UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Security architecture for Fog-To-Cloud continuum system

Sarang Kahvazadeh

Security architecture for Fog-To-Cloud continuum system

Universitat Politècnica de Catalunya

Departament d' Arquitectura de Computadors

Thesis presented in fulfilment of the requirements for the degree of Doctor for the Universitat Politècnica de Catalunya

Research Group: CRAAX

PhD Student: Sarang Kahvazadeh

Advisor: Xavier Masip-Bruin

Co-Advisor: Eva Marín-Tordera

July, 2019

Acknowledgements

Firstly, I want to thank my partner, Lidia, for being always by my side on moments of happiness and difficulties, always manifesting love and patience. I want also to extend my acknowledgement to all the other members of my family, who were not physically present but have always supported me on my decision of studying abroad, especially my parents, Reza and Mahbobeh, my brother and my sister Behrang and Sara.

In addition, I thank all the friends I met in Vilanova i la geltru, for the relaxing moments and for being like a family to me during the last four years.

I thank my advisor Xavi and co-advisor Eva for the valuable tutorship and high availability for both meeting and reviewing my papers. This is further extended to my colleagues, who I have worked in collaboration since my first year in CRAAX, contributing for the publication of several ideas. I must also thank all CRAAX students. The good working environment has enabled me to go through this process as smoothly as possible.

Barcelona, July 2019

Sarang Kahvazadeh

Abstract

Nowadays, by increasing the number of connected devices to Internet rapidly, cloud computing cannot handle the real-time processing. Therefore, fog computing was emerged for providing data processing, filtering, aggregating, storing, network, and computing closer to the users. Fog computing provides real-time processing with lower latency than cloud. However, fog computing did not come to compete with cloud, it comes to complete the cloud. Therefore, a hierarchical Fog-to-Cloud (F2C) continuum system was introduced. The F2C system brings the collaboration between distributed fogs and centralized cloud. In F2C systems, one of the main challenges is security. Traditional cloud as security provider is not suitable for the F2C system due to be a single-point-of-failure; and even the increasing number of devices at the edge of the network brings scalability issues. Furthermore, traditional cloud security cannot be applied to the fog devices due to their lower computational power than cloud. On the other hand, considering fog nodes as security providers for the edge of the network brings Quality of Service (QoS) issues due to huge fog device's computational power consumption by security algorithms. There are some security solutions for fog computing but they are not considering the hierarchical fog to cloud characteristics that can cause a no-secure collaboration between fog and cloud. In this thesis, the security considerations, attacks, challenges, requirements, and existing solutions are deeply analyzed and reviewed. And finally, a decoupled security architecture is proposed to provide the demanded security in hierarchical and distributed fashion with less impact on the QoS.

Table of Contents

Abstract 2
List of Figures: 6
List of Tables: 8
List of Acronyms: 9
Chapter.1 Introduction 11
1.1 Fog-To-Cloud system 11
1.2 Problem statement 11
1.3 Thesis motivation and objective 12
1.4 Thesis structure 12
Chapter.2 Fog-to-Cloud scenario 14
2.1 Cloud computing 14
2.2 Fog computing 17
2.3 IoT and edge devices 19
2.4 Fog-to-Cloud (F2C) continuum system 20
Chapter.3 Fog-to-Cloud basic security consideration 24
Chapter.4 Fog-to-Cloud attacks 28
4.1 Cloud attacks 28
4.2 Fog attacks 29
4.3 Edge attacks 30
4.4 Most potential attacks in F2C system 32
Chapter.5 Fog-To-Cloud security requirements 35
5.1 Cloud security requirements 35
5.2 Fog security requirements 38
5.3 IoT devices security requirements 41
5.4 F2C combined security requirements 45
Chapter.6 Fog-To-Cloud security challenges and directions 49
Chapter.7 Existing security proposals 56
7.1 Existing cloud layer security proposals 56
7.1.1. Authentication and key management solutions 56

7.1.2. Access control solutions 59
7.1.3. Secure storage and data protection solutions 61
7.1.4. Malicious, intrusion and anomaly detection solutions 62
7.1.5. Cloud security solutions conclusion 63
7.2 Existing fog layer security proposals 64
7.2.1. Authentication and key management solutions 64
7.2.2. Access control solutions 66
7.2.3. Secure storage and data protection solutions 67
7.2.4. Malicious, intrusion and anomaly detection solutions 68
7.2.5. Fog security solutions conclusion 69
7.3 Existing IoT layer security proposals 70
7.3.1. Authentication and key management solutions 70
7.3.2. Access control solutions 75
7.3.3. Malicious, intrusion and anomaly detection solutions 76
7.3.4. IoT security solutions conclusion 78
Chapter.8 F2C Distributed Security Architecture: Proposal 81
8.1 Distributed Security Architecture 81
8.2 Use case: The F2C and Security Architecture benefits in Critical Infrastructure 85
8.3 Authentication in Security Architecture 89
8.3.1 CAUs in mF2C 89
8.3.2 CAUs as distributed authenticators 91
8.4 Key distribution and management in Security Architecture 92
8.5 Access control and distributed data management in Security Architecture 97
8.6 Decoupled proposed security architecture vs embedded 104
8.6.1 ECF 105
8.6.2 DCF 106
Chapter.9 Results/ evaluation 110
9.1 Authentication 110
9.2 Key management 111
9.3 Access control 114
9.4 Decoupled security architecture vs embedded 118
Chapter.10 Conclusion 124

References...126

List of Figures:

Figure 1. F2C continuum system 21
Figure 2. Fog-cloud layer security 24
Figure 3. Device-fog layer security 25
Figure 4. Secure mobility 26
Figure 5. Most potential attacks in F2C system 33
Figure 6. Distributed Security Architecture 81
Figure 7. Distributed security architecture in F2C system 82
Figure 8. Number: Fog-to-Cloud communication 84
Figure 9. Number: Fog-to-Fog communication 84
Figure 10. Security Architecture in CIs 87
Figure 11. Security Architecture in CIs 87
Figure 12. Security Architecture in Smart City 88
Figure 13. Authentication workflow in mf2 90
Figure 14. Authentication workflow in CAUs as authenticator 92
Figure 15. Cloud key management workflow 95
Figure 16. Distributed key management and authentication (DKMA) 96
Figure 17. Cloud key management and authentication 97
Figure 18. Proposed DKMA 97
Figure 19. Proposed Architecture 98
Figure 20. a) Storing edge device' resource information workflow 101
Figure 21. b) Accessing edge device's information in same fog area 101
Figure 22. c) Accessing edge device's resource information from different fog areas 102
Figure 23. Algorithm of securely storing resource information 103
Figure 24. Algorithm of secure retrieve of resource information 103
Figure 25. Embedded security architecture (ECF) 104
Figure 26. Decoupled transversal security architecture (DCF) 104
Figure 27. The ECF workflow 106
Figure 28. The DCF workflow 108
Figure 29. Smart city test-bed 110
Figure 30. Key distribution and authentication delay comparison 112
Figure 31. Network Time Delay 112
Figure 32. Network overhead comparison (Kbytes) 113
Figure 33. Penetration test over authentication and TLS communication 115
Figure 34. Data Storing: Traditional Cloud vs CAU-based Distributed Database 117
Figure 35. Data Retrieving: Traditional Cloud vs CAU-based Distributed Database 117
Figure 36. Security topologies: a) Non secure F2C (nF2C); b) Embedded CAUs F2C (ECF); c) Decoupled CAUs F2C (DCF). 118
Figure 37. Service allocation time delay: (a) Service A time delay; (b) Service B time delay; (c) Service C time delay 120
Figure 38. Service blocking: (a) service A; (b) service B; (c) service C 121
Figure 39. Statistical analysis (Time Delays) 122

Figure 40. Statistical analysis (Service Blocking) 122

List of Tables:

Table 1. Security attacks by architectural layer 31
Table 2. Security requirements in different layers 35
Table 3. Most potential security requirements in F2C 45
Table 4. Security challenges in F2C 50
Table 5. Cloud security solutions 64
Table 6. Fog security solutions 69
Table 7. IoT security solutions 78
Table 8. Security Architecture Advantages 85
Table 9. ECDSA Algorithm sign description 93
Table 10. Authentication time delay 111

List of Acronyms:

F2C	Fog-to-Cloud
IoT	Internet of Things
QoS	Quality of Service
NIST	National Institute of Standards and Technology
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
API	Application Interface
VM	Virtual Machine
DoS	Denial of Service
DDoS	Distributed Denial of Service
OS	Operating System
SQL	Structured Query Language
RFID	Radio Frequency Identification
XML	eXtensible Markup Language
CIA	Confidentiality, Integrity, and Availability
ID	Identification
KGC	Key Generator Center
CA	Certificate Authority
CPU	Central Processing Unit
TLS	Transport Layer Security
RSA	Ron Rivest, Adi Shamir and Leonard Adlema
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Authentication
ECDH	Elliptic Curve Diffie-Hellman
PKI	Public Key Infrastructure
PKG	Private Key Generator
MD5	Message Digest algorithm
USB	Universal Serial Bus
SDI	Security Device Issuer
e-ID	Electronic Identification
IMEI	International Mobile Equipment Identification
CP-ABE	Ciphertext Policy Attribute Based Encryption
ABS	Attributed Based Signature
HABE	Hierarchical Attribute Based Encryption
KP-ABE	Key Policy attribute Based Encryption
EACF	Extensible Access Control Framework
SD	Software Defined
SDN	Software Defined Networking
SEC	Security
DTLS	Datagram Transport Layer Security
SHA	Secure Hash Algorithm
ID- AVL	Identification Attribute-Value Pair

HTTPS	Hyper Text Transfer Protocol Secure
NED	Network Edge Device
CoAP	Constrained Application Protocol
HIP	Host Identity Protocol
TTP	Trusted Third Party
MAC	Message Authentication Code
ACK	Acknowledgment
CH	Cluster-Head
RA	Registration Authority
DNS	Domain Name System
CAU	Control-Area-Unit
CI	Critical Infrastructure
CSR	Certificate Signature Request
DKMA	Distributed Key Management and Authentication
SSL	Secure Sockets Layer
AES	Advanced Encryption Standard
ECF	Embedded CAUs in Fog nodes
DCF	Decoupled CAUs from Fog nodes
RP	Raspberry Pi
MS	millisecond
S	Second
V4	Version 4
GB	Giga Byte
MB	Mega Byte
KB	Kilo Byte
TB	Tera Byte
RAM	Random Access Memory
TCP	Transmission Control Protocol
PC	Personal Computer
SD	Secure Digital
GHz	Gigahertz
nf2c	Non-Secure Fog-to-Cloud

Chapter.1 Introduction

1.1 Fog-To-Cloud system

By growing devices connected to the internet rapidly such as mobiles, tablets, sensors, actuators, etc, cloud computing [1], [2] was emerged to provide huge computational power, network and storage for processing, filtering, aggregating and storage to the huge amount of produced data by devices. However, cloud computing is located far from distributed devices, and then it cannot provide low latency and real-time processing for all distributed devices.

Therefore, fog computing [3] is merged into the system for bringing cloud characteristics closers to the users and devices. Fog computing provides distributed computing, network, and storage closer to the users in a virtualized or non-virtualized environment. Fog computing facilities data filtering, processing, aggregating, and storage closer to the users. Fog devices physically can be considered as set-top-boxes, access points, routers, switches, base stations, smart phones, tablets, and etc. It is worth to mention that fog computing was not introduced to compete with cloud, fog computing is completing cloud. Therefore, a hierarchical Fog-To-Cloud (F2C) [4] computing system was introduced to bring fog and cloud into the one framework. In this hierarchical architecture, distributed fog devices can be clustered into the different layers for providing computing, network and storage. In the F2C architecture, fog devices provide initial data processing, filtering, and storage for devices and then aggregated data can be sent to the cloud for more processing and for storing.

The F2C system has hierarchical characteristics such as device might send services to the nearby fog device, if corresponding fog device has enough capacity then it provides service execution, otherwise services might be allocated in upper layers (fog devices or cloud) for service execution. The F2C system has many challenges ongoing such as fog devices discovery, resource categorization, resource allocation, service execution, security and etc. In this thesis, the security in F2C is analyzed deeply and it is proposed a security architecture for handling security in the F2C system with a hierarchical and distributed nature.

1.2 Problem statement

The traditional cloud as a centralized and distanced component provides security for the hierarchical F2C system, but it can bring issues such as be a single-point-of-failure, scalability, and quality of services issues such as time delay, etc. On the other hand, existing cloud security solutions cannot be applied into the fog devices due to their lower computational power than cloud. Even, the existing fog security solutions without considering the whole hierarchical F2C system can bring challenges and issues such as not secure coordination between the layers.

In the bottom layer of F2C, there are distributed low computational power devices that are called internet of things (IoT) [5]. IoT devices are not capable of handling their security, they rely on

other components in upper layers for security provisioning. Some existing solutions are using cloud as security provider for IoT devices, although, scalability issues might arise due to the growing number of devices and even, because cloud as centralized entity can be considered as a single-point-of-failure. On the other hand, other existing solutions are using fog devices as IoT security providers. In this case and although security provisioning uses fog device's computational power, it might cause quality of service (QoS) degradation. Therefore, designing a security architecture for handling distributed devices in a hierarchical F2C system is a though challenge.

1.3 Thesis motivation and objective

In this thesis, the theoretical objectives are:

1- Security requirements and challenges are analyzed deeply in all layers of F2C.
2- According to the requirements, security considerations for F2C are illustrated.
3- Security attacks in all F2C layers are described and analyzed.
4- Existing security solutions for all layers in the F2C continuum are reviewed and analyzed.

The Technical Objectives are:

1- A novel security architecture is designed to be adopted into the F2C system.
2- Authentication is implemented in security architecture in distributed fashion (distributed authenticators) to provide authentication with less impact on QoS.
3- Key management is implemented in security architecture in distributed fashion (distributed key managers) to handle F2C system key management securely with less impact on QoS.
4- Access control and secure distributed data storage are implemented in security architecture for providing distributed secure data storage closer to the users.
5- Finally, the security architecture is decoupled and puts transversal to the F2C system for providing security functionalities with less impact on QoS

1.4 Thesis structure

In this section, the structure for the rest of the thesis is presented in details.

Chapter 2: Presents all layers in the combined F2C.

- Section 2.1 describes cloud layer in F2C scenario and the cloud features and weaknesses points.
- Section 2.2 presents fog layer in F2C and fog features and disadvantages.
- Section 2.3 describes IoT layer in F2C system.
- Section 2.4 presents the whole combined layers in F2C system.

Chapter 3: Presents the most basic and potential security consideration for F2C system.

Chapter 4: Analyses most possible attacks in all different layers of F2C.

- Section 4.1 illustrates the most possible attacks in cloud layer of F2C.

- Section 4.2 presents the most possible attacks in fog layers of F2C.
- Section 4.3 presents the most possible attacks in IoT layer of F2C.
- Section 4.4 illustrates the most potential attacks in combined all layers of F2C.

Chapter 5: Illustrates most potential security requirements in all layers of F2C.

- Section 5.1 describes most potential security requirements in cloud layer of F2C.
- Section 5.2 presents most potential security requirements in fog layers of F2C.
- Section 5.3 shows most potential security requirements in IoT layer of F2C.
- Section 5.4 describes most potential security requirements in combined all layers of F2C.

Chapter 6: Describes security challenges and directions for the F2C system.

Chapter 7: Reviews and analyzes possible security solutions in all different layers of F2C.

- Section 7.1 reviews and analyses solutions in cloud layer of F2C.
- Section 7.2 reviews and analyses solutions in fog layers of F2C.
- Section 7.3 reviews and analyses solutions in IoT layer of F2C.

Chapter 8: Proposes a security architecture for handling authentication, key management, and access control in hierarchical F2C system.

- Section 8.1 describes the security architecture proposal.
- Section 8.2 illustrates the proposed security architecture benefits in critical infrastructures.
- Section 8.3 presents the authentication process in the security architecture.
- Section 8.4 describes key management in the security architecture.
- Section 8.5 presents access control and secure distributed storage in the security architecture.
- Section 8.6 provides analysis between decoupled security architecture from F2C components versus embedded security architecture in fog devices.

Chapter 9. Summarizes the proposed security architecture and then concludes.

Chapter.2 Fog-to-Cloud scenario

The fog-to-cloud (F2C) continuum system has different layers such as cloud, fog and edge (IoT devices). For describing the F2C system in a sophisticated way, all the layers will be discussed and analyzed in details and finally putting all layers together to define the hierarchical F2C system.

2.1 Cloud computing

The cloud computing [1], [2], [6], [7] concept was established first by the national institute of standards and technology (NIST) [8]: Cloud computing is a model to make able ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources such as networks, storage, applications and services that can be provided and released with minimal management effort or service provider interaction. Cloud computing facilitates service provisioning to the external users by using internet technologies. The cloud computing can be considered as a collection of interconnected and virtualized computers that unified as a one computing resources based on service level agreement between services providers and consumers. Cloud computing provides access to virtualized resources such as computers, networks and storage. The cloud computing is conceptually centralized and able to handle huge volume of data processing, filtering, aggregating, storage, and network facilities for huge amount of consumer in the scalable way. In a nutshell, cloud computing provides broadband network, data center, virtualization, web technology, multitenancy, and facilitates service execution and delivery.

The cloud computing provides three type of services such as:

- Infrastructure as a Service (IaaS): It is the base of the cloud which provides processing, storage, network resources, and other computational functionalities to the consumer. All the IaaS's users can deploy arbitrary applications, any type of software, operating systems, and services that able to scale up and down dynamically and rapidly. The users and costumers have the control over operating systems, storage, deployed applications, and some part of system administration parts. IaaS is responsible for providing cloud' virtualized environment for users and customers.
- Platform as a Service (PaaS): It provides customers and users an environment and solution to develop, test, and deploy their applications. In this case, users have no control and power over the network, servers, operating system and storage provided by cloud infrastructure. Although, the users have control over their deployed applications. In a nutshell, PaaS facilitates platform management and maintenance for the users that wish to develop, deploy, and test their applications.
- Software as a Service (SaaS): In the SaaS model, users are able to use applications running on a cloud infrastructure. The applications providers make them accessible to the users and customers through an interface. For example, the applications are available through a web browser for providing cloud services. The users and costumers have no control over

infrastructure such as network, system, servers, operating systems, storage, and even no control over the platform. This SaaS makes users and customers free from the need of installing software locally and consuming their resources.

The cloud computing can be accessible and deployable through four types of model such as public, private, community and hybrid. These type of cloud accesses can be deployed into the organizations according to their policies and management. In the following, the four type are described in details:

- Public cloud: In this type, all the resources such as computing, storage, network, virtualization, and applications are provided by third party service providers and all resources are accessible for public users over the internet. It is less expensive rather than others types for users, but it is less secure.
- Private cloud: In this type, cloud environment and infrastructure are operated merely for an organization. Private one can only be used by one organization, one company or one of its customers. In this type of cloud, data resources and applications are only accessible to specific and permitted clients. Compare to public one, private cloud is more expensive but more secure.
- Community cloud: The organizations with the same or similar requirements such as security policies and compliance consideration share this type of cloud (Community). Community cloud facilitates organizations with the same or similar requirements to share computing resources, data storage and other capacities for integrating their companies. The community cloud can be managed by organizations member or a third party. This type is less expensive for integrating companies with similar requirements rather that each one of them have private cloud.
- Hybrid cloud: It is the combination and integration of two or more above of the models (public, private, and community). All the cloud environment and infrastructure are managed and handled by mix of external and internal cloud members. Sensitive data are stored internally and takes pack up externally in the public cloud in case of system failures. In the case of insensitive data, all are public.

The most potential cloud computing characteristics, features and advantages are [1], [9], [2], [6], [7] :

a. On-demand self-services: In cloud computing, services and computing resources can be accessible and provided automatically to the users without human-service provider interaction.
b. Scalability and elasticity: Cloud resources can be scaled up and down flexible due to business dynamic, organization's and user's needs. Users can access current and historical data anytime easily and fast. Organization and users can easily scale up and down the cloud resources according to their policies and needs.
c. Resource pooling: cloud service providers and administration provide and supply their pool of resources for users and clients. For example, a cloud server as a standalone server might host many users and clients in its virtual environment provided by cloud administrative.
d. Fast deployment: All the cloud' users and client can get application running quickly due to manageable and maintenance environment

e. Accessing resources easily: The cloud services can be accessible anywhere and anytime by multiple users. Users even can easily access and make configuration on the services host by cloud.
f. Huge amount of storage: The cloud can provide almost unlimited data processing, filtering and storage with low cost. Cloud is pay-as-you-go online computing system that means user pay only for the amount of storage that they use.
g. Cost: Cloud service provider or third party can provide computing infrastructure for organization instead of they purchase all their computing infrastructures.
h. Data recovery: All the historical data from organizations or even simple user will be stored in the cloud huge data center, therefore, it can prevent any disaster in recovery.
i. Application updates: In cloud infrastructure, any software provided by the cloud will be updated to the last version automatically.
j. Facilitating collaboration and integration: In the organization, any staff according to their restriction can access, edit and share data among others. Even, by using community cloud, some organizations can integrated by their interest and policies.
k. Multitenancy: In the cloud, a centralized data center can host multiple services by multiple providers. This characteristic facilitates management and interactions between different service providers.

Cloud computing provides and facilitates computing capabilities for the organizations and users, although, this computing system has disadvantages as will mention in the below [1], [9], [2], [6], [7] :

- Centralized computing: Cloud computing conceptually is centralized. This might cause single point of failure. If the cloud being down, failed, or it is attacked; it might not work properly for the organizations. Or even if the cloud as a conceptually centralized gets an attack, it might cause a fundamental disaster because all data stored in the cloud.
- Distance: The cloud computing conceptually is far from users. This distance might cause some security attacks such as man-in-the-middle and etc. even in some case might effect on the Quality of service.
- Security: The cloud nature makes data stored accessible on the internet anywhere. This might cause that unauthorized users to get access to this sensitive data. Even though cloud has huge computational power to provide security over the data and the network, however the conceptually centralized (single point of failure) and the distance makes cloud security on the risks.
- Interoperability and portability: The lack of appropriate application interface (APIs) between clouds with different service provides, makes the data movement and applications between clouds difficult and in some case impossible. Nowadays, there are not any interoperability and adequate APIs between clouds with different service providers.
- Connectivity issues: Sometimes, cloud ‘services cannot be done due to low internet connection or due to which is called downtime. Cloud services are based on internet and sometimes connectivity might be low for getting services to be executed.
- Availability and Reliability: Cloud services must be available and reliable all the times and always. Most of the time, this feature is in a danger due to failures in connectivity, possible

attacks, conceptually distance cloud data center. In the case of any failure, cloud service provider must take adequate actions.

- Inefficient use of network bandwidth: The cloud computing is capable of doing huge volume of computation and processing for big data, although, sending all data from edge of the network to the cloud causes network traffic increasing significantly. For sending all generated data at the edge of the network to the cloud, a huge volume of network bandwidth is needed.
- Latency: The conceptually distanced cloud makes impossible real-time processing for some application such as gaming, smart homes and traffics with higher latency.

Most of the disadvantages in cloud computing can be overcome by utilizing fog computing closer to the users. In the next sub-section, fog computing will be analyzed and discussed in details.

2.2 Fog computing

The fog computing concept was introduced by cisco [3] in 2012, fog computing concept is a decentralized computing system at the edge of the network (closer to the users). The fog computing provides distributed computing, network, and storage closer to the users in a virtualized or non-virtualized environment. The fog layer can be considered as a middleware between the edge devices, users and cloud. Fog can be considered as an extension of cloud at the edge of the network that facilitates computing, network and storage in a distributed fashion and closer to the users. Fog computing facilities data filtering, processing, aggregating, and storage closer to the users. Fog devices physically can be considered as set-top-boxes, access points, routers, switches, base stations, smart phones, tablets, end devices, and etc.

Most of the mentioned cloud computing challenges were mentioned in the previous section can be overcome by fog computing features and advantages. Fog computing advantages, characteristics and features are as below [10], [11], [12], [13], [14]:

- Geographically distributed (Geo-distributed): Fog devices which provide distributed computing can be routers, cell towers, road-side units, base stations, and etc. These distributed devices can provide and facilitate geo-distributed communication, network, storage, and computing.
- Mobility: Geo-distributed nature of fog computing facilitates mobility. Some edge devices such as phones, tablets, cars with on-board units, etc. are on the move. Fog devices have inter communication, therefore, they can handle and facilitate data handover for devices on the move.
- Reduce load on the cloud: In the traditional cloud concept, all the produced data by devices must be transferred to cloud servers for aggregating, filtering, processing, and storage. Fog computing can be considered as a middleware between end devices and cloud. Fog facilitates data aggregation, filtering, processing, and storage closer to the users and then the processed, aggregated, and filtered data can be sent to the cloud for more investigation and storage. Therefore, the cloud over the cloud servers can be minimized by fog devices.

- Data computation offloading: In the traditional cloud, for uploading and sending data such as video-stream, huge bandwidth is required. The fog computing can facilitate data computation by compressing the data before sending it to the cloud.
- Low-latency: The fog computing makes service execution closer to the users at the edge of the network, therefore, compare to cloud (conceptually distanced from users) provides services with lower latency and time delay.
- Real-time processing: The fog locations at the edge of the network where is closer to the users and its distributed nature, makes possible the real-time decision making and processing. Therefore, all the users-fogs interactions occur in real-time.
- Heterogeneity: Fog computing is considered as a middleware between users and cloud in a distributed nature. All fog nodes must have inter-communication to provide resource orchestration, exchange of information, perform load balancing and finally provide heterogeneity into the system.
- Location-awareness: Fog computing is located at the edge of the network in a distributed nature, therefore, it supports mobility and can track users location in real-time.
- Scalability: Fog computing can provide distributed computing, network, and storage for the large-scale edge devices and bring scalability into the system.
- Interoperability: Fog computing is capable of working under different fog service providers. Therefore, all fog service providers have inter-communication to bring interoperability into the system.
- Cost and lower expenses: The operation done by fog computing is lower in terms of cost and expenses compare to the cloud because the process in fog is done at the edge of the network and it saves the needed network bandwidth.
- Data security and privacy: Fog computing provides computation, network, and storage closer to the users, therefore, users can have more control than cloud over their own data in terms of security and privacy.

Although, fog computing has some unsolved challenges and disadvantages such as [15], [16], [17], [18], [19]:

- Security issues: The fog computing characteristics such as its distributed nature and the fact of being closer to the users makes it more vulnerable than cloud. Even, the traditional existing security mechanism for cloud cannot be applied to the fog system due to its distributed nature and because it has lower-computational power than cloud.
- Control and management over resources: The fog nodes must provide the required resources for computation, network, storage, latency and bandwidth at the edge of the network and also ensuring the expected QoS. In some case, the inherent mobility associated to fog causes that those metrics dynamically change and even the fog environment most of the time is virtualized which causes additional latency therefore, controlling and managing resources at the dog computing is one of the main challenges and issues.
- Energy management: The distributed characteristics of fog makes energy consumption higher than centralized cloud, therefore, one of the issue is optimizing energy at the edge of the network.

- Virtualization choose: One of the main characteristics in fog nodes is the use of a virtualized environment, then choosing the best virtualized environment (hypervisor or container) is one the challenges for designing the desirable fog computing system.
- Providing low-latency: Low-latency is one the fundamental advantages of fog computing. Although, the desirable low-latency for users in some case such mobility, providing desirable resources for tasks execution, handling distributed data aggregation and processing are challengeable.
- Lower-resource fog devices: All the user devices and network management devices with computational power can be considered as fog devices. Although, these devices have lower computational power than cloud, therefore, efficient job allocation and chosen desirable policies are needed.
- Power limitation: All the devices such as smart phones, tablets, and laptops can act as fog devices, although, these devices are power-limited. Tasks which are running in these devices may not be completed due to devices can be power off because of power-limitation.
- Connectivity: Most of the fog devices at the edge of the network uses wireless technology for connectivity. Although, edge devices such as Internet of things (IoT) devices, sensors, actuators, and etc. might lose their connectivity to fog devices by increasing number of connected devices to the fog. Therefore, an estimation for fog devices connectivity capabilities must be done.
- Scalability: the huge number of edge devices generates huge amount of data at the edge of the network. The produced data needs to be processed, filtered and aggregated then it requires huge amount of resources (processing capabilities and storage). Therefore, fog devices at the edge of the network must be selected carefully for handling huge amount of generated data.
- Distributed architecture: The distributed fog nature makes it vulnerable in case of redundancy. Therefore, the framework must be designed for reducing the redundancy.
- Device's heterogeneity: At the edge of the network, several devices are naturally heterogeneous. Therefore, the fog computing environment must consider device's heterogeneity while developing fog applications.

As mentioned above, there are some of the issues and challenges in fog computing which can be overcome by utilizing the hierarchical F2C system. In the next section, the IoT and edge devices layer will be described. Then, finally putting all layers together, the F2C continuum system will be described and analyzed.

2.3 IoT and edge devices

The Internet of things (IoT) [5] , [20] concept is referring to make devices such as sensors, actuators, and etc. connected without human interaction. The IoT makes all devices and humans connected to each other. One of the main motivations of the IoT is to make interoperable communication and connections between devices at the edge of the network such as smart devices, sensors, PCs, cars, and etc. However, most of the IoT devices and edge devices have not enough

computational power, battery power, memory network, and storage. Therefore, fog computing (fog devices with more computational power at the edge of the network) and cloud (conceptually distanced centralized huge datacenters) can provide data processing, filtering, aggregation and storage for IoT and edge devices. There are many challenges in IoT such as real-time communication and processing, security and privacy, low-computational devices, the distributed nature of IoT, etc. which can be overcome by using collaborated Fog-to-Cloud (F2C) continuum system.

In the next section, the hierarchical F2C system considering all mentioned layers above such as cloud, fog and IoT will be discussed, analyzed and illustrated.

2.4 Fog-to-Cloud (F2C) continuum system

Putting all the layers mentioned above together and according to the fact that fog computing comes to complete cloud computing rather than compete, the Fog-to-Cloud (F2C) Computing has been proposed in [4] intended to enable a coordinated management of resources available at both fog and cloud. By means of a hierarchical layered-architecture, network controllers are deployed in a distributed fashion enabling a multi-layer resource allocation as well as the distributed and parallel service execution in fog resources, cloud, or both. Therefore, service demands are mapped into fog or cloud resources according to their suitability and availability to meet the expected QoS requirements. The integrated and combined F2C system can provide higher performance, higher energy efficiency, real-time processing, faster responding, scalability and localization for IoT and edge devices due to their distributed, hierarchical and combined characteristics.

The F2C is a hierarchical multi-layered architecture conceived to cover a broad area with plenty of computing devices. The hierarchical distributed nature of this architecture allows combining the advantages of both computing paradigms, i.e., proximity at the edge of the network by fog and high performance at the cloud, while the coordinated management of the whole system allows the feasibility of providing optimal resource allocation that meets the expected services QoS requirements. The F2C ecosystem is shown in Figure 1. The whole area of coverage is organized in fog areas, which include the set of resources (nodes) located inside that area. The exact scope of an area is a topic of current research, and affects the scalability of the system. One fog node at each area is selected to become the manager of the area for handling other fog devices and edge devices (IoT devices). The fog node as the manager is a node with certain features, such as enough computing and networking capabilities to manage its area, and good network access. The responsibilities of such fog node are managing the devices inside the area as well as coordinating with higher level layers. In this figure, the fog nodes are connected, and managed, by the Cloud layer, thus crafting the hierarchical architecture. Obviously, the Cloud layer has enough capacity to perform a higher level management of the fog nodes set.

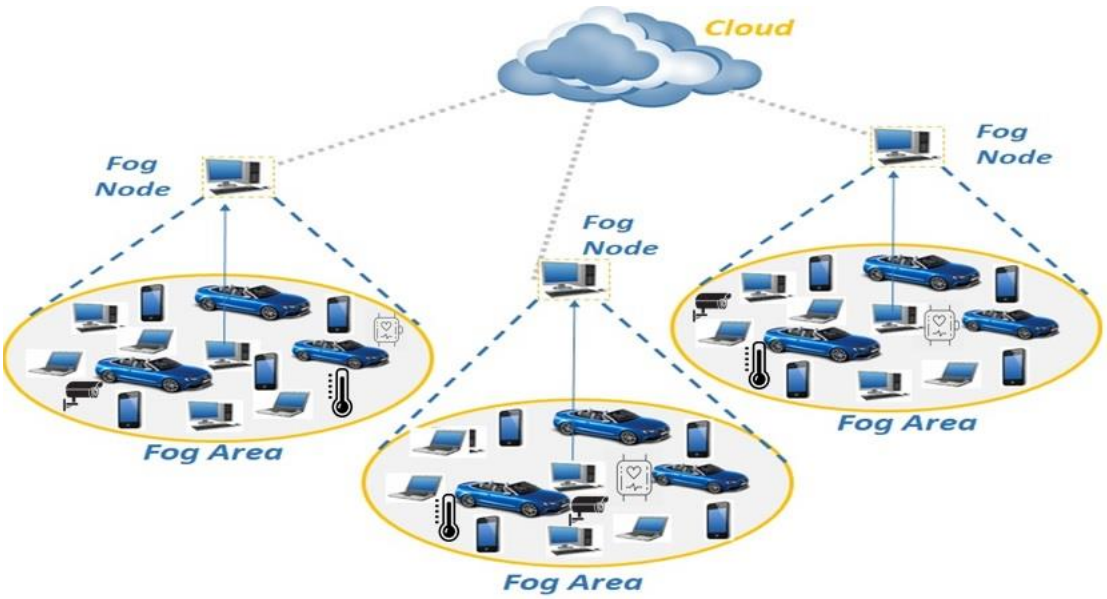

Figure 1. F2C continuum system

Additionally, in a large scenario with millions of IoT and edge devices and spanning several squared kilometers, such architecture could increase the number of layers in order to facilitate an efficient coordination between nearby areas and, thus, becoming a multi-layered architecture. The multi-layered hierarchy guarantees the scalability of the system, as well as an efficient service management.

In this scenario, and the set of resources (nodes) organized in layers and areas, users share their resources to the F2C system, but also become F2C clients requesting the execution of services or applications. To take advantage of the execution of services in this combination of the different computing paradigms, fog, edge and high performance at cloud, it is necessary a system controlling and managing the execution of services. The outlined characteristics in the execution of a service is mentioned below:

- Launching the service: The service can be requested to the system in any node belonging to it.
- Hierarchical search of resources: If the service is requested to a specific node:
  When the node is a normal node, if it has enough resources to execute the service, it will be executed in this node; otherwise the request will be forwarded to its fog node as the manager (higher layer).
  If the node is a fog node as the manager, it will also check if it has enough resources, but in this cases considering the resources of all the nodes belonging to the area it is controlling. Again, if in the area there are enough resources the service will be executed in the nodes of the area; otherwise the service will be forwarded to the higher layer, in the case of Figure.1 to the cloud layer, but with more hierarchical layers to the corresponding upper layer.

- Mapping of services and resources: The previous description about the hierarchical search of resources will be based on the smartness to map services into fog or cloud resources according to their capabilities, availability, expected QoS requirements, etc.
- Distributed and parallel execution: The F2C system must allow the distributed execution of services. Services can be monolithic applications or services divided into subservices or tasks. When a service allows its division into tasks, the F2C system must perform the best division into tasks and also assign the tasks to the more suitable resources. Moreover, this distributed execution may be also parallel in some services. Taking advantage again of the large number of nodes, different tasks of a service can be executed in different nodes. The F2C will have a runtime controller controlling the synchronized execution of tasks.

Other main aspects of the F2C easing the distributed execution of services in this ecosystem are:

- Resource discovery: Nodes can be on the move in the city, such as mobile phones. It must exist a mechanism to mutual discover between fog node managers and normal nodes.
- Identification: Nodes participating in the system must be uniquely identified.
- Sharing model: Users sharing their devices in the system should indicate the amount of resources they want to participate (memory, storage, etc.)
- Handover: As it is mentioned, nodes can be on the move, belonging to an area, and disappearing of this area after a time. There should be a handover mechanism to reallocate tasks being executed in this on move devices.

The F2C system is bringing management coordination between fog and cloud. However, there are many challenges and issues to provide F2C system. One of the main challenges in F2C scenario is bringing security into the system. The F2C system has a hierarchical and distributed nature with the combination of the cloud huge data centers, fog devices with enough computational power and low-computational power, and finally edge and IoT devices with limited computational power. The mentioned F2C system characteristics make it so vulnerable to the attacks. There are many existing security solutions for cloud which cannot be applied in fog and IoT due to the lower computational power; and even existing fog security solutions without considering the whole combined F2C system cannot cover the security requirements of the F2C system. The motivation here is to analyze the security considerations, attacks, requirements and challenges in the existing solutions, and finally to propose a novel distributed and decoupled security architecture for F2C systems for handling authentication, key management, access control, and encryption/decryption with the desired QoS. In the next chapter, the most potential security considerations for F2C will be discussed and analyzed.

Chapter.3 Fog-to-Cloud basic security consideration

In this chapter, we analyze the basic security view in the hierarchical F2C scenario to illustrate the basic security considerations that must be applied into the F2C continuum system.

In the F2C system, there are distributed fog nodes as distributed managers to provide computation, network, and storage for other fog devices, edge and IoT devices closer to the users for their corresponding areas. The distributed fog nodes and cloud in the combined F2C system must communicate securely to avoid any passive and active attacks such as man-in-the-middle, masquerade, etc. Fog nodes need to connect to the cloud (not the fake or malicious), and in parallel, F2C cloud must connect to the trustable fog nodes (not the fake or malicious). The fog nodes are distributed in a distributed way to provide management to the nodes in its area, moreover this can be exploited to bring security in a distributed way to the F2C system. To bring security in all the fog and cloud layers some steps are needed such as it is illustrated in Figure 2.

Each fog node must be securely discovered and then perform mutual authentication with cloud by providing credentials and identities to provide system and data integrity and confidentiality. After performing authentication, cloud provides keys for fog nodes. Fog nodes can use these keys to encrypt and decrypt exchange information with cloud, for preventing attackers to eavesdropping, modifying, or deleting exchange information between cloud and fogs. The network technologies between cloud and fog nodes such as wired, wireless, etc. must be secured to avoid any passive and active attacks. It means cloud and fog nodes must exchange information in secure channels (all blue lines in the Figure 2). And the final step is that cloud acts as access control and provides access to F2C cloud and data centers for the distributed fog nodes to processing, aggregating, filtering, and storing information according to their attributes and preventing any unauthorized access.

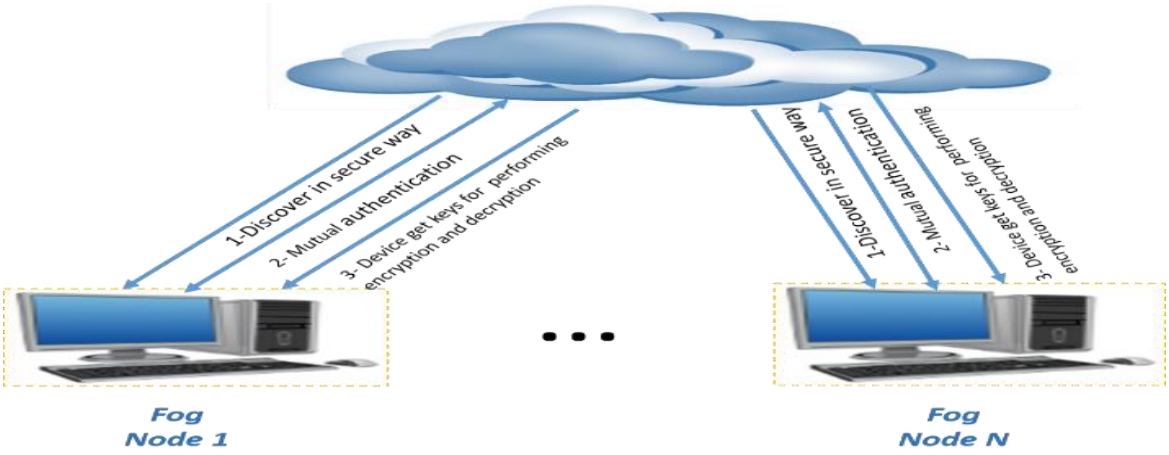

Figure 2. Fog-cloud layer security

In parallel, after fog nodes are properly installed and authenticate to the F2C cloud, they get authorization to provide computation, network, storage, and shareable computational environment to IoT devices at the edge of the network in a distributed way. In this case (Figure 3), when a device arrives to the fog area, first must be discovered by fog nodes securely. Then, fog nodes with devices must be mutually authenticated by providing credential and identities to bring data and system integrity and confidentiality at the edge of the network. After authentication, fog nodes can generate and distribute keys to devices to be used for encryption and decryption. All exchange information between devices (such as IoT devices) and fog nodes must be encrypted to prevent attackers from eavesdropping, modify, or delete the information. All information exchange must occur in secure channels for different network technologies (blue lines in Figure 3). At the end, fog nodes can act as access control for devices to prevent any unauthorized access to the F2C system at the edge of the network.

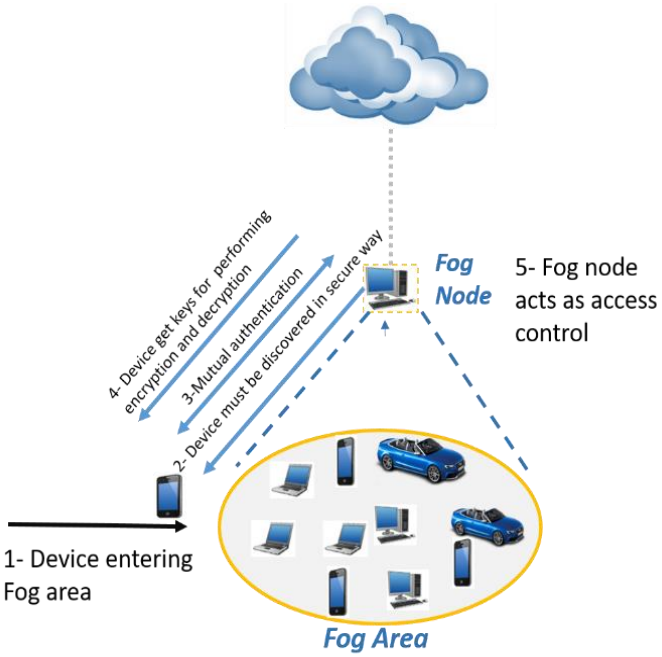

Figure 3. Device-fog layer security

At the edge (in a fog area) there are different type of devices such as fog devices, edge devices and IoT devices, managed by fog nodes as managers. These devices might be on the move. Therefore, secure inter-communication between fog nodes is a must to provide secure handover. For example in Figure 4, a device (car) that was connected to the fog area 1 to a fog node, it leaves and heads to the fog area 2. When the car arrives to the fog area 2, it must be authenticated and start executing services securely. This is possible thanks to the secure handover provided by secure fog node inter-communication to prevent any unauthorized or attacker eavesdrop and modified exchange information.

All the involved components and their communications in the F2C system can get attacked. Therefore, the next chapter deeply analysis all possible attacks in different F2C layers.

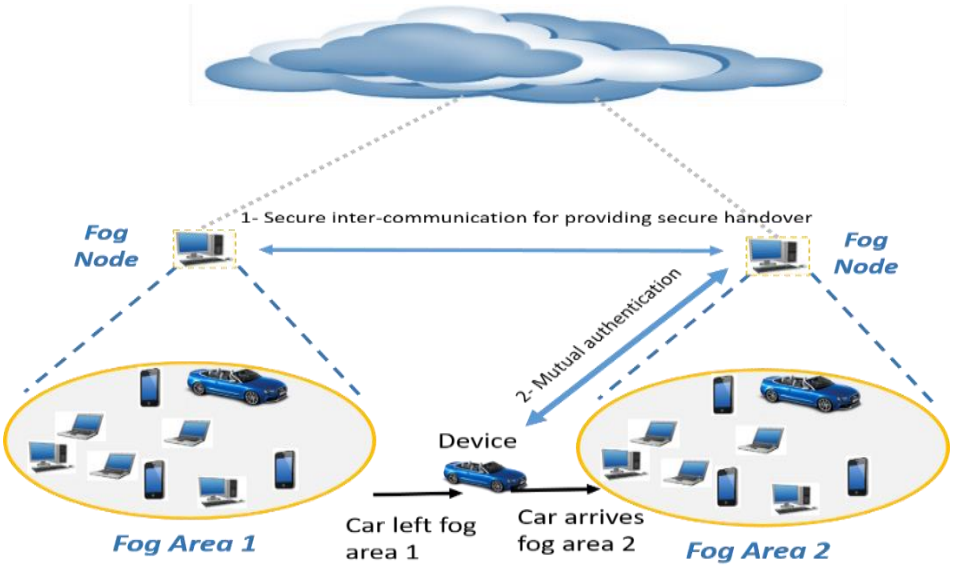

Figure 4. Secure mobility

Chapter.4 Fog-to-Cloud attacks

In the last chapter, the basic security consideration was described for F2C system. This chapter provides all possible attacks on each layer in F2C and finally illustrates the most potential and important attacks to overcome in a hierarchical F2C continuum system (Table 1**).**

4.1 Cloud attacks

The attacks on the cloud layer in F2C system are described in the following articles, [9], [21], [22], [23], [24], [25], [26], [27]and the type of attacks are:

Backdoor channel attacks: attackers take remote access to the compromised system. Attackers take control over victim's resource by using backdoor channel, then the attacker can use victim's resource to launch a zombie attack, even they can disclose private victim's information.

Malware injection: Hackers can inject malware application, services, or virtual machines into the cloud system or datacentres to interrupt the whole system.

Virtualization attack: There are two types of virtualization attacks including VM escape and rootkit in hypervisor.

In **VM escape**, the attacker runs a program in a VM and breaks the isolation layer in order to run with hypervisor's root privileges instead with the VM privileges, which allows the attacker to interact with the hypervisor. Therefore, attacker gets access to the host OS and other VMs running on the physical machine.

Rootkit in hypervisor: VM-based rootkit initiates a hypervisor including the existing host OS to a VM. In reality host OS does not exist; however, the new initiated guest OS assumes that it is running as the host OS with the corresponding control over resources. Hypervisor produces a channel to execute unauthorized code into the system which attacker get control over running VM on the host machine and activities manipulation on the system.

Denial of service (DoS): Attackers can affect the availability of the cloud and prevent legitimate users to access to cloud by jamming or flooding requests to the server.

Man in the middle attack: If a secure channel between cloud and users is broken, attackers are able to access data exchange.

Metadata spoofing: An attacker can modify web service's description languages where descriptions of services are stored.

Malicious insider: Person who is an employee in the cloud organization can use their privileges to disclose private information.

Phishing attack: Attackers can manipulate the web link and redirect users to a fake one to get user's private information. An attacker may use cloud services to host a phishing attack site to hijack accounts and services of other users in cloud.

SQL injection: Attackers can inject malicious data into the SQL and get the private information or interrupt the whole SQL.

Sniffer attack: The attacker tries to read the content of a network packet, or to derive partial information (e.g. number of letters in a password).

Zombie attack (DoS/DDoS): Through the Internet, an attacker tries to flood the victim by sending requests from innocent hosts (normal host not the fake one) in the network. There are 2 types of Zombie attacks; the first is when an attacker floods a large number of requests via a zombie (innocent host) to affect availability of cloud services. The second case is when a huge number of requests overloads cloud to be exhausted which can cause DoS and DDoS attack. DDoS is a type of DoS attack where multiple compromised systems are used to target a single system causing a DoS attack.

Spoofing attack: This occurs when an attacker impersonates to be a legitimate cloud user with the intention of stealing sensitive information or launching the attack to the whole cloud system.

The cloud characteristics are high computation, storage, and network, therefore we have to be able to provide also high security. There are many cloud security solutions in the market that can be applied, however there are so many challenges still unsolved that will be discussed in chapter 6.

4.2 Fog attacks

The fog computing inherits cloud characteristics to provide computing, network and storage closer to the users. Therefore, there are some mutual attacks in both layers. Some important attacks in fog are [28], [29], [30], [31], [32], [33]:

Man in the middle: If a secure channel between fog nodes, users, and servers is broken an attacker will be able to access data exchange.

Virtualization attack: Fog inherits the virtualization attacks from cloud due to their similar characteristics. This attack is already described in the cloud section in details.

DoS/DDoS attack: Attackers can affect the availability of fogs and prevent legitimate users from accessing to fog servers. These attacks are described at the cloud part.

Malware injection: Hackers can inject malware data, services, or virtual machines into the fog system to interrupt the whole system.

Gateway attack: Gateways are acting as bridge between fog and cloud. An attacker can get control over gateways to disclose fog information, interrupt the fog system, use that gateway to launch zombie attacks, etc.

Spoofing attack: This occurs when an attacker impersonates to be a legitimate fog device, user, or server to steal data or launch attack to the fog system.

Due to its mobility nature, one of the main security issues in fog is secure mobility and a secure handover. However, many challenges are yet unsolved in the fog security area, and unfortunately, most of the cloud security solutions cannot be applied to the fog scenarios due to their low computation and storage capabilities and high mobility. All the security challenges in fog will be discussed in the next chapter 6.

4.3 Edge attacks

Some vital attacks in this layer are [5], [34], [35], [36], [37], [38], [39]:

Hardware attack: It's a malicious modification of an integrated circuit. An attacker can access to data and software running on the integrated circuit.

Cryptanalysis attacks and side channel attacks: In this type of attacks, attacker by analysing the cryptography that is used in edge device can obtain cipher text or plain text and at the end can get the encryption key used in the algorithm.

Denial of service attack: There are 3 types of attacks: 1. Battery draining: edge nodes have a small battery with limited energy capacity. An attacker may disable battery and cause node failure. 2. Sleep deprivation: an attacker sends a set of legitimate requests to the power-battery limited energy capacity edge node to interrupt the device. 3. Outage attack: it happens when an edge node stops performing normal operations. It causes devices stop functioning.

Physical attacks/ tampering: The attacker with a physical access may get valuable information, tamper with the circuit, modify programming and change the operating system because edge devices are in the physical environment where physical access may be possible.

Node replication attack: The attacker replicates node identification and enters a new malicious node to the system. It affects network performance.

Camouflage attack: Attackers hide an authorized edge node or insert a counterfeit edge node to catch, modify or redirect packets.

Corrupted/malicious node: Attackers take access to the network by corrupting a legitimate edge node or by injecting a malicious node to the system to access to other nodes.

Tracking: A fixed radio-frequency identification (RFID) tag has a unique identifier that can be read by nearby unauthorized readers, therefore attackers use a large number of RFID readers of this unique identifier to access into the system and get authorization.

Inventorying: An attacker can obtain a manufacturer code and product code and other valuable information that are attached to the RFID tags to use for other attacks such as impersonating attacks.

Tag cloning: Attackers impersonate RFID tags to get access to private information

Counterfeiting: Attackers manipulate tags by modifying their identity.

Eavesdropping: Attackers intercept, read, and save message for future analysis to launch more attacks.

Due to the edge devices' characteristics, cloud solutions or even fog security solutions cannot be always applied to them, and new solutions should be designed. The security requirements will be discussed in the next section 5.

Cloud security attacks	Fog security attacks	Edge security Attacks
Backdoor channel attacks	Man in the middle	Hardware attack
Virtualization attack	Virtualization attack	Cryptanalysis attacks and side channel attacks
Denial of service (DoS)	DoS/Ddos attack	DoS/DDoS attack
Malware injection	Malware injection	Physical attack/tampering
Metadata spoofing	Spoofing attack	Node replication attack
Malicious insider	Gateway attack	Camouflage attack
Phishing attack	Most of the cloud attack can be happen in Layer 1 (Layer 1 inherits security challenges from cloud)	Corrupted/malicious node
SQL injection		Tracking node
Sniffer attack		Inventorying attack
Zombie attack (DoS/DDoS)		Tag cloning
Man in the middle		Counterfeiting
Spoofing attack		Eavesdropping

Table 1. Security attacks by architectural layer

4.4 Most potential attacks in F2C system

In the previous sub-sections, all the most possible attacks in the different layers of F2C were described. Now putting altogether, there are many potential security vulnerabilities in F2C-like systems, paving the way for attackers to launch attacks in different layers in the system. In this section, we identify most potential attacks to be faced by F2C-like systems as illustrated in Figure 5, all grouped into three categories, as follows.

Man-in-the-middle attack: Attackers can take the network control between devices at different levels (IoT devices, fog nodes, fog nodes and cloud) to either eavesdrop communication, modify information or even to inject malicious information and code into the system. For example, attackers can obtain the identity of a F2C component and then impersonate it to be an eligible component. Due to the obtained identity, the attacker can impersonate a fog node (malicious fog node) thus getting devices and users information and locations. Also, an attacker can impersonate users and devices to take information or gain access to services it is not authorized to. In upper layers, i.e. fog-cloud communication, an attacker can impersonate fog node or even cloud to launch a man-in-the-middle attack. In all these cases, attackers can launch the attack in passive (eavesdropping without changing information) and active (information modification, manipulation and malicious injection) ways. This type of attack effects the integrity and confidentiality of any F2C-like system (see Figure 5.A).

Denial of service and Distributed denial of service (DoS and DDoS): In this case, attackers either launch multiple service requests to the fog node or perform a jamming wireless communication between fog node-devices to deplete the fog node resources and consequently making it down. An attacker can use legitimate devices, such as IoT devices, fog devices or fog nodes to launch DoS and DDoS using their identities. DoS and DDoS attacks can also occur in upper layers such as fog node-cloud. As a consequence, attackers successfully prevent legitimate users and devices from accessing services provided by a fog node or even by cloud (see Figure 5 .B). In short, this attack severely affects the availability of the F2C system.

Database attacks: In a F2C system, databases may meet a hierarchical architecture, keeping for example one centralized at cloud and some other locally distributed at fog layers. If an attacker can access to these databases, it can modify, manipulate and even leak the data, what may have a high impact on the total system performance. Database attacks may be internal –coming from F2C service providers–, or external –legible and illegible users. This attack intensely effects the F2C integrity and confidentiality (see Figure 5.C).

Thus, proposing a solution for F2C undoubtedly requires a strong background on possible security attacks in each layer and security aspects in the cloud, fog and IoT devices. In the next section, security requirements in each layer of F2C system and security challenges will be analyzed and illustrated.

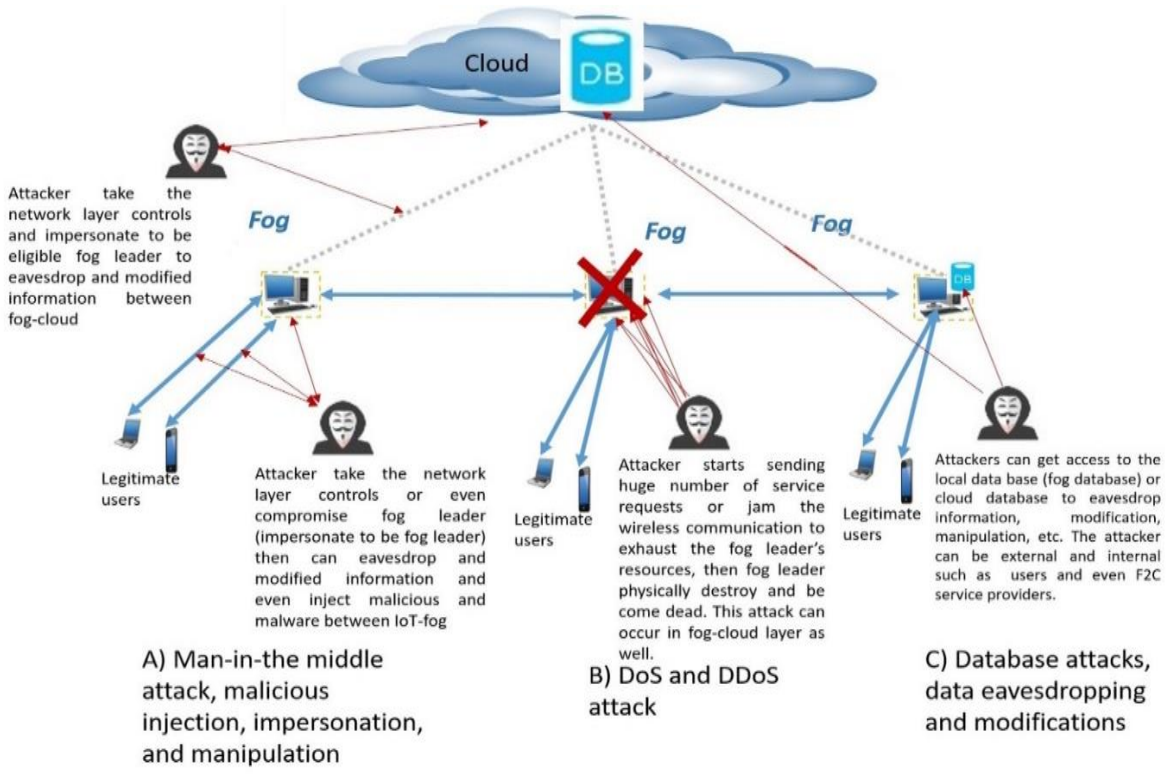

Figure 5. Most potential attacks in F2C system

Chapter.5 Fog-To-Cloud security requirements

F2C as described in previous section includes 3 layer (cloud, fog node (Leader), and Edge devices). We analyses each layer separately to discover most potential F2C security requirements. Table 2 illustrates security requirements in different layers and finally combined F2C system, and in the next subsections 5.1, 5.3 and 5.4 we detail all these security requirements.

Cloud security requirements	Fog security Requirements	IoT security Requirements	Fog-to-cloud security requirements
Authentication and authorization	Authentication and authorization	Authentication and authorization	Authentication and authorization
Key management	Key management	Key management	Key management
Identity management	Identity management	Identity management	Identity management
Policy management	Access control	Access control	Access control
Logging protection mechanism	Privacy	Secure bootstrapping	Data security (Storage, sharing, aggregation, and search)
Access control	Data security and protection (storage , sharing, aggregation)	Data security (storage, sharing, search)	Integrity, confidentiality and availability
Trust	Secure gateway	Integrity, confidentiality and availability	Network security (End-to-end security) Secure communication
Data and storage protection	Intrusion detection	Lightweight protocol design	Intrusion detection
End-to-end encryption	Virtualization security	Secure shareable computation	Trust
API security	Integrity, confidentiality and availability	Malicious IoT device detection	Heterogeneity (secure multitenancy)
Web application security	Lightweight protocol design	Privacy	Privacy (Data, location)
Federation of security among multi clouds	Secure service discovery	Distributed security architecture	Secure virtualization
Heterogeneity	Trust and secure shareable computation	Trust	Secure sharing computation and environment
Integrity, confidentiality and availability	Malicious fog nodes detection	Intrusion detection	Monitoring
Intrusion detection	Distributed security architecture	Secure mobility	Secure front end and back end
Location privacy preserving	Secure multitenancy	Scalability	Forensic analysis
Human resources security	Monitoring	IoT device join and leave	Security management
Secure multitenancy	Secure mobility	Non-Repudiation	Distributed security Architecture
Proper information / asset management	Forensic analysis	Robustness	Secure logging mechanism
Secure virtualization	Trust	Resiliency	Malicious IoT and fog nodes detection
Secure sharing environment	Fog nodes join and leave	Self-organization	Fog and IoT devices join and leave
Monitoring	Network security	Software and firmware security	Secure service discovery
Secure front end and back end	Security management	Network service security	Lightweight protocol for IoT
Risk analysis		Cryptographic security	Secure devices bootstrapping
Service availability		Secure communication	Secure mobility
Recovery management		Security monitoring	
Forensic analysis		End-to-end security	
Secure parallel application		Fault tolerance	
Secure web servers		Energy efficient security	
Secure proxy		Security for handling IoT big data	
Secure trusted third party			
Security management			
Network security			

Table 2. Security requirements in different layers

5.1 Cloud security requirements

The cloud security requirements are considered and analyzed according to ([25], [40], [41], [22], [42], [21], [43], [24]). From these references, we infer the main security requirements in cloud:

1. Secure storage: All data stored at cloud must be encrypted and shared only with authorized users.
2. User and device authentication and authorization: All devices and users such as fog nodes (layer 1 and layer 2), IoT devices must be authenticated to access to the cloud, to prevent

undesired information disclosure to unauthorized users. An open challenge here is to design a distributed authentication mechanism for this hierarchical F2C system.

3. Key management: A key management mechanism to handle key distribution to fog nodes (layer 1 and layer2), fog users, and IoT devices is mandatory to encrypt messages and thus provide secure communication. One of the main challenges is how to distribute and manage keys in a hierarchical and distributed F2C system. A distributed key management system is needed for hierarchical F2C system.
4. Identity management: Fog nodes, cloud services, servers, clouds and all entities must have a unique identity to be recognizable by the system and parties. Identity must not disclose user private information.
5. Policy management: well-structure policies for security provisioning must be defined by F2C cloud for fog layers.
6. Logging protection mechanisms: A secure password-based or other type of logging strategy needed to protect user private information. F2C provider must provide a proper secure way to the users and fog layers for accessing F2C cloud to avoid somebody steal or eavesdrop on user's activities, transactions, and credential.
7. Access control: A well-secure access must be defined for users to prevent hackers and attackers to access to the infrastructure. In hierarchical F2C system, a distributed access controls might be needed.
8. Trust: F2C cloud service providers must be trustable enough for users and fog layers to store their data in the infrastructure.
9. Data protection: All data processing, aggregation, storing must be encrypted and protected from unauthorized users. Data must transfer in encrypted way to the F2C users to do not disclose to unauthorized users.
10. End-to-end encryption: F2C cloud must provide secure end to end communication for protecting data against leakage or breach. F2C cloud must provide end to end communication to their lower layer (fogs in layer 1 and 2).
11. Application programming interface security: Software application communication can be defined by a set of protocols and standards through Internet. Cloud APIs provide all the infrastructure, platform and software service levels communication: i) Platform as a Service API provides access to the service; ii) Software as a Service provides the software application API connection with cloud, and; iii) Infrastructure as a Service provides access and management to resources such as network and VMs.
12. Web application security: Some critical applications, such as banking must have a high secure web quality to avoid attackers to gather any user information.
13. Federation of security among multi clouds: When multiple clouds are federated or some services from different clouds are needed, their security requirements must be federated
14. Heterogeneity: When different service providers deliver a huge amount of services using different technologies, the heterogeneity problem arises, such as no security compatibility at software and hardware levels.
15. Integrity: This refers to data and system integrity. Information can only be changed in an authorized manner. Integrity provides accurate and reliable information between cloud components.

16. Confidentiality and privacy: Access in F2C cloud must be restricted to those authorized to view the data. It prevents user private information disclosure.
17. Availability: This requirement means that cloud and network systems work properly without interruption, problems or possible bugs.
18. Intrusion detection: A well-structure intrusion detection mechanism must be defined for the cloud layer. External or internal attackers can attack or inject malicious information to the F2C system, therefore intrusion mechanism must be implemented in cloud layer to detect any malicious behavior on cloud layer and even below layers (Cloud layer).
19. Location privacy preserving: In the F2C system, fog nodes provide real-time local services, data management and content distribution in geographically distributed fashion. However, users might not want to disclose his/her geo-location. Therefore, location privacy preserving is needed to be applied in both fog layers.
20. Human resources security: F2C cloud provider must provide human resources to make whole cloud environment securely works.
21. Secure multitenancy: F2C cloud provider must ensure secure sharing of computational resources, services, and application with other F2C tenants.
22. Proper information/ asset management: F2C cloud provider must define a proper countermeasure about location of sensitive asset/information, physical control over data storage, reliability of data backup, recovery.
23. Secure sharing environment (secure hypervisor and orchestration): Cloud relies on virtualized resources and environment. F2C cloud provider must provide a secure virtualized environment to be shared by F2C cloud users.
24. Scanning or monitoring: Scanning or monitoring must be done in F2C cloud to detect malicious users, malicious fogs, external and internal attackers, and behavioral analysis.
25. Virtual machine isolation: When virtual machines spread in cloud environment, F2C cloud provider must configure the isolation process to avoid data leakage and VM attacks.
26. Virtual machine monitoring: Host machines play a control role in virtual environment. Host machines in F2C cloud must provide secure monitor of all applications running on VMs.
27. Secure virtualization: virtualization is one important aspect that cloud provides to the users. Therefore, all virtualized operating system running on guest VMs must be secured. All virtual machine elements must be secured. F2C provider must be aware of all security tools and techniques while virtualization is deployed. The virtual network security and hypervisor technologies must be evaluated. The virtual machine configuration and installation must be done carefully.
28. Secure user front end and back end: F2C cloud provider must implement a proper secure back end and front end for F2C users to prevent any data leakage and attacks to user's private information.
29. Service availability: F2C cloud provider must provide securely availability into the services to avoid any availability attack such as DDoS.
30. Risk analysis: F2C cloud provider must provide a proper risk analysis in lower layers in the whole F2C system and define policies to be applied into the system. This risk analysis can act as a control framework to mitigate security risks in the F2C system.

31. Backup and recovery management: F2C cloud provider has to provide service availability even when the F2C data center is down. Therefore, F2C provider must define proper recovery methods with application restoration, privacy, confidentiality, and normal system operational characteristics.
32. Forensic analysis: In F2C systems, techniques and tools must be applied to analyze text indexing, users logging records, and network traffic to avoid cybercrimes.
33. Secure parallel application: The F2C system must provide mutual authentication between applications while parallel applications executions occurs.
34. Secure web servers: F2C system must provide secure web servers and web servers' integrity to prevent eavesdropping, malicious injection, and malicious XML in the F2C system.
35. Secure proxy server: F2C cloud must apply secure proxy server to avoid different types of attacks on proxy server.
36. Secure trusted third party: any compromise or internal attack in trusted third party can impact in user's data leakage which is a big threat to the F2C system. Therefore, the F2C cloud must provide a proper secure trusted third party to be able to authenticate, authorize, and audit the confidential data for fog layers.
37. Security management: Well-defined security requirement, policies, security controls configuration, and feedback to the F2C security management from security controls must be included into the security management block in F2C cloud.
38. Security performance tradeoff optimization: one of the important F2C cloud tasks is delivering services. Therefore, service level agreements must include performance, reliability, security, and violation penalty. Tradeoff between security and performance must be defined due to implementing high level of security which consumes more resources, could impact on QoS and performance.
39. Network security: all the network and communication in the F2C system must be secure. All data transmission from cloud to the fog layers must be encrypted to avoid malicious activities.

5.2 Fog security requirements

Security requirements analysis in fog layers are based on different works in the literature ([44], [31], [45], [32], [46], [47], [33])

1. Authentication: All components, such as fog nodes, fog servers, gateways, etc. need to be authenticated. Authentication allows only authorized components to communicate and obtain data. Fog users, fog devices and cloud service providers must be authenticated to provide a trust F2C environment before any user accesses to any of the services. One of main challenges here is to authenticate constrained IoT device to fog nodes; so, fog nodes must have the ability to authenticate IoT devices for end-to-end communication. Without any proper authentication mechanism such as identity authentication, external attackers may attack to the resources of services and infrastructure.

2. Privacy: In F2C systems, distributed fog nodes locally to provide computational power might bring privacy challenges such as data leakage and private information leakage. Fog user's private information must be anonymous or confidential. Confidential information can be shared only to the authorized components. Fog nodes in layer 1 and layer 2 act as intermediate nodes between IoT and cloud. Therefore, during packets forwarding between all layers, packets must be encrypted and anonymize to prevent private information disclosure.
3. Access controls: Access control must be defined in fogs components to restrict unauthorized users to obtain critical information. Without proper access control mechanism, external attackers may get unauthorized access to the services, user's personal accounts, and infrastructure. Each fog node must use this access control mechanism to prevent any information disclosure to unauthorized users. In hierarchical F2C systems, distributed access control mechanisms may be needed.
4. Data protection: All data processing, communication and storage at fog must be encrypted to be protected against attackers. One of the main challenges on fog nodes is to identify sensitive information from large volume of the data that produced the lower layer (IoT and edge devices). For protecting sensitive information, IoT-fog, fog-fog, fog-cloud data communication must be encrypted.
5. Secure gateway: All gateways must be protected against attackers by a well-defined security strategy and protocol.
6. Intrusion detection: A well-structure intrusion detection mechanism must be defined for the fog system. External or internal attackers can attack or inject malicious information to the F2C system, therefore intrusion mechanism must be implemented in fog layer 1 and fog layer 2 to monitor and analyze traffic and behavior of fog nodes and IoT devices to detect any malicious behavior on fog layers and even lower layer (IoT layer).
7. Virtualization security: Fog inherits some security challenges such as virtualization security from cloud and brings them next to the users. A security mechanism in the fog layer must be defined to protect from these virtualization attacks.
8. Identity management: Fog users, devices, servers must have unique identities to be recognizable. A secure identity management must be defined and implemented for both fog layers, layer 2 and layer 1 (and the security management mechanism in each layer may differ due to the different capabilities of the devices in each layer.)
9. Integrity: It means both, data and system integrity. Information can only be changed in an authorized manner. It provides accurate and reliable information between fog components. Attackers may modify, delete or disclose data on fog nodes if data integrity is not provided on fog nodes.
10. Confidentiality: Access must be restricted to those authorized to view the data. It prevents user private information disclosure. It assures that only authenticated users can access information.
11. Availability: All network and fog systems must be available and work properly without interruption, problems or possible bugs.

12. Lightweight protocol design: By using protocols that consume huge computational process, it may cause huge service delivery delays to the F2C users. Therefore, lightweight protocol could be designed for fog nodes.
13. Secure data storage: In F2C systems, data produced by IoT devices can be stored or aggregated to send to cloud for efficient data management. So, data storage on fog nodes must be secure to not disclose any user's information or data to the unauthorized users.
14. Secure data sharing: Data must transfer from the lowest layer (IoT) to fog nodes, from fog-to-fog nodes, and finally from fog to cloud in an encrypted way to prevent modification, eavesdropping or disclose data transfer.
15. Secure data aggregation: After fog nodes in different levels receive data, they must aggregate it in a secure way to prevent data leakage, privacy-preserving, and communication overhead reduction.
16. Secure service discovery: Each fog node must ensure service provision to authorize F2C system users in their corresponding area.
17. Trust computation: In hierarchical F2C systems, cloud might give computational responsibility to distributed fog nodes. However, how cloud can verify distributed fogs and can provide trust computational capabilities to the users is one of the main challenges.
18. Secure shareable computation: IoT devices have limitation on computational power, therefore they may send data to the upper layer, fog nodes, to be processed, aggregated or stored. The main challenges in this scenario are providing a secure way to handle this shareable computation in a distributed way to not disclose any private information.
19. Malicious fog nodes detection: Distributed fog nodes are so vulnerable to external and internal attacks. Therefore a mechanism is needed to detect malicious fog nodes in layer 1 and 2, and to be revoked from the F2C system.
20. Distributed security architecture and mechanism: The F2C system in fog nodes layer 1 and layer 2 has distributed characteristic. Traditional security architectures and mechanisms are not sufficient and comprehensive for dynamic F2C system. Therefore, it's essential to design new distributed security architecture and mechanism for the F2C system to handle fog nodes (in both layers) security provisioning.
21. Secure multitenancy: each fog nodes (layer 1 and 2) must ensure secure sharing of computational resources, services, and application with other F2C tenants.
22. Scanning or monitoring: Scanning or monitoring must be done in both fog layers to detect malicious users, malicious IoT devices, external and internal attackers, and behavioral analysis.
23. Secure mobility: IoT devices, fog users, fog nodes might be dynamic. Fog nodes in the F2C system must be able to provide secure mobility and secure handover through secure inter-communication between fog nodes in the F2C system.
24. Secure communication: In the F2C system, fog nodes have communication with IoT devices and fog nodes have inter-communication to manage resources and computation locally. Therefore, all these communications must be secure to avoid any eavesdropping or attack.
25. Key management: one of the main challenges in the F2C system is how to provide key distribution, key update, and key revocation for distributed fogs to be secure and efficient.

Keys must be provided for every component to encrypt and provide end-to-end secure communication. Fog nodes must be able to get keys from the F2C provider, and in parallel also to provide keys for constrained IoT devices. Key management must be secure and efficient in terms of time-delay, number of messages, and energy consumption.

26. Fog forensic analysis: Each fog nodes in the F2C system must have forensic analysis to obtain log data, data integrity, and multi-tenancy.
27. Trust: In the F2C system, trust is a one of the main challenges due to the distributed nature of fog nodes. Trust environment must be provided in the F2C system for avoiding malicious attacks.
28. Fog nodes join and leave: Fog nodes might join to the F2C system and leave continuously. One of the big challenges here is how to handle security, authentication, and privacy issues for these dynamic fog nodes (joining and leaving) in the F2C system. Traditional security models are not suitable for this F2C dynamic system, therefore, a new security model must be designed to cope with the F2C needs.
29. Secure virtualization: in the F2C system, fog nodes in layer 1 and 2 may provide virtualization. Therefore, fog nodes must provide secure virtualization environment the same as in cloud layer to avoid malicious VM, virtualization attacks, and to prevent when attackers may take control over hardware or operating system to launch attacks.
30. Network security: all the network and communication in the F2C system must be secure. All data transmission between fog nodes in different fog layers must be encrypted to avoid malicious activities.
31. Security management: a well-structured security management must be designed to handle the security requirements, configurations, and policies in the fog layers.

5.3 IoT devices security requirements

IoT-devices security requirements are listed according to the proposals in ([25], [40], [45], [48], [36], [37], [49], [50], [38], [51], [52], [39])

1. Authentication and authorization: All edge devices such as IoT devices must be authenticated to communicate with fog, with cloud and to each other. IoT devices have not enough computational power to implement cryptography and to do authentication, therefore, fog nodes in layer 1 and layer 2 must provide an authentication mechanism for these constrained devices in the F2C system.
2. Access control: a well-structure access policy must be defined to the edge devices (IoT devices). Due to the constrained nature of IoT devices, fog layers must provide a new distributed access control mechanism for IoT devices. Access control defines who or what can be view or use F2C resources.
3. Secure bootstrapping mechanism: A secure authenticated registration and initialization for bootstrapping edge devices must be defined. In the F2C system, fog layers must be able to provide registration, authentication, and in case of mobility handover, and finally secure

bootstrapping for IoT devices. During device bootstrapping, all private information such as private keys and pre-shared keys must kept secure to not be stolen.

4. Data security: The huge volume of data produced by IoT devices must be secured in communication and processing. Data must be encrypted for communication between edge devices to ensure only authorized devices and users can access to the F2C system and services. Fog layers must provide encryption and secure communications for IoT devices.
5. Identity management: All devices should have a unique identity which must be kept secure from unauthorized users. The main challenge is huge number of distributed devices such as fog nodes, edge and IoT devices in F2C system. Identity management and authentication are so integrated therefore, identity updates and revocation in malicious situation is needed. Another main challenge that may arise is the need of identifying IoT devices, and being able to be searched and accessed in secure manner.
6. Integrity: The meaning is the same as previously described in cloud and fog; it provides accurate and reliable information between edge devices.
7. Availability: The network and edge devices must be available and work properly without interruption, problems or possible bugs.
8. Confidentiality: Access must be restricted to those unauthorized to view the data. It prevents user private information disclosure. It assures that only authenticated users can access to the information.
9. Lightweight protocol design: IoT devices suffer from low computational, network and storage capabilities. Therefore, IoT devices are not be able to process protocols that needs huge computational process. For this reason, new lightweight protocols must be designed for IoT devices.
10. Secure data search: IoT devices must encrypt data before sending it to fog nodes in the upper layer. Users and IoT devices must provide secure index when they upload to the fog nodes.
11. Secure shareable computation: IoT devices have limitation on computational power, therefore they may send data to upper layer, fog nodes, to be processed, aggregated or stored. The main challenges here are providing a secure way to handle this shareable computation in a distributed way to not disclose any private information.
12. Malicious IoT device detection: Distributed IoT devices are so vulnerable to external and internal attacks. They cannot implement essential cryptography mechanism to prevent attacks due to their low computational, network, and storage capabilities. Therefore, a mechanism is needed to detect malicious IoT devices to be revoked from the F2C system. These mechanisms may be applied to upper layers to detect malicious IoT devices.
13. Privacy: IoT devices might produce sensitive private information such as the data from body-sensors; but also non-private information such as pollution rates. How to be able to distinguish between these produced data, and how to provide data privacy to private information to not disclose it to unauthorized users are main challenges. Some other challenges is privacy issues in data, location, and usages privacy. In the F2C system, distributed fog nodes in the different layers must be able to provide data privacy to IoT devices due to their more powerful computational capability; and fog nodes collaboration for aggregating received data from IoT devices.

14. Distributed security architecture and mechanism: IoT devices are distributed in nature and with low computational, network, and storage capabilities. Traditional security architectures and mechanisms are not sufficient and comprehensive for this dynamicity of the F2C system. Therefore, it's essential to design new distributed security architecture and mechanism for the F2C system to handle IoT devices security provisioning.
15. Trust: Establishing trust between IoT devices and between IoT devices and fog layers must be done to provide a secure F2C environment and sustain security and reliability in the F2C services. Due to the IoT devices constrained capabilities, fog layers must provide trust mechanisms to them.
16. Intrusion detection: It must be applied to detect malicious IoT devices and provide reports about these detections. However, IoT devices cannot implement intrusion detection techniques due to their low computational power. Therefore, fog layers must provide intrusion detection to IoT devices.
17. Key management: The fog layer must provide key generation, distribution, update, and revocation mechanisms for IoT devices in the F2C system due to the edge and IoT device's low computational power.
18. Secure mobility: IoT devices might be dynamic. Fog nodes in the F2C system must be able to provide secure mobility for these IoT devices, while IoT devices are constrained to provide secure mobility by themselves.
19. Scalability: With the growing number of IoT devices, traditional security schemes suffer from scalability issues. Therefore, a new scalable security scheme must be designed for the F2C system to handle this huge volume of IoT devices.
20. IoT devices join and leave: IoT devices might join to the F2C system and leave it continuously. One of the big challenges here is how to handle security, authentication, and privacy issues for these dynamic IoT devices (joining and leaving) in the F2C system. Traditional security models are not suitable for this F2C dynamic system, therefore, a new security model must be designed to cope with the F2C requirements in terms of dynamicity.
21. Anonymity: In the F2C system, source of produced data must be anonymous to provide privacy in the F2C system. So, IoT devices need to provide this anonymity.
22. Non-repudiation: If fog nodes in the upper layer receive messages from IoT devices, then IoT devices cannot deny they did not send the message to the fog nodes in the F2C system.
23. Robustness: In the F2C system, the IoT network must work properly and keep alive even during some malicious incidents.
24. Resiliency: the security scheme in IoT devices must protect against attacks, even if some IoT devices or the F2C system are compromised. Huge number of constrained IoT devices gives the opportunity to attackers to launch attacks. Therefore, resilience mechanisms against attacks and failures must be applied into the F2C system to have recovery mechanism in order of maintain the F2C operations during failures and attacks.
25. Self-organization: If IoT devices or fog nodes in upper layers got compromised, other IoT devices or collaborator IoT devices must be maintained secure due to their self-organized characteristics.
26. Software and firmware security: IoT devices software updates must be done properly and securely. The idea is that the attacker must not get any sensitive information such as

cryptographic credential and also must not get update software's configuration during updates.

27. Security in different types of connectivity: in the F2C system, IoT devices might use wireless, wired, private, and public network to connect to fog nodes. All this various types of connectivity must be secure to provide data integrity and quality of service to the F2C system.
28. Network service security: network services in the F2C system must be secure, otherwise, IoT devices would be inaccessible to F2C users.
29. Cryptographic security: IoT devices have a constrained nature. Therefore, fog nodes must provide appropriate cryptographic security for IoT devices in F2C system.
30. Secure communication: IoT devices communications and IoT device-fog layers' communication must be secure in the F2C system.
31. Security monitoring: IoT devices interaction must be monitored by the upper fog layer, fog nodes, to track malicious activity or attackers in the F2C system.
32. End-to-End security: All IoT devices communication, IoT devices user communication, and IoT devices-fog layer communication must be secure in the F2C system. All these communications must be secure in the F2C system to avoid eavesdropping, modifications, or tampering on data exchange. One of the main challenges is because IoT devices use different characteristics and communication technologies, therefore, establishing secure communication is very challenging. Fog nodes in the F2C system may be able to provide end-to-end secure communications for IoT devices.
33. Fault tolerance: IoT devices must have defense mechanism to repel attacks and recover from damage. Fog nodes must provide fault tolerance to IoT devices in the F2C system.
34. Energy efficient security: IoT devices can use efficient cryptographic modules and fog nodes can provide cryptography to IoT devices to reduce energy consumption in the lowest layer of F2C system.
35. Security for handling IoT big data: all IoT devices in the F2C system provide data. All this data must be transferred, maintained, and synced in a secure way. Fog nodes in different layer might be helpful to provide security in this big data.
36. Secure service and resource discovery: Service and resource discovery between F2C users and IoT devices (services requester and target IoT devices) must be secure and authenticated. Fog layers in the F2C system must provide a mechanism to query and match resource and service directories. In this case fog layers and IoT devices must be authenticated mutually in the F2C system to match services and resources in a secure manner.
37. Security management: With the growing number of IoT devices and their constrained characteristics, a well-structured security management must be designed to handle security requirements, configurations, and policies. In the F2C system fog layers must provide this distributed security management for the IoT layer.
38. IoT devices physical security: IoT devices must be tamper resistant hardware to prevent be tampered or be cloned in F2C system. In this case, if one device is compromised, it should have a resiliency mechanism that must not affect the other devices such as other IoT

devices and fog nodes in F2C system. On the other hand, compromised devices must be detected and blocked by fog nodes in upper layers in the F2C system.

39. Network security: all the network and communication in the F2C system must be secure. All data transmission from IoT devices to fog layers must be encrypted to avoid malicious activities.

5.4 F2C combined security requirements

In this sub-section, we identify the most potential security requirements for combined fog-to-cloud system (Table 3) according to the previous different layers' security requirements analysis.

Security requirements in combined F2C system	Description
Authentication and authorization at the whole set of layers	Authentication must be done for all participant components in F2C systems to provide integrity and secure communication. A hierarchical authentication may be considered, in short, cloud authenticates fog leaders, and fog leaders authenticate edge devices (fog nodes and IoT devices).
Appropriate key management strategy	F2C systems must include a well-defined key management strategy for keys distribution and update as well as for key revocation.
Access control policies to reduce intrusions	Access control must be supported at cloud level and distributed access controls at fog layers
Providing confidentiality, integrity, and availability (the CIA triad) as a widely adopted criteria for security assessment	In a F2C system, user's information must stay private not to be disclosed to unauthorized users (confidentiality), information must be complete, trustworthy and authentic (integrity), and finally the whole system must work properly, reacting to any disruption, failure or attack (availability).
All network infrastructure must be secure	All components in a F2C system (users, devices, fog leaders, fog nodes, and cloud) must communicate through secure channels regardless the specific network technology used to connect (wired, Bluetooth, wireless, ZigBee, etc.).
All components must be trustable	In the proposed hierarchical approach, the set of distributed fog leaders act as a key architectural pillar enabling data aggregation, filtering, and storing closer to the users, hence making trustness mandatory for fog leaders.
Data privacy is a must	Data processing, aggregation, communication and storage must be deployed not to disclose any private information, or produce data leakage, data eavesdropping, data modifications, etc. To that end, data must be encrypted, and data access must not be allowed to unauthorized users. Moreover, assuming mobility a key bastion in F2C systems other particular privacy related issues come up, such as for example geo-location.
Preventing fake services and resources	Fake scenarios are highly malicious in F2C systems, hence some actions must be taken to prevent that to happen, such as services and resources must be discovered and identified correctly and services and resources allocation must be done securely.
Removing any potential mobility impact on security	Fog nodes and IoT devices might be on the move, thus demanding the design of secure procedures to handle mobility related issues, such as devices handover.

Table 3. Most potential security requirements in F2C

1. Authentication and Authorization: In the F2C system, all components in the different layers including cloud, fog, and edge devices must be authenticated. Authentication must be done

for all participant components in F2Csystem to provide integrity and secure communication.

2. Key management: A well-defined key management for distributing keys, update keys, and revocation must be designed for the F2C system. F2C has distributed and hierarchical characteristics, therefore, distributed key management must be applied to the F2C system.
3. Identity management: All users, devices, services, etc. in the F2C system must have unique identity to be used in authentication and validation. The distributed characteristics of the F2C system arises the need for distributed identity management. All identities must be managed by means of assigning IDs, update IDs, and IDs revocation.
4. Access control: a well-defined access control in cloud and distributed access controls in the fog layers must be applied to the F2C system.
5. Integrity, confidentiality, and availability: In the F2C system, systems and data must be integrated (all components must be authenticated to each other), user's information must keep private to not disclose to unauthorized users, and finally the system must work properly without any disruption or failure.
6. Network Security (End-to-End security): In the F2C system, users, devices, fog nodes, and cloud must have secure connectivity. For example, a sensor providing private information must be able to communicate with fog nodes in a secure channel. Data over channels must be encrypted to prevent disclosing information.
7. Intrusion detection: Intrusion detection mechanisms must be applied in cloud as the centralized point and also in parallel, distributed in the fog layers of the F2C system.
8. Trust: Trust between all components in F2C system such as cloud, fog nodes, and IoT devices must be applied.
9. Heterogeneity (Secure multitenancy): The F2C system may have different service providers that deliver a huge amount of services using different technologies, therefore, the heterogeneity problem arises, such as no security compatibility at software and hardware levels. A secure multitenancy must be provided in the F2C system according to service provider's agreements.
10. Privacy (Data, location): In the F2C system data processing, aggregation, communication, storage must be done in a secure way to not disclose any private information, or to not produce any data leakage, data eavesdropping, data modifications, etc. All data in channels must be encrypted and access to data must not be disclosed to unauthorized users. In parallel, F2C users may not want to disclose their geo-location, therefore, location privacy preserving must be implemented in the F2C system.
11. Secure virtualization: one of the main advantages of the F2C system is bringing virtualization next to the users by means of the fog layers. Cloud and fog layers in the F2C system must be able to provide secure virtualization to prevent any virtualization attacks.
12. Secure sharing computation and environment: The F2C system allows the fog layers to share their computational power to lower levels (IoT devices) with low processing power. This shareable computation in the F2C system arises security and privacy concerns. Therefore, security strategies must be applied into the F2Csystem to provide a secure shareable F2C environment.

13. Monitoring: A well-structure security monitoring in cloud and also a distributed security monitoring for the distributed fogs nodes must be applied in the F2C system to analyze traffics and other variables and to detect malicious activities.
14. Secure front end and back end: In the F2C system, the provider must implement a proper secure back end and front end for the F2C users; and also for cloud and fog layers to prevent any data leakage and attacks to user's private information.
15. Forensics analysis: In the F2C system, techniques and tools must be applied to analyze text indexing, users logging records, and network traffic in all layers (cloud, fog, and edge devices) to avoid digital crimes and cybercrimes.
16. Security management: Well-defined security requirements, policies, security controls configuration, etc. must be applied in the hierarchical F2C system. One of main challenges here is the management of security in the different and distributed fog layers and edge devices.
17. Distributed security architecture: Traditional centralized security architectures cannot provide the expected security level. This is because the the hierarchical nature of the F2C system with distributed fog layers. Therefore, a new distributed security architecture must be designed to be applied to the F2C system.
18. Secure logging mechanism: a well-structured logging mechanism should be defined for cloud, fog, devices, and users in the F2C system.
19. Malicious IoT devices and fog nodes detection: All IoT devices and fog nodes behavior must be monitored to detect any malicious activity and revoke that devices when it is needed. Then a distributed malicious device detection mechanism must be designed for the F2C system. Without this revocation mechanism, a fog node or IoT device might be attacked or to be used for launching attacks, therefore, they must be detected and revoked.
20. Fog nodes and IoT devices secure join and leave: Both fog nodes and IoT devices join and leave the F2C system, therefore for both cases a secure joining (authentication, secure communication, access controls, and other security provisioning.) and a secure leaving (session revocation, key updates, and key and identity revocation if they leave the F2C system) must be applied in the hierarchical F2C system.
21. Secure service discovery: In the F2C system, service discovery and allocation to the authenticated resources must be done in a secure way.
22. Lightweight protocol for IoT: IoT devices have not enough computational capabilities to provide security by cryptography. Therefore, fog nodes might supply the cryptography needs for IoT devices in the F2C system and use lightweight cryptography protocols for IoT devices.
23. Secure device bootstrapping: All devices in the F2C system must be bootstrapped in a secure way to participate in the system. Any failure in bootstrapping might compromise the device in Fthe 2C system. In the F2C system, fog nodes and IoT device must be bootstrapped in an authenticated way. During device bootstrapping, all private information such as private keys and pre-shared keys must kept secure to not be stolen.
24. Secure mobility: Fog nodes and IoT devices might be on the move (mobility), therefore, secure handover and secure mobility must be applied in the F2C system.

Chapter.6 Fog-To-Cloud security challenges and directions

This chapter identifies the most potential security challenges for the combined F2C system according to different layers' security requirements mentioned in the previous chapter and also according to the F2C hierarchical and distributed characteristics. All security challenges are illustrated in Table 4.

Security Area	Challenge	Description
Trust & Authentication	Trust Authentication Key management Identity management	Authentication is mandatory to prevent unauthorized users to access the system. The authentication mechanism needs identity or certificate to be verified and give users authorization to be involved into the system. Trust can be established between components after their authentication. Trust is one of the key component for establishing security between distributed fog nodes. Then, keys for encryption and decryption process can be distributed for components. Both Keys and identities need to be generated as unique, updated, and revoked during attacks, therefore, in F2C system handling key and identity management are the bottleneck due to hierarchical nature and huge number of distributed low-computational IoT devices. For Authentication and establishing trust, the traditional cloud as centralize point cannot be sufficient in F2C system due to distrusted nature. Therefore, as the main challenge here, trust and authentication must be redesigned to be handled in F2C system in hierarchical and distrusted way.
Access Control & Detection	Access Control Intrusion detection mechanism Malicious IoT and fog device detection	Access control is used to put rules that who and what can access the resources. In the case, the unauthorized users access the resources, intrusion and malicious device detection is needed. In case of access control and intrusion detection, handling the huge number of distributed IoT devices and fog nodes is one the main challenge in F2C security. Therefore, a need rises to re-design access control and intrusion detection in distributed way to be handled in F2C system
Privacy & Sharing	Privacy Secure sharing computation and environment	Privacy means that all the user's private information should not be disclosed to the others. In F2C system, fog nodes in hierarchical way would share their resources to users and IoT devices with low-computational power to run services. In this case, one of critical issues is how to handle user's, IoT devices', and Fog device's privacy in hierarchical F2C system without disclosing any critical information about each one of them to each other or even others.
End-to-end Security	Network security Quality of Service Heterogeneity	Providing secure end-to-end communication between all components in a F2C system is one the challengeable issue due to different network protocols, huge amount on distributed devices at the edge of the network, and

	Secure visualization	hierarchical F2C architecture. To provide secure communications, initially each one of the participant devices in F2C system must bootstrap in secure way. Fog nodes can be host virtualization environment to run the services, therefore, secure virtualization is a must at fog layers. All the secure communications must be monitored to detect any abnormal or malicious activities. All fog and cloud providers must set agreement to provide secure communications between their components in F2C system. At the end, a most challengeable secure communication issue is to design a new distributed security architecture to handle end-to-end security with less impact on the Quality of service.
	Monitoring	
	Centralized vs distributed security management	
	Secure devices bootstrapping	
Mobility support	Secure mobility	In the F2C system, devices such as IoT devices, mobiles, cars, etc. are dynamic. The devices are on the move. All devices arrive to the fog nodes must be securely discovered. A device joining in F2C system for the first time and even the existing device join the fog area must be done in secure way. Then, a securely leaving fog area to join another area must be considering as well. The most challengeable secure mobility issues are using cloud as single point of failure and even bring scalability issues. In hierarchical F2C system, a new distributed security must be design to handle device discovery, joining and leaving, mobility, and handover in secure way.
	Secure devices joining and leaving	
	Secure discovery and allocation	

Table 4. Security challenges in F2C

Authentication: With the growing number of IoT devices at the edge of the network, the use of the conceptually centralized cloud for handling devices and users authentication cannot be sufficient for the F2C system, due to the huge number of messages going and coming to/from the cloud. Even if cloud is down, compromised or gets an attack, the whole F2C system will be compromised. Authentication in any system is one the essential security requirement to bring system and data integrity. The distributed nature of the F2C system demands a distributed authentication mechanism. On the other hand, taking advantage of the hierarchical nature of F2C, cloud can manage the authentication fog nodes, and then these authenticated fog nodes can handle IoT devices and users authentication in their area in a distributed fashion. But the main question here is "how can these hierarchical authentication processes be done and which type of authentication for each layer can better fit." Authentication mechanism must be redesigned to fulfill the distributed and hierarchical nature of the F2C system.

Key management: The F2C system has distributed characteristics. Therefore, a centralized key generator center (KGC) can bring challenges into the system due to huge number of transferred messages (scalability issues) and even the whole system can be affected, if the centralized KGC is compromised, fails or is attacked. Distributed key management for generating, assigning, updating and revoking keys must be designed for the combined F2C system. However, there are many challenges such as:

- How can the huge number of IoT devices with low computational power get keys to encrypt data?
- Is there the possibility that fog nodes can handle distributed key management for IoT devices and users?"
- Does the hierarchical nature of F2C system mean that cloud can act as key manager for fog layers and fog nodes act as distributed key managers for their areas?" If the answer is yes, "which type of key management algorithm (symmetric or asymmetric) for each F2C layer must be applied?"

Identity management: The primary challenge here is assigning IDs to devices with low computational power. The centralized identity manager in the cloud for assigning, updating, and revoking IDs for the huge number of distributed devices cannot be adequate for the F2C system. On the other hand, some devices have low computational power; therefore they cannot store the whole ID. So, some questions raise such as:

1. Is it suitable to divide the whole ID into fragments and use different fragment's size for each layer [53]?
2. How must IDs be divided into fragments to be stored in each layer?
3. Which IDs fragment size can be stored in each layer?
4. How will distributed IDs manager in fog layer be secured?

Access control: The main challenge here is, "how to design a hierarchical access control such as, centralized in the cloud and distributed in fog layers for controlling and managing devices and users access or put restrictions on the accesses? In case of the F2C system, a hierarchical access control such as distributed access controls at fog layers for devices and users at the edge of the network and centralized access control in cloud layer for fog nodes must be reconsidered.

Intrusion detection mechanism: The centralized intrusion detection for the huge number of participant devices in the F2C system brings challenges such as, malicious activities or nodes cannot be detected due to the huge volume of traffic analysis or even if the intrusion detector collapse or it is compromised, then, the whole system can fail for detecting malicious devices.

Quality of service (QoS): On one hand, the F2C system brings its hierarchical architecture to execute services in fog, cloud or both, providing the required QoS for users and system. On the other hand, implementing security in F2C impacts on the QoS regarding service execution time delay, bandwidth, CPU, service allocation time, etc. due to the huge volume of computational cryptographies requirements when implementing security. The F2C system demands both, to have the required level of security and also QoS. The main challenges are: "If distributed fog nodes handle required security for end devices, QoS can be met?" Fog nodes have other responsibility such as data collection, aggregating, filtering, storing, service execution, etc. Therefore, "when embedding security in fog nodes, QoS can be met?" Then, the main question here is: "how can both demands: security level and QoS be met in the F2C system?"

Network security: In the F2C system, different technologies such as wireless, wired, zigbee, bluetooth, etc. may be deployed. The main challenge here is implementing network security for all types of network communication technologies to provide a secure system. Besides, different security protocols for different technologies in the F2C system might impact each other. Traditionally the way that cloud handles the security cannot be sufficient for the F2C system due

to the single point failure and conceptually distanced from devices. Therefore, network security must be re-designed to handle all types of network communication technologies to provide end-to-end security in the F2C system, and to avoid the negative impact among them.

Trust: In the F2C system and it's the distributed nature, distributed trust establishment at the edge of the network is needed. Fog consortium [54] aims to provide the hardware root of trust into the fog nodes to provide security in the IoT layer. This hardware root of trust is still under development. There are some challenges such as cost might be expensive and even if a fog node is compromised or attacked, how security for its IoT devices in its area can be provided is unknown. Blockchain as new distributed trust establishment may be effective to be applied into the F2C system.

Heterogeneity: F2C systems might have more than one service provider and cloud. The main challenge is that how the different security strategies are applied for different service providers in the F2C system, and how they can be compatible with each other. In the F2C system, all service providers must get service security level agreement, and put efforts to provide compatible hardware and software security to ensure the F2C system security.

Privacy: Data anonymization and data privacy must be applied to the F2C system to protect user's private information. However, one of the main challenges here is "which data anonymization can be applied to the F2C system to bring balanced privacy and data utilization?" In the combined F2C system, fog nodes closer to users can provide privacy preserving because the data can be analyzed, aggregated, and stored closer to the users. Another main challenge here is that users might do not want to disclose their location; on the other hand, in case of attacks, the F2C system must locate attackers. Privacy in different layers must be analyzed, privacy concerns must be identified, and finally, data and location privacy must be applied appropriately without impacting the whole security in the hierarchical F2C system. In the F2C system privacy and security must be provided being compatible with each other.

Secure visualization: In the combined F2C system, fog nodes bring virtualization closer to the end users. In fact, virtualization environments are so vulnerable to virtualization attacks such as virtual machine scape, hypervisor attacks, etc. If the hypervisor of a fog node is attacked or becomes controlled by the attacker, the whole virtualization environment gets compromised. Therefore, virtualization must be implemented in a secure way for the hierarchical F2C system.

Secure sharing computation and environment: In the combined F2C systems, fog nodes share their computational environment with IoT devices with low computational power. Hence, attackers can get an advantage to fake themselves as legible devices (both as IoT devices and as fog nodes) to launch passive and active attacks. Two main challenges arise here, first, "how can a fog node share its resources in a secure way with low computational power devices?" and second "how can IoT devices trust fog nodes to outsource their service execution to them?" Trust establishment and the ability to distinguish between legible and illegible all devices is paramount here. Therefore, threat models and security analysis for the hierarchical shareable F2C environment must be done in each layer at an early stage, and all needed security requirements such as authentication, privacy, etc. must be provided before any device share their computational power with the system.

Monitoring: With the vast number of devices in a distributed nature at the edge of the network, centralized monitoring at the cloud is not adequate for the combined F2C system. The main question here can be "which monitoring strategy or strategies must be applied to monitor the huge number of distributed devices and the huge traffic analysis to detect malicious activities?" Therefore, a distributed monitoring implemented in fog nodes to detect any malicious or abnormal behavior at the edge of the network, combined with centralized monitoring at the cloud for fog layers are needed to be designed and implemented.

Security management: The main challenges and questions here are, "can cloud as a centralized concept be sufficient to act as a security manager for the hierarchical F2C system?" The answer to this question is no, cloud as a centralized point failed to provide security and resist or prevent several appeared attacks in past decades. Therefore, a question arises, as "What security management strategy must be taken into account for the distributed fogs and IoT devices?" Therefore, to tackle these challenges, a new security management strategy, such as distributed security manager at fog layers and centralized at cloud, can be a suitable for the current security management in the F2C system.

Centralized vs. distributed security architecture: With the growing the number of devices at the edge of the network and their distributed nature and mobility, traditional centralized security architecture for handling all system security is not sufficient. Then, "How and which distributed architecture can be fit and adequate to provide reasonable F2C security level? "Distributed security architecture is a must to apply into F2C system, but the main challenge raises as a complexity issue due to handling distributed security provision. Therefore, a new security architecture must be re-designed to handle hierarchical nature of the F2C system and in parallel handling required security for distributed devices with both low and high-level computational power.

Malicious IoT and fog device detection: The massive number of devices at the edge of the network gives the opportunity to attackers for launching attacks or faking devices to be legible to eavesdrop the system. If the device gets compromised or attacked, it must be detected and revoked from the system. The primary challenge is that "how can malicious devices in different layers be detected and what strategy or algorithm can be fit to detect the malicious devices on real-time processing and revoke them?"

Secure devices joining and leaving: The centralized cloud providing secure join and leave for the huge number of devices in different layers cannot be sufficient for F2C systems. Even, using cloud for handling this considerable number of devices joining and leaving the F2C system can bring scalability issues. A hierarchical strategy can be useful to be applied here, such as cloud can manage secure joining and leaving of fog nodes, and in parallel, each fog node can control secure joining and leaving of devices (IoT devices) in its area. On the other hand, in the F2C system, fog nodes should have secure intercommunication to provide secure devices handover in another area in case of mobility.

Secure discovery and allocation: All resources, services, and devices must be discovered in a secure way. Services must be allocated to resources, previously authenticated. Hence, different challenges arise: "how can services and devices be discovered in an authenticated secure way in the hierarchical F2C system?", "how can services be allocated to the corresponding authenticated

resources securely?", "are fog nodes getting responsibility for securely discovering devices and allocating services to authenticated resources in a distributed fashion?", considering the different technologies such as Wi-Fi, zigbee, bluetooth, etc. "which strategy can be applied in the F2C system to provide secure discovery for all mentioned technologies?" According to these challenges, "can a strategy for hierarchical resource and service discovery, as well as allocation in a secure authenticated fashion be re-designed for the combined F2C system?"

Secure devices bootstrapping: All components in the combined F2C system must bootstrap in a secure way by getting public and private parameters. In this scenario, traditionally centralized cloud or centralized trusted authority for bootstrapping the huge number of devices in the F2C system cannot be affordable. The main issues can be mentioned such as: "which component must take cloud responsibility for bootstrapping devices at the edge of the network?", "can we apply a strategy where the cloud bootstraps fog nodes and fog nodes bootstrap devices in their area in a distributed fashion?", and finally handling the bootstrapping of the huge number of devices at the edge of the network is the main challenge in the combined F2C system.

Secure mobility: By growing the number of devices at the edge of the network which are on the move, the main challenge arises when trying to handle secure handover and secure mobility. The centralized and remote cloud cannot handle secure mobility for distributed devices at the edge of the network. Fog nodes closer to the users might handle the secure mobility issues. Therefore, distributed fog nodes must have secure intercommunication to provide secure handover for devices on the move. But the main challenges here are, "if a fog node is on the move as well, who is providing secure mobility and secure handover for it and how is fog node handling secure mobility and secure handover for devices on the move?"

To tackle all the questions and challenges mentioned above, proper security threats analysis must be done for the F2C system then a distributed security architecture with respect to hierarchical fashion must be redesigned to provide the identified security requirements for the combined F2C system. In the next chapter, all the possible exiting proposal for providing security to a F2C system in different layers will be analyzed.

Chapter.7 Existing security proposals

In this chapter, most potential existing security solutions for different layers of F2C system are described and analyzed. Considering that, although traditional cloud security protocols may theoretically provide some security to fog computing systems, the constraints on processing capacities of the edge devices undoubtedly limit the efficiency of such existing protocols. On the other hand, security initiatives designed for fog computing cannot be applied to cloud due to they are designed for edge device for limited capabilities and cannot meet the huge amount of processing and storage cloud requirements. In addition, the design of secure fogs and clouds with existing security solutions without considering the coordinated nature of F2C (interoperability, heterogeneity, etc.), may cause additional security problems when considering the whole set of resources envisioned in F2C.

In this section, existing security proposals will be reviewed and analyzed. Cloud, fog, and IoT security proposals are revisited to illustrate that these solutions are not compatible with the combined F2C system without considering its hierarchical characteristic.

7.1 Existing cloud layer security proposals

Here, the most potential existing security solution in cloud for providing authentication, key management, access control, and intrusion detection are reviewed and finally in security solution's conclusion all the reviewed works are analyzed to be F2C adaptable or not as illustrated in Table 5.

7.1.1. Authentication and key management solutions

In [55], authors propose using integrating username/password, bio-metric fingerprint and one-time password into the centralized authenticator server for client-cloud authentication and key management. Their proposal provides mutual authentication, privacy preserving, and access keys protection and management. [56] Proposes a user-cloud authentication scheme using smart-card and hash function that provides mutual authentication, privacy, and data integrity. In [57], authors propose zero knowledge data privacy for outsourcing data from cloud with a new key-updates strategy. Their strategy uses homomorphic authenticator, short signature scheme, and unidirectional proxy re-signature for key updates. This proposal eases key updates by not downloading the whole file with user's key changes but only a file tag that decreases communication overhead. Their proposal provides privacy and key-updates.

Proposal in [58] uses combined identity authentication and quantum key distribution to provide mutual authentication and session key agreements for users to access cloud services. Cloud server takes the responsibility of registering users and store authentication parameters. Then, the user and cloud server will be mutually authenticated. Their proposal provides mutual authentication, authorization, confidentiality, identity and access management, forward secrecy, anonymity, availability and it is secured against all types of attacks.

[59] proposes an authentication mechanism for cloud-mobile system. They propose two solutions:

1. Mobile-cloud establishes a transport layer security (TLS) communication, cloud provider asks authentication center, then at the end cloud-mobile exchanges information to be checked and authenticated.
2. There is no authentication center in this solution, mobile gets and uses a chip for calculating authentication information at registration, then establishes a TLS connection with cloud and exchanges information to be checked and authenticated.

[60] proposes three solutions for cloud computing security which includes:

1- User-cloud authentication mechanism which uses authorization token with blind signature protocol (RSA) that provides user privacy by not disclosing user' identity.
2- Multi-keyword searches by using bloom filter's bit pattern.
3- It provides cloud search security against insider threats by using hash functions.

Their proposal provides high cloud security level.

The work in [61] proposes cloud-centric authentication as a service for multi-level systems. Their proposal components include a centralized cloud service provider as a certificate authority and key management facility for managing all users and IoT devices, wearable nodes for producing data and information, wearable network coordinator (Intermediate level) for managing IoT devices, and finally users who request IoT device information. Cloud service provider uses PKI (elliptic curve cryptography, ECC); the user and cloud service provider signatures generation and verification is done by elliptic curve digital signature authentication (ECDSA), key agreement between cloud service provider and wearable network coordinator is done by means of elliptic curve diffie-Hellman (ECDH). Their proposal provides scalability and effectiveness.

[62] proposes a cloud security mechanism by using trusted third part and applying identity-based encryption and MD5 message digest algorithm. In their proposal, a client uploads his/her file encrypted on a server, the cloud admin generates a hash value for the client's file, the cloud admin sends the file's ID and the hash value to the trusted third party, then, trusted third party receives client's request to audit the file, and finally the trusted third party verifies or rejects the user's file.

Authors in [63] propose the use of a hash-based selective disclosure and chebyshev chaotic for local mutual authentication between mobile and wearable devices, and in parallel, merkle hash tree based selective disclosure mechanism is used for remote authentication between wearable devices, mobile and cloud. Their architecture includes a user who has a mobile phone that can connect to a cloud server, smart devices (smart glass, smart watch, and etc.) that communicates with the mobile phone, and finally a centralized cloud server that provides remote data storage and outsourcing services. Their proposal provides confidentiality, integrity, mutual authentication, forward security, and privacy preserving.

[64] uses a centralized key management server to exchange keys between clients, data owners, and cloud storage. The procedures are: the client requests to cloud storage, cloud storage sends back encrypted data to clients, finally the client using its own private key decrypts the data. The proposal uses homomorphic encryption techniques (RSA and paillier algorithms) to ensure addition and

multiplication on homomorphic. Their work provides data integrity and confidentiality; and allows to maintain and upgrade databases to the cloud.

Work in [65] uses a two-factor authentication based on identity-based encryption by using a Universal Serial Bus (USB). Their system includes a private key generator (PKG) for issuing user' private keys, security device issuer (SDI) for providing user's security device, data sender and data receiver, and finally a cloud server for storing data and acting as proxy. A user joins to the system, gets private key from PKG and security device from SDI. After this, data sender encrypts the data according to the receiver's ID, then uploads in the cloud server, the cloud server re-encrypts according to data receiver; and then, data receiver gets data from the cloud server and decrypts by using its private key and security device. Their proposal provides confidentiality and recovery mechanism for stolen/lost security device.

In [66], authors provide cloud security by symmetric and asymmetric algorithms combination. In their model data encryption is done by a symmetric algorithm, and in parallel, symmetric key distribution is implemented by asymmetric algorithms. Their proposal provides data confidentiality and integrity.

[67] uses a centralized unified cloud authenticator as middleware between cloud service provider and users. This centralized authenticator provides user' credentials, hashing credentials, connection management and monitoring, authenticating user-cloud and finally service management for user logging. Proposal in [68] is a proxy re-encryption for cloud computing. Their proposal uses a centralized proxy as middleware between users and cloud service provider for key generation, access control, and re-encryption. The procedure is that a user gets keys from the proxy, encrypts his/her data and sends to proxy; then another user wants to access to this data and sends a request access to the proxy, the proxy checks the access list and then re-encrypts and passes the data to this new use.

[69] provides identity-based privacy-preserving auditing scheme capable of recovering messages due to a verification failure in the cloud system. Their proposed system has different components such as:

1. The cloud server for storing, serving stored data, and updating data on storage.
2. The user: client who stores his/her data on cloud.
3. A trusted third auditor: A centralized auditor which has communication with the cloud server to check data integrity and user validation.

Their work is robust, secure against forgery and replays attacks.

Work in [70], proposes two solutions:

1. The first provides a key management and authentication by means of elliptic-curve diffie-hellman and symmetric bivariate polynomial based secret sharing with and without trusted third party for cloud security.
2. The second is an extension on provided cloud security to handle multi-server cloud computing security.

Their proposals provide user-server mutual authentication, authentication and key recovery on multi-server cloud, high level of security against malicious activity (insider and outsider attackers) server side attack, and client attack.

[71] proposes the use of an electronic-ID (e-ID) that contains human and machine readable values, such as picture, finger prints, name, address, biometric, nationality, cryptography keys and digital certificate for multi-cloud identity management and authentication. Users-cloud authentication can be handled by e-ID due to their pre-loaded certificate and keys. The proposal provides authentication and identity management.

Authors in [72] propose a security architecture for multi-factor authentication in cloud systems. Their system components are: cloud service provider, cloud access management, cloud administrator, smart phone, and an email-id. In their authentication process, they use multi methods such as secret key, one-time password, and international mobile equipment identification (IMEI). Their workflow is as following: users register at cloud access management server (users give identification information such and email-id and IMEI), then users log into cloud access management system by their provided identifications, and finally cloud access management server may or may not give access to the users after checking their identifications and cloud resources security levels (high, medium, and low). Secret key factor authentication and arithmetic captcha expression is implemented for low cloud resources security. For medium one, proposal uses arithmetic captcha and one-time password. And finally the proposed architecture uses arithmetic captcha, one-password, and IMEI for high resources security authentication.

7.1.2. Access control solutions

Authors in [73], propose using online/offline attribute based encryption for data sharing. Their proposal provides confidentiality, collusion resistance, online/offline encryption, public ciphertext test, and security against chosen-ciphertext attacks. [74] proposes attribute based access control in mobile cloud computing using anonymous cipher-text attribute based encryption (CP-ABE) and match-then-decrypt technique. Their proposal includes attribute authority for generating system public parameters and keys, cloud service provider for managing, storing, and controlling access to encrypted data, data owner for defining access policies before uploading data, and finally mobile consumer to anonymously access the encrypted data on cloud.

[75] proposes a secure cloud computing architecture using a trusted central authority that provides secret and public parameters for distributed lower-layer domain authority, distributed domain authority for managing and generating public and secret parameter for their corresponding user's domain, cloud provides semi-trusted data storage and collaboration, data owner outsources data to cloud after encryption, and finally users that must validate some attributes to access data. Their proposal is based on a Cipher-text Policy Attribute Based Encryption (CP-ABE), Attribute Based Signature (ABS), and finally Hierarchical Attribute Based Encryption (HABE).

Work in [76] proposes an extensible access control framework (EACF) for cloud systems. Their proposal reduces unauthorized access chances and provides reliability for accessing authorized services in cloud.

In [77], attribute-based access control is proposed using a ciphertext-policy attribute based encryption and hierarchical access tree. This work eases the data owner to define the access control policy and provides efficient keys and user revocation.

[78] presents a multi-user access control policy for querying and accessing data on the cloud system. Their proposal has a database administrator that acts as proxy encryption; this administrator makes one round of encryption and then cloud service provider makes another round of encryption. The administrator is responsible for defining access policies to the encrypted database on cloud and cloud service provider makes the authorization check. In their proposal, a centralized key management authority is responsible for generating and giving keys to the cloud users, cloud database administrator, and cloud service providers.

Authors in [79] provide a secure storage and access control mechanism in cloud according to the storage's location. Their system has four components such as:

1. Cloud service providers: different service providers are responsible for storing data. Storage nodes might be in different region.
2. Cloud users: Users which are using cloud storage. Users must identify which region are allowed to store data and only the storage node in that specific region can process their data.
3. Region servers: These distributed servers are responsible for authenticating and controlling their corresponding storage nodes.

This proposal makes sure that cloud user' data storage and processes must be done only if user specified location validation. Authors propose a new transformable attribute based encryption for accessing to the cloud system.

Authors in [80] propose a role-based access control for cloud storage security. Their system components are:

1. Set of data owners who wants to store private data on cloud.
2. Sets of users who wants to access those private data on cloud.
3. Role manager that assigns roles to users and according to the roles (by comparing roles qualification) verifies or revokes users. Role manager and user authentication is assumed to be done before their communications.
4. Group administrator is a centralized trusted party to provide public parameters, roles, and keys for users.

Group administrator, role manager, and user communication is over secure channels. So, users got decryption keys from group administrator at the registration step, then users must show credentials to the role manager for illustrating their qualification to join that role, at the end users get access to ciphertext in cloud and decrypt text by means of the provided key. They propose a hierarchical role based encryption that prevents malicious cloud provider,

[81] presents a new key-policy attribute-based encryption (KP-ABE), where ciphertexts are tagged with a set of attributes and private keys. A well-defined hierarchical access control is applied to their system. This access control checks which user (according to his/her sets of attributes and private key) is able to decrypt which ciphertexts.

[82] proposes a key-aggregate cryptography system for data sharing in cloud. In their proposal, the data owner encrypts the message by means of the public-key and class (identifier of ciphertext). Therefore, ciphertexts are classified into different classes. Then secret keys (aggregator keys) for the different classes are extracted by data owner's master secret key. These aggregator keys must be sent to data receivers by a secure channel such as email. Then data receivers can decrypt data on the cloud by means of the key aggregators.

[83] proposes verifiable searchable encryption with aggregate keys in cloud storage. In their proposal, the search keys and verification tokens are aggregated into one key over a subset of an owner's documents set. Data users can easily access data by decrypting and verifying with the one single aggregator key. Their proposal provides scalability.

[84] presents a privacy-preserving access control in cloud systems named cloud mask. Their system architecture includes:

1. Document manager: It manages subscriptions and performs policy-based documents encryption.
2. Cloud data service: It stores encrypted documents.
3. Identity providers: They are responsible for issuing certified identity tokens and identity attributes for users.

Cloudmask is implemented by oblivious commitment based envelope protocols and broadcast group key management. Their proposal provides user's privacy by not disclosing user's identity attributes to document manager and cloud service data. All documents will be stored in shapes of sub-documents in cloud data service without disclosing the contents. And finally cloud data service provides restriction for accessing to these subdocuments to authorized users without knowing their identity attributes. Their proposal provides privacy and security in attribute-based access control.

7.1.3. Secure storage and data protection solutions

In [85], authors propose a cryptography strategy for securing distributed data storage on cloud by using alternative data distribution, secure efficient data distribution, and efficient data conflation algorithms. Their approach provides privacy, avoids malicious access and low computation time,

In [86], a secure storage on cloud propose. The proposal uses RSA for encrypting data and MD5 message digest (before storing data) for digital fingerprinting. Their work provides secure data storage on the cloud, making inaccessible the data to the attacker.

In [87], authors propose securing data storage on smart devices by means of a multi-cloud architecture. In their architecture, cloud users take the responsibility of data encryption instead of

cloud, users keep only one segment of data (last segment of data), users compressed data, split data segments, keep last segment on their device, encrypt other segments of data and finally encrypt the data distributed to multi-cloud to be stored. Their proposal provides privacy and it is a well-strategy for authentication. Authors in [88], propose two approaches to secure cloud computing. Their first approach is that cloud users keep their private and sensitive data secure on their secure region (locally) and the second approach is using a centralized secured trusted authority to store sensitive data and only is able to decrypt and re-encrypt user's data. Although, they do not describe how a secured region can be implemented locally. Authors in [89] implement a secure cloud storage by using a centralized cloud broker acting as access control (for reading and writing data on cloud) and a trusted center for auditing and monitoring (it raises alarms in case of any violation). Their proposal provides integrity, confidentiality, and freshness of data in cloud storage.

[90] provides cloud storage security in terms of remote data integrity checking and data dynamics. The system components are: clients who wants to store data on the cloud, cloud storage server that has high storage and computation capabilities managed by the cloud service provider, and a third party auditor for security checks on behalf of clients. The proposal uses merkle hash tree based with block tag authenticator for providing efficient data dynamics, bilinear aggregate signature to provide efficient third party auditor multi-task handling.

7.1.4. Malicious, intrusion and anomaly detection solutions

[91] provides different virtualized intrusion detection systems (IDS) solutions for cloud computing.. Authors in [92] analyses where the location of virtualized intrusion detection systems can be implemented. IDS can be in every virtual machine (VM) on all the devices, or in a centralized separated VM on cloud, or finally separated VMs on the gateways. They implement proxy virtualized IDS on gateways, so each gateway acts as proxy for IDS.

Work in [93] implements a hypervisor-based virtualized intrusion detection in the core network, for cloud system. Their proposal has 3 components and includes:

1. Controller node: It gathers and analyses data from endpoints in real-time. It also analyzes signatures of suspicious activity.
2. Endpoint nodes: They can be a physical system that contains a hypervisor or an API to the hypervisor. The API acts as middleware between hypervisor and the controller node to pass data to be analyzed.
3. Notification service: It gives alerts to the system that a signature of attack has been detected.

Their system works in the way that endpoints gather data from every virtual machine running in cloud from the hypervisor and passes it to a centralized controller node to be analyzed. If an attack signature is detected, the notification service provides an alarm to the system. Their proposal is able to detect denial of service attacks.

In [94], a software defined system (SDs) was proposed to decouple control and data plane. Their proposal uses 3 types of centralized controllers such as SDN controller for managing network, SDStore for controlling storage, and SDSec controller for handling encryption/decryption, DoS detection and defining access policies.

7.1.5. Cloud security solutions conclusion

Proposal	Authentication and key management	Access control and secure storage	Malicious and intrusion detection	F2C capable
[55]	✓			NO
[56]	✓			NO
[57]	✓			NO
[58]	✓			NO
[59]	✓			NO
[60]	✓			YES
[61]	✓			NO
[62]	✓	✓		NO
[63]	✓			NO
[64]	✓	✓		NO
[65]	✓	✓		NO
[66]	✓			NO
[67]	✓			NO
[68]	✓	✓		NO
[69]	✓	✓		NO
[70]	✓			YES
[71]	✓			NO
[72]	✓			NO
[73]		✓		NO
[74]	✓	✓		NO
[75]	✓	✓		YES
[76]		✓		NO
[77]	✓	✓		YES
[78]	✓	✓		NO
[79]		✓		YES
[80]	✓	✓		NO
[81]	✓	✓		YES
[82]		✓		YES
[83]	✓	✓		YES
[84]	✓	✓		YES
[85]		✓		NO
[86]		✓		YES
[87]		✓		YES
[88]		✓		YES
[89]		✓		NO
[90]	✓	✓		NO
[91]			✓	NO

[92]			✓	YES
[93]			✓	NO
[94]			✓	NO

Table 5. Cloud security solutions

Most of the reviewed proposals above are using some kind of centralized component such as: centralized authenticator, centralized certificate authority, centralized key manager, centralized server for handling access control, centralized attribute authority for handling access control, centralized component for generating public and private parameters, or centralized component for intrusion and anomaly detection. These components could act as a single point of failure, which means that if any of the mentioned centralized component gets attacked, compromised or even failed; the whole system can fail and or be compromised.

On the other hand, in some of the reviewed authentication proposals, the use of a smart card, USB, or e-ID for handling authentication were proposed, however, this type of solutions cannot be applied into the IoT devices, due to their low computational power, and even the main proposals' weakness is losing those smartcards, USB or e-id. Therefore, most of the cloud security solutions are not sophisticated enough to adapt and implement to the hierarchical F2C system due to F2C distributed nature. Moreover these cloud security solutions are not compatible with the low computational power of devices at the edge of the network in the F2C system.

Finally also, some of the mentioned proposals can be considered for the distributed F2C, because they use distributed components, for example for handling authentication, key management, access control, or intrusion detection; these types of proposals could be considered and applied in the F2C system for handling security between cloud and fog. But even in that case, of the reviewed proposal that are capable of being part of the fog-cloud security in F2C, these proposals still are not considering the whole F2C security architecture from IoT devices at the edge, fog, and finally the cloud all together. A distributed security architecture is a must for handling security from the edge to the cloud by means of security by design.

7.2 Existing fog layer security proposals

In this section, the existing security solutions in fog computing for providing authentication, key management, access control, malicious and intrusion detection are revisit and in sub section fog security solutions conclusion all reviewed proposals are analyzed. Most the existing fog security solutions are different to the cloud security solutions due to the distributed and low computational power devices in fog computing as illustrated in Table 6.

7.2.1. Authentication and key management solutions

[95] proposes secure publish-subscribe fog computing systems using elliptic curve cryptography. In their proposal, devices (IoTs) and the broker (Fog node) communication occurs in a secure way. Fog broker has key management, subscription, and publication module. Key management provides

public and private parameters, authentication is done by subscription, and finally publication encryption and delivering to subscribers are done by the publication module. Their proposal provides mutual authentication, integrity and confidentiality for the fog-IoT system by using distributed brokers (fog nodes) and using elliptic curve cryptography in publish/subscribe system.

Authors in [96] propose end-to-end security mobility-aware for IoT-cloud system using fog computing. In their work, smart gateways (Fog layer) provide device-users-cloud authentication and authorization remotely by using certificate-based datagram transport layer security (DTLS) handshakes, end-to-end security between end users and devices that is provided based on session resumption, and finally a robust secure mobility that is implemented by a secure inter-connection between smart gateways (Fogs). Their proposal provides confidentiality, integrity, mutual authentication, forward security, scalability and reliability.

[97] proposes an anonymous and secure data aggregation mechanism for fog systems. Their system model is a centralized system manager for generating public parameters and helping other components to generate public/private keys. Terminal devices are users and IoT devices and a fog node is the middleware between terminal devices and public cloud server for storing, communicating, controlling, configuring, measuring and managing terminal devices at the edge of the network. And finally a cloud service provider for storing and analyzing data that comes from fog nodes and terminal devices. Public key cryptography infrastructure is applied to the system and gives all components certificate and public/private keys. All IoT devices and users get authenticated and authorization from the cloud service provider to upload their data. After fog nodes and IoT devices are authenticated, the IoT devices encrypt data and upload the ciphertext to fog nodes. Fog nodes aggregate all received ciphertexts into one ciphertext and upload on cloud. Finally cloud service providers decrypts the ciphertext and gets plaintext. All terminal IoT devices are anonymous to fog nodes, and both of them in some conditions can be revoked

[98] implements an edge-fog mutual authentication scheme. It proposes that fog users get long-term master key from registration authority (cloud) in the registration phase. In parallel, the registration authority sends encrypted and signed registered fog users-IDs to fog servers to be stored. Therefore, fog users that arrive to fog servers can mutually authenticate by showing their IDs in an encrypted way. Their proposal uses symmetric encryption and hash function such as SHA-1 or SHA-256. The work in [99] uses two middle-wares, one between IoT and fog and other between fog and cloud. IoT-fog middleware uses session resumption, intermittent security using device IDs, Datagram transport layer security (DTLS), and pre-shared key for securely exchange data between IoT devices and fogs. Another middleware between fogs-cloud uses authentication scheme, bulk encryption, message authentication code, TLS or DTLS, and pre-shared key to provide secure fogs-cloud communication.

The proposal in [100] presents a variant of password authenticated key exchange protocol without password sharing for fog systems. In their proposal, a trusted authority provides public parameters in offline mode, two fog-users by using their wireless communication when they are proximate to each other, then they obtain a short-password by means of a key exchange algorithm (Diffie-Hellman key exchange), finally, the two proximate fog-users are authenticated to establish secure channel.

7.2.2. Access control solutions

Work in [101], presents a new privacy-preserving data aggregation for fog-IoT systems. Their system has:

1. A centralized trusted authority for bootstrapping, key distribution and management for all devices
2. Control center is a middleware between fogs and could for analyzing data.
3. Fog devices are between IoT devices and the control center to do some aggregation process.
4. IoT devices that have not enough computational power.

Their proposal implements privacy preserving data aggregation by homomorphic Paillier encryption, chines remainder theorem, and one-way hash chain techniques.

[102] presents secure fog-cloud communication provisioning by combining ciphertext policy attribute-based encryption key exchanges and digital techniques. A centralized key generator server generates and distributes keys for all the components in the system and cloud defines the access structure. Ciphertexts are attached with sets of attribute, then fog nodes can get a shared key for providing secure communication to cloud, if and only if, it satisfies the corresponding sets of attributes.

[103] provides attribute-based data sharing for fog computing. In their proposal, data owners assign sets of attributes to each file. Fog servers have a copy of the group attribute history list that contains historical attributes and used proxy re-encryption keys. Proxy re-encryption keys give re-encryption responsibility to fog nodes from data owner without disclosing data contents. Data users can access data if and only if it satisfies the sets of attributes. Their proposal uses key policy attribute based encryption and proxy re-encryption for providing access control and secure communication.

[104] provides access control based on ciphertext policy attribute-based encryption with outsourcing attribute updates for the fog system. In their proposal, fog nodes responsibilities are the generation of parts of ciphertext and decrypting parts of ciphertext, data owner defines access policies and structure and generates part of the ciphertext then send to fog nodes, a centralized key authority for providing public and private parameters and even attributes and keys updating, Cloud service provider that provides data storage, and finally data user that must satisfy sets of attributes to access data on cloud or changing data on fog.

Work in [105] presents a privacy preserving public-subscribe scheme in fog computing. In their work, fog nodes act as brokers between users. Brokers receive user's notification event then analyze the attributes and data to match notification and users. Their proposal is using notification event production, content-based event privacy protection, and event matching. It also uses the U-Apriori algorithm to mine top-K items such as attributes form uncertain datasets, exponential and Laplace mechanism for privacy preserving, and fog nodes (brokers) uses the mined top-K items to match publisher-subscriber. [106] presents privacy-preserving fine-grained access control by using user-level key management and update mechanism for fog-cloud computing systems. User-level

key management provides data owners to set a number of keys regardless of the number of outsourced data files. It provides user-level public keys updating face in cloud without knowledge of the private keys. The proposal provides fog devices to deduplicate encrypted data within the same domain of data owners while preserving privacy and reduces communication overhead and computation overhead.

Authors in [107] proposes revocable data sharing by using distributed attribute authorities for controlling access. Their proposal has components such as:

1. Centralized certificate authority: It is responsible for providing unique identifier for each users and each attribute authority.
2. Distributed attribute authorities: They provide issuing, revoking, and updating fog user's attributes. They are responsible for generating public and private attribute keys.
3. Data owners: They define their access control policies by using attribute authority and encrypted data by their policies.
4. Cloud server: It stores data and provides the access mechanism for the users.
5. Data users: They can decrypt data and access data after verifying access attributes control policies.

Their proposal provides multi-authority ciphertext policy attribute-based encryption and distributed access control for distributed fogs.

[108] proposes secure a data access control for fog-IoT systems. Their proposal system has: a centralized attribute authority for generating public and private parameters, a cloud service provider to provide data storage and signature verification for accepting or updating ciphertext, fog nodes that are responsible for generating part of ciphertext and upload whole ciphetrtext to the cloud service provider, data owner who defines the access and update policies for the gathered IoT's data to generate ciphertext with fog nodes, and finally users that must satisfy the access policy by sets of their attributes for accessing data. Their proposal is based on ciphertext policy attributed based encryption and attribute based signature.

7.2.3. Secure storage and data protection solutions

[109] proposes using two databases (big data and decoy big data storages) for securing fog-cloud system. A decoy database is created by using decoy techniques and fog computing. Decoy database is the fake database which has fake data, therefore, gives attacker's believes that they access user's database. All authorized and unauthorized users get access first into the decoy database (located in fog) then only authorized users get authentication to access real databases (located in cloud) by security challenges verification. They use user profile algorithm, decoy database algorithm, key exchange (elliptic curve diffie-hellman) algorithm, and bilinear pairing cryptography to implement their system.

[110] presents secure storage by integration of fog and cloud computing. In their system, IoT devices produce data to be stored in cloud, first these data go to edge servers (Fog) to be aggregated and transformed into a unified framework to be processed. Then edge servers send aggregated and unified data to a centralized proxy server between fog and cloud. Proxy server constructs an index

structure and encrypts data by maintaining security and searchability. Then, encrypted data is stored in cloud, and users can access to data by building a trapdoor with the help of the proxy server and users send trapdoor to cloud. Cloud searches encrypted data according to the trapdoor and then sends encrypted data to users. Finally, users with their secret keys decrypt the data. The proxy server uses ID-AVL tree to provide search efficiency over encrypted data.

7.2.4. Malicious, intrusion and anomaly detection solutions

A hierarchical anomaly detection is proposed in [111]. In their proposal, IoT-devices with low computational power send data to the fog layer, fog layer implements anomaly detection in a distributed way for end devices, and finally all aggregated data is sent to a centralized cloud to be stored and cloud do centralize anomaly detection for fog layers. Their proposal uses hyper ellipsoidal clustering algorithm for IoT devices clustering and an anomaly scoring mechanism (Ellipsoidal neighborhood outlier factor) for detecting anomalies in clusters. It also provides detecting location of anomalies, hierarchical anomaly detection for all layers, latency and energy consumption reduction.

[112] presents a mechanism for identifying malicious edge devices in fog computing. In their proposal, a framework on top of the fog layer is implemented by secure load balancer. Secure load balancer includes intrusion detection mechanism and two-stage Markov model. This Secure load balancer watches over the IoT devices and categorize them into legitimate devices, sensitive devices, under-attack devices, and hacked-devices, by means of the two-stage Markov model and move the edge devices monitoring into a virtual honeypot device. Then, the virtual honeypot device monitors malicious edge devices and detect them by intrusion detection mechanism with efficient rates according to IoT device's classification.

Work in [113] proposes the use of a hierarchical defense mechanism against distributed denial of services (DDoS) attacks. In their proposal, fog layer consists of virtual local nodes, and the cloud layer has a virtual central node. Local nodes have anti-spoofing module and detection engine mechanisms. A copy of packets in network are read by anti-spoofing modules to validate source IP address of outgoing packets. In case of spoofing detection, packet are dropped. Otherwise packets are move for further classification continuously. Detection mechanism observes data periodically to find out abnormal traffic. Local nodes periodically aggregate statistical data from their area and send it for further investigation to the central node in cloud. Then central node classifies it and detects abnormal behavior by further analyzing. Their proposal provides DDoS detection locally and on the cloud.

7.2.5. Fog security solutions conclusion

Proposal	Authentication and key management	Access control and secure storage	Malicious and intrusion detection	F2C capable
[95]	✓			YES
[96]	✓			YES
[97]	✓			YES
[98]	✓			NO
[99]	✓			NO
[100]	✓			YES
[101]	✓	✓		NO
[102]	✓	✓		NO
[103]	✓	✓		YES
[104]	✓	✓		NO
[105]		✓		NO
[106]	✓	✓		YES
[107]	✓	✓		YES
[108]	✓	✓		NO
[109]	✓	✓		NO
[110]		✓		NO
[111]			✓	YES
[112]			✓	YES
[113]			✓	YES

Table 6. Fog security solutions

Some of the fog security solutions mostly rely on the cloud for providing authentication, key management, access control, and anomaly detection where any failure on the cloud can affect the whole system. Also, some of reviewed proposals are using a centralized component such as: a centralized proxy, centralized CA, centralized middleware, centralized authenticator, centralized key generation center, centralized attribute authority, centralized SDN controller, or centralized cloud for providing authentication, key management, access control, and anomaly detection. These centralized components are acting as a bottleneck to the whole system due to the single point of failure they represent, and even due to scalability issues. Any compromised, attack, or failure in those centralized components for handling security could cause the whole system failure.

On the other hand, in some of the proposals, the use of a smartcard or USB were proposed to provide authentication, but the smartcard-based authentication is so vulnerable due to missing or stolen cards.

Also, it is worth to mention that most of the reviewed cloud security solutions and even existing cloud solutions cannot be applied to the fog systems due to fog devices computational power are so lower than the cloud.

Finally, some of the reviewed proposals are using distributed authenticators, key managers, access controls, or distributed nodes for anomaly detection. These type of proposals could be

implemented in fog layers of the F2C system, although, their proposals are not considering the whole hierarchical F2C system from cloud to the edge of the network.

7.3 Existing IoT layer security proposals

In this sub-section, the most potential security solutions in IoT infrastructure for providing authentication, key management, access control, and anomaly and intrusion detection are reviewed and in IoT security solutions conclusion sub-section all reviewed proposals are analyzed as illustrated in Table 7.

7.3.1. Authentication and key management solutions

Locally centralized and globally distributed authentication and authorization is proposed in [114] for IoT devices. Their work uses Auth (open source software java in github.com/iotauth) for fog-IoT local authentication and authorization while handling trust with other distributed Auths globally. Their system works in this way: devices register, then Auth takes credential of registered devices and their access policies locally in its database. Session keys are distributed to registered devices to provide devices authorization for specific activities. Auths uses HTTPS for global communication. Then, if a device wants to communicate with another device, their corresponding Auth provides authentication and authorization between the two devices in different areas. Their proposal provides scalability, resilience, and distributed trust.

[115] presents secure data sharing and searching for a cloud-edge-IoT system. In their work, distributed edge servers are semi-trusted to provide secure data sharing and searching, and all edge servers act as secret key generators to distribute and manage keys for the IoT devices and themselves. All users register at edge servers by means of the username and the password share data, downloading, searching and retrieving it. Key generators in edge servers provide two types of secret key encryption and public key encryption. So, the user provides his/her username and password in a nearby edge server, then the user sends data to the edge server, and finally the edge server encrypts and signs the data to be uploaded in cloud storage. For data searching and sharing, users must verify their username, password and signature; then edge servers decrypt data for them.

[116] proposes the use of a trusted network edge device (NED) as a middleware between IoT devices and cloud. A centralized NED is a local network or cloud-based solution that acts as a proxy for IoT devices. The centralized NED can provide authentication, bandwidth control, logging, malware detection, IP-tables firewall, re-encryption, and virtual private network for IoT devices.

[117] presents a smart-card based authentication strategy for cloud-IoT system. Their proposal uses distributed cloud servers and a centralized controller server for handling user's authentication.

Proposal is protecting system against user anonymity problem, off-line password guessing attack, insider attack, and user impersonation attack.

Work in [118] presents a smart-card authentication mechanism for users in IoT systems. Their proposal has the next components, such as: users, authentication server, and IoT server. Users log into the IoT server by presenting their smartcard, then authentication server verifies IoT-server and users. Users and IoT-server request to authentication server to be registered, and then authentication server gives a smartcard to user and required values to IoT-server for logging and authenticating users. When the user presents his/her smartcard to the IoT server, authentication server performs verification and all components such as IoT-server, user, and authentication server generate session key.

A secure IoT architecture is designed for end-to-end security in [119]. Security architecture includes IoT-devices with low computational power, IoT-broker by handling multiple communication protocols and proxies, IoT-application to provide services to users, and IoT certificate authority (CA) for generating certificates for all components. IoT- CA generates and issues certificates for all verified components with their attributes, IoT devices and IoT broker use HTTP; IoT broker and IoT device use constrained application protocol (CoAP). In their system, attribute exchange occurs in a secure way by attribute-based encryption, moreover, HTTP and CaAP uses TLS or DTLS to provide secure communication.

[120] presents a scalable key establishment and management for IoT devices. In their system architecture, a centralized trusted anchor is responsible for generating and providing keys for the clients; resource servers are constrained devices that produce information, and client wants to get resource server's information. The work proposes two symmetric approaches, first one uses TLS/DTLS and second one uses DTLS/TLS/Trusted third party (TTP).

An efficient collaborative host identity protocol (HIP-based) key establishment is proposed in [121] for secure communication between IoT constrained devices. Their work has the next components:

1- Constrained devices that are not capable of key generation and cryptographic implementation
2- Proxy nodes are nearby devices and able to handle cryptographic' constrained devices need. All distributed proxy nodes have secure communication by using TLS or IPsec.

Therefore, when two constrained devices want to establish a secure communication, proxy nodes nearby them will handle key and secure communication establishment through a secure inter-communication between proxy nodes.

[122] proposes an IoT devices and gateway authentication mechanism by using a conceptually centralized software defined networking (SDN) controller, which is physically distributed at the edge, and also an identity-based authentication. In their system, they assume all the components have IPv6 addresses and are capable of TCP-IP protocol; then the key establishment is done by elliptic curve cryptography. The authentication is done by:

1- SDN controller assigns virtual identity to IoT devices and keys for gateway and nodes.
2- Gateway sends its ID with public key to controller.
3- Controller sends back to the gateway the certificate signed that can be used for gateway-controller authentication.
4- Node send its identity encrypted by controller's public key to the gateway.
5- Gateway checks preliminary the ID and then forward the ID to controller.
6- SDN Controller decrypts the messages and checks if node's ID exists in its database.
7- If ID exists, then controller generates IPv6 and public key for the node and encrypts by means of the gateway's public key.
8- Gateway receives the messages and decrypts and stores the node's IPv6 and node's public key.
9- Gateway sends messages to the node with node's IPv6, node's public key, and gateway's public key encrypted by node's private key.
10- The node receives the message and then decrypts and stores the gateway's public keys.
11- Node send IPv6 signed by the gateway's public key, the gateway receives and checks against the local database and then replies encrypted by node's public key.
12- For mutual authentication, nodes send again message encrypted by the gateway's public key, then gateway decrypts and validates.
13- Gateway-node are mutually authenticated.

Secure end-to-end key establishment for constrained IoT devices is proposed in [123]. Their system components are:

3- High constrained IoT devices that are not capable of cryptographic implementation.
4- Less constrained devices that can implement cryptographic algorithms and key management.
5- Devices without any constrained which act as remote servers or workstations.

In their proposal, high constrained devices get help of nearby trustable less constrained device for providing keys to establish secure communication with remote server. Therefore, distributed trustable less constrained devices act as middleware between high constrained devices and remote server to handle key management and secure communication establishment.

An IoT-device and cloud server authentication mechanism is proposed in [124]. Their proposal components are:

1- IoT-devices are collecting and gathering data to be analyzed and stored.
2- Local processing unit which acts as a mobile-gateway for receiving, aggregating, and passing data to cloud server.
3- Cloud server is a secure trustable server for storing and processing collected data.

A trustable centralized public key generator provides public and private parameters and keys for all three components in a secure channel. Cloud server and local processing unit authenticate and

provide secure channel at first step, then IoT devices and local processing unit authentication process occurs. In the IoT devices and local processing unit authentication phase, the local unit first checks IoT device parameters (when it receives IoT device parameters) with cloud server through a secure and authenticated channel. Their work uses crypto-primitives for implementing the proposed security achievement.

[125] proposes a secure authentication and access mechanism for IoT devices. The proposed work uses low-power radio wakes up radio for changing IoT device from passive to active state, symmetric encryption key for generating session keys and message authentication code (MAC) secret key for data authenticity. In their proposal, wake up radio in receiver node receives a session token (only valid for one usage) from main radio in receiver before data transmission. Sender satisfies the authentication process by sending the ID to receiver wake up; then the token is passed for validation and receiver wake up sends parameters to receiver main radio for generating hashed value of group key communication. At the end, sender sends MAC with data to be checked in the receiver main radio, if its valid, main radio sends ACK.

[126] [127] propose using hybrid cryptography (symmetric and asymmetric encryption together) to provide IoT device security.

[128] proposes a partial key distribution and authentication mechanism for IoT devices. Their work uses a centralized certificate authority to generate and send certificates for all the components such as sensors, cluster-heads and base station. It provides inter and intra sensors secure communication and authentication in different clusters. Each cluster head has secure communication with base station, when sensors send their credential and identity to the cluster-head to be checked, cluster-head sends those parameters to the base station in a secure channel for validation; if valid, then base station selects a list of partial keys and sends them in a secure way to their corresponding cluster-head, then the cluster-head disseminates the encrypts the list to all its members. Sensors in the same cluster can have secure communication and authentication due to their partial keys; and sensors in the different clusters can establish authentication and secure communication through their cluster-heads and base stations.

A remote security management server is proposed in [129] to handle distributed IoT-devices security. The proposed security management server provides identity management, authentication, access control, secure storage, privacy, crypto module, key management and distribution, secure channel, firmware update module, trusted remote control module, security policy, vulnerability assessment, monitoring, intrusion detection, and intrusion response. In their workflow, an IoT device starts to register and gets identification and authentication to the gateway through the security management server.

[130] proposes a lightweight datagram transport layer security (DTLS) for IoT devices by adding a centralized trusted party for handling pre-shared keys. The proposal uses trusted third party to exchange pre-shared key and the packet transformation in DTLS compressed by IP Header compression and Next Header compression schemes. First, TTP and server get key agreement then TTP and client share mutual key for client's credential authentication to ensure secure client-server communication. Client sends request to the server, the server compares authentication key for

validation, if it matches, the server continues the process, and here the client hello packet is compressed by mentioned mechanisms.

[131] presents an efficient authentication mechanism for IoT devices by combining elliptic curve Qu-Vanstone implicit certificate with datagram transport layer security. Their system uses a centralized certificate authority in cloud for generating and providing certificates, public and private parameters and keys for IoT devices; then IoT devices can authenticate with gateway, IoT devices can authenticate each other in the same cluster or from different clusters.

[132] proposes to the integration of SDN-IoT system and uses blockchain to provide security to the whole system. They analyze SDN approach for IoT security and then apply blockchain to establish trust in the whole system. Blockchain is a mechanism that uses cryptographic hash and eliminate single point central verification authority failure.

[133] provides authentication and secure communication for IoT devices by combining user's passwords, hash values, and timestamp. The proposal uses passwords and hash values for authenticating devices based on time that integrate public key algorithm, hash value, and password. Timestamp in their proposal, varies each session and user's password with hash value operations. They insert session key and public key to transmit different values in each session.

A pervasive authentication protocol and key establishment is proposed in [134] for wireless sensor networks and IoT systems. A centralized certificate authority (CA) provides X.509 certificate for end-users and distributed cluster-heads (CHs) by datagram transport layer (DTLS) for providing end-to-end secure communication. After cluster-heads are authenticated by the centralized CA, they can act as distributed CA for their clusters to issue certificates for their child's nodes. Therefore, users that want to access to the information provided by sensor, first must establish DTLS with its CH, authenticate themselves with the CH according to the obtained certificate from the centralized CA, then CH issues a certificate for authenticating and establishing secure communication between its own child and users. The sensors in different clusters can have secure communication through their corresponding CHs secure communication. Their proposal uses IPv6 as identity, security scheme provided by MAC layers in IEEE802.15.4, DTLS based on elliptic curve cryptography, RSA and X.509 certificate.

[135] proposes a mutual authentication scheme between IoT devices and cloud servers by using elliptic curve cryptography and hypertext transfer protocol (HTTP) cookies. In their proposal, IoT devices register themselves with the cloud server and the server stores cookies on the IoT devices at registration process, IoT devices send a logging request and then IoT devices and server are mutually authenticated by using elliptic curve cryptography (ECC).

A lightweight collaborative key establishment is proposed in [136]. In their system architecture, there are highly constrained resources that cannot perform needed cryptography procedures and less constrained devices that can act as proxies in a distributed fashion for high constrained devices. The high constrained devices offloaded their cryptography needs, such as key establishment and authentication to proxies (less constrained devices). First high constrained devices and proxies agree on session keys for establishing a secure communication, then proxies can do encryption and authentication for those constrained devices.

An authentication and access control mechanism is presented in [137] based on elliptic curve cryptosystem (ECC), key establishment and role-based access control mechanisms. In their system architecture, local registration authorities (RAs) provides registration and assign IDs to their corresponding nearby IoT devices. The process is as following. Users request to get access to an IoT device, the IoT device sends to the RA an authentication requests, RA requests user's ID from user, user sends it with cloud server information, RA verifies user' cloud server information and sends ID verification request to cloud server, cloud server sends user challenge to check validation, if user answers the challenge correctly, then cloud server sends to RA that ID is ok, finally RA informs to the IoT device by user ID and issues session key to the user.

7.3.2. Access control solutions

[138] proposes the use of conditional identity-based broadcast proxy re-encryption to provide security in an IoT-cloud system for data collection, access and storage. Their encryption method uses a centralized key generator to provide private keys for all users and devices.

A certificateless searchable public key encryption scheme is proposed for IoT in [139]. Their proposal system components are:

1. Key generation center: A centralized key generation center is generating keys and generating receiver's and server's partial-private-keys.
2. Data owner: Data owner encrypts data and index of keywords in data by use of the public key of receiver and server.
3. Receiver: Receivers are data users who get partial private key from the key generation center and then receivers generate trapdoor of keywords for searching and send it to the cloud server.
4. Cloud server: It gets partial private key from key the generation center. It provides computing, storing, and searching for users.

[140] proposes using proxy re-encryption to provide IoT devices security. In their proposal, IoT-devices perform an encryption and provide "n" re-encryption keys, then send keys to the proxy server. Their proposal uses attribute based encryption. Therefore, user A wants to share data, user A re-encrypts data and passes to the proxy server, the proxy server generates ciphertext that allows user B to decipher text, then user B wants to access data's user A, user A and user B must have the same attribute to get decryption keys from the proxy server, and finally user B decrypts data and accesses it.

A collaborative key-policy attribute-based encryption (KP-ABE) is proposed for an IoT-cloud system in [141]. The proposed work has different system components including:

1. IoT devices (constrained device): They must send sensitive data while they are not capable of handling KP-ABE.
2. Unconstrained devices: They are able to handle KP-ABE operation. These devices might be servers or devices belonging to the same local infrastructure.

3. Remote servers (cloud): They are able to store received encrypted data from all type of devices and make available for any request from users due to their high computational and storage power.

In their proposal, unconstrained devices act as assistants for unconstrained devices for handling encryption and authentication data to be send to remote servers for storing. The work uses access tree, bilinear maps, and KP-ABE algorithms.

A lightweight no-pairing attribute-based encryption implemented by elliptic curve cryptography for IoT security and privacy is presented in [142]. Their work provides security based on elliptic curve computational Diffie-Hellman problem, more efficient than other attribute based encryption proposals and with low cost computational.

A RSA-based ciphertext-policy attribute based encryption with AND-gate access structure mechanism is proposed in [143] for IoT devices. Their proposal uses a high-level security in terms of key generation, encryption, decryption, and access control mechanism with less secret key size than other proposal and ciphertext without using bilinear maps.

7.3.3. Malicious, intrusion and anomaly detection solutions

[144] presents an integrated mechanism of software defined network (SDN) with IoT to defend against malicious activities and attacks. Their system use an open-flow protocol and Opflex to define policy security rules in distributed way for IoT devices and the centralized SDN controller to establish secure communication between distributed clusters and controlling open-flow messages and all the system components. If controller suspects about a malicious IoT device according to the device information, it revokes that device and the device's flow previously predefined in open-flow.

[145] provides security in IoT systems by merging a software-defined network (SDN) into the architecture. Their proposal architecture components are:

1. SDN controller: It provides IoT-controller and the SDN objects communication, routing algorithms, management and monitoring of traffic, and provides authentication and security policies.
2. IoT-controller: It receives connection's requests from IoT agents. IoT-controller decisions making is under SDN-controller supervision.
3. IoT-agents: They are distributed devices responsible for gathering, collecting, analyzing, and capturing data from the environment.

[146] presents flow-based traffic analysis for IoT devices at the edge of the network by using software defined networking (SDN) controllers. Their proposal implements distributed SDN controllers at the edge of the network where open-flow messages are transferred into them for traffic analysis to detect abnormal behavior or malicious attackers. It mitigates denial of service (DoS) attacks.

[147] proposes a software-defined network (SDN) mechanism for providing IoT-security. Their proposal has three components:

1- SDN edge nodes (fog nodes) are distributed to handle services at the edge of the network and ease data processing, analyzing, and storage by open-flow switch and virtual machines running different services.
2- SDN controller is a middle-ware between fog nodes and end-to-end security application to ease secure communication in north bound interface communication.
3- End-To-End security application has three components:
    a. Collector that gathers the data and flow information from SDN-controller.
    b. Anomaly detection module that provides analysis of the gathered data for detecting anomalies.
    c. Anomaly mitigation module for neutralizing identified attacks.

[148] proposes a new SDN approach for providing security in IoT systems. The proposed work uses distributed SDN controllers above the SDN-IoT gateways with specific different roles includes:

1- Intrusion controller: It handles traffic monitoring, managing each flow's rules, intrusion detection and mitigation, secure routing (some keys might be needed from key controller).
2- Key controller: It implements key management (for both symmetric and asymmetric), key distribution, handling needed cryptographic operations, and key distribution for intrusion controller (then a communication between intrusion controller and key controller is needed).
3- Crypto controller: It provides integrity, confidentiality, privacy, authentication, and identity management. Most of the services provided by crypto controller needs keys to establish security and encryption, then key controller and crypto controller needs inter communication.

An Intrusion detection mechanism is proposed in [149] for IoT devices. The mechanism uses distributed devices capable of analyzing network traffic that act as middleware between IoT devices and cloud to detect intruders. These devices analyze traffic locally generated by IoT devices and send gathered traffic to an embedded software analyzer for analyzing domain name system (DNS) flows for searching indicators of DNS anomalies.

An IoT security framework is proposed in [150] for anomaly detection and attacks mitigation. Their framework includes end-nodes to provide and propagate information, network layer for information transmission from/to end-nodes, service layer as a middleware between the application layer and the network layer, and finally an application layer for providing user's services. Their framework recognizes threat models and vulnerabilities in each layer and proposes using anomaly behavioral analysis intrusion detection mechanism for protecting end-nodes.

An anomaly detection system is proposed for monitoring IoT devices in [151]. They use a controller near edge devices to receive data collected by edge devices, then it checks the data with previous sensor's history according to values such as data type, data source and destination, their current time and location, restriction on values, business rules of system, user's behavior, edge device's geo location and their movements, and internal consistency and databases conformity to detect any anomalies.

7.3.4. IoT security solutions conclusion

Proposal	Authentication and key management	Access control and secure storage	Malicious and intrusion detection	F2C capable
[114]	✓			YES
[115]	✓	✓		YES
[116]	✓		✓	NO
[117]	✓			NO
[118]	✓			NO
[119]	✓			NO
[120]	✓			NO
[121]	✓			YES
[122]	✓			YES
[123]	✓			YES
[124]	✓			NO
[125]	✓	✓		YES
[126]	✓			YES
[127]	✓			YES
[128]	✓	✓		YES
[129]	✓			NO
[130]	✓			NO
[131]	✓			NO
[132]	✓			YES
[133]	✓			YES
[134]	✓			YES
[135]	✓			YES
[136]	✓			YES
[137]	✓			YES
[138]		✓		NO
[139]	✓	✓		NO
[140]	✓	✓		NO
[141]	✓	✓		YES
[142]	✓	✓		NO
[143]	✓	✓		YES
[144]			✓	NO
[145]	✓		✓	NO
[146]			✓	YES
[147]			✓	YES
[148]	✓		✓	YES
[149]			✓	YES
[150]			✓	YES
[151]			✓	YES

Table 7. IoT security solutions

In the reviewed IoT security solutions, some of them are using a centralized component such as a centralized authenticator, centralized CA, centralized trusted third party, centralized remote security server, centralized proxy or proxy server, centralized attribute authority, centralized IoT

controller or a centralized SDN controller for providing authentication, key management, access control, and anomaly detection. In these proposals, any failure, attack, or compromise to the centralized component can cause failure in the whole system. Another bottleneck for these proposals is the scalability, due to the difficulty for handling the security of that huge amount of IoT devices at the edge of the network by a single component.

Also, some of the proposals use a smartcard or USB that are so vulnerable to card missing/stolen issues

There are other reviewed IoT security solution that are using distributed components for authentication, key management, access control, and intrusion detection. But these proposals do not consider the whole F2C architecture from bottom to up (IoT-Fog-Cloud) and most of the proposals has issues with balancing quality of service QoS versus security in their system.

Finally, it is worth to mention that IoT devices have low computational power without capabilities to provide security on their own. Therefore, some other reviewed proposals suggest to use distributed fog nodes for IoT devices' security provisioning. Although, fog nodes have other responsibilities such as service allocation and execution and then running security algorithms can consume huge amount of fog node's computational power and effect on QoS.

According to all reviewed proposals in the different layers (IoT-fog-cloud), in the next chapter, a novel security architecture is proposed to handle authentication, key management, and access control in a distributed fashion. The novel security architecture is capable of overcoming the reviewed and analyzed proposals weaknesses (mentioned in the conclusions subsection for each layer) in terms of avoiding the single point of failure, considering the whole hierarchical F2C architecture by means of security by design, and finally also providing balance between security provisioning versus QoS.

Chapter.8 F2C Distributed Security Architecture: Proposal

In this section, a novel security architecture which can properly provide security in the F2C continuum system will be proposed.

8.1 Distributed Security Architecture

The F2C system has a hierarchical and distributed nature. Therefore, if we want to provide security by design in the F2C, a new security architecture must be designed to provide security from scratch into the system. In this thesis, a novel distributed security architecture is proposed (already published in [152] [153]) to provide hierarchical and distributed security in the F2C system. The proposed security architecture has two components such as the F2C controller at the cloud and distributed control-area-units (CAUs) at the edge of the network, closer to the fog nodes and users. Figure 6 illustrates the decoupled security architecture. The components' description and functionalities are as following:

1. F2C controller: This component is located at the cloud. The F2C controller acts as a master and centralized certificate authority to the distributed CAUs at the edge of the network. In the initialization of the system, the F2C controller provides all distributed CAUs public and private parameters and makes mutual authentication. Then, all the CAUs after authentication, get authorization to provide security functionalities for their corresponding areas.
2. Control-Area-Units (CAUs): All the distributed CAUs at the edge of the network get mutual authentication for having secure inter-communication. After authentication and getting authorization from the F2C controller, all the CAUs at the edge of the network must provide security such as authentication, key management, authorization, access control, secure channel establishment, and etc. for the devices in their corresponding areas.

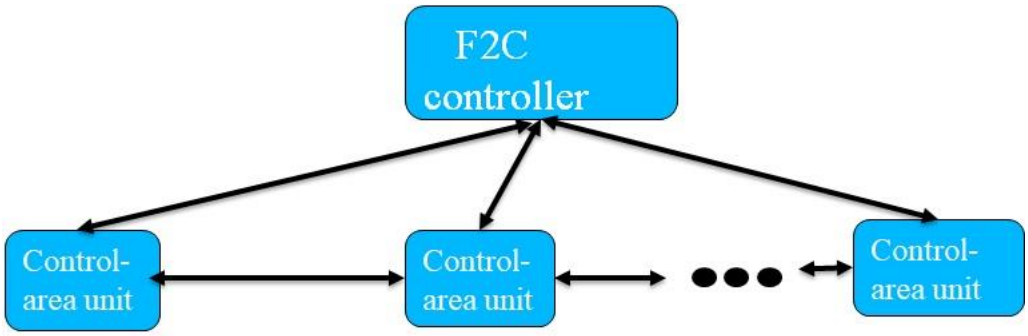

Figure 6. Distributed Security Architecture

This security architecture is applied into the F2C system as illustrated in Figure 7. In this architecture, the components in different layers are:

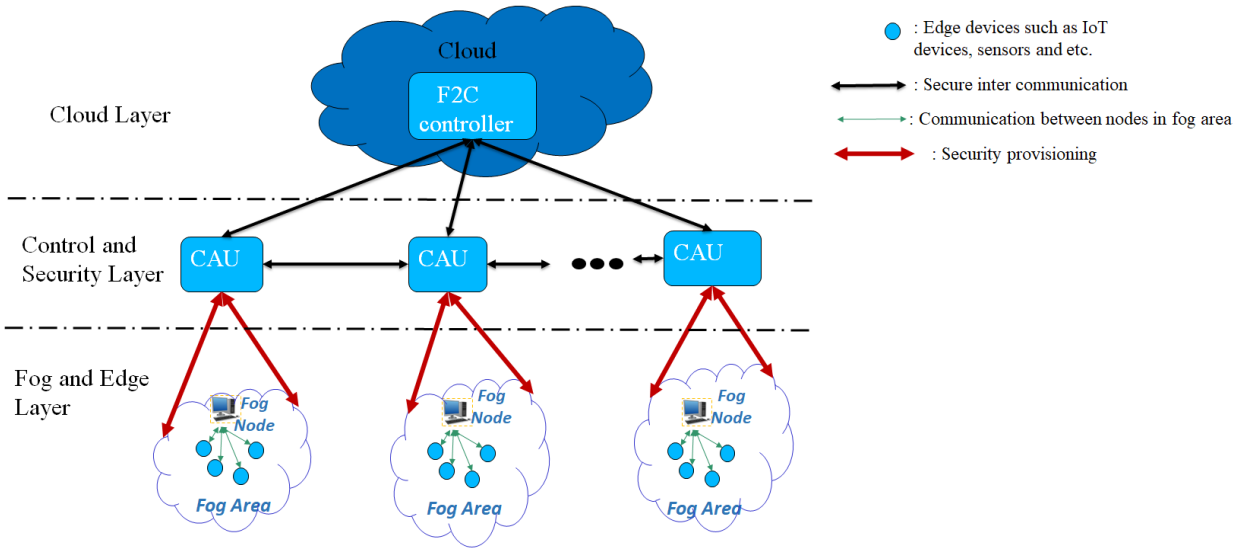

Figure 7. Distributed security architecture in F2C system

Cloud Layer:

- Cloud: The centralized cloud is responsible for huge data processing, aggregating, filtering and finally storing the historical data for the F2C system. The cloud has powerfull devices and huge data centers for doing all these functionalities. However, the cloud in the F2C system cannot provide security for the huge number of distributed devices and located far from devices. Therefore, in our architecture a centralized F2C controller is located at cloud to provide authentication and authorization for chosen computational power devices which are called control-area-unit (CAUs).
- F2C controller: As mentioned above, it acts as master to distributed CAUs and provides authentication and authorization to them. In an initialization phase, the F2C controller gives authorization to CAUs for security provision for devices at the edge.

Fog and Edge Layer:

- Fog Area: Fog areas can be considered as distributed clusters that include a cluster-head (fog node) and slave nodes such as IoT devices. Fog area is located at the edge of the network and includes devices with computational power such as fog nodes to provide data processing, filtering, aggregating, and storing data before sending to the cloud for more processing. In the fog area, there are also devices with low-computational power such as IoT devices, sensors, actuators, and etc. The low-computational power devices only can gather and send information to the fog nodes for more processing. In this case, for huge numbers of distributed devices in the different fog areas, a distributed security provisioning is a need. The distributed CAUs located at the edge of the network nearby the fog areas. Therefore, CAUs can provide security functionalities in their corresponding areas.

Control and Security Layer:

- Control-Area-Units (CAUs): As mentioned above, the distributed CAUs after authentication with the F2C controller and themselves (secure inter-communication), get authorization to provide distributed security functionalities for their corresponding fog areas. CAUs are cable to provide security for both, low-computational power and with-computational power devices.

It is worth to mention some characteristics of the proposed security architecture here:

1. All the CAUs after getting authorization from the F2C controller work independently from it. Then if the F2C controller is down or fails, all distributed CAUs can work properly without any compromising.
2. In the distributed CAUs, if one of the CAUs or more is down, other CAUs can work properly without compromising.
3. In the fog area, if the corresponding CAU is down or fails, the fog nodes can communicate directly to cloud (F2C controller) to discover nearby CAU for security provisioning till new proper CAU will be selected.
4. In the security architecture, the distributed CAUs have secure inter-communication. Therefore, security architecture handle secure mobility and handover between fog areas.

In the secure F2C architecture, all the communication must be done through CAUs and the F2C controller from the edge of the network to the cloud and even all the node-to-node, node-to-fog and fog-to-fog communications must be done through CAUs. The CAUs act as distributed security controllers at the edge of the network and have the all control over the distributed nodes for providing security functionalities such as authentication, key distribution and management, access control, encryption and decryption for the low-computational power devices, secure channel establishment, etc. In the following, the two most important and potential communications steps are described in simple fashion:

• Fog-to-Cloud communication: First, all types of devices such as IoT devices, sensors, fog nodes, etc. are registered at cloud. The communication between registered edge devices is controlled by the fog node, their corresponding CAU and the F2C controller for providing secure communication from edge to the cloud in combined the F2C architecture as illustrated in Figure 8. For example, for providing secure communication in fog area 1 to cloud, the device will be checked by fog node 1, fog node 1 checked by CAU 1, and CAU 1 will be checked by the F2C controller. Therefore, after all authentication, authorization, access control and secure channel establishment between all components, the device can securely communicate hierarchically from edge to the cloud in the F2C system. In the figure, all the black lines are security functionalities and security information to provide secure communication between device and cloud (blue line).

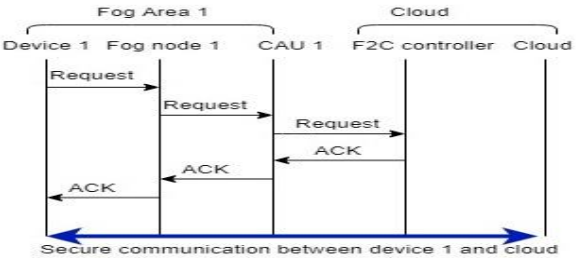

Figure 8. Number: Fog-to-Cloud communication

• Fog-to-Fog communication: All the CAUs in the initialization phase are authenticated and got authorization from the F2C controller, therefore, all CAUs have secure inter-communication after the initialization phase. Then, two fog nodes that are registered in different CAUs or two nodes in different fog areas establish a secure communication through their distinct CAUs. For example, one device in fog area 1 wants to establish a communication with a device in fog area 2. As shown in Figure 9, device 1 takes approval from fog node 1, and fog node 1 from CAU 1 under authorization of the F2C controller. In parallel, similar procedure takes place for the device in fog area 2. Then, after authentication, authorization, access control and secure channel establishment all included components in this communication by CAU 1 and CAU 2, the CAU's secure inter communication provides secure communication between device in fog area 1 with device in fog area 2.

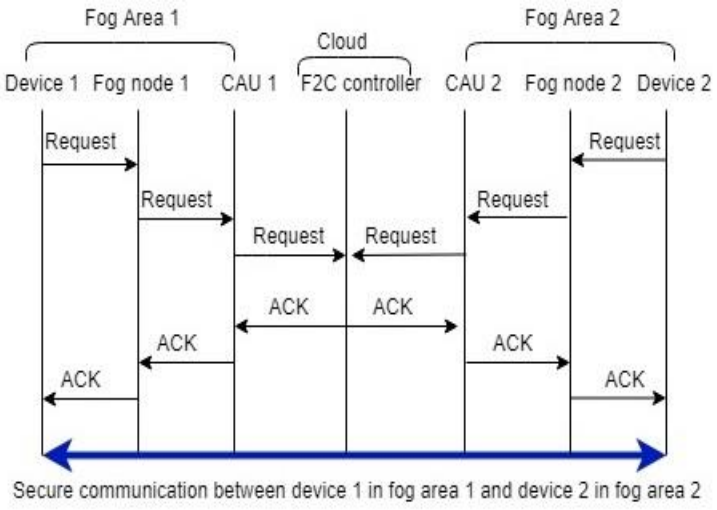

Figure 9. Number: Fog-to-Fog communication

All the above communication (black lines in figures) will be described in detail in the next sections for illustration each one of the security functionalities such as authentication, key management, secure channel establishment and access control that are implemented in CAUs in a distributed and hierarchical fashion.

The security architecture has many advantages that are illustrated in Table 8.

Security Architecture Advantages
Security management in distributed and hierarchical fashion
Distributed security provisioning
Efficient key management in distributed fashion
Less time-delay authentication mechanism at the edge of the network

Using hybrid cryptography for different layers (for example using symmetric for IoT devices and asymmetric for fog nodes and cloud)
Providing distributed authenticator for handling authentication rather than using a centralized one (single point of failure)
Handling security for edge and IoT devices with low computational power
Providing secure mobility and handover
Providing security with less impact on quality of service (QoS)
Decreasing the number of messages going to cloud
Scalability

Table 8. Security Architecture Advantages

In the next section, the usages of the security architecture will be described in critical infrastructures as an example.

8.2 Use case: The F2C and Security Architecture benefits in Critical Infrastructure

Critical infrastructures (CIs) [154] play a vital role in the world impacting on the economy, security, and health provisioning. CIs are a set of assets, be it either physical or virtual, providing country's essential requirements and directions; and any failure on them can cause a disaster in terms of security, economy or health.

Nowadays, the Internet of things (IoT) concept is merged into different CIs [154] such as hospitals, transport, nuclear plants, etc. Many sensors and actuators are utilized in CIs to facilitating the collection of distributed information from different locations to be analyzed for CIs. For example, a hospital uses distributed temperature sensors to collect temperature information for providing comfortable environment for patients. A nuclear system uses sensors and actuators to collect information from nuclear station to be checked and preventing any nuclear radiation. The huge volume of data produced by IoT devices in CIs must be filtered, aggregated, and stored, thus requiring the right technology and infrastructure to do so. Cloud computing, as a pay as you go online system provides datacenters for data processing, filtering, and storing. However, the conceptually far cloud cannot provide real-time processing, required by CIs to provide services for people. Then, fog computing appeared as a new concept which can be merged along with cloud to be used by CIs.

Fog provides real-time processing, geo-distribution, security, etc., by handling services closer to the users. The fog computing concept was introduced as a complementary architecture leveraging cloud computing, rather than to compete with it. From this idea, emerges, the F2C computing continuum system. This combined system allows services that demand real-time processing to use

fog, and in parallel, services demanding huge volume of data processing to use cloud. This hierarchical architecture can be merged into the CIs to facilitate their dependency interactions and services executions. CIs can benefit from the F2C system for using hierarchical fog nodes in their system. It facilitates the services execution for CIs, without, or even increases the security and the privacy of CIs' data. Data from sensors must be analyzed in CIs to detect possible risks. With the proposed deployment of fog nodes and fog areas, this data does not need to be sent to cloud through Internet to be processed. The cluster of devices in a fog area, under the control of a fog node, can handle the execution of sensitive services in CIs. In that sense, higher privacy is guaranteed. This is especially important in CIs, because data treated is particularly sensitive, both in terms of security (for example information about a nuclear station) and also in terms of privacy (for example patients' data in a hospital).

On the other hand, if data is processed close to the sources of data, real-time processing is also guaranteed, and finally, network traffic to cloud is also decreased. Apart of the advantages of handling CIs services in a F2C system, the security itself should be managed by a distributed system instead of being managed by cloud. Certainly, it is widely accepted that a key challenge in the CIs world is security. Potentially, CIs are so dependable to bring safety and security for people. However, the larger the number of things (IoT devices) in CIs are, the larger the security and privacy risks will be. Indeed, IoT devices have limited computational power to handle cryptography and security provision by themselves. Therefore, IoT devices can be used by attackers to launch the attack or get access to the collected information. This type of devices can be hacked or attacked in terms of passive and active attacks. In 2017, one attack to a hospital in England stopped the hospital network system for 24 hours, in this case casualties might be so terrible due to human's life losses.

Some researchers rely on the emerging "fog computing" concept because security can be handled closer to users (enhancing then the privacy as well) and in a distributed fashion. Nevertheless, in any case centralized cloud and distributed fogs must be coordinated to deliver a safe and secure system. CIs must consider a new strategy for handling security in this distributed and hierarchical fashion, because the centralized cloud as a single point of failure cannot be sufficient for handling security in dependable CIs. Therefore, the proposed security architecture for F2C scenario can be applied into CIs to bring more safety and security. In the following, the benefits of using the proposed security architecture in CIs will be illustrated and discussed in details.

In a critical infrastructure scenario that includes different CIs such as, smart healthcare, smart factory, smart transportation, etc. (Figure 10), the security architecture can be applied as a transversal architecture to provide security requirements in a distributed fashion. In this scenario and with the security architecture proposed, CAUs are deployed in the different areas handling the security of all type of CIs; or in an even more decoupled architecture with specialized CAUs for each one of the CIs. In this second approach of specialized CAUs, each smart component in each CI can use a certain number of specialized CAUs for each one of the CIs (smart Health, smart Transportation, etc.) as it is shown in Figure 10. For example, smart healthcare might use CAUs for handling security according to the huge number of IoT devices in their environment. All distributed CAUs have secure inter-communication. In case of a CAU failing, being compromised

or attacked, the F2C controller at cloud may substitute the nearest and safest CAU in that area till a new security controller will be selected.

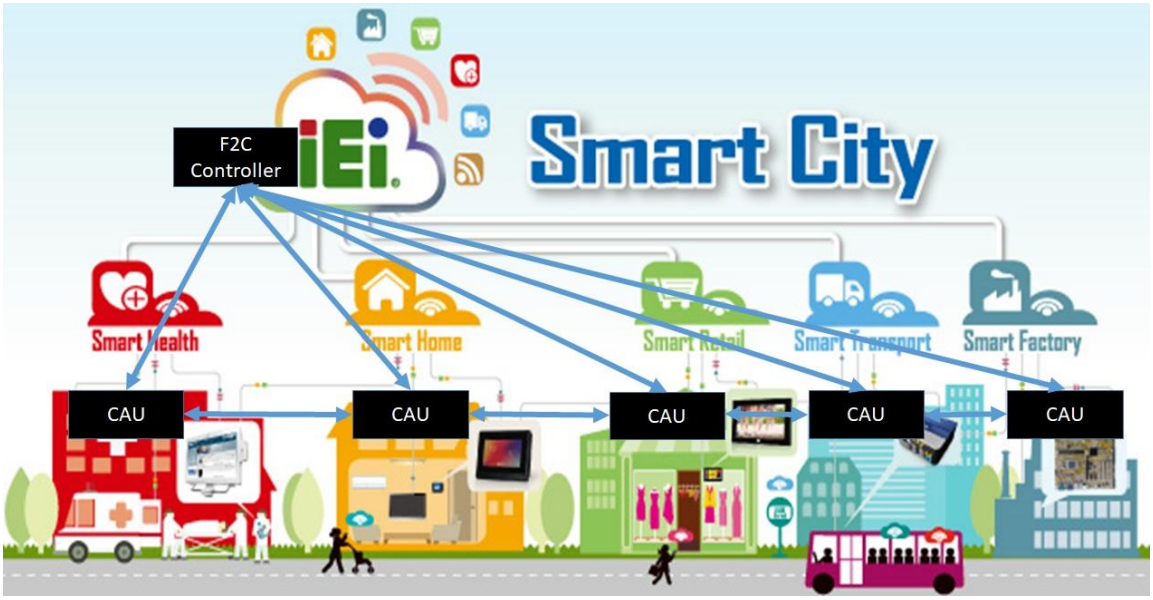

Figure 10. Security Architecture in CIs

This intercommunication between CAUs can also help to detect, counteract and react to possible cascading effects. The CIs are so dependable to each other. Therefore, any failure or compromise in one of them might effect on the others. The proposed transversal architecture using distributed security controllers can bring secure dependency into the CIs (Figure 11). Each one of the CIs might use different numbers of CAUs due to their infrastructure's need. All CAUs get authenticated and authorized from the F2C controller at cloud, therefore, they have secure inter-communication. This distributed CAUs can bring trust into the CIs. For example, in case of accident, a CAU in healthcare can communicate with CAU in transportation securely for getting patients information before patient arrives to hospital. Or in case of transportation accidents, CAU in transportation system can securely communicate with emergency services to provide safety and security for people.

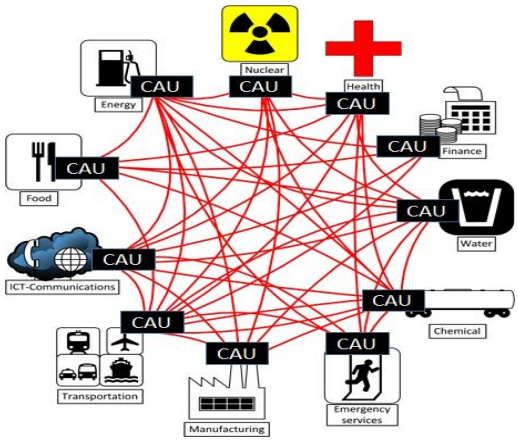

Figure 11. Security Architecture in CIs

One of the critical infrastructure issues is handling the new smart city concept due to transportation, healthcare, etc. where any failure or attack cause disaster in terms of human lives. When trying to deploy this security architecture in this smart city scenario with different CIs, the CAUs might be embedded inside of different smart city's component with high computational power. However, these components might have other critical responsibility such as real-time service execution, real-time data processing with low-latency, data aggregation and storing. Therefore, CAUs embedded into the smart city' devices might not be so suitable due to the high security processing usages which can impact on QoS in the smart city service to be executed. A decoupled transversal security architecture as another dimension with separated components must applied into the system to bring safety and security with demanded QoS. This second approach and applied to a smart city scenario is shown in Figure 12. The growth of IoT devices in a smart city for collecting information allows the execution of services aiming at easing the people's lives. With this amount of IoT devices security is being a challenge. One of challenges is how to provide security requirements for IoT-devices with low computational power. Here, the proposal is to distribute CAUs into the city. Therefore, each CAU provides IoT-devices' security requirements in its area. All CAU have secure inter-communication with each other. In case of failing a CAU, it is compromised, or attacked, the nearest safest CAU is used as backup to provide security in that area till a new security controller is selected. The distributed security controllers (CAUs) are capable of providing security requirements such as key management, authentication, intrusion detection, abnormal behavior detection and etc. for distributed low-computational IoT devices in smart city. Therefore, smart city concept can be developed to "smart secured city" to ease people's lives with safety and security.

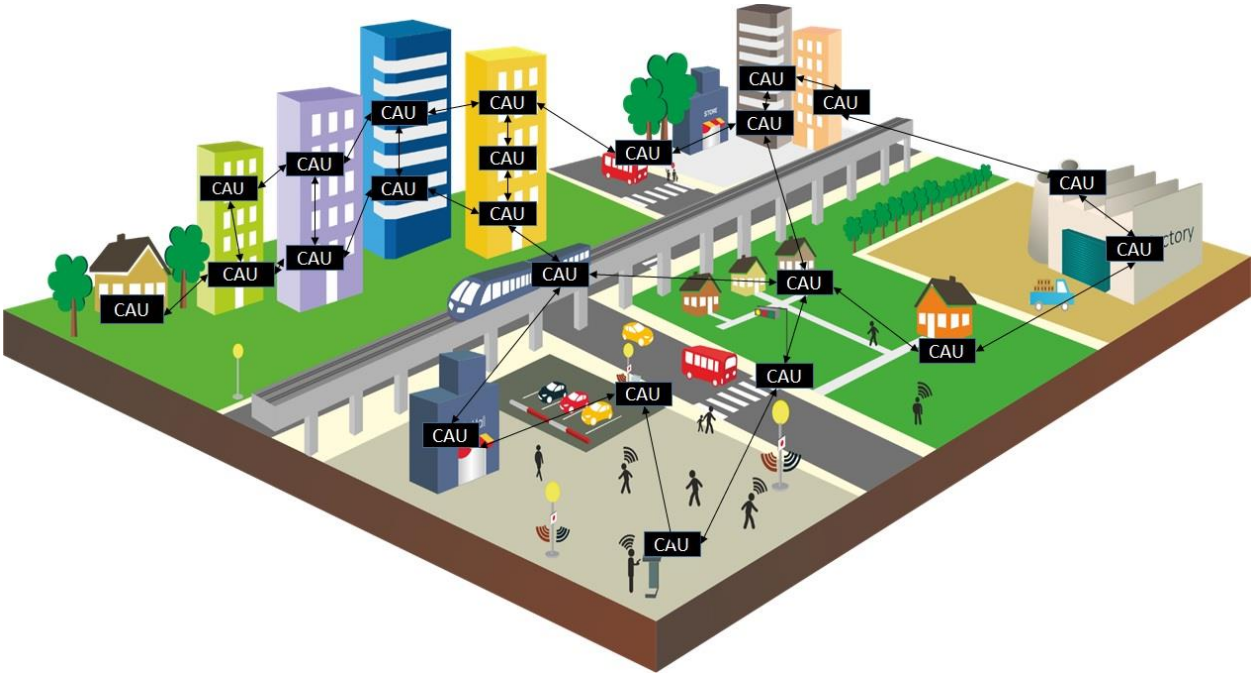

Figure 12. Security Architecture in Smart City

In the previous scenario, the benefit of using the proposed security architecture has been discussed for critical infrastructures and smart cities. In the next sections, the security architecture implementation will be illustrated in details.

8.3 Authentication in Security Architecture

As mentioned in security consideration and requirements chapter, authentication is one of the most essential and primal security requirements which must be implemented in any proposed security architecture. Authentication is the process of showing and proving that the users, devices, and all components in the system are truly the ones that they claim. Authentication prevent any unauthorized users to access the system.

In case of the F2C system, all the devices and components such as cloud, fog nodes, and edge and IoT devices must be mutually authenticated. In order to provide authentication in the F2C scenario, two implementations are done such as:

1- In the mF2C project [155], the security architecture (CAUs) act as smart-gateways and bridge between cloud and devices at the edge for getting certificate and authentication process in the early stage.
2- The CAUs act as distributed authenticators and provide certificates and authentication process in their corresponding areas. In the following, both scenario will be illustrated and discussed.

8.3.1 CAUs in mF2C

The mF2C project [155], proposes a F2C combined architecture in a hierarchical way, where N-numbers of fog layers are allocated for facilitating service execution and delivery to the users. In this project, the proposed security architecture is implemented in iteration-1 for facilitating authentication process. In Figure 13, the implemented authentication process with the help of CAUs is shown.

Authentication in mF2C is described as follows:

1- Initialization phase: In this process, all distributed CAUs authenticate and establish secure channel with certificate authority (CA) in the cloud.
    1- CAU sends certificate signature request (CSR) and its id (CAU-id) to the CA.
    2- CA sends CAU-id to the corresponding component such as id provider in cloud.
    3- Id-provider in the cloud checks the CAU-id existence and it is validated and exists.
    4- Id provider in cloud sends validation to the CA.
    5- CA sends signed certificate to the CAU.
    6- CAU and CA are authenticated and transport layer security (TLS) established for providing CAU-CA secure channel.

It is worth to mention, after fog nodes (see Fig. 7) selection in the fog areas is done, all the previous process will occur between each one of the fog nodes and the corresponding CAU, to provide fog node-CAU authentication and TLS establishment in initialization phase.

2- Edge device authentication process:

7- Edge device is registered in cloud.
8- Id provider in the cloud, generates device-id.
9- The id-provider sends device-id to the edge device.
10- In parallel, the device-id is sent to the CAU by id-provider for local id validation.
11- The edge device comes to the fog area and is discovered by the fog node.
12- The edge device sends CSR and device-id to the CAU.
13- CAU checks the device-id existence for validation.
14- If the device-id exists and it is validated then CAU sends CSR to the CA.
15- CA signs certificate and sends signed certificate to the CAU.
16- CAU sends signed certificate to the edge device.
17- In parallel, CAU sends device-id to the fog node.
18- Edge device and fog node are authenticated and establish TLS.

This authentication process uses distributed CAUs that act as smart gateway and bridges for id validation and facilitating communication between edge devices and CA.

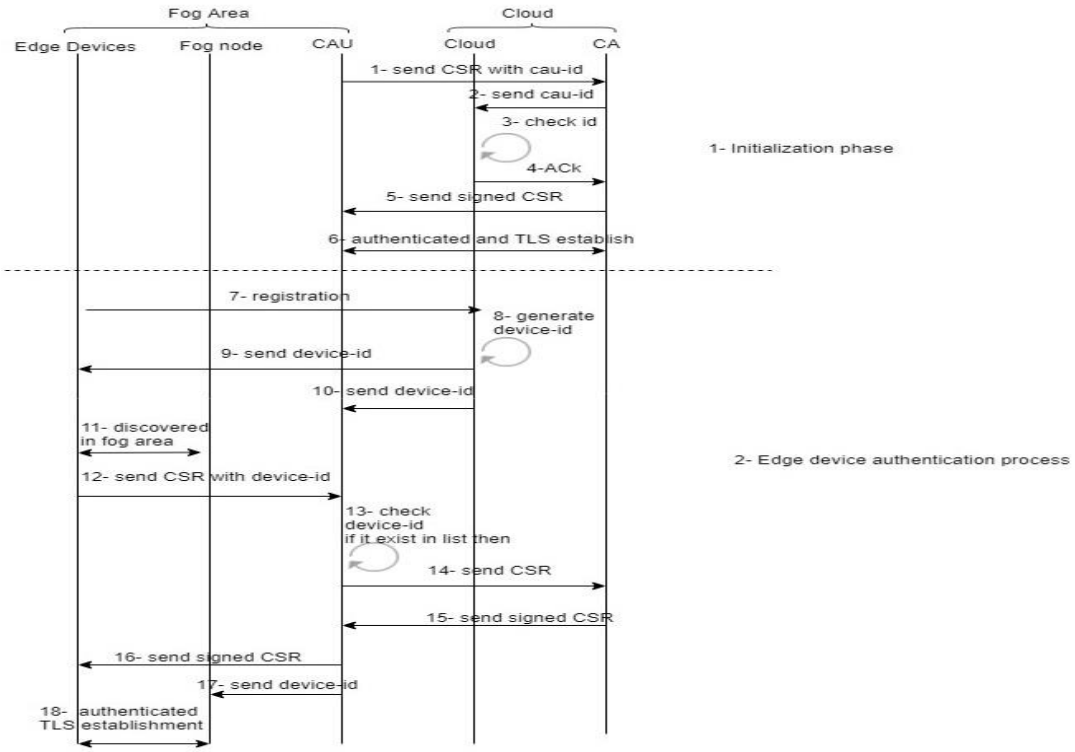

Figure 13. Authentication workflow in mf2

8.3.2 CAUs as distributed authenticators

After developing the previous implementation, the security architecture deployed in the F2C system provides authentication in a distributed way rather than rely on the CA in the cloud. Whereas, in this new implementation, the F2C controller at cloud acts as CA for providing authentication and TLS establishment for distributed CAUs at the edge of the network. Then, all the CAUs act as distributed authenticator (distributed CA) for their corresponding fog areas. The Figure 14 illustrates the workflow for the implementation.

Authentication in distributed authenticators (CAUs) scenario is described as following:

1- Initialization phase: In this process, all distributed CAUs are authenticated and establish a secure channel with certificate authority (CA) in the cloud.
    1- CAU sends certificate signature request (CSR) and its id (CAU-id) to the F2C controller.
    2- F2C controller checks the CAU-id existence in the list for validation, if exists then, goes to the next step. (After id provider generate ids for CAU, it sends to the F2C controller.).
    3- F2C controller sends signed certificate to the CAU.
    4- CAU and F2C controller are authenticated and a transport layer security (TLS) is established for providing CAU-F2C controller secure channel.

It is worth to mention, after fog nodes selection (see Fig. 7) in the fog areas, all the mentioned process will occur for each one fo the fog nodes and the corresponding CAU, to provide fog node-CAU authentication and TLS establishment in the initialization phase.

2- Edge device authentication process:
    5- Edge device is registered in cloud.
    6- Id provider in the cloud, generates device-id.
    7- The id-provider sends device-id to the edge device.
    8- In parallel, the device-id is sent to the CAU by id-provider for local id validation.
    9- The edge device comes to the fog area and is discovered by the fog node.
    10- The edge device sends CSR and device-id to the CAU.
    11- CAU checks the device-id existence for validation. If the device-id exists and is validated then, goes to the next step.
    12- CAU signs the certificate and sends signed certificate to the edge device.
    13- In parallel, CAU sends device-id to the fog node.
    14- Edge device and fog node are authenticated and establish a TLS connection.

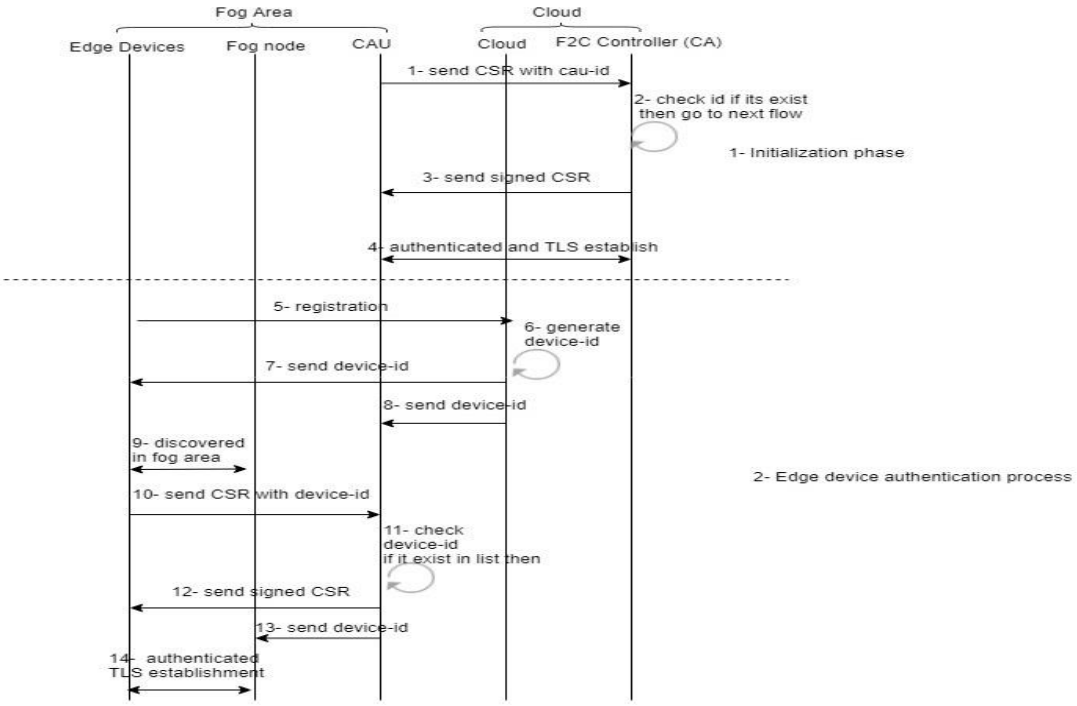

Figure 14. Authentication workflow in CAUs as authenticator

The both scenarios are implemented in our test-bed, the obtained results and comparisons between two scenarios are shown in section 9.1.

8.4 Key distribution and management in Security Architecture

After authentication, key distribution and key management is one of the vital issues in the F2C system due to its hierarchical and distributed nature. In the traditional fashion, a centralized component such as trusted third party (TTP) that could be inside or outside of the cloud is responsible for providing public and private information such as pair of keys, etc. This centralized remote component is not suitable for the hierarchical and distributed F2C system because the centralized TTP is a single-point-of-failure that any attack, failure, and compromise, can affect the whole system. Even, the distanced TTP for handling huge number of devices at the edge of the network effects on the QoS in terms of time delay, network overhead, and bandwidth. This centralized TTP must provide key management such as update keys and even delete the keys in the case of attacks; however, handling the key management process for the huge number of devices at the edge of the network with a centralized component is not enough efficient.

Some devices at the edge of the network such as fog nodes in the F2C system can generate keys due to their powerful computational power, although, some devices such as IoT devices, sensors, actuators, and edge devices have low-computational power and need a trusted component or components for their key generation, distribution, and management.

In the following, the security architecture proposed in this thesis is adapted to the F2C scenario for providing key management (it is published in [156]) in details; and an implementation is emulated in the mentioned smart city test-bed. The evaluation is a comparison between traditionally centralized TTP and cloud handling key management, against the security architecture proposal for distributed key management and authentication that is called DKMA that uses CAUs as distributed key managers and message authenticator.

For the both, the cloud key management and the security architecture key management, the elliptic curve key distribution and signature for managing keys and message authentication is used for sake of comparison in both scenario. The key Elliptic Curve Cryptography (ECC) advantage is that it provides same security guarantees as Public Key Infrastructure (PKI) and other cryptographies with less key size. Here briefly, the Elliptic curve digital signature algorithm (ECDSA) is discussed and illustrated.

ECDSA is known to be an efficient secure certificate-signing algorithm, used in several TLS libraries, such as OpenSSL and GnuTLS. ECDSA depends on modular arithmetic operations on elliptic curves as defined by the equation (1)

$$Y^2 \equiv x^3 + ax + b \bmod p \qquad (1)$$

The curve includes three parameters: p, a large prime number defining the curve finite field Fp, and the coefficients, a and b [157] , [158]. Table 9 lists all parameters description as used in the algorithm.

Signs	Description
Fp	the curve finite field
p	a large prime defining the curve finite field Fp
a and b	coefficients
G	is the base point of the curve
n	Is the order of G
h	Is the cofactor
sk	Private key
Pk	Public key
k	A uniform random number in the range [1, p-1]
(u,v)	Curve points
r	One part of signature
s	Other part of signature
d	Private key used for signing
z	Hash of the message to be signed
(r,s)	Signature made public

Table 9. ECDSA Algorithm sign description

The ECDSA algorithm features the two following functionalities:

1. Key generation: It is based on elliptic curve diffie-hellman, which is used for encryption and decryption.

   1.1. The private key *sk* is a random integer chosen from $\{1,\ldots,p-1\}$

1.2. Public key pk is calculated from the curve point multiplication $pk = sk \times G$.

2. Signature and verifying: Used for authentication.

2.1. Signing:

- Take a random integer k chosen from range of $\{1,\ldots,p-1\}$
- A curve point is calculated by: $(u,v) = k \times G$.
- One part of signature is $r = u \bmod n$.
- If $r=0$, then choose another k and try again.
- The other part of signature is $s = k^{-1}(z + rd) \bmod n$.
- If $s=0$, then choose another k and try again.
- The pair (r,s) is the signature.

2.2. Verifying: The signer's public key pk, the (truncated) hash z and, obviously, the signature (r,s) are required.

- Calculate the integer $u_1 = s^{-1}z \bmod n$.
- Calculate the integer $u_2 = s^{-1}r \bmod n$.
- Calculate the point $P = u_1G + u_2\,pk$.

The signature is valid only if $r = u \bmod n$.

The mentioned algorithm is implemented in both traditional cloud and the proposed security architecture for comparing both scenarios. In the following, implementation workflows for both scenario will be shown.

Cloud key management and authentication: In this traditional key management steps are (Figure 15):

1. Device registers in the cloud.
2. Identity provider in the cloud generates device-id.
3. The device-id is sent to device by cloud.
4. The device sends keys and signature request with its id to the cloud.
5. The cloud checks the device-id, if it exists then generates public key, private key and signature for device.
6. The cloud sends keys and signature to device.
7. The device sends message authentication request signed by signature to cloud.
8. The cloud checks the signature validation if its valid then
9. Cloud sends validation to the device.

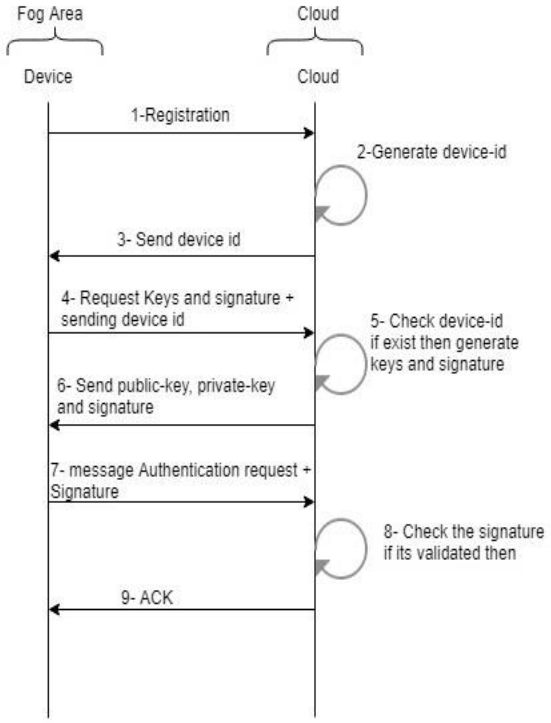

Figure 15. Cloud key management workflow

Distributed key management and authentication (DKMA) proposal (it is published in [156]): The proposed DKMA according to the security architecture adopted to F2C scenario have the following steps (Figure 16):

1- Initialization phase:
    1- CAU requests keys and signature by sending the request with its id to the F2C controller at the cloud.
    2- The F2C controller checks CAU-id, if it exists then generates keys and signature.
    3- F2C controller sends public key, private key and signature to the CAU.
    4- CAU sends message authentication request signed by signature to the F2C controller.
    5- F2C controller checks signature validation, if it is valid then goes to step 6.
    6- F2C controller sends ACK to the CAU.
2- Device key management and authentication: after the initialization phase, all distributed CAUs get authenticated and authorized from the F2C controller to provide key distribution, management and message authentication to their corresponding areas. The steps device gets keys and message authentication are as following:
    7- Device registers in the cloud.
    8- Identity provider in the cloud generates device-id.
    9- The device-id is sent to device by cloud.

10- In parallel, the device-id is sent to CAU for validation part.
11- The device sends keys and signature request with its id to the CAU.
12- The CAU checks the device-id, if it exists then generates public key, private key and signature for device.
13- The CAU sends keys and signature to device.
14- The device sends message authentication request signed by signature to CAU.
15- The CAU checks the signature validation if it is valid then goes to step 16
16- CAU sends validation to the device.

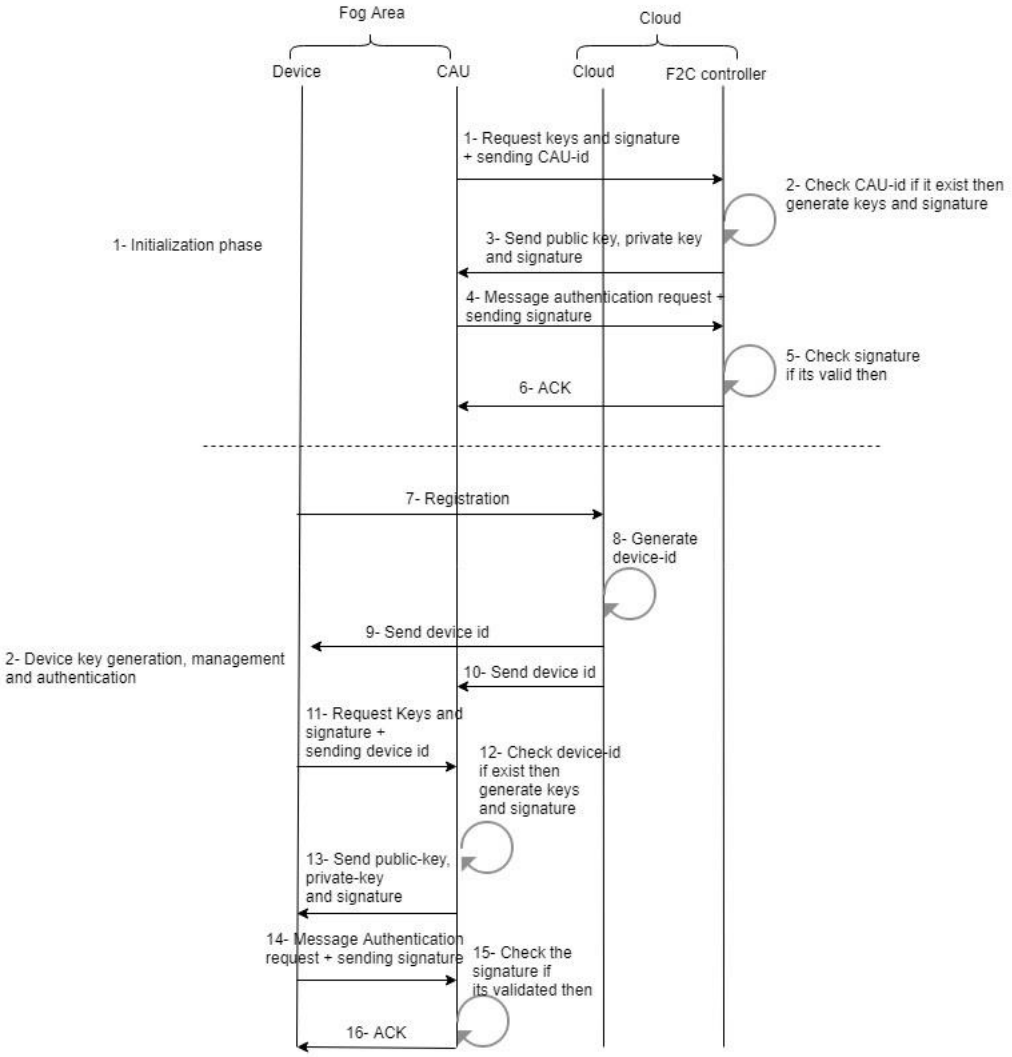

Figure 16. Distributed key management and authentication (DKMA)

As described in the above, now in Figure 17 and Figure 18 both scenarios are shown.

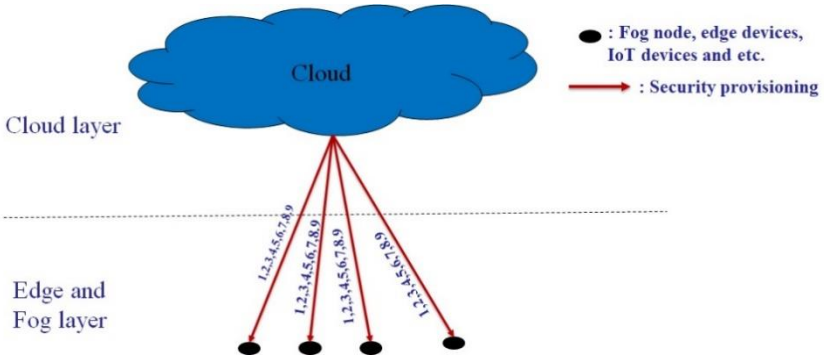

Figure 17. Cloud key management and authentication

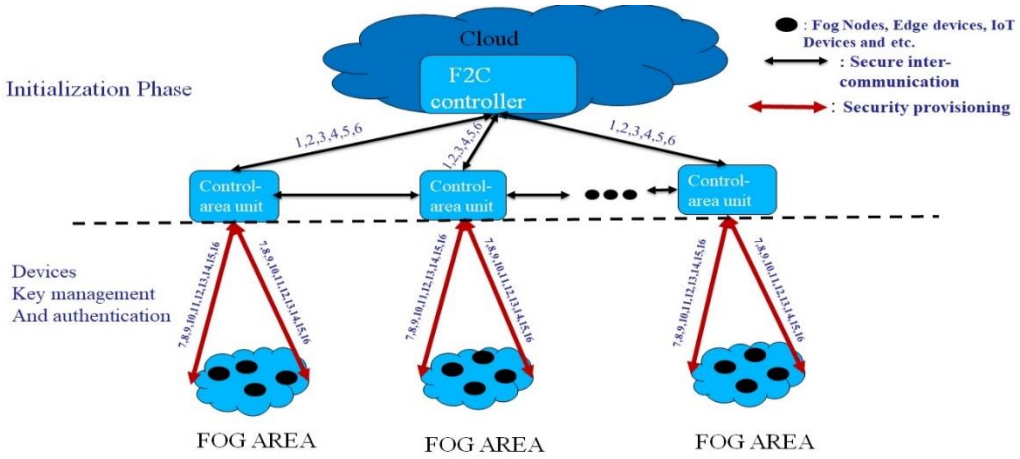

Figure 18. Proposed DKMA

The both scenario are implemented in the test-bed. The obtained results and comparison are shown in section 9.2.

8.5 Access control and distributed data management in Security Architecture

In F2C system, the devices at the edge of the network such as sensors, actuators, even some fog nodes have low-computational power. The devices such as sensors and actuators are continuously capturing, producing, and sending data to the fog node in the fog layer for processing, aggregating, filtering, storing and after a while sending historical data to the cloud for more processing. Usually, the fog node has so many functionalities such as service allocation and execution for the users, therefore, using fog node as data storage effect on the QoS due to lower computational power compare to the cloud. Even, a selected fog node to manage a fog area can be substituted to another fog node with more computational power, then a data storage synchronization between fog nodes is a must.

On the other hand, taking into account the distributed nature of the F2C system, using cloud as traditional data manager and storage for the whole system is not enough efficient. In this case, handling a huge amount of data in the F2C system become a big issue. A new strategy must consider to provide distributed data management for the F2C system taking advantage of the huge number of distributed devices at the edge of the network.

Another considerable issue about data management in the F2C system is be sure that data transfer and storage are secure. The security in data management means that all captured data must be sent to the corresponding component for processing and storage in a secure channel and authenticated way. After data storage, a user who wants to access to this data must satisfy access roles to prevent any unauthorized user to access to the data. One of the most essential data that must be secured is device's resource information due to critical information about devices such as ID, IP address, etc. Even, it is also clear that this information and data are quite essential for managing the whole system. Mostly, they are sensitive in nature and eventually, unauthorized access, processing and tampering of this information and data might affect the service execution. Unfortunately, that might degrade the performance and efficiency of the whole system. For that reason, it is necessary to ensure the security and reliability of this highly distributed system. Therefore, a novel secure distributed data storage in the fog layers for F2C system is proposed according to the mentioned security architecture (CAUs) to make sure that all capture data and resource information are secure with ensuring QoS in the F2C system.

Proposed Architecture:

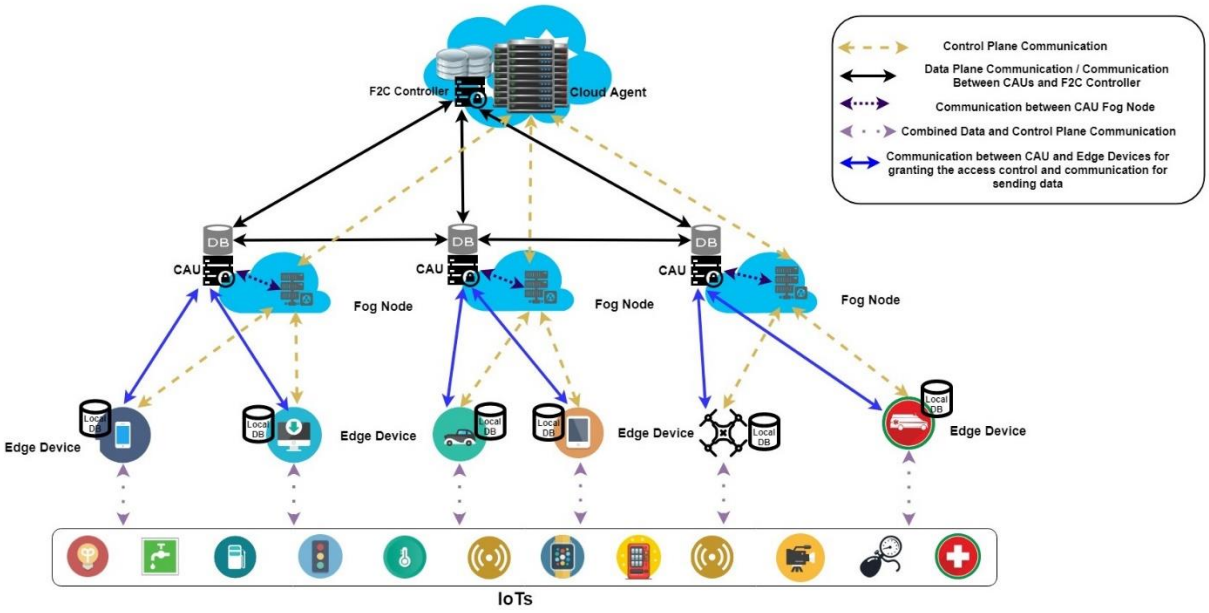

Figure 19. Proposed Architecture

As mentioned most of the edge devices such as IoT devices, sensors, actuators and even fog nodes have not enough computational power to provide multiple service functionalities such as

reasonable security and QoS. Therefore, decouple and distributed secure database is proposed for ensuring secureness and reliability in the F2C system. In the architecture, the distributed CAUs at the edge of the network are responsible for implementing security functionalities for their corresponding fog areas. At initialization phase, all CAUs get authenticated and take authorization from the F2C controller at the cloud to provide security functionality at the edge of the network. After authentication and authorization, all the CAUs-CAUs and CAUs-F2C controller have secure inter-communication (black lines in the Figure 19). The CAUs act as distributed authenticator at the edge of the network and provide authentication for their corresponding fog area's components (all the blue lines in Figure 19). Also, CAUs act as distributed key manager and capable of generating, distributing and managing keys for the device with low computational power that are not capable of generating keys.

As mentioned, CAUs have secure inter communication over secure channel, therefore, it is pretty relevant to implement a distributed database over the CAUs. The distributed database is adapted into the CAUs and CAUs provide security functionality such authentication, secure channel establishment, encryption and decryption over data and access control in a distributed fashion for their corresponding fog areas. The CAUs perform encryption to store data in a secure way so even if attacker try to eavesdrop or modify data (passive and active attacks), data would be protected.

Therefore, CAUs are distributed trusted components for providing a distributed database and finally adapting the security architecture with a F2C controller at cloud and distributed CAUs at edge of the network, which are setting up a secure distributed database in the hierarchical F2C system. It is worth to mention that this architecture can be used for all types of data, but in the thesis, edge device resource information is used for illustrating the importance of secure distributed data base. In the following all the communication between components are illustrated in different workflows.

Proposed Workflows:

The proposed architecture includes three work flow such as:

a. Storing edge device's resource information
b. Accessing resource information in the same fog area
c. Accessing resource information in different fog areas

All the communication between components such as Cloud, F2C controller, CAUs, fog nodes, and edge devices are over the transport layer security (TLS) secure channel. All the CAUs act as distributed authenticator that are using X.509 public key certificates with RSA. All CAUs can perform access control (Role-Based) to prevent any unauthorized access to distributed database and even do encryption/decryption by AES advanced encryption standard (AES) algorithm.

a. Storing edge device's resource information: The workflow steps are shown in Figure 20:

- Edge Device authentication and information creation in distributed database:
  1- Edge device is registered in cloud.
  2- Cloud generates edge device id.
  3- Cloud sends device-id to the edge device.
  4- In parallel, the device id is sent to the F2C controller and all CAUs.

5- Edge device sends CSR to its corresponding CAU with its id.
6- CAU checks if the device-id exists, and then signs the certificate,
7- CAU sends signed certificate to edge device.
8- Edge device and CAU are authenticated and a user 'information is created for edge device in CAU.
9- Edge device and CAU establish TLS communication.
10- CAU sends edge device' public key and edge device' id to its database.
11- CAU creates the roles for accessing the edge device resource information in its database.
12- The user' information of edge device is replicated in all CAUs database.
13- CAU sends ACK to the edge device.

- Storing resource information:
  14- Edge device sends its resource information to the CAU.
  15- CAU checks id and authentication process if it is valid goes to step 16.
  16- CAU finds the user of edge device in its database.
  17- CAU checks the edge device access and permission if it's valid then.
  18- CAU encrypts resource information.
  19- CAU stores encrypted resource information in its database.
  20- The encrypted resource information is replicated in all CAUs database.
  21- CAU sends ACK to edge device

b. Accessing edge device's resource information in same fog area: The workflow steps are as follows (Figure 21):

- Initialization phase:
  1- Edge device resource information was uploaded into the CAU' database according to workflow a.
  2- Fog node sends CSR with its id to the corresponding CAU in the fog area.
  3- CAU checks the id if it is exist then signs the certificate.
  4- CAU sends signed certificate to the fog node.
  5- Fog node and CAU are authenticated and establish TLS communication.
- Accessing edge device's information:
  6- Fog node sends request for accessing edge device resource information to CAU.
  7- CAU checks access control list and permission if it is satisfied then goes to step 8.
  8- CAU finds edge device user 'information in its database.
  9- CAU queries the resource information in its database.
  10- CAU decrypts the resource information.
  11- CAU sends edge device resource information to fog node over a secure channel.

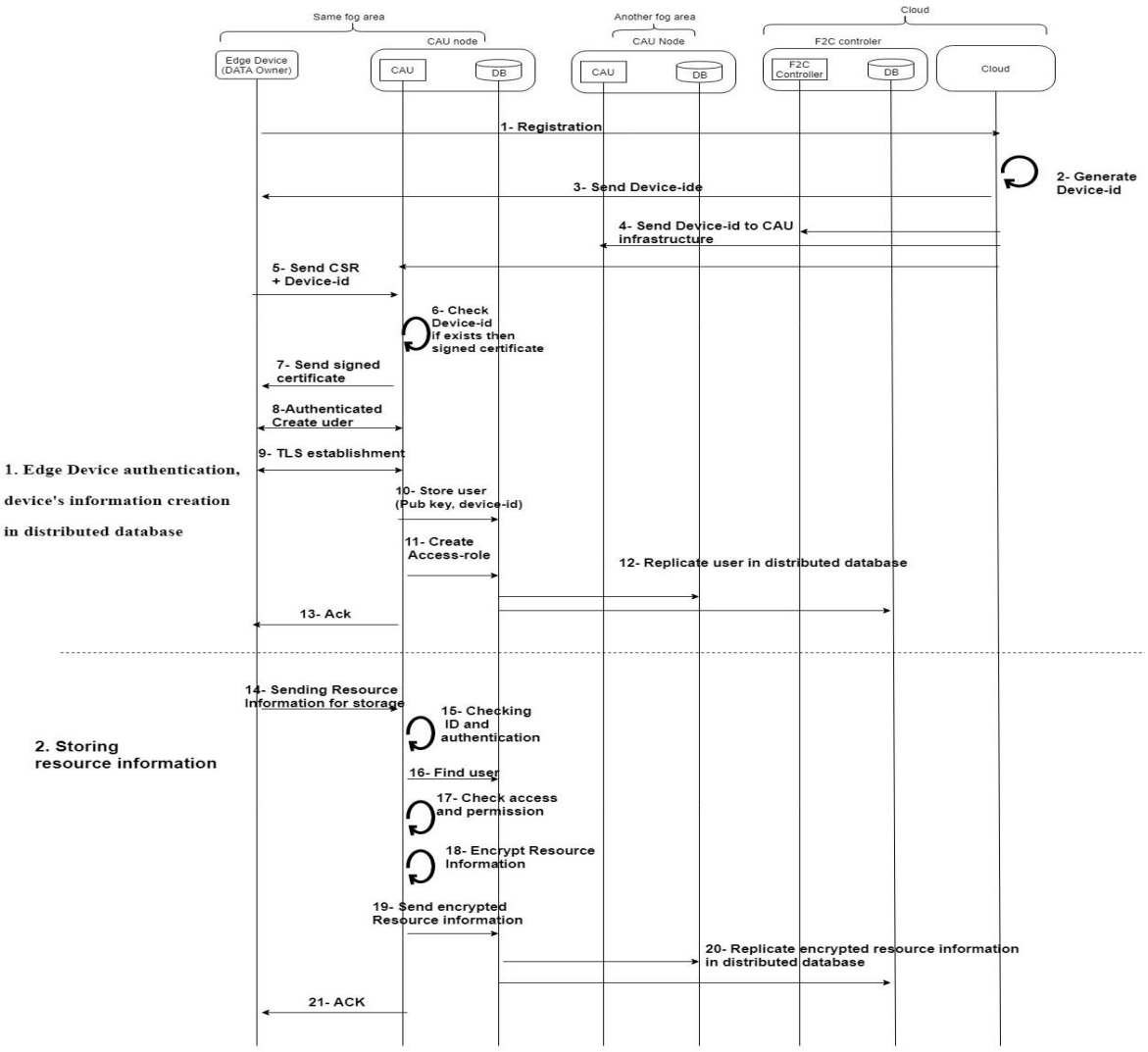

Figure 20. a) Storing edge device' resource information workflow

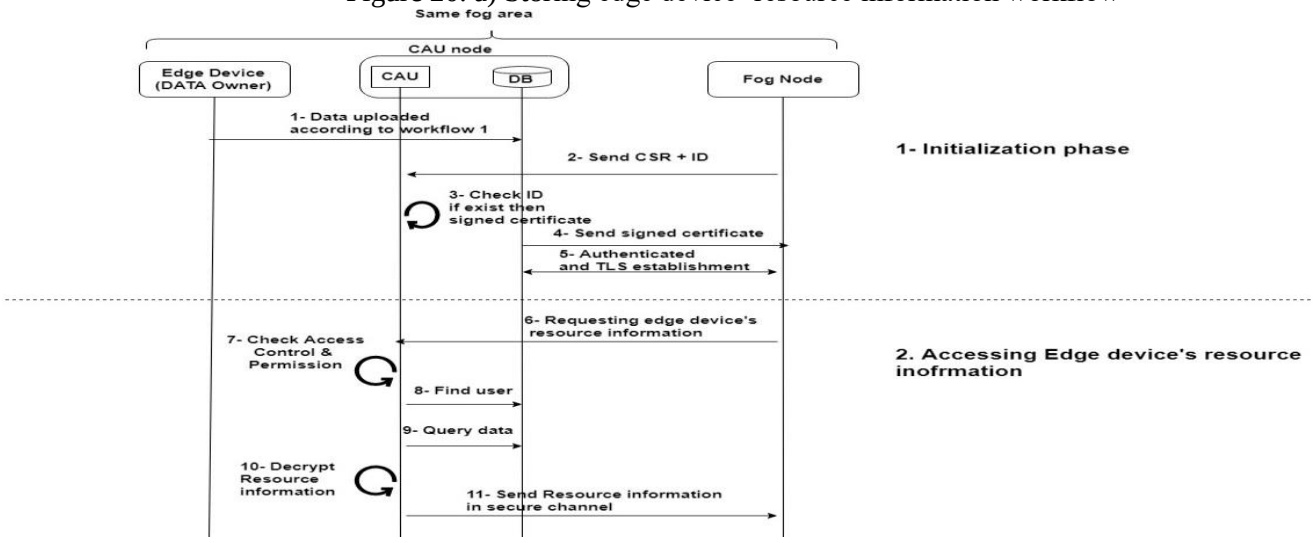

Figure 21. b) Accessing edge device's information in same fog area

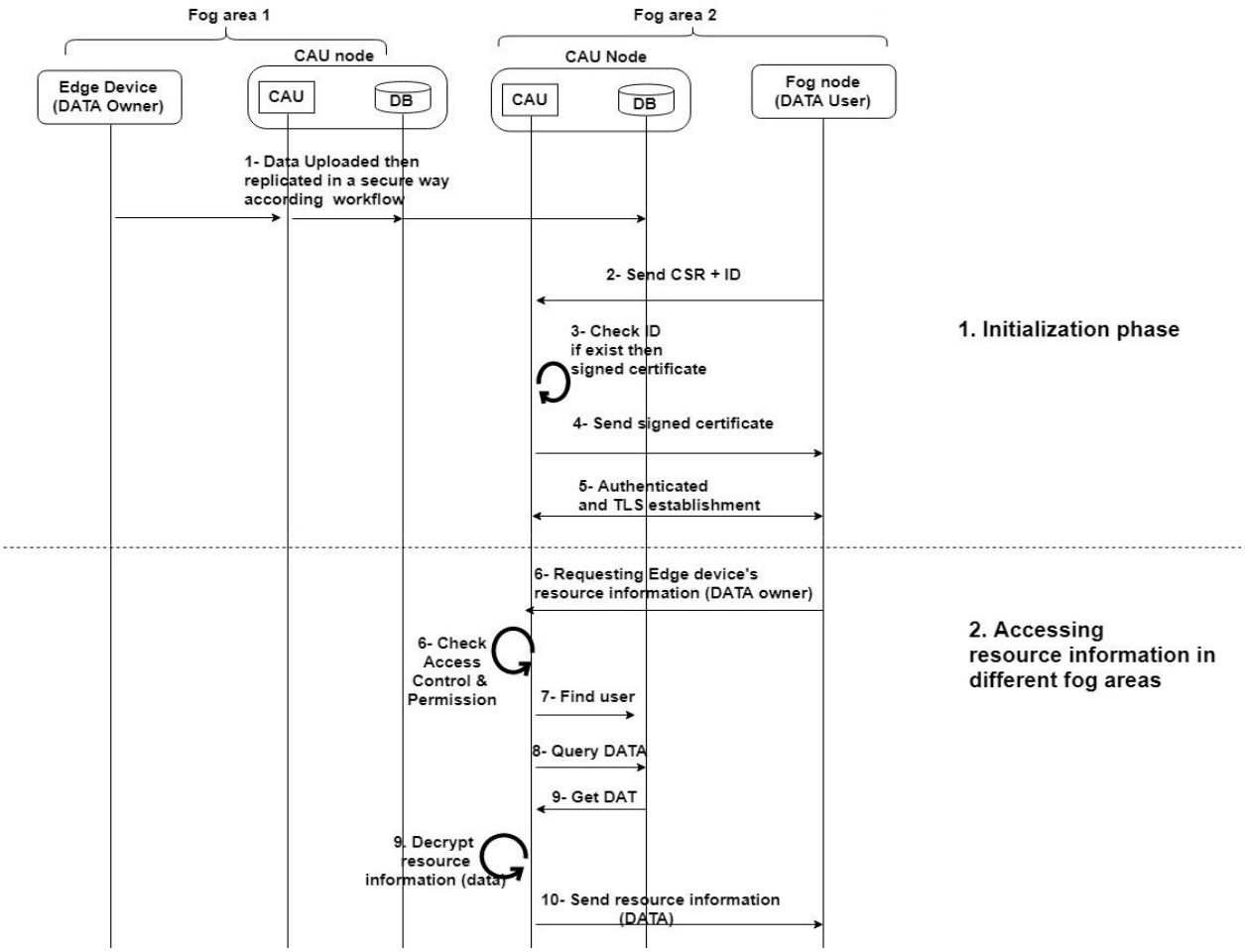

Figure 22. c) Accessing edge device's resource information from different fog areas

c. Accessing edge device's resource information from different fog areas: The workflow steps are as follows (Figure 22):

- Initialization phase:
  1- Edge device resource information was uploaded and replicated in secure way in all CAUs database according to workflow a.
  2- Fog node in fog area 2 sends CSR and its id to the corresponding CAU.
  3- CAU checks id if it exists then signs the certificate.
  4- CAU sends signed certificate to fog node.
  5- Fog node and CAU are authenticated and establish TLS communication.
- Accessing edge device's resource information from different fog areas: As edge device resource information in fog area 1 is uploaded and replicated in the all CAUs database, fog node in area 2 can access edge resource information through CAU in fog area 2.
  6- Fog node sends request to access edge device information in fog area 1 to its corresponding CAU.
  7- CAU checks access control list and permission if it is validated then goes to step 8.
  8- CAU finds edge device user's information in its database.
  9- CAU queries its database.

10- CAU gets edge resource information from its database.
11- CAU decrypts resource information.
12- CAU sends edge resource information to the fog node over secure channel.

Algorithm 1 Securely storing of resource's information (data)

Input: (to store) $pubKey$, $column\_family$, $index$, $data$
Output: *Store the data to the distributed database*
Initialisation : *Authentication of Fog Node and secure TLS connection establishment between Fog Node and CAU*
PROCEDURE: *Securely_data_storing_in_distributed_db*

```
putData(pubKey, column_family, index, data):
checkAuth(pubKey)
userID = queryUser(pubKey)
if userID != None then
## User is registered on the system ##
    hasPerm = checkPermissions(column_family, userID);
    if hasPerm == True then
        edata = encryptData(data);
        res = queryInsert(column_family, index, edata);
        if res == 'OK' then
            return True & store data in the distributed database;
        end if
    end if
else
    return False;
end if
```

Figure 23. Algorithm of securely storing resource information

Algorithm 2 Secure retrieval of resource's information (data)

Input: (to retrieve) $pubKey$, $column\_family$, $index$
Output: *Retrieve the data from the distributed DB*
Initialisation : *Authentication of Fog Node and secure TLS connection establishment between Fog Node and CAU*
PROCEDURE: *Secure_data_retrieve_from_distributed_db*

```
queryData(pubKey, column_family, index):
checkAuth(pubKey)
userID = queryUser(pubKey)
if userID != None then
## User is registered on the system ##
    hasPerm = Access_control_list(column_family, userID, role);
    if hasPerm == True then
        edata = encryptData(data);
        res = querySelect(column_family, index);
        if notres.Empty() == True then
            data = decryptData(edata);
            return data;
        end if
    end if
else
    return "";                    ## return empty string ##
end if
```

Figure 24. Algorithm of secure retrieve of resource information

The main objective of the proposed secure data storage and management is handling properly the huge amount of data or even resource information in the F2C system in a secure way. In the following algorithms, all security functionalities (Figure 23) and data management tasks (Figure 24) are summarized according to the workflows.

The both workflow are implemented in the smart city test-bed. The obtained results and workflow analysis are shown in section 9.3.

8.6 Decoupled proposed security architecture vs embedded

The security architecture proposal can be implemented as embedded CAUs in fog nodes (ECF) as illustrated in Figure 25 (Section 8.1) or a decoupled security architecture, decoupled CAUs from fog node (DCF) (it is published in [159]). DCF is adapted into the F2C system in a way that a centralized F2C controller at cloud and distributed decoupled CAUs as distributed transversal security providers at the edge of the network provide security functionalities to the F2C system as illustrated in Figure 26 (All the red lines are security provisioning. Section 8.5). In this section, both scenario are implemented and adapted to a smart city scenario for comparing and evaluating that which one of the scenarios have less impact on the QoS by providing demanded security in the F2C scenario.

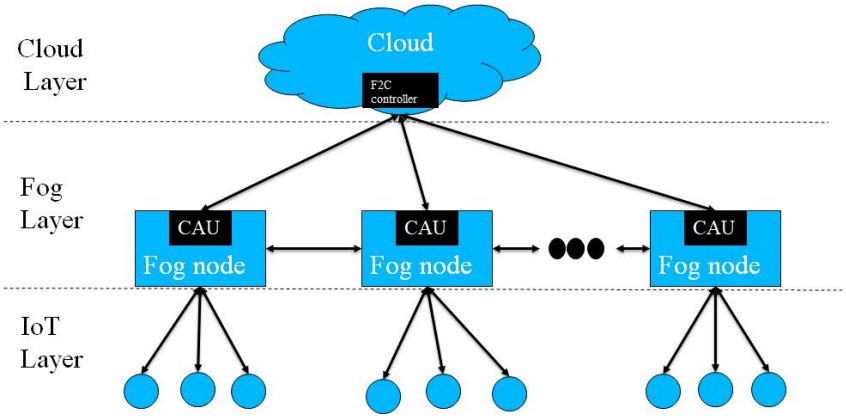

Figure 25. Embedded security architecture (ECF)

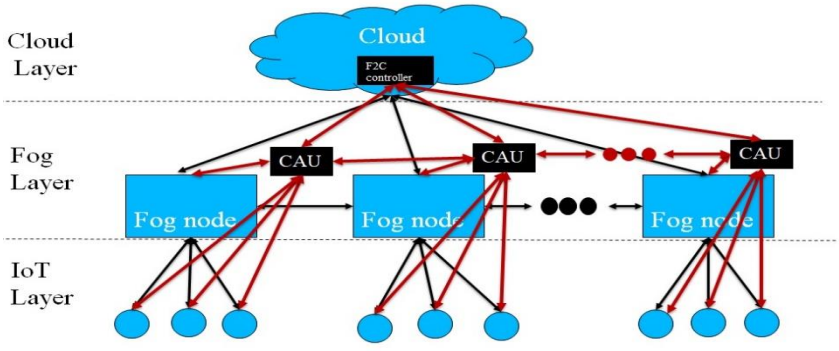

Figure 26. Decoupled transversal security architecture (DCF)

For implementing both ECF and DCF, the proposed workflows are illustrated and described in the details in the next subsections.

8.6.1 ECF

- ECF workflow:
  In the ECF workflow (Figure 27), the 4 phases are:
  0. Initialization phase: In this phase the embedded CAU inside the fog node is authenticated with the F2C controller in cloud and gets authorization to provide security in its corresponding area as follows:
     1- CAU sends CSR with its id to the F2C controller.
     2- F2C controller checks the CAU-id if it exists then signs the certificate.
     3- F2C controller sends signed certificate to the CAU in the fog node.
     4- CAU in fog node and F2C controller get authenticated and establish TLS communication.
  1. Authentication phase: Edge device-CAU authentication process is done in this phase:
     5- Edge device does registration in the cloud.
     6- Identity provider inside cloud generates device-id.
     7- Cloud sends device-id to the edge device.
     8- In parallel, cloud sends device-id to CAU for local validation.
     9- Edge device sends CSR with its id to CAU.
     10- CAU checks id if it exists then signs the certificate.
     11- CAU sends signed certificate to edge device.
     12- Edge device-CAU get authenticated and establish TLS communication.
  2. Key management phase: In this phase, edge devices that are not capable of generating keys, then they request CAU for key generation.
     13- Edge device sends keys request to the CAU.
     14- CAU generates public key and private key pairs by elliptic curve algorithm.
     15- CAU sends key pairs to the edge device.
  3. Service requests, allocation, and execution in a secure manner:
     16- Edge device sends service request in an encrypted way.
     17- CAU decrypts service request and sends service execution request to the corresponding components inside of fog node.
     18- After executing services by fog node, the results are re-directed to the CAU, then CAU encrypts and sends service results to the edge device.
     19- Edge device decrypts service results.
     20- Edge device sends ACK to CAU.

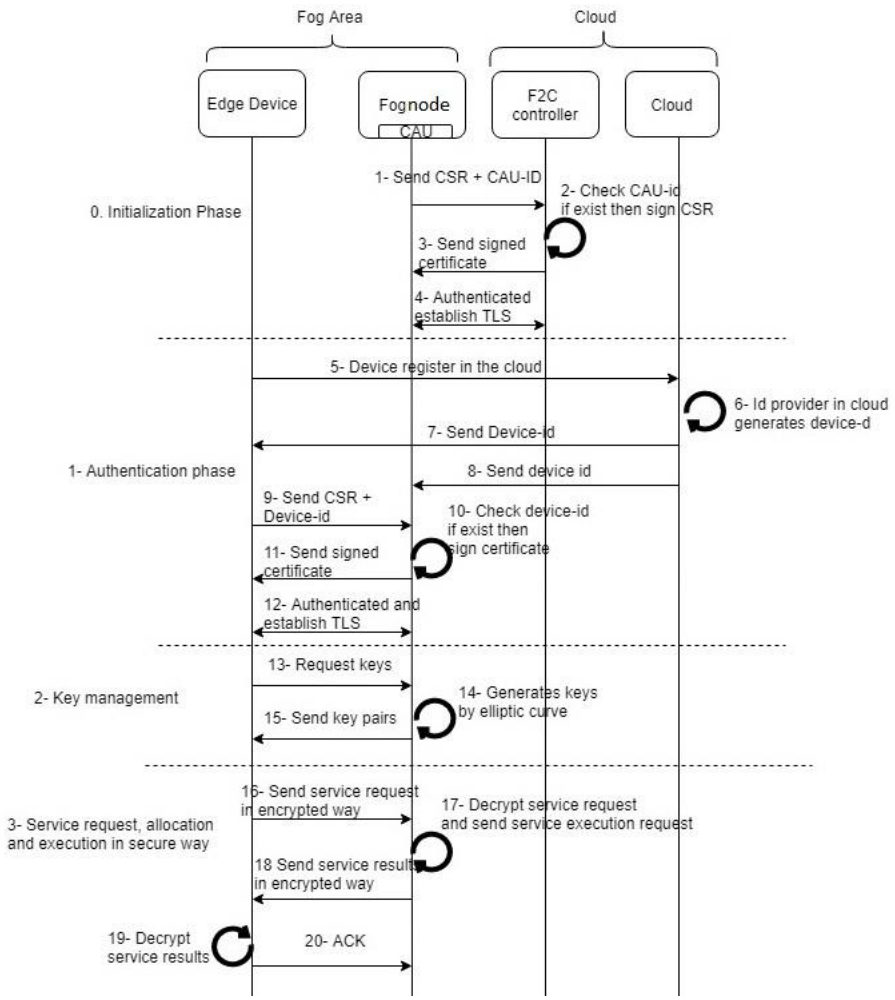

Figure 27. The ECF workflow

8.6.2 DCF

- DCF: In the DCF workflow (Figure 28), the 5 phases are:

0. Initialization phase: The CAU-F2C controller authentication process. After this process, CAU gets authorized to provide security in its fog area.
    1- CAU sends CSR with its id to the F2C controller.
    2- F2C controller checks id if it exists; and then it signs the certificate.
    3- F2C controller sends signed certificate to CAU.
    4- CAU-F2C controller get authenticated and establish TLS.

0.1. CAU-fog node authentication phase:
    5- Fog node do registration in cloud.
    6- Identity provider in cloud generates id for fog node.
    7- Cloud sends id to fog node.
    8- In parallel, cloud sends fog node-id to CAU for local validation.
    9- Fog node sends CSR with its id to its corresponding CAU.
    10- CAU checks id if it exists; and then signs the certificate.
    11- CAU sends signed certificate to fog node.

12- Fog node-CAU get authenticated and establish TLS.

1. Edge device authentication phase:
    13- Edge device do registration in the cloud.
    14- Identity provider in cloud generates id for edge device.
    15- Cloud sends id to edge device.
    16- In parallel, cloud sends edge device-id to CAU for local validation.
    17- Edge device sends CSR with its id to CAU.
    18- CAU checks id if it exists; and then signs the certificate.
    19- CAU sends signed certificate to edge device.
    20- Edge device-CAU get authenticated and establish TLS.
2. Key management phase:
    21- Edge device sends keys request to CAU.
    22- CAU generates public and private key pairs by elliptic curve algorithm.
    23- CAU sends keys to edge device.
3. Service request, allocation, and execution in a secure manner:
    24- Edge device sends service request in encrypted way to the fog node.
    25- Fog node sends encrypted service request to CAU.
    26- CAU decrypts service request.
    27- CAU sends service execution request to fog node in a secure channel.
    28- Fog node executes service and gets results.
    29- Fog node sends service results to CAU in a secure channel.
    30- CAU encrypts service results.
    31- CAU sends encrypted service results to edge device.
    32- Edge device decrypts service results
    33- Edge device sends ACK to fog node and CAU

The both workflow are implemented in the smart city test-bed. In section 9.4, obtained results and comparison between workflow are illustrated.

The next section is analyzed and illustrated the obtained results for the all mentioned security functionalities in the security architecture proposal.

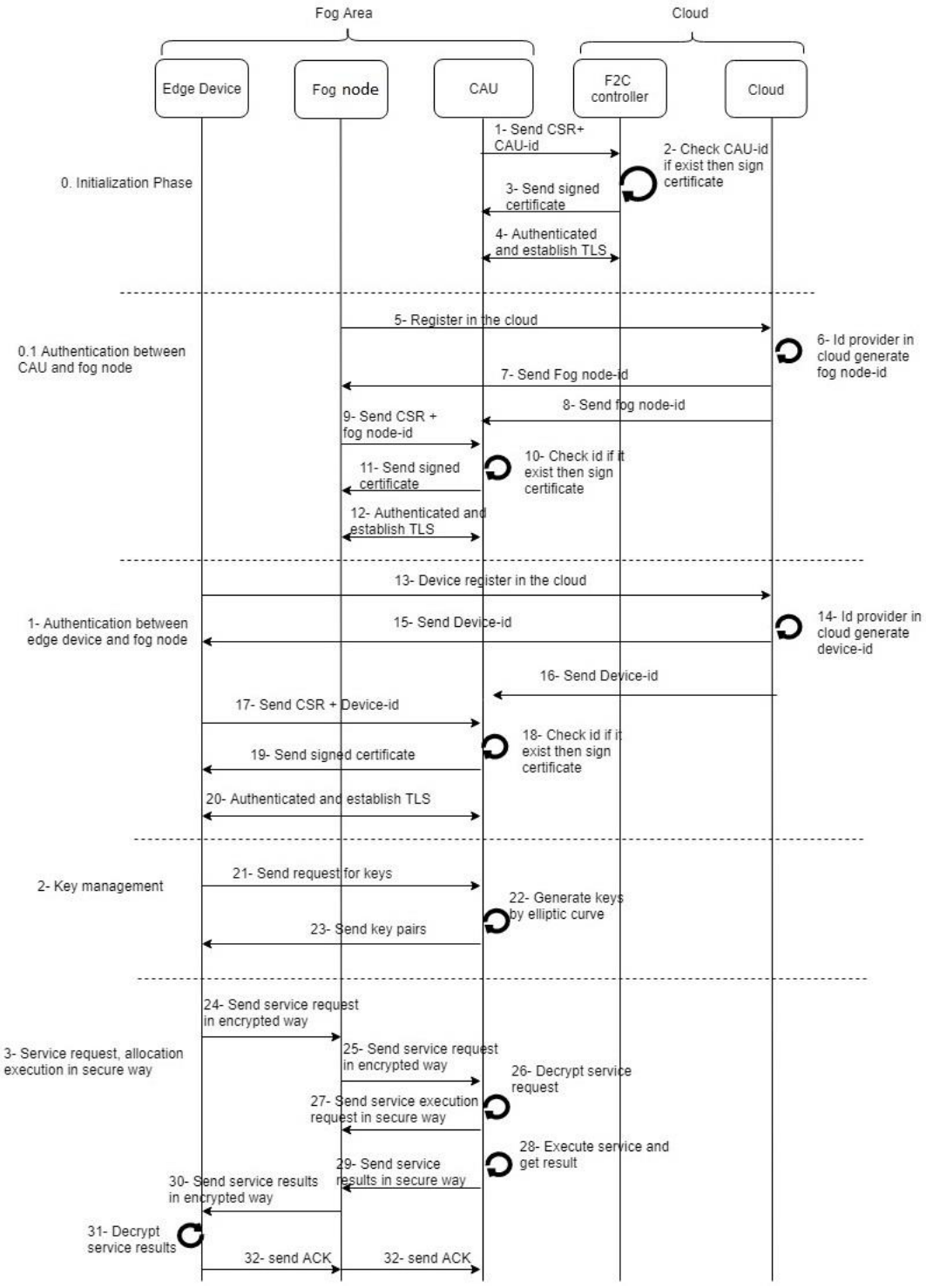

Figure 28. The DCF workflow

Chapter.9 Results/ evaluation

In this chapter, the security functionalities such as authentication, key management, access control, and ECF vs DCF scenario mentioned in chapter 8 are implemented in our smart city test-bed (Figure 29). In the following, all the obtained results in the security architecture proposal are illustrated.

Figure 29. Smart city test-bed

9.1 Authentication

Both authentication processes (in section 8.3.1 and 8.3.2) are adapted to the F2C scenario in the smart city testbed. The testbed emulates a smart city scenario in a 25m2 space and includes a variety of devices, such as traffic lights, street lights, vehicles and buildings, all with an embedded Raspberry Pi 3 (RP). The testbed is illustrated in Figure 29.

The both mentioned scenarios are implemented in python 3 and adapted into the test-bed. X.509 public key infrastructure (PKI) standard implemented in CA (Scenario 1), F2C controller and CAUs (Scenario 2). In the first scenario, cloud services such as identity provider and certificate authority are hosted on a server machine with Intel Xeon family E5-2620 V4 series (clock speed @3GHz), 96GB RAM, 1TB Hard Drive running on Ubuntu 16.04LTS Linux, a RP implemented as fog node (traffic light), a RP acting as CAU in the fog area and RPs acting as edge devices (buildings).

In the second scenario, the cloud services such as identity provider and F2C controller (CA) are hosted in a server machine with Intel Xeon family E5-2620 V4 as well, a RP acts as fog node, a RP acts as CAU (authenticator), and a RP acts as edge device.

For both implemented scenario, the results obtained are illustrated in Table 10. The authentication' time for centralized CA is 86.567ms and for distributed CAUs is 8.288ms. Therefore, the conclusion here is that using distributed CAUs as distributed authenticator closer to the users and proximate to the edge of the network provides more efficiency in terms of authentication time compare to using traditional cloud, which is in CIs is one of the main keys. Even, the distanced CA in cloud is a single-point-of-failure, therefore, using distributed CAUs as authenticators rather than using a centralized cloud as authenticator provides less probability of security risks.

Authentication done by CA (Scenario 1)	Authentication done by CAU (Scenario 2)
86.567 ms	8.288 ms

Table 10. Authentication time delay

9.2 Key management

For the key management experimental result as mentioned in section 8.4, ECDSA algorithm is implemented in both scenarios such as centralized key management and DKMA (Figure 17 and Figure 18) for the sake of comparison and even elliptic curve provides authentication and key management in less key size secure same as other algorithm. The Test-bed scenario is the same as the smart city test-bed mentioned in previous section. The current test-bed deployment leverages an access point providing connectivity to the environment through the same network. Traffic to the cloud is sent through a router along the link to cloud. A frontend is also deployed to manage the test-bed settings, as well as to show an overview of the different trials running in the test-bed. Regarding the network analysis, the test-bed also includes some scripts for packets tracking, thus getting updated information about the network state using a packet catcher (e.g., tcpdump for Linux scenarios) and application logs.

1- Cloud scenario: A single PC provides key distribution and authentication. A Fujitsu Primergy TX300 S8 is hosting 100 virtual devices. In this case, the PC acts as key and signature generator, distributer, manager and authenticator for the 100 virtualized device.
2- In the distributed (DKMA) approach, One PC acts as cloud and F2C controller, five computers are acting as distributed CAUs. All CAUs are authenticated in the initialization phase, getting the authorization from cloud. Each CAU provides key and signature generation, distribution, management and authentication for groups of 20 devices. So there are 5 distributed CAUs controlling 20 virtual devices each.

The mentioned two workflows are implemented and analyze the two scenarios, comparing them in terms of key distribution and authentication delay, network delay, and network overhead, on the test-bed described above.

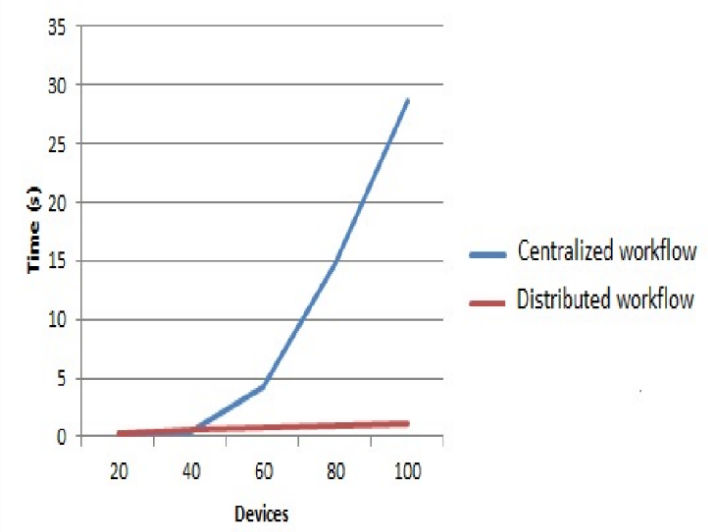

Figure 30. Key distribution and authentication delay comparison

Figure 30 illustrates the comparison results obtained from both workflows in terms of key distribution and authentication delay. A substantial reduction in the delay for the proposed DKMA distributed approach is shown. Indeed, while the time grows exponentially with the number of devices for the centralized approach, it keeps almost flat for the distributed one, reaching the maximum reduction when considering 100 devices, from 28.69s to 1.0942s.

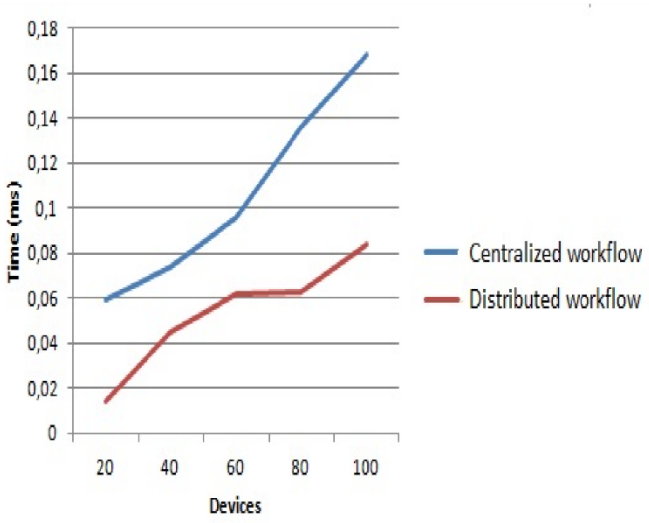

Figure 31. Network Time Delay

The Figure 31 similarly illustrates the obtained results for both workflows in terms of network delay, which is computed by dividing the Round Trip Time (RTT) by 2. The obtained result shows an incremental value for the delay reduction when using the distributed approach, reaching just the half (from 168ms to 84ms), when considering 100 devices.

Devices	KB Centralized	CAU	KB DKMA
100	223,83	5	44,77
80	179,06	4	44,77
60	134,30	3	44,77
40	89,53	2	44,77
20	44,77	1	44,77

Figure 32. Network overhead comparison (Kbytes)

Figure 32 shows the results comparison between the scenarios in terms of the network overhead. The whole set of messages in Kilo-Bytes (KB) forwarded throughout the network. The assumption here according to the defined policy is per CAU can manage up to 20 devices for security provisioning. Certainly, for the distributed scenario (DKMA), the network overhead (KBs) is not changing while for the in the centralized approach grows as the number of devices increase. Therefore, the network overhead will not change for the proposed DKMA distributed approach, as long as we can keep the assumed CAUs deployment policy.

Finally, the total number of messages for both scenarios are measured and analyzed. Here, the number of message is only analyzed, not message size. The total number of messages per device to get keys, signature and finally authenticate for both workflows in implementation part is 27 messages. In the first workflow, the cloud provides a centralized key distribution, management and authentication for every device, therefore number of messages going to cloud is (27*number of devices). However, when deploying the DKMA distributed controller's workflow, the number of messages is reduced up to (27*number of control area units). In the implementation part, 100 devices are used in centralized simulation and 20 devices are controlled by 5 CAUs then number of message go to cloud are:

1. Centralized: 27*100=2700 number of messages
2. Distributed (DKMA): 27*5= 135 number of messages

As a summary of the obtained and presented results, the conclusion can be the proposed DKMA is more sophisticated and efficient with less impact on the QoS to be deployed in the F2C scenario for key distribution, management and authentication rather than the centralized one, while keeping the same security level. Although, it is worth to mention that by distributing key manager at the edge of the network, the single point of failure of the centralized key management is overcome. One of the main advantages of using the hierarchical key management such as, centralized in cloud for CAUs and distributed CAUs as key manager at the edge of the network, is the possibility of using different symmetric and asymmetric algorithms for different layers according to the devices computational power.

9.3 Access control

For validating the secure distributed data management proposal (Section 8.5), the implementation of all workflows (Figure 20, Figure 21, and Figure 22) is adapted to the smart city testbed. In the smart city testbed, there are different fog areas, in each area a fog node is selected for managing the fog area resources, service allocation, execution, and etc. In parallel, for each area a CAU is selected for handling security in their corresponding area. On the top of the smart city a cloud and F2C controller as master of CAUs is implemented.

The cloud services are hosted on the server machine with Intel Xeon family E5-2620 V4 series (clock speed @3GHz), 96GB RAM, 1TB Hard Drive running on Ubuntu 16.04LTS Linux, and the F2C controller is hosted in this machine. In the testbed, all the CAUs and fog nodes are relatively small computing devices. The CAUs are implemented in the Raspberry Pi3 B+ model. The Raspberry Pi3 B+ models are coming with Cortex A53 @ 1.4GHz processor, 1GB SD-RAM and each of them having a 64GB micro-SD card and running on Ubuntu 16.04LTS Linux. All the fog nodes are implemented on the Raspberry Pi Zero model. They are coming with 1GHz single-core CPU and 512MB RAM. For storage purpose, the 8GB microSD for each of the fog devices is considered.

Considering all the CAUs and the F2C Controller, we implemented the distributed database, over the network. For distributed data base creation over the CAUs and F2C controller, the containerized Apache Cassandra *(Dockerized-Cassandra)* is implemented. Basically, for performing the tests, the multi-datacenter, and multi-node based Cassandra cluster over the considering distributed framework have been implemented. All CAUs and F2C controller are implemented in Golang for providing the security functionalities. The authentication mechanism in CAUs is using X.509 public key cryptographic, data encryption and decryption by using AES algorithm and finally providing access control by using role-based approach.

In this section, a preliminary security analysis over the proposed secure distributed data management will be discussed and a penetration test is done over the TLS and authentication mechanism to illustrate that the security mechanism in distributed fashion works properly. Then, a comparison between the proposed architecture and traditional cloud is done to illustrate the efficiency of the proposed secure distributed database.

Security analysis: In the proposed architecture, the security functionalities such as authentication, secure channel, encryption and access control are implemented.

1- Authentication: All distributed CAUs act as distributed authenticator using X.509 public key certificate with RSA that prevents any unauthorized user or device to enter in the F2C system. The distributed security provision rather than using a centralized approach gives some privileges such as: preventing man-in-the middle attack by decreasing the distance, because the authentication is performed closer to the users, also decreasing authentication's time delay by bringing closer the authentication to the users, facilitating scalability issues by means of the distributed authenticator nodes in CAUs, bringing trust by using

distributed CAUs as authenticator nodes; and finally, facilitating QoS in fog nodes by decoupling the security functionalities in the CAUs node.

2- Secure channel: All CAUs provide secure channel over TLS communication after the authentication process. Therefore, it prevents any type of active or passive attacks during edge devices-CAUs-fog nodes-cloud communications because all data are passing over a secure channel
3- Encryption: Distributed CAUs are capable of doing encryption over data before storing them in their database by using AES. Therefore, all the data in CAUs storage are encrypted and then it prevents any type of database attacks.
4- Access control: CAUs consist on two components such as, CAU that provides security functionalities and Cassandra database for providing distributed secure data storage. CAUs have access control functionalities by access control role-based. Therefore, any device or user who wants to access to encrypted data stored in CAUs must satisfy access control role-based. This functionality prevents any unauthorized user or device to access the data stored in CAUs.

All the listed security functionalities are implemented in the security architecture (F2C controller and CAUs) to provide secure data management for the F2C system. The authentication and TLS communicates is tested by kali tools [160]. As illustrated in Figure 33, authentication and TLS communication provided by CAU are tested and proved completely secure.

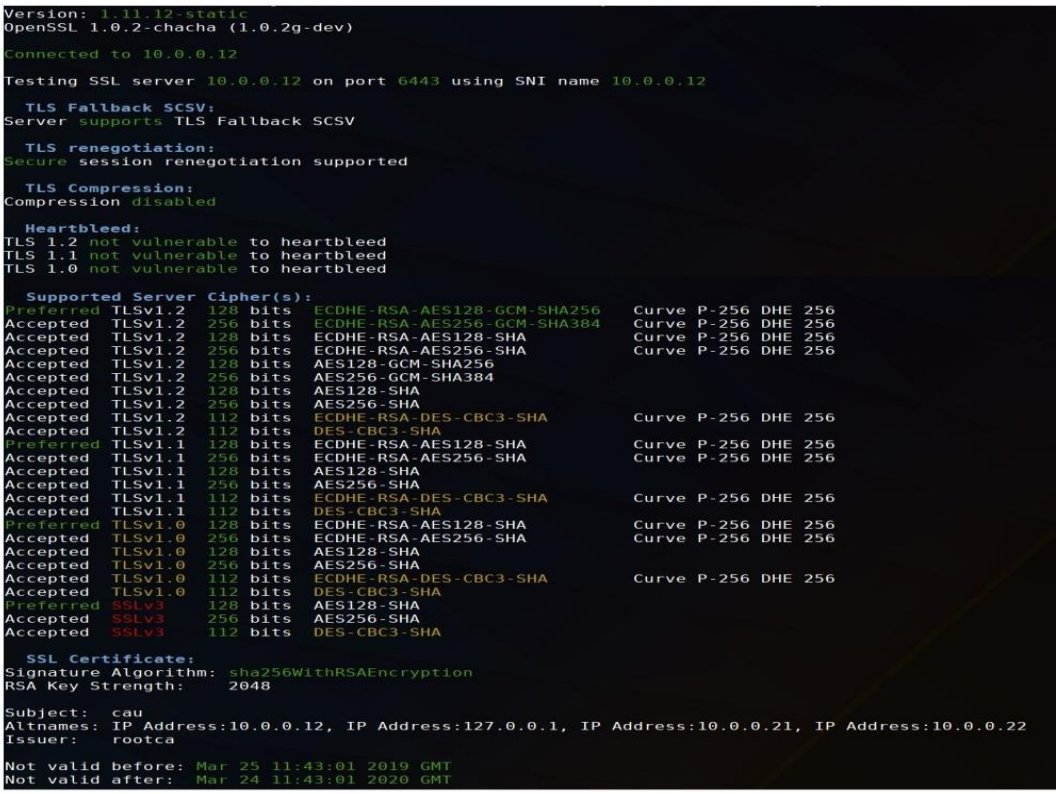

Figure 33. Penetration test over authentication and TLS communication

Data transmission impact (For data storing and data retrieving): Here, a comparison between traditional cloud secure data storing/retrieving and the proposed secure distributed database is performed and analyzed. The traditional cloud one uses the same workflow and algorithms that it is used in CAUs for sake of comparison between the centralized and distributed approaches. Then by comparing these two cases, the evaluation is showing the efficiency of the proposed model.

So for that purpose, the test is performed on the various amount of distinct number of data packets. Each of the data packets contains information about the current state of participating resource information. The mainly consideration is the current state of participating resource information, for storing and retrieving purpose. The size of each data packet is mostly between the 10KB to 20KB. The whole measurement over both scenarios considers the whole workflow as described above that include security functionalities such as authentication, TLS establishment, encryption/decryption, access control, and finally data storing and retrieving.

Figure 34 illustrates the comparison of the obtained results for data storing between traditional cloud scenario (red line) and the proposed distributed data management (blue line). Obviously, cloud scenario has more network delay and bandwidth constraint because cloud is distanced and far away from the edge of the network. In the traditional cloud, the security functionalities cause delays not only because the distance, but also for handling the huge number of data going to the cloud. Therefore, the cloud scenario for storing data takes longer time compared to the proposed distributed data management. For sake of evaluation, the test performed is to store 1000 distinct data-packets for the both scenarios. Cassandra is using the consistent hashing algorithm to distribute the data over the cluster, so that also helps to speed up the data storing procedure. As illustrated in the obtained results (Figure 34) the proposed model is more efficient in terms of data transmission time compare to the cloud.

In Figure 35, data retrieving obtained results is shown for tested cloud scenario (red line) and the proposed model (blue line). For both scenarios, the data retrieving test is started with 10 thousand distinct data packets and ended-point for the evaluation is performed the test on 1.28 million data-packets. The obtained results on data retrieving in our proposed model shows that on 80 thousand distinct data packets, the response time has been raised up and after that for the next couple of tests, the query response time becomes more identical. The reason is because when increasing the number of data-packets from 80,000 to 1,60,000; then it might happen that, the data packets have not been uniformly distributed over the cluster. So, that is the main reason for jumping up the query-response time. Interestingly, after reaching more than 6,40,000 data packets, the query-response time is getting more constant, but with a bit higher response time. As illustrated in Figure. 31, traditional cloud has higher response time compared to our proposed model in data retrieving. When the data searching test on ten thousand distinct data-packets is performed, in the traditional cloud system; the response time is 6.9291 seconds. As the number of data-packets increases, the response time is also increased. After a certain amount of data packets (1,60,000), the query response time becomes more constant; because of the uniform distribution of data among the cloud resources. For the proposed architecture, as the number of data packets increase, the respond time is also increased. After 1.600.00 data packets (in 8.0340s), query response time become more constant.

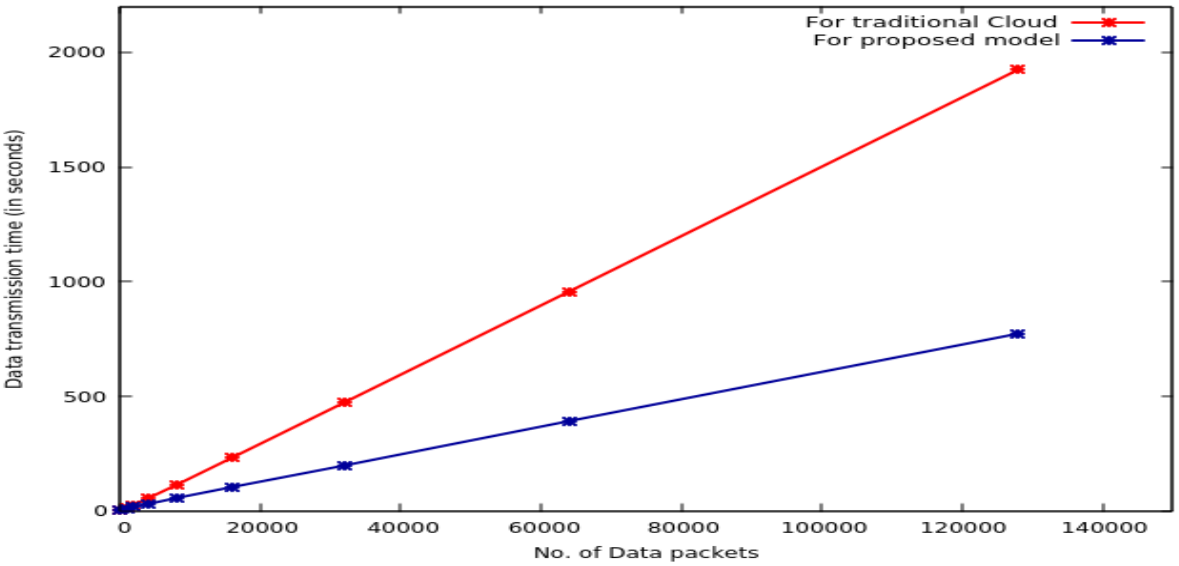

Figure 34. Data Storing: Traditional Cloud vs CAU-based Distributed Database

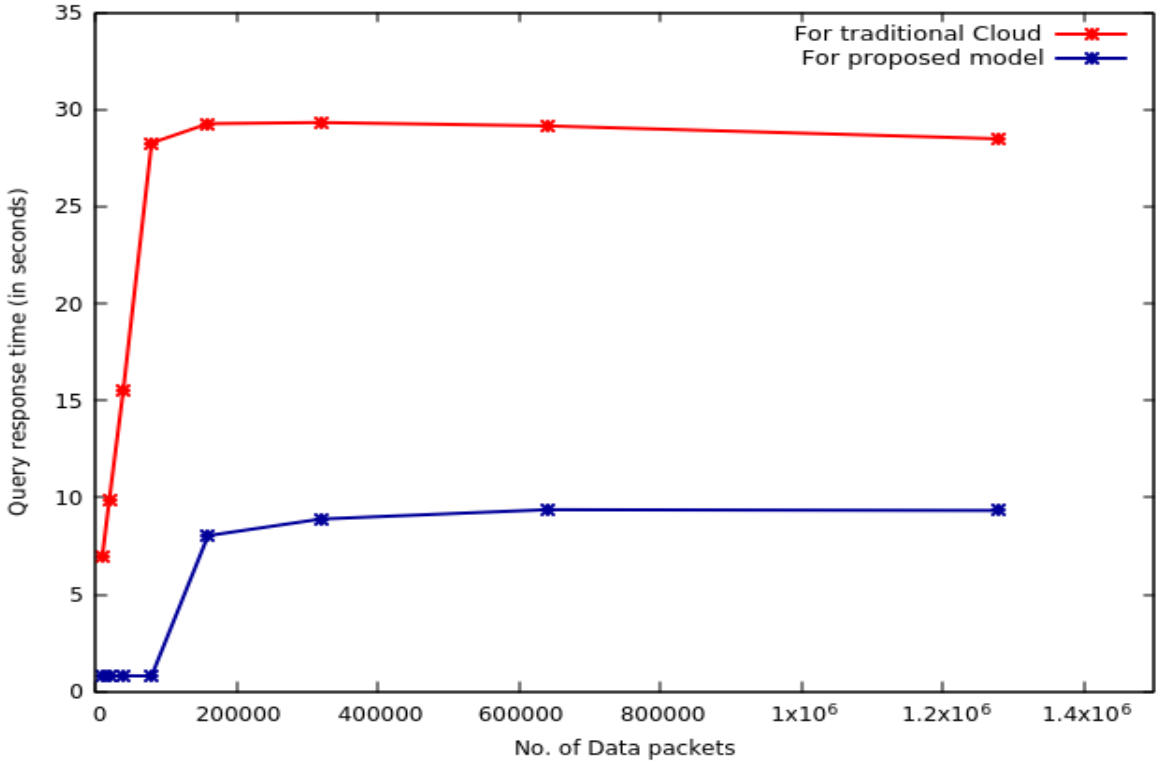

Figure 35. Data Retrieving: Traditional Cloud vs CAU-based Distributed Database

According to the mentioned obtained result, the conclusion is that the proposed model is more efficient compared with the traditional cloud in terms of secure data storing and secure data retrieving in the F2C scenario.

9.4 Decoupled security architecture vs embedded

For both ECF and DCF implementations mentioned in section 8.6.1 and 8.6.2 (Figure 27 and Figure 28), the following procedures are done:

- Edge device-fog node, fog node-cloud, CAU-F2C controller, and CAU-fog node communication over the transport layer security (TLS).
- The F2C controller (CA) at cloud is implemented by X.509 public key certificates for CAUs authentication. CAUs take authorization from the F2C controller (CA) to provide authentication and key distribution to the edge devices using X.509 and elliptic curve cryptography.

It seems evident that such a list of procedures shown in the workflows will affect the quality delivered to users mainly due to the required processing capacity (CPU), memory support (RAM), also turning into a remarkable time consumption. Thus, the main objective here is to evaluate the effects of adding security guarantees on the QoS for a F2C system. For this purpose, the both implemented workflows and a non-secure F2C system (nF2C) (in this scenario services are executed without any security implementation) are adapted to the smart city test-bed to evaluate delay and service.

The three different security approaches, namely nF2C, ECF and DCF are deployed in the testbed as shown in Figure. 36. The testbed emulates a smart city scenario as discussed in section 8.3

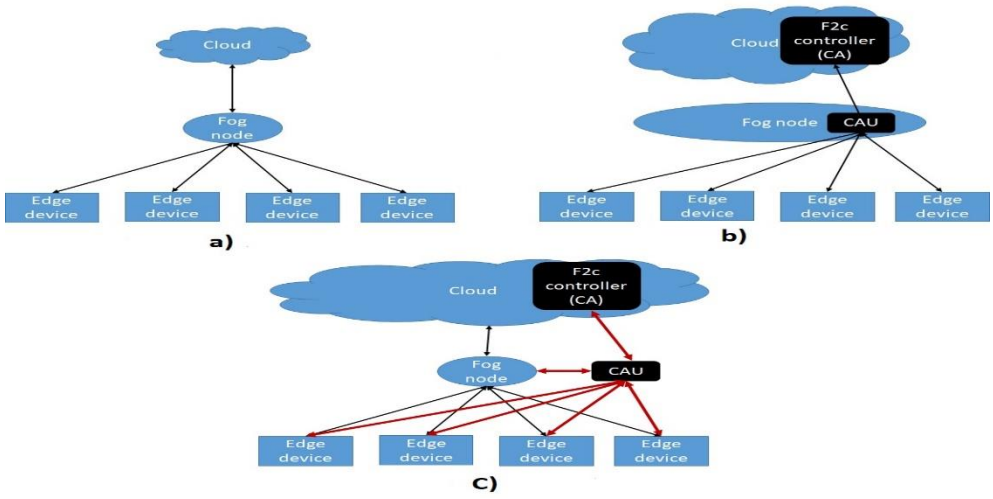

Figure 36. Security topologies: a) Non secure F2C (nF2C); b) Embedded CAUs F2C (ECF); c) Decoupled CAUs F2C (DCF).

a) The non-secure F2C scenario (nF2C), is built upon a Fujitsu Primergy TX300 S8 acting as cloud, a Raspberry Pi 3 (RP) implemented as fog node (Traffic Light), and 4 RPs acting as edge devices (Vehicles and Buildings).
b) The embedded CAUs scenario (ECF) deploys a Fujitsu Primergy TX300 S8, serving as both cloud and an embedded F2C controller that also plays as a certificate authority, a RP acting as a fog node and CAU (Traffic Light), controlling security for the corresponding area, and finally 4 RPs serving as edge devices (Vehicles).
c) The proposed decoupled security strategy (DCF), consists of a Fujitsu Primergy TX300 S8 acting as both cloud and embedded F2C controller (acting as a certificate authority), a RP acting as fog node (Building), 4 RPs serving as edge devices (Vehicles), and one RP acting as CAU (security provider for the corresponding area) decoupled from the fog node (Building).

For the sake of diversity, presented results are obtained over three distinct city services, each one with different requirements regarding the computational power or the immediate memory capacity:

1. Service (A) analyses a set of devices in the city and produces a score for each one, returning the best scored device. This service requires computational power (i.e., high CPU demand) to achieve the objective in a short space of time, and it is using protected information from the devices like their available resources.
2. Service (B) collects information from a set of sensors in the city in a defined period of time (i.e., high memory demand), and computes some statistical results at the end, returning the digested information to the user.
3. Service (C) computes all possible paths for a given set of vehicles in the city and their expected destination. The service tries to avoid congestion by selecting the best set of paths (i.e., high CPU and memory demand). When the model is computed, the result is sent to the city manager to trigger the necessary changes in the city and send the information to the vehicles.

The analysis between the three different services when the three different security strategies are deployed, in terms of both, service allocation (delivery time delay required by the ECF and DCF workflows) and services blocking (number of non-delivered services), are shown in the Figure 37 and Figure 38 to illustrate how the proposed solution impacts on the delivered QoS.

In terms of services allocation time delay, Figure 37 represents the overall delay when allocating 25, 50, 75 and 100 services. Analysing the nF2C scenario, the obtained results shows that although the time delay increases for all services, DCF keeps lower than the other two strategies (ECF and nF2C) where security are considered. Indeed, Figure 37a illustrates how service A time delay increases smoothly with the number of service requests due to the lower impact on memory, CPU, and network. Differently, Figure 37b shows how service B time delay increases with the number of service requests due to its high memory needs. Finally, service C time delay increases severely due to its high memory utilization as shown in Figure 37c. Analysing the ECF strategy, the observation shows that time delay notably increases for all services, which seems reasonable due to the specific time consuming tasks associated to security provisioning (i.e., authentication, key distribution, encryption, and secure channel processing). Hence, as expected, providing security in a F2C system notably impacts on the delivered quality in terms of service execution and allocation time delay. Nevertheless, Figure 37 also illustrates that when deploying DCF the time delay is substantially reduced compared to the one shown by ECF and close to the one presented by nF2C for Service C.

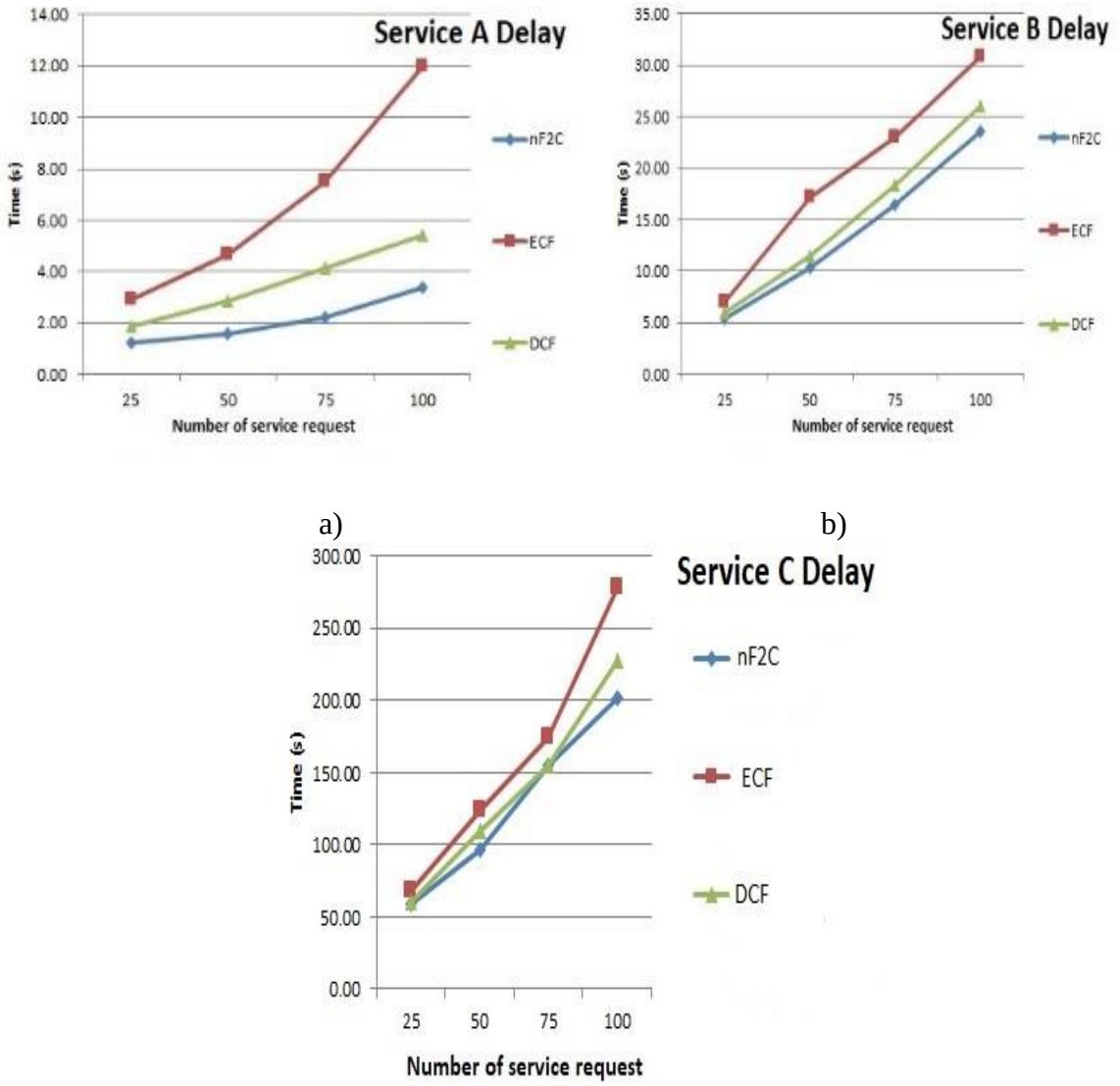

Figure 37. Service allocation time delay: (a) Service A time delay; (b) Service B time delay; (c) Service C time delay

Figure 38 represents the service blocking, standing for the set of denied service requests sent by edge devices to their fog node –the consideration is that the denial of a service is motivated by the fog node not having resources enough (CPU, memory, or network). The obtained results prove that when no security is applied (nF2C) the blocking keeps insignificant (0 for services A and C and very low for service B due to its high demanding CPU). However, when security is applied through the ECF strategy, the blocking starts increasing smoothly but exponentially when the number of requests grows, hence driving non-negligible effects on the delivered QoS. In the DCF strategy, service blocking gets a bit higher than nF2C while providing demanded security.

Nevertheless, Figure 37 and Figure 38also show that the deployment of the DCF strategy presents values similar to the nF2C scenario, hence removing the negative effects of the ECF strategy.

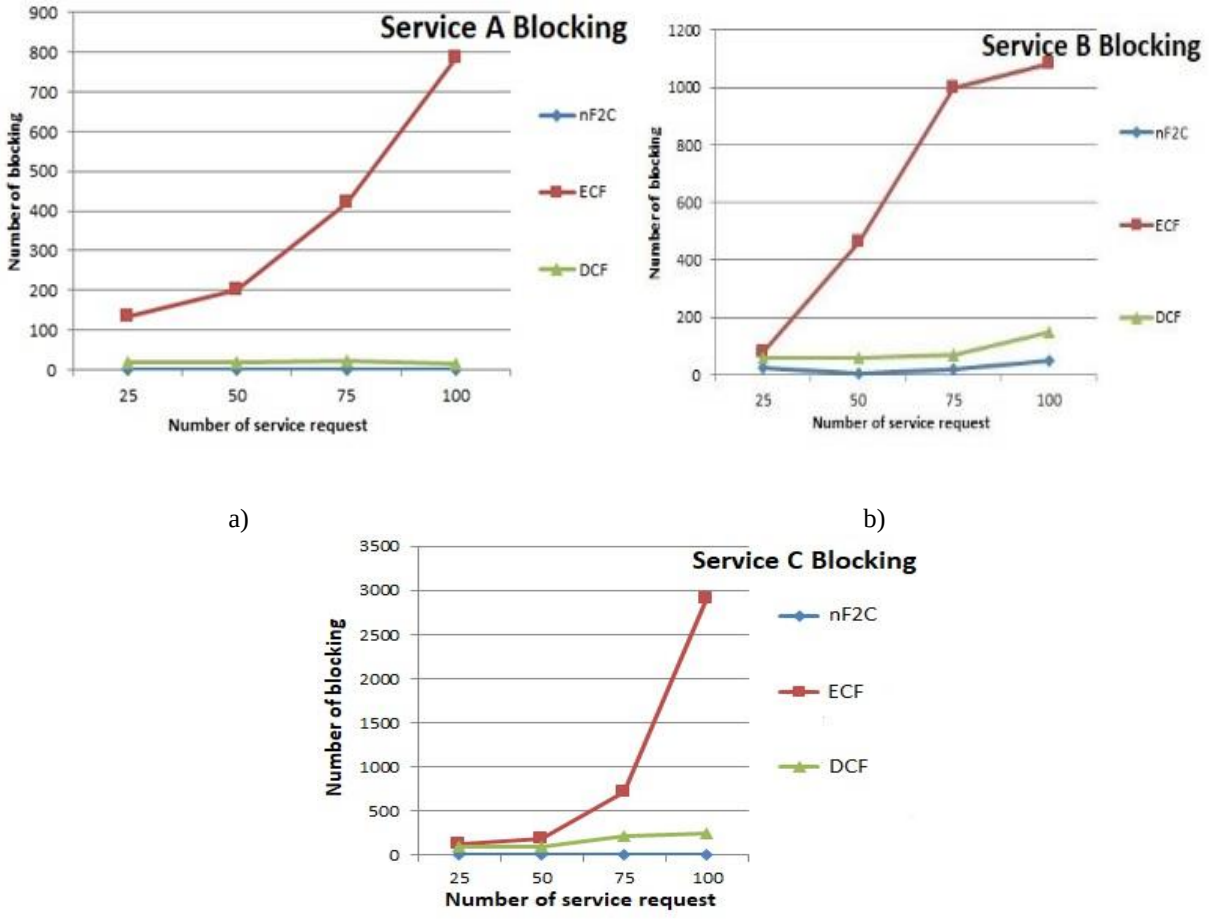

Figure 38. Service blocking: (a) service A; (b) service B; (c) service C

From an statistical analysis, the implemented workflows (ECF and DCF) and nF2C, are run for each type of service (A, B, C) by (25, 50, 75, 100) deploying the three different security strategies, 20 times in terms of both service allocation and services blocking. Figure 39 and Figure 40 show all the obtained results for each type of service, in terms of time delay and service blocking. The obtained results prove that each sample was fitted to Normal distribution by the Ryan-Joiner test (p-value >0.1). The population parameters for these distributions by statistics and 95% confidence intervals was estimated. And finally, means difference and variance ratio tests were applied to compare the corresponding population parameters. The level of significance was set at 99% for all the tests. The obtained result shows that when deploying the DCF strategy the time delay mean and service blocking mean are substantially reduced compared to the one shown by ECF **(μDCF < μECF)**, while standard deviations could be considered of equal order of magnitude (σDCF = σECF). All statistical analyses were done using Minitab, Inc.

Sample	Mean	StDev	95% CI for μ
nF2C 25 A	1.2020	0.0822	(1.1635, 1.2405)
nF2C 50 A	1.4695	0.0516	(1.4454, 1.4936)
nF2C 75 A	2.1385	0.0634	(2.1088, 2.1682)
nF2C 100 A	3.3695	0.0519	(3.3452, 3.3938)
ECF 25 A	2.97800	0.04408	(2.95737, 2.99863)
ECF 50 A	4.6065	0.0559	(4.5803, 4.6327)
ECF 75 A	7.4030	0.0709	(7.3698, 7.4362)
ECF 100 A	12.380	1.809	(11.533, 13.227)
DCF 25 A	1.7650	0.0550	(1.7393, 1.7907)
DCF 50 A	2.7935	0.0632	(2.7639, 2.8231)
DCF 75 A	4.0830	0.0516	(4.0588, 4.1072)
DCF 100 A	5.3895	0.0634	(5.3598, 5.4192)
nF2C 25 B	5.1495	0.0715	(5.1160, 5.1830)
nF2C 50 B	10.2645	0.0416	(10.2450, 10.2840)
nF2C 75 B	16.3710	0.0533	(16.3461, 16.3959)
nF2C 100 B	23.3725	0.1381	(23.3079, 23.4371)
ECF 25 B	6.9710	0.0566	(6.9445, 6.9975)
ECF 50 B	17.0860	0.0499	(17.0627, 17.1093)
ECF 75 B	22.9000	0.0550	(22.8743, 22.9257)
ECF 100 B	29.52	6.05	(26.69, 32.36)
DCF 25 B	5.9570	0.0728	(5.9229, 5.9911)
DCF 50 B	11.5880	0.0878	(11.5469, 11.6291)
DCF 75 B	18.3325	0.1166	(18.2779, 18.3871)
DCF 100 B	25.9655	0.1008	(25.9183, 26.0127)
nF2C 25 C	58.9965	0.0603	(58.9683, 59.0247)
nF2C 50 C	96.2790	0.0713	(96.2456, 96.3124)
nF2C 75 C	155.158	0.109	(155.107, 155.208)
nF2C 100 C	202.311	0.086	(202.271, 202.351)
ECF 25 C	65.62	11.17	(60.39, 70.85)
ECF 50 C	124.116	0.071	(124.083, 124.149)
ECF 75 C	174.943	0.085	(174.903, 174.983)
ECF 100 C	278.120	0.078	(278.084, 278.157)
DCF 25 C	59.5800	0.1152	(59.5261, 59.6339)
DCF 50 C	110.046	0.200	(109.953, 110.140)
DCF 75 C	154.989	0.078	(154.953, 155.025)
DCF 100 C	217.1	45.9	(195.7, 238.6)

Figure 39. Statistical analysis (Time Delays)

Sample	Mean	StDev	95% CI for μ
nF2C 25 A	0.000000	0.000000	(0.000000, 0.000000)
nF2C 50 A	0.000000	0.000000	(0.000000, 0.000000)
nF2C 75 A	0.000000	0.000000	(0.000000, 0.000000)
nF2C 100 A	0.000000	0.000000	(0.000000, 0.000000)
EFC 25 A	125.47	9.83	(120.73, 130.21)
EFC 50 A	199.00	10.84	(193.78, 204.22)
EFC 75 A	412.74	13.78	(406.10, 419.38)
EFC 100 A	776.37	11.81	(770.68, 782.06)
DFC 25 A	19.632	3.095	(18.140, 21.123)
DFC 50 A	16.000	3.215	(14.451, 17.549)
DFC 75 A	21.368	3.531	(19.667, 23.070)
DFC 100 A	14.000	3.859	(12.140, 15.860)
nF2C 25 B	16.68	5.45	(14.06, 19.31)
nF2C 50 B	8.474	3.580	(6.748, 10.199)
nF2C 75 B	19.00	4.98	(16.60, 21.40)
nF2C 100 B	48.84	8.46	(44.76, 52.92)
EFC 25 B	81.21	8.05	(77.33, 85.09)
EFC 50 B	464.58	11.81	(458.89, 470.27)
EFC 75 B	1000.32	9.29	(995.84, 1004.79)
EFC 100 B	1081.21	14.11	(1074.41, 1088.01)
DFC 25 B	57.16	9.28	(52.69, 61.63)
DFC 50 B	62.53	7.89	(58.72, 66.33)
DFC 75 B	70.89	8.06	(67.01, 74.78)
DFC 100 B	159.58	13.44	(153.10, 166.06)
nF2C 25 C	0.000000	0.000000	(0.000000, 0.000000)
nF2C 50 C	0.000000	0.000000	(0.000000, 0.000000)
nF2C 75 C	0.000000	0.000000	(0.000000, 0.000000)
nF2C 100 C	0.000000	0.000000	(0.000000, 0.000000)
EFC 25 C	129.42	13.36	(122.98, 135.86)
EFC 50 C	183.53	10.64	(178.40, 188.65)
EFC 75 C	712.42	10.67	(707.28, 717.57)
EFC 100 C	2909.16	13.49	(2902.66, 2915.66)
DFC 25 C	99.95	5.70	(97.20, 102.69)
DFC 50 C	98.42	7.84	(94.64, 102.20)
DFC 75 C	217.74	7.95	(213.90, 221.57)
DFC 100 C	253.74	10.19	(248.82, 258.65)

Figure 40. Statistical analysis (Service Blocking)

The obtained results and statistical analysis are shown for both, time delay and service blocking, the conclusion is that the DCF security architecture is providing security to the F2C system with less impact on the QoS. Therefore, DCF is most sophisticated and a more reasonable security architecture for F2C, providing security with the demanded QoS.

Chapter.10 Conclusion

By growing numbers of devices rapidly at the edge of the network, a new hierarchical computation model (F2C) was proposed for facilitating service discovery, categorization, allocation and execution. For providing security in F2C system, all the involved layers such as cloud, fog layer, and IoT must be analyzed in cooperative way. Traditional cloud security provider is not suitable for the F2C system due to be a single-point-of-failure (conceptual centralized cloud), and because handling such a huge number of devices at the edge can bring scalability issues. On the other hand, fog devices act as security provider for the edge device might bring QoS issues due to medium computational power of fog device and security provisioning consume huge amount of fog device's resources. Even, existing cloud's security solution cannot be applied into the fog devices because fog devices has less computational power than cloud.

Therefore, in this thesis, all security requirements, challenges, considerations, attacks and existing solution in all F2C layers are deeply analyzed as theoretical objectives. Finally, as technical objective, a novel decoupled transversal security architecture has been proposed. The security architecture has a centralized component (F2C controller) at the cloud and distributed security controllers (CAUs) at the edge of the network for providing security for fog and edge devices. The implemented CAUs are able to provide authentication, key management, secure channel establishment, access control and secure storage for all their corresponding fog and edge devices. According to obtained results in chapter 9, CAUs provide authentication in less time, key management in less time and less network overhead, access control (in terms of data storing and data retrieving) less time compare to the centralized cloud, and finally the DCF architecture provides security with less impact on the time delay and service blocking (less impact on QoS). The proposed security architecture provides the demanded security with less impact on QoS as shown and proved by obtained results in chapter 9.

The proposed security architecture is capable of providing distributed security that revoke the single-point-of-failure for the F2C system. The architecture improves QoS compared to the centralized traditionally cloud as security provider. Although, still some issues remaining such as CAU's discovery, CAU's selection, and the number of devices that CAU can handle, etc. The mentioned issues are ongoing research for our future work.

The outcomes of thesis's contribution are:

1. S.Kahvazadeh, V.Barbosa, X.Masip-Bruin, E.Marín-Tordera, J.Garcia, R.Diaz, "Securing combined Fog-to-Cloud System through SDN approach", CrossCloud 2017
2. S.Kahvazadeh, V.Barbosa, X.Masip-Bruin, E.Marín-Tordera, J.Garcia, R.Diaz, "An SDN-based Architecture for Security Provisioning in Fog-to-Cloud (F2C) Computing Systems", FTC 2017
3. S.Kahvazadeh, X.Masip-Bruin, R.Diaz, E.Marín-Tordera, A.Jurnet, J.Garcia, "Towards an Efficient Key Management and Authentication Strategy for Combined Fog-to-Cloud Continuum Systems", Things, CIoT 2018
4. S. Kahvazadeh, X. Masip-Bruin, R. Díaz , E. Marín-Tordera, A. Jurnet, J. Garcia, A. Juan, E. Simó, Balancing Security Guarantees vs QoS Provisioning in Combined Fog-to-cloud systems, 10th IFIP International Conference on New Technologies, Mobility & Security
5. S.Kahvazadeh, X.Masip-Bruin, Eva Marín-Tordera, A.Gómez, Securing Combined Fog-to-Cloud Systems: Challenges and Directions, FTC 2019
6. S.Kahvazadeh, X.Masip-Bruin, Pau Marcer, Eva Marín-Tordera, Deploying Fog-to-Cloud Towards a Security Architecture for Critical Infrastructure Scenarios, IOsec 2019
7. A.Gómez, X.Masip-Bruin, E.Marín-Tordera, S.Kahvazadeh, J.Garcia, "A Hash-Based Naming Strategy for the Fog-to-Cloud Computing Paradigm", F2C-DP Workshop, Euro-Par 2017, LNCS 10659, Santiago de Compostela, September 2017
8. A.Gómez, X.Masip-Bruin, E.Marín-Tordera, S.Kahvazadeh, J.García, "A Resource Identity Management Strategy for Combined Fog-to-Cloud Systems", IoT-SoS 2018
9. A.Gómez, X.Masip-Bruin, E.Marín-Tordera, S.Kahvazadeh, "A Novel and Scalable Naming Strategy for IoT Scenarios", FTC 2018
10. A.Gómez, X.Masip-Bruin, E.Marín-Tordera, S.Kahvazadeh, Resource Identification in Fog-to-Cloud Systems: Toward an Identity Management Strategy", Journal of Reliable Intelligent Environments 2018
11. Souvik Sengupta, Sarang Kahvazadeh, Xavi Masip-Bruin, Jordi Garcia, SFDDM: A Secure Distributed Database Management in Combined Fog-to-Cloud Systems, CAMAD 2019

References

[1] Q Zhang, L Cheng, R Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications,* 2010.

[2] B.P Rimal, E. Choi, I. Lump, "A Taxonomy and Survey of Cloud Computing Systems," in *Fifth International Joint Conference on INC, IMS and IDC*, 2009.

[3] F Bonomi, R Milito, J Zhu, S Addepalli, "Fog computing and its role in the internet of things," in *MCC workshop on Mobile cloud computing*, 2012.

[4] X Masip-Bruin, E Marín-Tordera, G. Tashakor, A. Jukan, GJ. Ren, "Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems," *IEEE Wireless Communications ,* 2016.

[5] FA Alaba, M Othman, IAT Hashem, F Alotaibi, "Internet of Things security: A survey," *Journal of Network and Computer Applications,* 2017.

[6] D. Puthal, B.P.S. Sahoo, S. Mishra, S. Swain, "Cloud Computing Features, Issues, and Challenges: A Big Picture," in *International Conference on Computational Intelligence and Networks*, 2015.

[7] I Attiya, X Zhang, "Cloud computing technology: Promises and concerns," *International Journal of Computer Applications,* 2017.

[8] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, D. Leaf, "NIST Cloud Computing Reference Architecture," *National Institute of Standards and Technology.*

[9] I Khalil, A Khreishah, M Azeem, "Cloud computing security: A survey," *Computers,* 2014.

[10] H Atlam, R Walters, G Wills, "Fog computing and the internet of things: a review," *big data and cognitive computing,* 2018.

[11] S Yi, C Li, Q Li, "A survey of fog computing: concepts, applications and issues," in *Workshop on Mobile Big Data*, 2015.

[12] SB Nath, H Gupta, S Chakraborty, SK Ghosh, "A survey of fog computing and communication: current researches and future directions," *Networking and Internet Architecture ,* 2018.

[13] P. Bellavista, J. Berrocal, A. Corradi, S.K Das, L.Foschini, A. Zanni, "A surevey on fog computing for Internet of Things," *Pervasive and mobile computing,* 2019.

[14] RK Naha, S Garg, D Georgakopoulos, PP Jayaraman, L. Gao, Y. Xiang, R. Ranjan, "Fog Computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access,* 2018.

[15] Z Hao, E Novak, S Yi, Q Li , "Challenges and software architecture for fog computing," *IEEE Internet Computing,* 2017.

[16] M Mukherjee, L Shu, D Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys & Tutorials ,* 2018.

[17] C Mouradian, D Naboulsi, S Yangui, RH. Glitho, MJ. Morrow, PA. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials ,* 2017.

[18] P Hu, S Dhelim, H Ning, T Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of Network and Computer Applications,* 2017.

[19] F Haouari, R Faraj, JM AlJa'am, "Fog Computing Potentials, Applications, and Challenges," in *International Conference on Computer and Applications (ICCA)*, 2018.

[20] S Li, L Da Xu, S Zhao , "The internet of things: a survey," *Information Systems Frontiers,* 2015.

[21] Mazhar Ali, Samee U.Khan, Athanasios V.Vasilakos, "Security in cloud computing: Opportunities and challenges," *Information science, ELSEVIER,* vol. 305, 2015.

[22] Saurabh Singh, Young-Sik Jeong, Jong Hyuk Park, "A survey on cloud computing security: Issues, threats, and solutions," *Journal of Network and Computer Applications (ELSEVIER),* vol. 75, 2016.

[23] A Singh, K Chatterjee, "Cloud security issues and challenges: A survey," *Journal of Network and Computer Applications,* 2017.

[24] Fariba Ghaffari, Hossein Gharaee, Mohammad Reza Forouzandehdoust, "Security considerations and requirements for Cloud computing," in *International Symposium on Telecommunications (IST)*, 2016.

[25] Jun Zhou, Zhenfu Cao, Xiaolei Dong, Athanasios V. Vasilakos, "Security and Privacy for Cloud-Based IoT: Challenges, Countermeasures, and Future Directions," *IEEE Communications Magazine,* vol. 55, no. 1, 2017.

[26] L Coppolino, S D'Antonio, G Mazzeo, L Romano, "Cloud security: Emerging threats and current solutions," *Computers & Electrical Engineering,* 2016.

[27] C Modi, D Patel, B Borisaniya, A Patel, M Rajarajan, "A survey on security issues and solutions at different layers of Cloud computing," *The Journal of Supercomputing,* 2013.

[28] R Roman, J Lopez, M Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems,* 2018.

[29] S Khan, S Parkinson, Y Qin , "Fog computing security: a review of current applications and security solutions," *Journal of Cloud Computing,* 2017.

[30] K Lee, D Kim, D Ha, U Rajput, H Oh, "On security and privacy issues of fog computing supported Internet of Things environment," in *6th International Conference on the Network of the Future (NOF)*, 2015.

[31] Jianbing Ni, Kuan Zhang, Xiaodong Lin, Xuemin Sherman Shen, "Securing Fog Computing for Internet of Things Applications: Challenges and Solutions," *IEEE Communications Surveys & Tutorials,* vol. 20, 2017.

[32] Mithun Mukherjee, Rakesh Matam, Lei Shu, Leandros Maglaras, Mohamed Amine Ferrag, Nikumani Choudhury, Vikas Kumar, "Security and Privacy in Fog Computing: Challenges," *IEEE Access,* vol. 5, 2017.

[33] Ivan Stojmenovic, Sheng Wen, Xinyi Huang, Hao Luan, "An overview of Fog computing and its security issues," *Wiley online library,* 2015.

[34] SA Kumar, T Vealey, H Srivastava, "Security in internet of things: Challenges, solutions and future directions," in *49th Hawaii International Conference on System Sciences (HICSS)*, 2016.

[35] V Adat, BB Gupta , "Security in Internet of Things: issues, challenges, taxonomy, and architecture," *Telecommunication Systems,* 2018.

[36] Subho Shankar Basu, Somanath Tripathy, Atanu Roy Chowdhury, "Design challenges and security issues in the Internet of Things," in *IEEE Region 10 Symposium (TENSYMP)*, 2015.

[37] V. Chellappan, K.M. Sivalingam, "Security and privacy in the Internet of Things," *Internet of Things Principles and Paradigms,* pp. 183-200, 2016.

[38] Md. Mahmud Hossain, Maziar Fotouhi, Ragib Hasan, "Towards an Analysis of Security Issues, Challenges, and Open Problems in the Internet of Things," in *IEEE World Congress on Services*, 2015.

[39] Jie Lin, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang, Wei Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," *IEEE Internet of Things Journal,* 2017.

[40] Syrine Sahmim, Hamza Gharsellaoui, "Privacy and Security in Internet-based Computing: Cloud Computing, Internet of Things, Cloud of Things: a review," *Procedia Computer Science, Elsevier B.V.,* vol. 112, 2017.

[41] Meryeme Alouane, Hanan El Bakkali, "Security, privacy and trust in cloud computing: A comparative study," in *Cloud Technologies and Applications (CloudTech)*, 2015.

[42] Gururaj Ramachandra, Mohsin Iftikhar, Farrukh Aslam Khan, "A Comprehensive Survey on Security in Cloud Computing," *Procedia computer Science (ELSEVIER B. V.),* vol. 110, 2017.

[43] K. Padmaja, R. Seshadri, "A Review on Cloud Computing Technologies and Security Issues," *Indian Journal of Science and Technology,* 2016.

[44] Bridget A. Martin, Frank Michaud, Don Banks, Arsalan Mosenia, Riaz Zolfonoon, Susanto Irwan, Sven Schrecker, John K. Zao, "OpenFog Security Requirements and Approaches," in *Fog world congress*, 2017.

[45] Arwa Alrawais, Abdulrahman Alhothaily, Chunqiang Hu, Xiuzhen Cheng, "Fog Computing for the Internet of Things: Security and Privacy Issues," *IEEE Internet Computing,* vol. 21, 2017.

[46] Yifan Wang, Tetsutaro Uehara, Ryoichi Sasaki, "Fog Computing: Issues and Challenges in Security and Forensics," in *IEEE 39th Computer Software and Applications Conference (COMPSAC)*, 2015.

[47] Bushra Zaheer Abbasi, Munam Ali Shah, "Fog computing: Security issues, solutions and robust practices," in *23rd International Conference on Automation and Computing (ICAC)*, 2017.

[48] Emmanouil Vasilomanolakis, Jörg Daubert, Manisha Luthra, Vangelis Gazis, Alex Wiesmaier, Panayotis Kikiras, "On the Security and Privacy of Internet of Things Architectures and Systems," in *International Workshop on Secure Internet of Things (SIoT)*, 2015.

[49] Mahmoud Elkhodr, Seyed Shahrestani, Hon Cheung, "The Internet of Things: New Interoperability, Management and Security Challenges," *International Journal of Network Security & Its Applications (IJNSA),* 2016.

[50] Arbia Riahi Sfar, Enrico Natalizio, Yacine Challal, Zied Chtourou, "A roadmap for security challenges in the Internet of Things," *Digital Communications and Networks,* 2017.

[51] Mario Weber, Marija Boban, "Security challenges of the internet of things," in *39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2016.

[52] Ali H.Ahmed, Nagwa M. Omar, and Hosny M. Ibrahim, "Modern IoT Architectures Review: A Security," in *8th Annual International Conference on ICT: Big Data, Cloud and Security (ICT-BDCS)*, 2017.

[53] A Gómez-Cárdenas, X Masip-Bruin, E Marín-Tordera, S Kahvazadeh, J Garcia, "A Hash-Based Naming Strategy for the Fog-to-Cloud Computing Paradigm," in *Euro-Par: Parallel Processing Workshops* , 2017.

[54] "https://www.iiconsortium.org/index.htm".

[55] Sabout Nagaraju , Latha Parthiban, "SecAuthn: Provably Secure Multi-Factor Authentication for the Cloud Computing Systems," *Indian Journal of Science and Technology,* 2016.

[56] Fan Kai, Deng Hai, Li Hui, Yang Yintang, "Privacy Protection Smartcard Authentication Scheme in Cloud Computing," *Chinese Journal of Electronics,* 2018.

[57] Yannan Li, Yong Yu, BoYang Geyong Min, Huai Wu, "Privacy preserving cloud data auditing with efficient key update," *Future Generation Computer Systems, elsevier,* vol. 78, 2018.

[58] Geeta Sharma, Sheetal Kalra, "Identity based secure authentication scheme based on quantum key distribution for cloud computing," *Peer-to-Peer Networking and Applications,* 2018.

[59] A Ahmed-Nacer, MAN Samovar, "Strong authentication for mobile cloud computing," in *international conference on new technologies for distributed systems (NOTERE)*, 2016.

[60] Chia-Mu Yu, Chi-Yuan Chen, Han-Chieh Chao, "Privacy-preserving multikeyword similarity search over outsourced cloud data," *IEEE systems journal,* vol. 11, 2017.

[61] I Butun, M Erol-Kantarci, B Kantarci, H. Song, "Cloud-centric multi-level authentication as a service for secure public safety device networks," *IEEE communications magazine ,* 2016.

[62] BP Gajendra, VK Singh, M Sujeet, "Achieving cloud security using third party auditor, MD5 and identity-based encryption," in *International conference on computing, communication and automation (ICCCA)*, 2016.

[63] Hong Liu, Huansheng Ning, Yinliang Yue, Laurence T. Yang, "Selective disclosure and yoking-proof based privacy-preserving authentication scheme for cloud assisted wearable devices," *Future Generation Computer Systems, Elsevier ,* vol. 78, 2018.

[64] M Marwan, A Kartit, H Ouahmane, "Applying homomorphic encryption for securing cloud database," in *IEEE International Colloquium on Information Science and Technology (CiSt)* , 2016.

[65] JK Liu, K Liang, W Susilo, J Liu, Y Xiang , "Two-factor data security protection mechanism for cloud storage system," *IEEE Transactions on computers,* 2016.

[66] Sana Belguith, Abderrazak Jemai, Rabah Attia, "Enhancing Data Security in Cloud Computing Using a Lightweight Cryptographic," in *International Conference on Autonomic and Autonomous Systems (ICAS)*, 2015.

[67] AC Donald, L Arockiam, "A secure authentication scheme for MobiCloud," in *International conference on computer communication and informatics (ICCCI)*, 2015.

[68] Abdul Nasir Khan, M. L. Mat Kiah, Sajjad A. Madani, Mazhar Ali, Atta ur Rehman Khan, Shahaboddin Shamshirband, "Incremental proxy re-encryption scheme for mobile cloud computing environment," *The Journal of Supercomputing,* 2014.

[69] MS Kiraz, I Sertkaya, O Uzunkol, "An efficient ID-based message recoverable privacy-preserving auditing scheme," in *Conference on privacy, security and trust (RST)*, 2015.

[70] CN Yang, JB Lai, "Protecting data privacy and security for cloud computing based on secret sharing," in *International symposium on Biometrics and Security Technologies*, 2013.

[71] J Gouveia, SM de Sousa, R Azevedo, "E-id authentication and uniform access to cloud storage service providers," in *Cloud computing technology and science*, 2013.

[72] RK Banyal, P Jain, VK Jain , "Multi-factor authentication framework for cloud computing," in *Computational Intelligence, Modelling and Simulation (CIMSim)*, 2013.

[73] Jin Li, Yinghui Zhang, Xiaofeng Chen, Yang Xiang, "Secure attribute-based data sharing for resource-limited users in cloud computing," *Computers & Security, elsevier,* 2018.

[74] Yinghui Zhang, Xiaofeng Chen, Jin Li, Duncan S.Wong, Hui Li, Ilsun You, "Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing," *Information Sciences, elsevier,* vol. 379, 2017.

[75] Qinlong Huang, Yixian Yang, Mansuo Shen, "Secure and efficient data collaboration with hierarchical attribute-based encryption in cloud computing," *Future Generation Computer Systems, Elsevier,* vol. 72, 2017.

[76] Faria Mehak, Rahat Masood, Muhammad Awais Shibli, Islam Elgedway, "EACF: extensible access control framework for cloud environments," *Annals of Telecommunications, Springer ,* 2017.

[77] J Hur, DK Noh , "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Transactions on Parallel and Distributed systems,* 2011.

[78] MR Asghar, G Russello, B Crispo, "Supporting complex queries and access policies for multi-user encrypted databases," in *ACM workshop on Cloud computing security*, 2013.

[79] Jingwei Li, Anna Squicciarini, Dan Lin, "SecLoc: Securing Location-Sensitive Storage in the Cloud," in *ACM Symposium on Access Control Models and Technologies*, 2015.

[80] Lan Zhou, Vijay Varadharajan, Michael Hitchens, "Enforcing Role-Based Access Control for Secure Data Storage in the Cloud," *The Computer Journal,* 2011.

[81] V Goyal, O Pandey, A Sahai, B Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *ACM conference on Computer and communications security (CCS)*, 2006.

[82] CK Chu, SSM Chow, WG Tzeng, J Zhou, RH Deng, "Key-aggregate cryptosystem for scalable data sharing in cloud storage," *IEEE Transactions on Parallel and Distributed Systems,* 2014\.

[83] Z Liu, T Li, P Li, C Jia, J Li, "Verifiable searchable encryption with aggregate keys for data sharing system," *Future Generation Computer Systems, Elsevier,* 2018.

[84] M Nabeel, E Bertino, M Kantarcioglu, B Thuraisingham, "Towards privacy preserving access control in the cloud," in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, 2011.

[85] Li Yibin, Keke Gai, Longfei Qiu, Meikang Qiu, Zhao Hui, "Intelligent cryptography approach for secure distributed big data storage in cloud computing," *Information Sciences, elsevier,* vol. 387, 2017.

[86] Nithya Chidambaram, Pethuru Raj, K. Thenmozhi, Rengarajan Amirtharajan, "Enhancing the Security of Customer Data in Cloud Environments Using a Novel Digital Fingerprinting Technique," *International Journal of Digital Multimedia Broadcasting ,* 2016.

[87] HS Alqahtani, P Sant, "A Multi-Cloud Approach for Secure Data Storage on Smart Device," in *Digital Information and Communication Technology and its Applications (DICTAP)*, 2016.

[88] Mohammed M. Dawoud, Gamal A. Ebrahim, Sameh A. Youssef, "A Cloud Computing Security Framework Based on Cloud Security Trusted Authority," in *10th International Conference on Informatics and Systems (INFOS)*, 2016.

[89] Carlos Andre Batista de Carvalho, Miguel Franklin de Castro, Rossana Maria de Castro Andrade, "Secure cloud storage service for detection of security violations," in *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2017.

[90] Q Wang, C Wang, K Ren, W Lou, J Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE transactions on parallel and distributed systems,* 2011.

[91] Josenilson Dias Araújo, Zair Abdelouahab, "Virtualization in Intrusion Detection Systems: A Study on Different Approches for cloud computing environments," *IJCSNS International Journal of Computer Science and Network Security,* 2012.

[92] U Oktay, OK Sahingoz, "Proxy network intrusion detection system for cloud computing," in *Technological Advances in Electrical, Electronics and computer engineering (TAEECE)*, 2013.

[93] J Nikolai, Y Wang, "Hypervisor-based cloud intrusion detection system," in *Computing, Networking and Communications (ICNC)*, 2014.

[94] Yaser Jararweh, Mahmoud Al-Ayyoub, Lo'ai Tawalbeh, Ala Darabseh, Houbing Song, "Software-defined systems support for secure cloud computing based on data classification," *Annals of Telecommunications springer,* vol. 72, 2017.

[95] AA Diro, N Chilamkurti, N Kumar , "Lightweight cybersecurity schemes using elliptic curve cryptography in publish-subscribe fog computing," *Mobile Networks and Applications,* 2017.

[96] SR Moosavi, TNGia, E Nigussie, AM Rahmani, S Virtanen, H Tenhunen, J Isoahoa, "End-to-end security scheme for mobility enabled healthcare Internet of Things," *Future Generation Computer Systems,* 2016.

[97] H Wang, Z Wang, J Domingo-Ferrer, "Anonymous and secure aggregation scheme in fog-based public cloud computing," *Future Generation Computer Systems,* 2018.

[98] MH Ibrahim, "Octopus: An Edge-fog Mutual Authentication Scheme," *International journal of network security,* 2016.

[99] B Mukherjee, RL Neupane, P Calyam, "End-to-End IoT Security Middleware for Cloud-Fog Communication," in *Cyber Security and Cloud Computing (CSCloud)*, 2017.

[100] Y Zhang, Y Xiang, W Wu, A Alelaiwi, "A variant of password authenticated key exchange protocol," *Future Generation Computer Systems,* 2018.

[101] R Lu, K Heung, AH Lashkari, AA Ghorbani, "A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced IoT," *IEEE Access ,* 2017.

[102] A Alrawais, A Alhothaily, C Hu, X Xing, X Cheng, "An attribute-based encryption scheme to secure fog communications," *IEEE Access,* 2017.

[103] A Alotaibi, A Barnawi, M Buhari, "Attribute-Based Secure Data Sharing with Efficient Revocation in Fog Computing," *Journal of Information Security,* 2017.

[104] P Zhang, Z Chen, JK Liu, K Liang, H Liu, "An efficient access control scheme with outsourcing capability and attribute update for fog computing," *Future Generation Computer Systems,* 2018.

[105] Q Wang, D Chen, N Zhang, Z Ding, Z Qin, "PCP: A Privacy-Preserving Content-Based Publish–Subscribe Scheme With Differential Privacy in Fog Computing," *IEEE Access,* 2017.

[106] D Koo, J Hur, "Privacy-preserving deduplication of encrypted data with dynamic ownership management in fog computing," *Future Generation Computer Systems, Elsevier,* 2018.

[107] K Fan, J Wang, X Wang, H Li, Y Yang, "Secure, efficient and revocable data sharing scheme for vehicular fogs," *Peer-to-Peer Networking and Applications,* 2017.

[108] Q Huang, Y Yang, L Wang , "Secure Data Access Control With Ciphertext Update and Computation Outsourcing in Fog Computing for Internet of Things," *IEEE Access,* 2017.

[109] HA Al Hamid, SMM Rahman, MS Hossain, A Almogren, A Alamri, "A Security Model for Preserving the Privacy of Medical Big Data in a Healthcare Cloud Using a Fog Computing Facility With Pairing-Based Cryptography," *IEEE Access,* 2017.

[110] J Fu, Y Liu, HC Chao, B Bhargava, Z Zhang, "Secure Data Storage and Searching for Industrial IoT by Integrating Fog Computing and Cloud Computing," *IEEE Transactions on industrial informatics,* 2018.

[111] Lingjuan Lyu, Jiong Jin, Sutharshan Rajasegarar, Xuanli He, Marimuthu Palaniswami, "Fog-Empowered Anomaly Detection in IoT Using Hyperellipsoidal Clustering," *IEEE Internet of Things Journal ,* 2017.

[112] AS Sohal, R Sandhu, SK Sood, V Chang, "A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments," *Computers & Security,* 2018.

[113] VL Nguyen, PC Lin, RH Hwang, "MECPASS: Distributed Denial of Service Defense Architecture for Mobile Networks," *IEEE Network,* 2018.

[114] H Kim, EA Lee, "Authentication and Authorization for the Internet of Things," *IT Professional,* 2017.

[115] MB Mollah, MAK Azad, A Vasilakos, "Secure data sharing and Searching at the edge of cloud-assisted Internet of Things," *IEEE Cloud Computing,* 2017.

[116] J Kuusijärvi, R Savola, P Savolainen, a Evesti, "Mitigating IoT security threats with a trusted Network element," in *Internet Technology and Secured Transactions (ICITST)*, 2016.

[117] R Amin, N Kumar, GP Biswas, R Iqbal, V Chang, "A light weight authentication protocol for IoT-enabled devices in distributed Cloud Computing environment," *Future Generation Computer Systems,* 2018.

[118] W Bae, J Kwak, "Smart card-based secure authentication protocol in multi-server IoT environment," *Multimedia Tools and Applications,* 2017.

[119] J Choi, Y In, C Park, S Seok, H Seo, H Kim, "Secure IoT framework and 2D architecture for End-To-End security," *The Journal of Supercomputing,* 2016.

[120] S Raza, L Seitz, D Sitenkov, G Selander, "S3k: Scalable security with symmetric keys—dtls key establishment for the internet of things," *IEEE Transactions on Automation Science and Engineering,* 2016.

[121] P Porambage, A Braeken, P Kumar, A Gurtov, M Ylianttila, "Efficient key establishment for constrained IoT Devices with collaborative HIP-based approach," in *Global Communications Conference (GLOBECOM)*, 2015.

[122] O Salman, S Abdallah, IH Elhajj, A Chehab, A Kayssi, "Identity-based authentication scheme for the Internet of Things," in *IEEE Symposium on Computers and Communication (ISCC)*, 2016.

[123] MA Iqbal, M Bayoumi, "Secure End-to-End key establishment protocol for resource-constrained healthcare sensors in the context of IoT," in *High Performance Computing & Simulation (HPCS)*, 2016.

[124] KH Yeh, "A secure IoT-based healthcare system with body sensor networks," *IEEE Access,* 2016.

[125] M Azarmehr, A Ahmadi, R Rashidzadeh, "Secure authentication and access mechanism for IoT wireless sensors," in *International Symposium on Circuits and Systems (ISCAS)*, 2017.

[126] A Yousefi, SM Jameii , "Improving the security of internet of things using encryption algorithms," in *IoT and Application (ICIOT)*, 2017.

[127] MS Henriques, NK Vernekar, "Using symmetric and asymmetric cryptography to secure communication between devices in IoT," in *IoT and Application (ICIOT)*, 2017.

[128] Q Mamun, M Rana, "A partial key distribution protocol for WSNs in distributed IoT applications," in *Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2017.

[129] S Yoon, J Kim, "Remote security management server for IoT devices," in *Information and Communication Technology Convergence (ICTC)*, 2017.

[130] A Haroon, S Akram, MA Shah, A Wahid, "E-Lithe: A Lightweight Secure DTLS for IoT," in *Vehicular Technology Conference (VTC-Fall)*, 2017.

[131] DA Ha, KT Nguyen, JK Zao, "Efficient authentication of resource-constrained IoT devices based on ECQV implicit certificates and datagram transport layer security protocol," in *Symposium on Information and Communication Technology (SoICT)*, 2016.

[132] C Tselios, I Politis, S Kotsopoulos, "Enhancing SDN security for IoT-related deployments through blockchain," in *IEEE conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2017.

[133] KH Han, WS Bae, "Proposing and verifying a security protocol for hash function-based IoT communication system," *Cluster Computing,* 2016.

[134] P Porambage, C Schmitt, P Kumar, A Gurtov, M Ylianttila, "PAuthKey: A pervasive authentication protocol and key establishment scheme for wireless sensor networks in distributed IoT applications," *International Journal of Distributed Sensor Networks,* 2014.

[135] S Kalra, SK Sood, "Secure authentication scheme for IoT and cloud servers," *Pervasive and Mobile Computing,* 2015.

[136] YB Saied, A Olivereau, D Zeghlache, M Laurent, "Lightweight collaborative key establishment scheme for the Internet of Things," *Computer Networks,* 2014.

[137] J Liu, Y Xiao, CLP Chen, "Authentication and access control in the internet of things," in *International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 2012.

[138] W Wang, P Xu, LT Yang, "Secure Data Collection, Storage and Access in Cloud-Assisted IoT," *IEEE cloud computing,* 2018.

[139] M Ma, D He, N Kumar, KKR Choo, J Chen, "Certificateless searchable public key encryption scheme for industrial internet of things," *IEEE Transactions on Industrial Informatics,* 2018.

[140] SH Kim, IY Lee, "IoT device security based on roxy re-encrytion," *Journal of Ambient Intelligence and Humanized Computing,* 2017.

[141] L Touati, Y Challal, "Collaborative KP-ABE for cloud-based internet of things applications," in *IEEE International Conference on Communications (ICC)*, 2016.

[142] X Yao, Z Chen, Y Tian, "A lightweight attribute-based encryption scheme for the Internet of Things," *Future Generation Computer Systems,* 2015.

[143] V Odelu, AK Das, MK Khan, KKR Choo, M Jo, "Expressive CP-ABE scheme for mobile devices in IoT satisfying constant-size keys and ciphertexts," *IEEE Access,* 2017.

[144] C Gonzalez, SM Charfadine, O Flauzac, F Nolot, "SDN-based security framework for the IoT in distributed grid," in *Computer and Energy Science (SpliTech*, 2016.

[145] F Al Shuhaimi, M Jose, AV Singh, "Software defined network as solution to overcome security challenges in IoT," in *International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2016.

[146] P Bull, R Austin, E Popov, M Sharma, R Watson, "Flow based security for IoT devices using an SDN gateway," in *Future Internet of Things and Cloud (FiCloud)*, 2016.

[147] R Vilalta, R Ciungu, A Mayoral, R Casellas, R Martinez, D Pubill, J Serra, R Munoz, C Verikoukis, "Improving security in Internet of Things with software defined networking," in *Global Communications Conference (GLOBECOM)*, 2016.

[148] K Kalkan, S Zeadally, "Securing Internet of Things (IoT) with Software Defined Networking (SDN)," *IEEE Communications Magazine,* 2017.

[149] Thales L. von Sperling, Francisco L. de Caldas Filho, Rafael Timóteo de Sousa, Lucas M. C. e Martins, Rodrigo L. Rocha, "Tracking intruders in IoT networks by means of DNS traffic analysis," in *Workshop onCommunication Networks and Power Systems (WCNPS)*, 2017.

[150] J Pacheco, S Hariri, "IoT security framework for smart cyber infrastructures," in *Work shop on Foundations and Applications of Self* Systems*, 2016.

[151] VA Desnitsky, IV Kotenko, S. B. Nogin, "Detection of anomalies in data for monitoring of security components in the Internet of Things," in *Soft Computing and Measurements (SCM)*, 2015.

[152] S Kahvazadeh, VB Souza, X Masip-Bruin, E Marn-Tordera, J Garcia, R Diaz, "Securing combined Fog-to-Cloud system Through SDN Approach," in *Crosscloud*, 2017.

[153] S Kahvazadeh, VBC Souza, X Masip Bruin, E Marin Tordera, J Garcia, R Diaz, "An SDN-based architecture for security provisioning in Fog-to-Cloud (F2C) computing systems," in *FTC*, 2017.

[154] T Simon, "Critical infrastructure and the internet of things," in *Cyber Security in a Volatile World,*, 2017.

[155] "https://www.mf2c-project.eu/".

[156] S Kahvazadeh, X Masip-Bruin, R Diaz, E Marín-Tordera, A Jurnet, J Garcia, "Towards An Efficient Key Management and Authentication Strategy for Combined Fog-to-Cloud Continuum Systems," in *Cloudification of the Internet of Things (CIoT)*, 2018.

[157] National Institute of Standards and Technology. 1999. Recommended, "http://csrc.nist.gov/groups/ST/toolkit/documents/ dss/NISTReCur.pdf".

[158] "nvlpubs.nist.gov/nistpubs/FIPs".

[159] S Kahvazadeh, X Masip-Bruin, R Diaz, E Marn-Tordera, A Jurnet, J Garcia, A Juan, E Simo, "Balancing Security Guarantees vs QoS Provisioning in Combined Fog-to-cloud systems," in *10th IFIP International Conference on New Technologies, Mobility & Security (NTMS)*, 2019 .

[160] "https://tools.kali.org/informationgathering/".

[161] I.Indu, P.M.Rubesh Anand, Vidhyacharan Bhaskar, "Encrypted token based authentication with adapted SAML technology for cloud web services," *Journal of Network and Computer Applications,* vol. 99, 2017.

[162] Masood Shah, Abdul Salam Shah, Imran Ijaz, "Implementation of User Authentication as a Service for Cloud," *International Journal of Grid and Distributed Computing,* 2016.

[163] Chaimae Saadi, Habiba Chaoui, "A new approach to mitigate security threats in cloud environment," in *Second International Conference on Internet of things and Cloud Computing*, 2017.

[164] S Chandrasekhar, M Singhal , "Efficient and scalable query authentication for cloud-based storage systems with multiple data sources," *IEEE Transactions on Services computing,* vol. 10, 2017.

[165] Vinothkumar Muthurajan, Balaji Narayanasamy, "An Elliptic Curve Based Schnorr Cloud Security Model in Distributed Environment," *The Scientific World Journal,* 2016.

[166] L Wang, K Chen, X Mao, Y Wang, "Efficient and provably-secure certificateless proxy re-encryption scheme for secure cloud data sharing," *Journal of Shanghai Jiaotong University (Science),* 2014.

[167] P Gharjale, P Mohod, "Efficient public key cryptosystem for scalable data sharing in Cloud storage," in *Computation of Power, Energy, Information and communication*, 2015.

[168] RK Lomotey, R Deters, "Saas authentication middleware for mobile consumers of iaas cloud," in *IEEE Ninth World Congress on services*, 2013.

[169] H Kim, S.C Timm, "X.509 Authentication and Authorization in Fermi Cloud," in *IEEE/ACM 7th International Conference on Utility and Cloud Computing*, 2014.

[170] C Basescu, A Carpen-Amarie, "Managing data access on clouds: A generic framework for enforcing security policies," in *Advanced information networking and applications (AINA)*, 2011.

[171] TD Dang, D Hoang, "A data protection model for fog computing," in *Fog and Mobile Edge Computing (FMEC)* , 2017.

[172] Y Jiang, W Susilo, Y Mu, F Guo, "Ciphertext-policy attribute-based encryption against key-delegation abuse in fog computing," *Future Generation Computer Systems, Elsevier,* 2018.

[173] X Yang, F Yin, X Tang, "A Fine-Grained and Privacy-Preserving Query Scheme for Fog Computing-Enhanced Location-Based Service," *Sensors,* 2017.

[174] T Wang, J Zeng, MZA Bhuiyan, H Tian, Y Cai, Y Chen, B Zhong, "Trajectory privacy preservation based on a fog structure for Cloud location services," *IEEE Access,* 2017.

[175] O Salman, I Elhajj, A Chehab, A Kayssi, "Software Defined IoT security framework," in *Software Defined Systems (SDS)*, 2017.

[176] D Wang, B Da, J Li, R Li, "IBS enabled authentication for IoT in ION framework," in *Global Internet of Things Summit (GIoTS)*, 2017.

[177] C Stergiou, KE Psannis, BG Kim, B Gupta, "Secure integration of IoT and cloud computing," *Future Generation Computer systems,* 2018.

[178] H Iqbal, J Ma, Q Mu, V Ramaswamy, G Raymond, D Vivanco, J Zuena, "Augmenting security of internet-of-things using programmable network-centric approaches: a position paper," in *Computer Communication and Networks (ICCCN)*, 2017.

[179] M Vučinić, B Tourancheau, F Rousseau, A Duda, L Damon, R Guizzetti, "OSCAR: Object security architecture for the Internet of Things," *Ad Hoc Networks,* 2015.

[180] L Barreto, A Celesti, M Villari, M Fazio, A Puliafito, "Identity management in iot clouds: A fiware case of study," in *Communications and Network Security (CNS)*, 2015.

[181] S Jebri, M Abid, A Bouallegue, "An efficient scheme for anonymous communication in IoT," in *Information Assurance and Security (IAS)*, 2015.

[182] K Sha, R Errabelly, W Wei, TA Yang, Z Wang, "EdgeSec: Design of an Edge Layer Security Service to Enhance IoT Security," in *International Conference on Fog and Edge Computing (ICFEC)*, 2017.

[183] WS Bae, "Verifying a secure authentication protocol for IoT medical devices," *Cluster Computing,* 2017.