks," in *IEEE Transactions on Wireless Communications*, vol. 8, no. 3, pp. 1086–1090, March 2009.

- [131] H. F. Huang, Y. F. Chang and C. H. Liu, "Enhancement of Two-Factor User Authentication in Wireless Sensor Networks," 2010 Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Darmstadt, 2010, pp. 27-30.
- [132] T.H. Chen, W.K. Shih, "A robust mutual authentication protocol for wireless sensor networks", ETRI Journal, vol. 32, Issue5, Oct. 2010, pp. 704-712.
- [133] M.K. Khan, K. Alghathbar, "Cryptanalysis and security improvements of two-factor user authentication in wireless sensor networks", Sensors journal 2010, vol 10, pp.2450–2459.
- [134] DJ He, Y. Gao, S. Chan, C. Chen, J.J. Bu, "An enhanced two-factor user authentication scheme in wireless sensor networks", Journal of Information Security and Applications, vol. 20, Issue C, February 2015, pp. 37-46.
- [135] A.K. Das, P. Sharma, S. Chatterjee, J.K. Sing, "A dynamic password-based user authentication scheme for hierarchical wireless sensor networks", Journal of Network and Computer Applications, vol. 35, Issue 5, September 2012, pp. 1646-1656.
- [136] M. Turkanovic, B. Brumen, and M. Holbl, "A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the Internet of Things notion," Ad Hoc Networks, vol. 20, pp. 96 – 112, 2014.
- [137] C. C. Chang and H. D. Le, "A Provably Secure, Efficient and Flexible Authentication Scheme for Ad hoc Wireless Sensor Networks," IEEE Transactions on Wireless Communications, vol. 15, no. 1, pp. 357–366, 2016.
- [138] C.-T. Li, M.-S. Hwang, and Y.-P. Chu, "A secure and efficient communication scheme with authenticated key establishment and privacy preserving for vehicular ad hoc networks," Computer Communications, vol. 31, no. 12, pp. 2803 – 2814, 2008
- [139] P. Gope and T. Hwang, "A Realistic Lightweight Anonymous Authentication Protocol for Securing Real-Time Application Data Access in Wireless Sensor Networks," in IEEE Transactions on Industrial Electronics, vol. 63, no. 11, pp. 7124-7132, Nov. 2016.
- [140] N. Khalil, M. R. Abid, D. Benhaddou, and M. Gerndt, "Wireless sensors networks for Internet of Things," in IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Singapore, 2014, pp. 1–6.
- [141] "Advanced Encryption Standard (AES)," FIPS PUB 197, National Institute of Standards and Technology (NIST), US Department of Commerce, November 2001. <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.
- [142] Y. Lu, L. Li, H. Peng, et al. "An Energy Efficient Mutual Authentication and Key Agreement Scheme Preserving Anonymity for Wireless Sensor Networks," Sensors, vol. 16, no. 6, article no. 837, 2016.
- [143] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining smart-card security under the threat of power analysis attacks," IEEE Transactions on Computers, vol. 51, no. 5, pp. 541–552, 2002.
- [144] "Secure Hash Standard," FIPS PUB 180-1, National Institute of Standards and Technology (NIST), US Department of Commerce, April 1995.
- [145] T. Team et al. Avispa v1. 1 user manual. Information Society, Technologies Programme (June 2006), <http://avispa-project.org>, 2006.
- [146] AlMajed, H.N.; AlMogren, A.S. Simple and Effective Secure Group Communications in Dynamic Wireless Sensor Networks. *Sensors* 2019, 19,
- [147] A. Mehdizadeh, F. Hashim, and M. Othman, "Lightweight decentralized multicast–unicast key management method in wireless ipv6 networks," Journal of Network and Computer Applications, vol. 42, 2014.
- [148] Zhu, B.; Susilo, W.; Qin, J.; Guo, F.; Zhao, Z.; Ma, J. A Secure and Efficient Data Sharing and Searching Scheme in Wireless Sensor Networks. *Sensors*, 2019, 19, 2583.
- [149] Tan, H.; Chung, I. A Secure and Efficient Group Key Management Protocol with Cooperative Sensor Association in WBANs. *Sensors* 2018, 18, 3930
- [150] Cheikhrouhou O. Secure Group Communication in Wireless Sensor Networks: A survey. Journal of Network and Computer Applications.
- [151] I.-C. Tsai, C.-M. Yu, H. Yokota, and S.-Y. Kuo, "Key management in internet of things via kronecker product," in Dependable Computing (PRDC), 2017 IEEE 22nd Pacific Rim International Symposium on. IEEE, 2017.

- [152] W. Ding *et al.*, "An Extended Framework of Privacy-Preserving Computation with Flexible Access Control," in *IEEE Transactions on Network and Service Management*. TNSM.2019.
- [153] M. Nabeel, N. Shang and E. Bertino, "Privacy Preserving Policy-Based Content Sharing in Public Clouds," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2602-2614, Nov. 2013.
- [154] X. Wang, J. Zhang, E. M. Schooler, and M. Ion, "Performance Evaluation of Attribute-Based Encryption: Toward Data Privacy in the IoT," in *IEEE ICC*, 2014.
- [155] S. Sciancalepore, A. Caposelle, G. Piro, G. Boggia, and G. Bianchi, "Key management protocol with implicit certificates for iot systems," in *Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems*. ACM, 2015, pp. 37–42.
- [156] Y. Tseng, C. Fan and C. Wu, "FGAC-NDN: Fine-Grained Access Control for Named Data Networks," in *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 143-152, March 2019.
- [157] Karuturi, N. N., Gopalakrishnan, R., Srinivasan, R., & Rangan, C. P. (2008). Foundations of Group Key Management-Framework, Security Model and a Generic Construction. IACR Cryptology EPrint.
- [158] <https://github.com/miracl/MIRACL>
- [159] LEVINE, D. "IIoT Challenges and Promises." *Machine Design*, vol. 88, no.7, pp.20-23, July 2016.
- [160] M. Khari, A. K. Garg, A. H. Gandomi, R. Gupta, R. Patan and B. Balusamy, "Securing Data in Internet of Things (IoT) Using Cryptography and Steganography Techniques," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 1, pp. 73-80, Jan. 2020
- [161] Lu, J.Z.; Chen, T.; Zhou, J.; Yang, J.; Jiang, J. An enhanced biometrics-based remote user authentication scheme using smart cards. In *Proceedings of the 2013 6th International Congress on Image and Signal Processing (CISP)*, Hangzhou, China, 16–18 December 2013
- [162] Zhu, H.; Lin, X.; Zhang, Y.; Lu, R. Duth: A user-friendly dual-factor authentication for Android smartphone devices. *Secur. Commun. Netw.* 2014, 8, 1213–1222.
- [163] Su, W.T.; Wong, W.M.; Chen, W.C. A survey of performance improvement by group-based authentication in IoT. In *Proceedings of the 2016 International Conference on Applied System Innovation (ICASI)*, Okinawa, Japan, 26–30 May 2016; pp. 1–4
- [164] Park, Y.; Park, Y. A Selective Group Authentication Scheme for IoT-Based Medical Information System. *J. Med. Syst.* 2017, 41, 48.
- [165] Anxi, W; Jian, S; Leiming Yan Yongjun Ren Qi Liu Year: 2018 A Practical Group Authentication Scheme for Smart Devices in IoT IOT EAI DOI: 10.4108/eai.5-3-2019.156719
- [166] T. Salman, M. Zolanvari, A. Erbad, R. Jain, and M. Samaka, "Security services using Blockchains: A state of the art survey," *IEEE Commun. Surveys Tuts.*, to be published, doi: 10.1109/COMST.2018.2863956
- [167] Chaudhary R, Jindal A, Aujla, et al. "BEST: Blockchain-based secure energy trading in SDN-enabled intelligent transportation system," *Computers and Security*, vol. 85, pp. 288-299, August 2019
- [168] P. K. Sharma, M. Chen and J. H. Park, "A Software Defined Fog Node Based Distributed Blockchain Cloud Architecture for IoT," in *IEEE Access*, vol. 6, pp. 115-124, 2018
- [169] Z. Zhou, B. Wang, M. Dong and K. Ota, "Secure and Efficient Vehicle-to-Grid Energy Trading in Cyber Physical Systems: Integration of Blockchain and Edge Computing," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 1, pp. 43-57, Jan. 2020,
- [170] J. Li, M. Wen and T. Zhang, "Group-Based Authentication and Key Agreement With Dynamic Policy Updating for MTC in LTE-A Networks," in *IEEE Internet of Things Journal*, vol. 3, no. 3, pp. 408-417, June 2016, doi: 10.1109/JIOT.2015.2495321.
- [171] L. Harn, "Group Authentication" in *IEEE Transactions on Computers*, vol. 62, no. 09, pp. 1893-1898, 2013. doi: 10.1109/TC.2012.251
- [172] A. Shamir, "How to Share a Secret," *Comm. ACM*, vol. 22, no. 11, pp. 612- 613, 1979.

- [173]Chien, H.Y. Group Authentication with Multiple Trials and Multiple Authentications. *Secur. Commun. Netw.* 2017, 2017, 3109624.
- [174]Z. Xiong, Y. Zhang, D. Niyato, et al. "When Mobile Blockchain Meets Edge Computing," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 33-39, Aug 2018
- [175]Hammi, M.T.; Hammi, B.; Bellot, P.; Serhrouchni, A. Bubbles of Trust: A decentralized Blockchain-based authentication system for IoT. *Comput. Secur.* 2018, 78, 126–142.
- [176]Wentong Wang, Ning Hu, Xin Liu. "BlockCAM: A Blockchain-based cross-domain authentication model." In 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC), pp. 896-901. IEEE, Jun. 2018.
- [177]Y. Yao, X. Chang, J. Mišić, V. B. Mišić and L. Li, "BLA: Blockchain-Assisted Lightweight Anonymous Authentication for Distributed Vehicular Fog Services," in *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3775-3784, April 2019, doi: 10.1109/JIOT.2019.2892009.
- [178]S. Guo, X. Hu, S. Guo, X. Qiu and F. Qi, "Blockchain Meets Edge Computing: A Distributed and Trusted Authentication System," in *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1972-1983, March 2020, doi: 10.1109/TII.2019.2938001.
- [179]V. Gupta, S. Gupta, S. Chang, and D. Stebila, "Performance analysis of elliptic curve cryptography for SSL." pp. 87-94.
- [180]R. Jiang, J. Luo, F. Tu, and J. Zhong, "LEP: a lightweight key management scheme based on ebs and polynomial for wireless sensor networks," in *Proceedings of the IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC '11)*, pp. 1–5, Xi'an, China, September 2011
- [181]Yuan Zhang, Xiaodong Lin, Chunxiang Xu. "Blockchain-based secure data provenance for cloud storage." In *International Conference on Information and Communications Security*, pp. 3-19. Springer, Cham, Oct. 2018
- [182]S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han and F. Wang, "Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2266-2277, Nov. 2019
- [183]Zhonglin Chen, Shanzhi Chen, Hui Xu, Bo Hu. "A security authentication scheme of 5G ultra-dense network based on block chain." *IEEE Access* 6 (2018): 55372-55379.
- [184]Aggelos Kiayias, Alexander Russell, Bernardo David, Roman Oliynykov. "Ouroboros: A provably secure proof-of-stake Blockchain protocol." In *Annual International Cryptology Conference*, pp. 357-388. Springer, Cham, Aug. 2017.
- [185]Raheem A Beyah, Cherita L Corbett, Janise McNair, *Book Security in Ad-hoc And sensor Networks*
- [186]R. Jiang, C. Lai, J. Luo, X. Wang, H. Wang, EAP based group authentication and key agreement protocol for 624 machine type communication, *International Journal of Distributed Sensor Networks*. (2013). 625 (doi.org/10.1155/2013/304601)
- [187]Zheng, X.; Zhu, Y.; Si, X. A Survey on Challenges and Progresses in Blockchain Technologies: A Performance and Security Perspective. *Appl. Sci.* 2019, 9, 4731. <https://doi.org/10.3390/app9224731>

Appendix 1

A. Transaction Process of the Blockchain

We describe the process of the transaction from generation until validation in the blockchain presented by the hyperledger fabric platform [87]:

- Client initiates the transaction carried out by a client application.
- Endorsing peers verify signature and execute the transaction.
- Client assembles endorsements into a transaction.
- Client disseminates the block to leader peers.
- Peers nodes validate and commit transaction.
- Peers nodes update their ledger.
- Peers nodes notify client with the update.

Figure A.1 illustrates the life cycle of the transaction process of the hyperledger fabric technology.

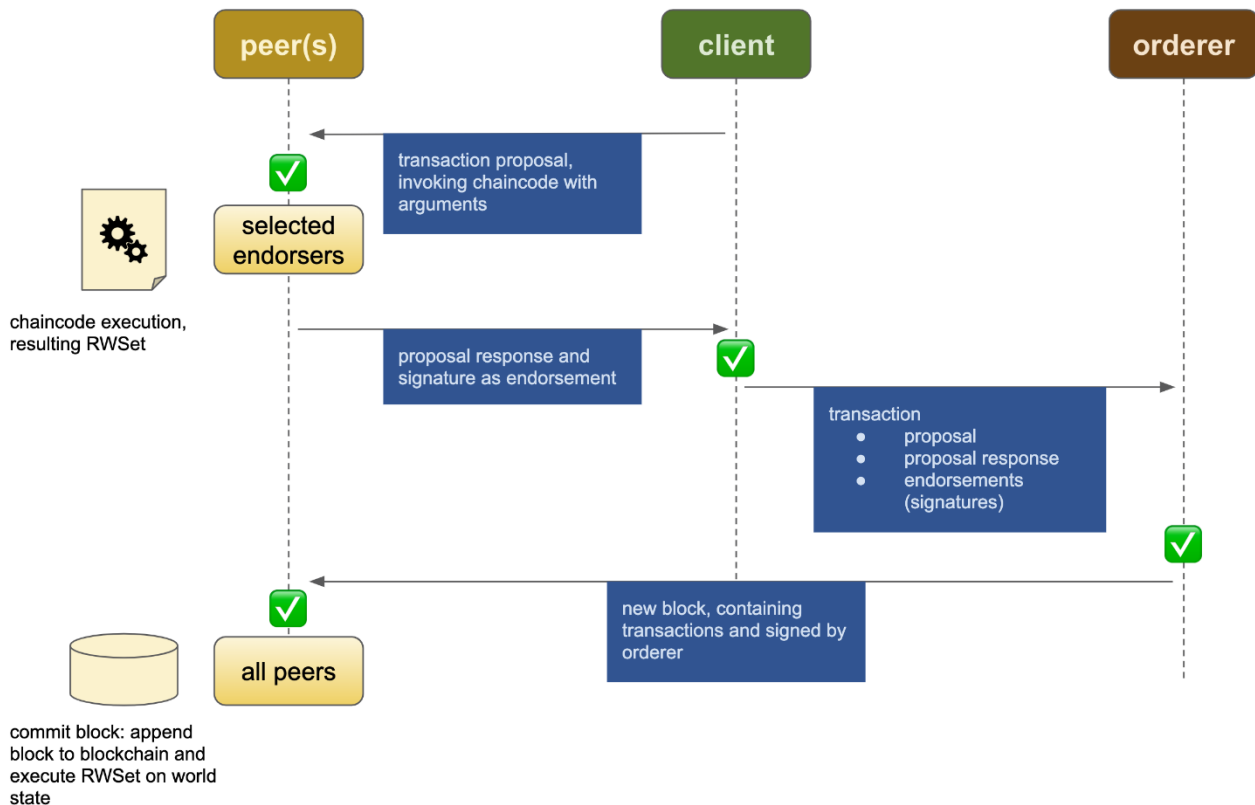


Figure. A.1: A life cycle of transaction process

B. Blockchain Types

The blockchain is classified into three categories:

- A **public blockchain** (also known as permissionless) is open for anyone to read, send or receive transactions and allows any participant to join the consensus procedure of deciding which blocks contain correct transactions and get added to the blockchain [86].
- A **consortium blockchain** placed certain constraints on write permissions such that only a pre-selected set of participants in the network can influence and control the consensus process, even though read is open to any participant in the network [86].
- **Private blockchain**, allows different level of writing permissions for users, so access is restricted strictly to some participant, even though its read permissions are open to the public or constrained to a subset of participants in the network [86].

Table A.1 summarizes these three categories:

Types Scenarios	Describe	Trusted authority number	Speed of consensus
A public blockchain	Anyone can participate, and it is accessible worldwide	0	Slow
A consortium blockchain	Controlled by pre-selected nodes within the consortium	≥ 1	Slight fast
A private blockchain	An organization controls writing rights	1	Fast

Table A.1: Blockchain types

C. Consensus Protocols of the Blockchain

All blockchain types rely on consensus protocols to synchronize replicas across the network. In what follows, we detail the three most known consensus protocols [89] :

- **Proof of Work (PoW):** is a consensus protocol designed for Bitcoin [88], aiming to reach a collective consensus on the bitcoin transaction's validity. This consensus is achieved by a subset of powerful nodes called the miners through solving a heavy mathematical puzzle. The rate of creating a new block to the blockchain is about 10 minutes, which is related to the time spent solving the proof of work challenge and the difficulty of the challenge. In the PoW is challenging and time consuming to predict the correct nonce for the appropriate hash target, while it is extremely easy to validate the hash result, which confirms the resilience to tampering attacks. However, the PoW protocol suffers from a too high computation complexity and a low probability of successful work proof generation.

- **Proof of State:** represents an alternative type of distributed consensus protocol, where only nodes who have locked up their capital as deposits (stake) are qualified to be chosen as miners or validators. In particular, the nodes that hold the highest stake are more likely to be selected for validating transactions and creating new blocks to add to the blockchain. Furthermore, all validators have known identities to allow the network to keep track of all legitimate validators. These identities are stable addresses in Ethereum. PoS breaks the dependency on rewards for security by promoting penalties-based solutions [86].
- **Practical Byzantine Fault Tolerance (PBFT):** is defined as an algorithm for practical BFT proposed to solve the problem of Byzantine generals [90]. PBFT achieves sub-millisecond increases in latency by processing thousands of requests per second. This protocol can work in a hostile environment as it tolerates Byzantine faults up to $1/3$ faults. In fact, it aims to reach a consensus by the collaboration of all honest nodes. The advantage of the PBFT protocol compared to PoW is reducing energy consumption. This protocol is very efficient, especially in private blockchains, as there is no need to perform heavy computations during the validation process, and it is legally distributed compared to the PoS protocol. However, the heavy communication overhead makes the classical PBFT only work with small consensus group sizes which affects the scalability issue.

Abstract :

The Internet of Things (IoT) represents the interconnection between the Internet and physical objects, places and environments. However, this extensive connectivity of IoT can be hampered by malicious interventions from cyber attackers. Thus, ensuring security for users and IoT devices remains a challenge, especially authentication and authorization, which are essential building blocks of the security process. This is due to the unconventional IoT characteristics, including scalability, heterogeneity, interoperability, and dynamic changes, which make the existing security measures inadequate. Indeed, these characteristics bring up several security requirements to consider when defining a secure IoT system. Thus, this dissertation focuses on designing a secure IoT system that achieves the main security concepts, namely the authentication and the authorization for a large-scale IoT environment. This IoT system provides an effective and efficient secure IoT solution that achieves a tradeoff between the security requirements and the network performance. To this end, we proposed a lightweight authentication scheme that meets the need for limited resources and the dynamic changes related to the one-to-one scenarios. This scheme is based on a token of identification to secure access during a prefixed predetermined time interval. We both developed a prototype of this solution for a smart hotel use case, and conducted experiments and simulations to show its effectiveness. Besides, to protect the sharing access between many users and devices, a decentralized access control mechanism was introduced using a decentralized group-key management to meet the scalability, heterogeneity, and dynamic changes issues. Furthermore, to ensure security for an extensive scale of communications based on groups, a distributed group authentication based on blockchain technology is adopted in order to meet the requirement of a large-scale heterogeneous environment. The blockchain provides our secure IoT system with a trustless, immutable, and distributed ledger that records users' information and traceability. Further, it facilitates the design of a distributed group authentication protocol without relying on a third party and eliminates the user re-authenticating process.

Keywords: Internet of things, authentication, authorization, cyber security, dynamic IoT, token, blockchain, decentralized system, distributed system.

Résumé :

L'Internet des objets (IoT) représente l'interconnexion entre l'Internet et des objets, des lieux et des environnements physiques. Cependant, cette connectivité étendue de l'IoT peut être entravée par des interventions malveillantes émanant de cyber attaquants. Ainsi, garantir la sécurité des utilisateurs et des appareils IoT reste un défi, en particulier l'authentification et l'autorisation, qui représentent des briques essentielles du processus de sécurité. Ceci est dû aux caractéristiques non conventionnelles de l'IoT, notamment la scalabilité, l'hétérogénéité, l'interopérabilité et les changements dynamiques, qui rendent les mesures de sécurité existantes inadéquates. En effet, ces caractéristiques font émerger plusieurs exigences de sécurité à prendre en compte lors de la définition d'un système IoT sécurisé. Ainsi, cette thèse se focalise sur la conception d'un système IoT sécurisé réalisant les principaux concepts de sécurité, à savoir l'authentification et l'autorisation pour un environnement IoT à grande échelle. Ce système IoT fournit une solution sécurisée efficace et efficiente qui réalise un compromis entre les exigences de sécurité et les performances réseau. Pour ce faire, nous avons premièrement proposé un schéma d'authentification léger qui répond au besoin de ressources limitées et aux changements dynamiques liés aux scénarios one-to-one. Ce schéma est basé sur un jeton d'identification pour sécuriser l'accès pendant un intervalle de temps prédéterminé. Nous avons à la fois élaboré un prototype de cette solution pour un cas d'utilisation d'hôtel intelligent, et mené des expériences et des simulations afin de monter son efficacité. En outre, pour protéger le partage d'accès entre de nombreux utilisateurs et appareils, un mécanisme de contrôle d'accès décentralisé a été introduit en utilisant une gestion décentralisée de clé de groupe afin de répondre aux problèmes de scalabilité, d'hétérogénéité et de changements dynamiques. De plus, pour garantir la sécurité pour des communications à grande échelle basées sur des groupes, une authentification de groupe distribuée basée sur la technologie blockchain est adoptée dans notre système IoT afin de répondre à l'exigence d'un environnement hétérogène à grande échelle. La blockchain fournit à notre système sécurisée un registre, immuable et distribué qui enregistre les informations et la traçabilité des utilisateurs. Entre autre, la blockchain facilite la conception du protocole d'authentification de groupe distribué sans dépendre d'un tiers et élimine le processus de réauthentification des utilisateurs.

Mots clés : Internet des objets, authentification, autorisation, cyber sécurité, IoT dynamique, jeton, blockchain, système décentralisé, système distribué.