Encryption as a Service (EaaS): Introducing the Full-Cloud-Fog Architecture for Enhanced Performance and Security

Amir Javadpour, Forough Ja'fari, Tarik Taleb, Chafika Benzaïd, Yang Bin, and Yue Zhao

***Abstract*—The main goal of Encryption as a Service (EaaS) is to deliver cryptography services to limited-resource devices. However, due to the massive number of devices connecting EaaS platforms, they face challenging issues, such as high service delays and uncovered requests. The existing EaaS architectures lack adequate taking advantage of both cloud and fog layers, by which the performance can be improved. Therefore, this paper proposes a novel EaaS architecture called full-cloud-fog that focuses on increasing the EaaS throughput by locating the frequently accessed components on the fog layer and resolving resource allocations utilizing the cloud nodes. We have analyzed the security aspects of the proposed architecture and then implemented it in a real testbed. The evaluation results show that the proposed full-cloud-fog architecture improves the EaaS throughput by 81%.**

***Index Terms*—Encryption as a Service, EaaS Architecture, Cryptography Service, Fog Computing, Cloud Computing, Improved Throughput.**

I. Introduction

THE rise of communication technologies, such as 5G and high-speed wireless networks, and the growth of smart devices making them available at any moment have increased the amount of transferred data through such networks[1, 2, 3]. To get a better comprehension of the data exchange scale, TechJury has analyzed the Internet of Things (IoT) networks and estimated that, by the end of 2025, 27 billion devices might connect to the Internet, and they may produce 79.4 zettabytes of data [4, 5, 6, 7, 8]. Protecting this large-scale data from being accessed by cyber criminals and intruders requires abundant resources for executing cryptography algorithms and altering it to another form that is readable only by the permitted parties. However, numerous resource-limited devices, especially in IoT networks, cannot execute such algorithms [9, 10, 11, 12]. Hence, outsourcing the encryption services is demanded.

EaaS is the provision of cryptography services, such as secure storage [13, 10] and access control [14, 15], without expecting the end-devices to be involved in the processing of data [16, 17, 18]. Different components, such as the key manager and the encryptor/decryptor, participate in an EaaS platform to deliver the encryption service. Based on the EaaS architecture, these components may be located on different layers, including the cloud, fog, and device layers [19].

The existing architectures that are proposed for EaaS platforms are not sufficient to take advantage of both cloud and fog layers. Some of them (e.g., the one proposed by Deb et al. [20]) have just focused on reducing the service delivery time by using the fog nodes, and the others (e.g., the architecture proposed by Yang et al. [21]) have only paid attention to successfully handling the service requests by utilizing the cloud nodes. To enhance the performance of the Eaas platform, we propose applying the cloud and fog layers together. Specifically, we propose a novel EaaS architecture that covers the shortages of the current architectures. The proposed EaaS architecture (namely, full-cloud-fog) places the most frequently accessed components on the fog layer, and then tries to resolve resources (overall locations) using the cloud nodes. The key contributions of this paper are summarized as follows:

- Proposing a novel EaaS architecture that reaches an impressive throughput by keeping the delay and the unresponded requests low. The most frequently accessed components are located at the edge layer, while the components at the cloud layer handle the shortage of resources. The existing approaches do not consider this aspect (i.e., taking the benefits of each layer).
- Analyzing the security aspects of the proposed architecture and investigating the vulnerabilities. This helps the users and the developers of an EaaS platform that works based on the proposed architecture consider the security risks of being served by it and enhance it to be more protected and popular.

A. Goals of full-cloud-fog architecture for Encryption as a Service

The fast evolution of information and communication technologies has ushered in the IoT, wherein many interconnected devices collaboratively generate and exchange unprecedented data. This transformative trend holds immense potential to revolutionize industries and enhance

Corresponding author: Amir Javadpour (a.javadpour87@gmail.com),
Amir Javadpour is with ICTFICIAL Oy, Espoo, Finland.
He was with the Faculty of Information Technology and Electrical Engineering, University of Oulu (e-mail: a.javadpour87@gmail.com).
Forough Ja'fari is with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran (azadeh.mth@gmail.com).
Tarik Taleb is with the Faculty of Electrical Engineering and Information Technology, Ruhr University Bochum, Bochum, Germany (e-mail: tarik.taleb@rub.de, talebtarik@gmail.com).
Chafika Benzaïd Faculty of Information Technology and Electrical Engineering, University of Oulu, Finland (Chafika.Benzaid@oulu.fi).
Bin Yang is with the School of Computer and Information Engineering, Chuzhou University, Chuzhou 239000, China (yangbinchi@gmail.com).
Yue Zhao is with National Key Laboratory of Security Communication, Chengdu 610041, China (yuezhao@foxmail.com).

the quality of human life. However, with the proliferation of IoT devices and the data deluge they generate, the critical challenge of securing sensitive information has come to the forefront. As data becomes the lifeblood of modern society, ensuring its confidentiality, integrity, and availability has become an imperative. Encryption stands as a cornerstone of data protection. Yet, IoT devices' resource limitations, the data's scale and diversity, real-time processing requirements, and the dynamic nature of IoT networks present a complex puzzle. This research endeavors to leverage the synergies of cloud and fog computing, matching efficiency with security, and bridging the gap between data-intensive IoT applications and robust encryption mechanisms. By unraveling the intricate challenges that underscore the need for our proposed architecture, we aim to offer a transformative approach that safeguards data and propels the growth and adoption of IoT technologies into a secure and interconnected future. Refining EaaS Architecture for IoT, The core objective of this research paper is to introduce an inventive EaaS architecture meticulously designed to tackle the distinctive obstacles innate to IoT applications effectively. Introducing the ”full-cloud-fog” architecture, a framework designed to optimize the potential of cloud and fog computing for EaaS platforms in IoT ecosystems. This architecture aims to combine the strengths of cloud and fog computing, overcoming the limitations of traditional EaaS architectures. By using cloud resources for complex computations and enabling real-time data processing at the fog layer, it strives to improve performance and enhance data security in IoT and Microservices applications [22, 23].

• Addressing IoT-Specific Challenges: The paper identifies and addresses the intricate challenges of implementing data encryption and protection in IoT applications. This includes tackling issues related to resource-constrained IoT devices, managing the scale and volume of data, ensuring real-time processing and low latency, accommodating device heterogeneity and standardization, and mitigating insider and outsider threats.

• Comparative Analysis and Evaluation: Another research goal is to compare the proposed full-cloud-fog architecture with existing EaaS architectures. By evaluating the proposed architecture's performance, efficiency, security, and suitability for IoT scenarios, the paper aims to demonstrate its advantages over other approaches.

• Security Analysis and Vulnerability Assessment: The paper will thoroughly analyze the security aspects of the proposed architecture. It will investigate potential vulnerabilities and threats the architecture may face, providing insights for users and developers to enhance security measures and address risks effectively.

• Practical Implementation and Evaluation: A practical implementation of the proposed full-cloud-fog architecture using a real-world testbed of LXC containers is planned. This implementation will validate the architecture's feasibility and effectiveness, providing credible results to demonstrate its performance.

This research paper seeks to present a novel EaaS architecture for IoT applications, addressing the challenges posed by IoT's unique characteristics. By conducting a comprehensive literature review, proposing an innovative architecture, and evaluating its performance, the paper aims to contribute valuable insights and advancements to the EaaS field in the IoT field.

B. Motivation for a Hybrid Cloud-Fog EaaS Architecture

IoT devices and applications have created a surge in demand for efficient and scalable encryption services. Traditional cloud-only solutions, while powerful, often face challenges in meeting the performance and latency requirements of real-time IoT applications. Conversely, fog computing, with its localized processing capabilities, offers potential benefits but may not fully address the scalability and resource requirements of large-scale deployments. Our motivation for proposing a hybrid cloud-fog EaaS architecture stems from addressing these challenges and leveraging the strengths of both paradigms.

-Resources shows that IoT applications, such as smart healthcare and autonomous vehicles, require low-latency responses due to their real-time nature. Cloud-only solutions, despite their high computational power, can introduce latency due to the distance between devices and cloud data centers. According to recent studies, the latency introduced by cloud-based processing can exceed acceptable thresholds for time-sensitive applications [17, 24, 25]. Fog computing mitigates this issue by processing data closer to the source, thereby reducing latency. However, fog nodes alone may struggle with high-demand scenarios, making a hybrid approach advantageous.

As IoT deployments scale, the resource demands for encryption services increase significantly. Full cloud-based solutions, while scalable, may encounter bottlenecks due to network congestion or resource limitations at the data center level. Contrariwise, fog nodes, being geographically distributed, can handle local requests efficiently but may face challenges in scaling resources dynamically. A hybrid architecture balances the load between fog and cloud resources, providing a scalable solution that can adapt to varying demands. Empirical studies have demonstrated that hybrid approaches can handle peak loads more effectively compared to purely cloud-based systems [26]

- Cost considerations are critical in designing large-scale EaaS platforms. Cloud resources, especially when handling massive volumes of requests, can become expensive. Fog nodes, being localized, can reduce the need for extensive data transmission and lower operational costs. A hybrid approach allows for cost optimization by leveraging fog computing for routine processing tasks and utilizing cloud resources for more intensive computations or overflow situations. Real-world implementations have shown that hybrid models can achieve cost savings of up to 25 % compared to purely cloud-based solutions [26].

- Security and Privacy are important in encryption services. Fog computing enhances security by processing sensitive data locally, reducing exposure to potential breaches during transmission. However, fog nodes may not always have acceptable computational power for complex encryption tasks. The hybrid architecture ensures that high-security requirements are met by leveraging cloud resources for intensive computations while maintaining data privacy through localized processing.

- Adaptability to Diverse IoT Scenarios
IoT environments are diverse, with varying requirements for connectivity, computational resources, and data processing. A one-size-fits-all approach may not be feasible. The hybrid cloud-fog architecture provides the flexibility to adapt to different scenarios by dynamically allocating resources based on the specific needs of the application. This adaptability is supported by case studies in diverse IoT deployments that highlight the benefits of hybrid solutions in meeting varied requirements [17].

- Implementing the architecture in a real Testbed
Implement the proposed architecture in a real testbed of LXC containers and evaluate it using different metrics. The implemented testbed is a small real-world sample, so the results are credible. To the best of our knowledge, the current research in this field does not have such implementation for their reported results.

C. Challenges in IoT Data Protection and Encryption

The proposed Full-Cloud-Fog EaaS architecture extends beyond general applications, as it intricately incorporates design principles to address the distinct features of IoT applications. Recognizing the unique requirements and challenges posed by IoT, our architecture has been meticulously crafted to ensure seamless alignment with the intricate characteristics of IoT ecosystems. By delving into the intricacies of IoT environments, we have enriched the architecture's framework to embrace the following key aspects:

• Resource Optimization for IoT Devices: IoT devices often contend with resource limitations, such as constrained computational power and energy efficiency concerns. With this in mind, our architecture strategically allocates tasks to both cloud and fog layers. This orchestration enables efficient resource utilization while minimizing latency and conserving energy, a critical concern in the IoT landscape.

• Scalability to IoT Growth: As IoT networks continuously expand in scale, accommodating growing numbers of devices becomes a critical requirement. Our architecture's modular design provides the flexibility needed for effortless scalability. This means additional fog nodes can be seamlessly incorporated to handle burgeoning device connections and a surge in service requests.

• Data Sharing Capabilities: IoT ecosystems frequently demand secure and controlled data sharing among devices. The architecture's adeptness in managing shared data access through encryption and meticulous key distribution is specifically designed to resonate with IoT's intricate data-sharing requirements.

• Mitigation of Insider and Outsider Threats: In an IoT landscape characterized by diverse devices and sprawling attack surfaces, security is paramount. Our architecture is thoughtfully fortified to counter both insider and outsider threats. Integrating encryption, secure connections, and advanced anomaly detection mechanisms at the component level ensures comprehensive protection against potential adversarial activities.

• Interoperability and Heterogeneity: The IoT landscape is rife with device heterogeneity and varied communication protocols. Our architecture's inherent flexibility accommodates this diversity by permitting customization of encryption parameters and interaction protocols, fostering seamless interoperability among disparate devices and platforms.

• Real-time Edge Decision Making: IoT scenarios often mandate swift decision-making at the edge. The architecture's fog layer augments this capability by enabling efficient computation of data sources. This facilitates real-time decision-making processes and enhances the architecture's adaptability to IoT's dynamic requirements.

D. The remainder of this paper

The remainder of this paper is as follows. section II categorizes the current EaaS architectures into four groups and reviews the researchers using each. The proposed architecture is thoroughly described in section III, and its security analysis is also provided. The performance of the proposed architecture is analyzed in section IV and the implementation results are reported. Finally, section VI presents the summary of this research and the suggested future works.

II. Literature Review

There are four types of components in an EaaS platform: **General management Component (GC), Key management Component (KC), Encryption Component (EC), and Decryption Component (DC)** [27]. When a request is transmitted to the platform, the GC is responsible for selecting the suitable KC and EC/DC, forwarding the request to them, and finally sending back the response. The KC performs the EaaS functionalities related to the cryptography key(s), such as generating the keys or verifying the signature of a message. The EC and the DC are dedicated to the processes requiring encrypting a plaintext and decrypting a ciphertext, respectively [20]. It is worth noting that a node in an EaaS platform can play the role of multiple components. For example, a Linux machine on the cloud layer can handle both key management and encryption processes, that means the KC and EC are located on this machine.

The architecture of an EaaS platform specifies the layers the four mentioned components are located on. These components can be located on the cloud layer, the fog layer, or even the device layer. While the cloud layer consists of central and powerful machines, the fog layer is a set of distributed machines with fewer resources. However, the fog layer is closer to the device layer, and hence, the average end-to-end delay between the device and the fog nodes is lower than that between the devices and cloud nodes [28].

We have categorized the different EaaS architectures currently existing in the literature, into four types: full-cloud, half-cloud, half-fog, and half-cloud-fog. The term full indicates that the device layer is not involved in hosting any of the mentioned EaaS components. On the other hand, the term half means that at least one of the EaaS components is located on the device layer. In the existing EaaS platforms with a half-X architecture, only the EC and DC are located on the device layer. Table I generally describes the different types of architectures.

The full-cloud architecture is a paradigm of efficiency, uniting all operational components within the cloud layer to create a streamlined cryptography process. In response to a cryptography request initiated by a device, the orchestration of transmission rests within the hands of the Global Coordinator (GC), responsible for routing it to the appropriate Key Coordinator (KC). The KC undertakes the pivotal role of managing key-related operations, forwarding the original request and essential key-specific information to the selected Encryption/Decryption Component (EC/DC). This EC/DC entity conducts the crucial cryptographic tasks and subsequently dispatches the final response to the awaiting GC, facilitating its relay back to the requesting device. This architectural model is underscored by its capacity to optimize the allocation of computational resources within the cloud layer, thereby engendering efficient processing and seamless transmission. Notably, Yang et al. [21] significantly contributes to this architecture by introducing a robust mechanism for secure key generation and management, thoughtfully distributed across multiple KCs to enhance data privacy through stringent access controls. This approach, in line with the full-cloud paradigm, accentuates the elevated security achieved through judicious resource segregation. Moreover, El Bouchti et al. [29] presents an EaaS platform tailored specifically for healthcare devices, wherein devices are empowered with direct control over their cryptographic processes. Similarly, Ibtihal et al. [30] brings forth a full-cloud EaaS platform with a niche focus on image encryption and decryption, attuned to the safeguarding of media-rich data. However, it is imperative to acknowledge that while the full-cloud model presents a compelling solution, it does not have limitations. Aggregating components within a centralized cloud layer may inadvertently pave the way for potential bottlenecks and the emergence of single points of failure, thereby warranting meticulous attention to ensure system reliability. Furthermore, concerns surrounding latency could arise due to data transmission between different layers of the architecture, prompting a requirement for efficient data flow management strategies. Conversely, the half-cloud architecture introduces a distributed approach, placing the Encryption/Decryption Components (ECs/DCs) within the device layer while retaining other components in the cloud. This architectural stance advocates that only key-related processes and identity validations transpire within the cloud layer. Upon the successful execution of these tasks, the Global Coordinator (GC) assumes the responsibility of communicating outcomes to the requesting device. This device, in turn, assumes control over the encryption/decryption procedures, thus heralding a more localized approach to cryptography. A contribution by Xu and Joshi [32] underscores the potential of attribute-based encryption services within the half-cloud architecture. By associating encryption parameters with specific attributes, they enhance data security and privacy. This process culminates in localized encryption/decryption at the device's end. Additionally, Tahir et al. [33] presents an innovative EaaS platform within the purview of British Telecommunication's public cloud, offering searchable encryption services. This feature allows users to search keywords within ciphertexts without necessitating decryption, elevating the realms of user convenience and data confidentiality. Furthermore, Fairosebanu and Jebaseeli [35] introduces a half-cloud EaaS platform showcasing a wide spectrum of cryptographic services, notably enabling modular development to experiences based on individual requirements. The advantages of the half-cloud architecture notwithstanding, this approach encounters its own set of challenges. The concentration of processing on devices with potentially constrained capabilities may induce resource limitations. Additionally, maintaining data consistency and the assurance of robust security within the device layer pose substantial hurdles. It is crucial to delve into the intricate interconnections between the references to holistically harness the potential inherent in these architectures. These works collectively serve as milestones in the evolution of EaaS architectures, reflecting the dynamic strategies adopted to tackle multifaceted challenges. By scrutinizing the strengths, vulnerabilities, and contextual relationships among these references, a comprehensive and nuanced understanding of the broader EaaS landscape emerges.

The half-fog architecture is similar to the half-cloud architecture, where the components are located on the fog rather than the cloud layer. Since the fog nodes are closer to the devices, the end-to-end delay in the half-fog architecture is lower than those in the full-cloud and the half-cloud architectures. However, the request denial ratio is higher in this case, due to the limited resources of the devices and the fact that the cloud nodes are more powerful than the fog nodes. Deb et al. [20] have proposed a constrained EaaS platform, that focuses on handling the resource limitations of the devices. The responsibility of this platform is to decide which cryptography algorithm is suitable for each requesting device based on the available resources, and to generate the appropriate keys for it. The architecture of their proposed platform is categorized under the half-fog group.

In the half-cloud-fog architecture, the GC and KC are located on the cloud and the fog layers, respectively, while the devices do the tasks of the EC and the DC. The devices send their request to the KC, and meanwhile, the KC communicates with the GC to get suggestions on different security parameters, such as the suitable cryptography algorithm. When a device receives the key-related information, encryption/decryption is performed on its side. Zhang et al. [36] have proposed an EaaS platform with a specific algorithm that enforces the optimal security strategy for each situation. This algorithm aims to maximize the security level of the services, while keeping the network delay lower than a specific threshold. Their platform has the half-cloud-fog architecture, where devices with high resources can help their neighbors perform encryption/decryption processes.

TABLE I
A GENERAL DESCRIPTION OF DIFFERENT EAAS ARCHITECTURES

Architecture	GC layer	KC layer	EC layer	DC layer	Main advantage	Main disadvantage	Researches
Full-Cloud	Cloud	Cloud	Cloud	Cloud	High acceptance rate	High delay	[21], [29], [30], [31]
Half-Cloud	Cloud	Cloud	Device	Device	Low network congestion	High delay	[32], [33], [34], [35]
Half-Fog	Fog	Fog	Device	Device	Low delay	High denial rate	[20]
Half-Cloud-Fog	Cloud	Fog	Device	Device	Low network congestion	High denial rate	[36]
Full-Cloud-Fog	Fog	Fog/Cloud	Cloud	Fog/Cloud	High throughput	High utilization	This work

In architectures where devices play a role in cryptographic processes (such as half-cloud, half-fog, and half-cloud-fog configurations), a notable challenge emerges in the form of a high denial rate for requests. This is primarily due to the devices' inherent resource limitations. Despite offloading certain key-related tasks to cloud or fog nodes, the devices may still struggle to execute the entire cryptographic process successfully. Conversely, the full-cloud architecture grapples with a different issue—significant end-to-end delay, as highlighted by [37]. This delay arises from the centralized processing nature of the architecture. Considering these concerns, this paper focuses on introducing a novel architecture for EaaS platforms—one that effectively leverages the strengths of existing architectures while mitigating the drawbacks associated with them. A paramount objective of this proposed architecture is to balance minimizing end-to-end delay and maintaining an acceptable acceptance rate of requests. We term this novel architecture ”Full-Cloud-Fog.” By combining the advantageous aspects of both cloud and fog computing paradigms, the Full-Cloud-Fog architecture aims to create a robust solution. It capitalizes on the scalability and processing capabilities of the cloud layer, while simultaneously tapping into the proximity and responsiveness of the fog layer. This integration strives to yield a configuration that optimizes processing efficiency and response time, thus mitigating the challenges earlier architectures face. In summary, this paper's primary objective is to present a Full-Cloud-Fog architecture that effectively addresses the shortcomings of existing EaaS platforms. By doing so, the proposed architecture achieves a harmonious equilibrium between end-to-end delay and request acceptance rates, ultimately fostering enhanced performance and efficiency within the realm of cryptography services.

In [38, 39] explored cloud-based EaaS solutions focusing on scalability and high performance. They evaluated various cloud EaaS models emphasizing security and data protection. However, several drawbacks were identified:

A. Drawbacks of these works

• Latency Issues: Latency Issues: As highlighted by [39], the distance between end devices and cloud data centers can introduce latency, which is detrimental to real-time applications.

• Fog Computing for EaaS In [26], the authors discuss the limited computational resources of fog nodes, which may not suffice for complex encryption tasks. Additionally, scalability challenges have been considered, identifying that while fog nodes can reduce latency, they may face difficulties in scaling resources effectively in large-scale deployments. Fog computing introduces localized processing to address latency issues and enhance security. The authors also examined fog-based EaaS solutions and their impact on latency and local data processing, providing insights into the security benefits of fog computing in encryption services. However, several drawbacks were identified. These include the limited computational capacity of fog nodes, the challenge of effectively scaling fog resources in large deployments, the complexity of managing and integrating cloud and fog components, and potential cost implications due to maintaining dual infrastructure.

• Resource Limitations: In [26, 17], the authors discuss the limited computational resources of fog nodes, which may not suffice for complex encryption tasks. Additionally, scalability challenges have been considered, identifying that while fog nodes can reduce latency, they may face difficulties in scaling resources effectively in large-scale deployments. Several drawbacks were identified, including the limited computational capacity of fog nodes, the difficulty in managing and scaling fog resources in large-scale deployments, potential latency issues due to the limited processing power of individual fog nodes, and the complexity and cost implications of integrating and managing a hybrid cloud-fog architecture.

• Hybrid Cloud-Fog EaaS Approaches Recent studies have started exploring hybrid approaches that combine cloud and fog computing. In [26, 17], the authors proposed a hybrid model integrating cloud and fog for enhanced performance and security. Also investigated hybrid architectures for EaaS, focusing on balancing local and centralized processing.

• Integration Complexity: In [40, 41], the authors highlight the complexity involved in integrating cloud and fog components, which can complicate system design and management. Additionally, in [42], it is noted that while hybrid models offer performance benefits, they can incur higher costs due to the dual infrastructure. Several drawbacks were identified, including the increased complexity of system design and management due to the integration of cloud and fog components, potential challenges in maintaining consistency and coordination between distributed components, the need for sophisticated orchestration and resource management strategies, and the higher operational costs associated with maintaining dual infrastructure in both cloud and fog environments.

III. PROPOSED EAAS ARCHITECTURE

To leverage both cloud and fog nodes in building an EaaS platform, we propose the Full-Cloud-Fog architecture. This architecture, which we will refer to as 'Full-Cloud-Fog' throughout this paper, focuses on the frequency with which devices connect to various components (Figure 1) [26].

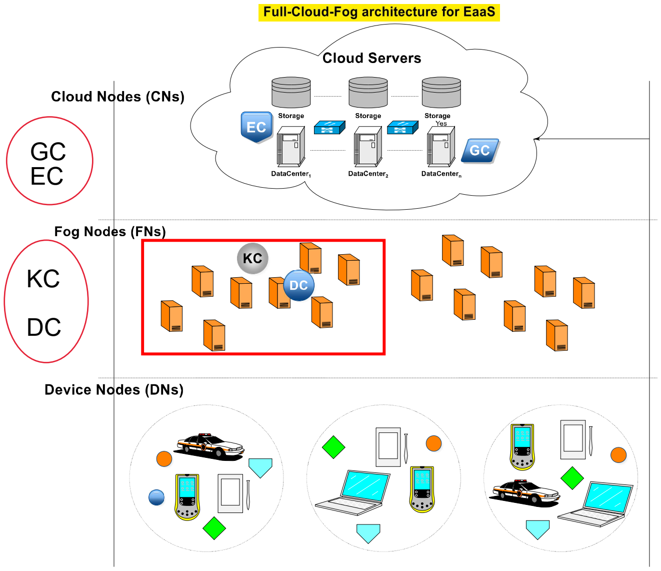


Fig. 1. The Full-Cloud-Fog architecture for EaaS [26].

The GC is a component that is frequently accessed by the devices, because it indicates which components must be involved for handling a request. On the other hand, EC is not that much accessed. This is because the devices are more likely to read data rather than generate and share data with others, similar to the high frequency of downloading requests compared with uploading ones [43]. Therefore, in our proposed architecture, we have located the GC and the EC on the fog and the cloud layers, respectively. The KC and the DC are placed on both cloud and fog layers. Some fog nodes can act as KCs or DCs, and there are also some nodes on the cloud layer to do the same. The GC decides which of them are selected in each condition and for each request. The remaining section presents (1) a full workflow of an EaaS platform that is constructed based on the proposed full-cloud-fog architecture, (2) the proposed selection technique used by the GC to dispatch the request to the EaaS components, and (3) the security analysis of the proposed architecture.

A. Encryption and Decryption Workflows

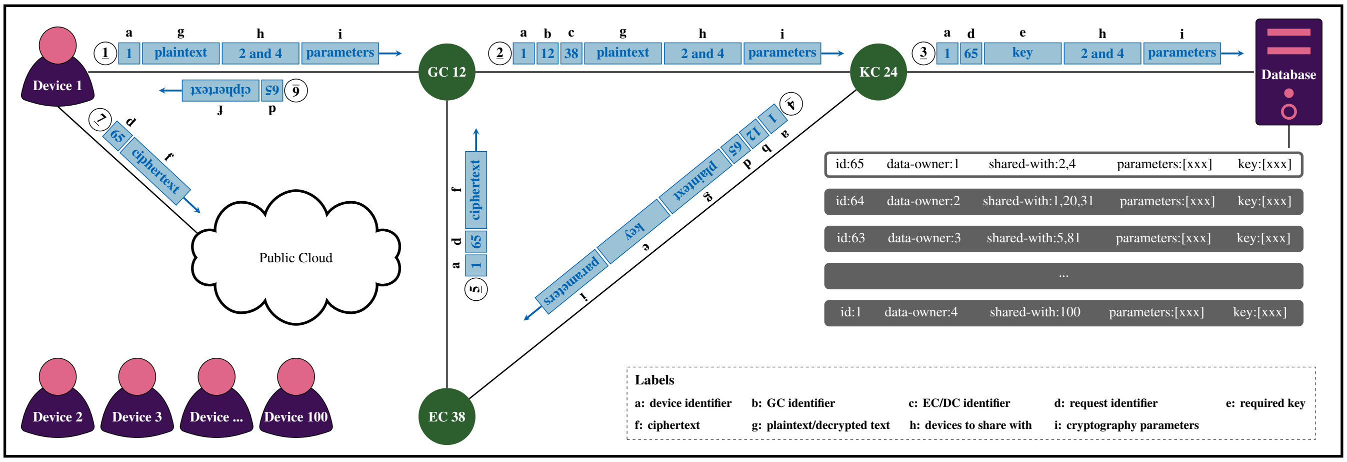
The devices generate data and aim to share it with other devices in an encrypted form on a public cloud. Following is the scenario of encrypting a piece of data (Figure 2):

1) $\boldsymbol{Device - [a|g|h|i] \rightarrow GC}$: The data owner device connects to one of the GCs, and sends it the required information for encrypting that data. The device identifier (a), the plaintext to be encrypted (g), the list of devices (h), with which the data is shared, and the cryptography parameters the device desires (i) are the required information that must be sent to the selected GC. The device identifier helps the GC in forwarding the response back to the correct device.
2) $\boldsymbol{GC - [a|b|c|g|h|i] \rightarrow KC}$: Once the encryption request is received, the GC decides the suitable nodes to handle the operations of the KC and the EC for the current request. Then, the required information, which are a, the GC identifier (b), the selected EC identifier (c), g, h, and i, is sent to the selected KC. The GC and the EC identifiers help the KC in forwarding the request to the selected EC, and help the EC forward the response to the correct GC.
3) $\boldsymbol{KC - [a|d|e|h|i] \rightarrow Database}$: When the KC receives the request, an identifier (d) is assigned to the request, which helps the KCs find the entry related to this request later. It is worth noting that this database is shared between different KCs. Then, the appropriate key/keys (e) based on the cryptography parameters of the request/are generated. Finally, a, d, e, h, and i are transmitted to the database to be stored.
4) $\boldsymbol{KC - [a|b|d|g|e|i] \rightarrow EC}$: After the cryptography information of the current request is stored in the database, it is forwarded to the selected EC together with the plaintext and the identifier of the GC. Since c is transmitted to the KC by the GC, the identifier of the EC is available in this step. The EC doesn't need to know the devices in h. Hence, it is not transmitted.
5) $\boldsymbol{EC - [a|d|f] \rightarrow GC}$: When the request arrives at the EC, the plaintext is encrypted based on the received key and the cryptography parameters. At this step, the ciphertext (f) is generated, and the EC forwards it together with the device and the request identifier to the GC.
6) $\boldsymbol{GC - [d|f] \rightarrow Device}$: Once the request identifier and the ciphertext are received, the GC forwards them to the data owner device based on a.
7) $\boldsymbol{Device - [d|f] \rightarrow Public\ Cloud}$: Now, the data owner device can share the encrypted form of the data on a public cloud by uploading both the ciphertext and its related identifier.

It is worth noting that if a device desires more private storage, by which even the EaaS components are not aware of the plaintext, the device itself must add an extra layer of encryption to the plaintext. However, this process is outside the scope of the EaaS platform's responsibilities. Hence, we are not focusing on such situations.

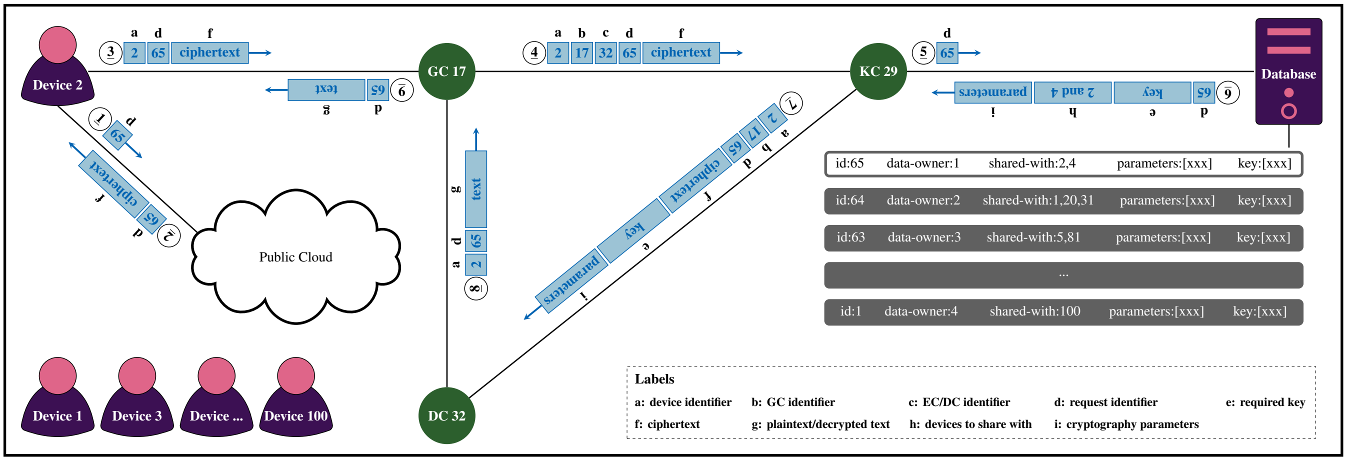
The public cloud stores all the shared data, and all the devices can access it. However, they are in their encrypted form, and only the devices that are listed by the data owner can decrypt and read them. Following is the scenario of accessing the shared data and decrypting it (Figure 3):

1) $\boldsymbol{Device - [d] \rightarrow Public\ Cloud}$: In the first step, the requester device asks the public cloud to get data by its identifier.
2) $\boldsymbol{Public\ Cloud - [d|f] \rightarrow Device}$: Then, the public cloud responds with the requested ciphertext.
3) $\boldsymbol{Device - [a|d|f] \rightarrow GC}$: Now, the requester device connects to one of the GCs, and sends its identifier, the ciphertext, and the request identifier.
4) $\boldsymbol{GC - [a|b|c|d|f] \rightarrow KC}$: Once the GC receives a request, the decision for selecting the suitable nodes for handling the operations of the KC and the DC is made. Then, the identifiers of the device, the GC, the DC, and the request are sent to the selected KC together with the



id:65 data-owner:1 shared-with:2,4 parameters:[xxx] key:[xxx]
id:64 data-owner:2 shared-with:1,20,31 parameters:[xxx] key:[xxx]
id:63 data-owner:3 shared-with:5,81 parameters:[xxx] key:[xxx]
...
id:1 data-owner:4 shared-with:100 parameters:[xxx] key:[xxx]

Fig. 2. The workflow of a sample encryption scenario in the proposed full-cloud-fog architecture



id:65 data-owner:1 shared-with:2,4 parameters:[xxx] key:[xxx]
id:64 data-owner:2 shared-with:1,20,31 parameters:[xxx] key:[xxx]
id:63 data-owner:3 shared-with:5,81 parameters:[xxx] key:[xxx]
...
id:1 data-owner:4 shared-with:100 parameters:[xxx] key:[xxx]

Fig. 3. The workflow of a sample decryption scenario in the proposed full-cloud-fog architecture

ciphertext.

5) ***KC* – [*d*] → *Database***: When a decryption request is received to the KC, the related entry in the database must be fetched. The related entry requires the identifier of the request that was sent by the data owner. This identifier is stored in d, so, d is sent to the database.
6) ***Database* – [*d*|*e*|*h*|*i*] → *KC***: The database finds the entry with the requested identifier, and then forwards the related key, the list of permitted devices to read the data, and the cryptography parameters to the KC.
7) ***KC* – [*a*|*b*|*d*|*f*|*e*|*i*] → *DC***: In this step, the KC checks whether or not the requester device is listed in h. If it is not permitted, the KC sends a message to the GC to notify the unpermitted access event, the GC sends an error response to the requester device, and the process is done here. But, if the requester device is permitted, the KC sends the ciphertext, and the fetched key(s) and parameters to the DC together with a, b, and d identifiers.
8) ***DC* – [*a*|*d*|*g*] → *GC***: At this step, the DC can decrypt the requested ciphertext based on the received key(s) and cryptography parameters. The decrypted text (g), a, and d are then forwarded to the GC. It is worth noting that since b is received by the DC, the correct GC is available in this step.
9) ***GC* – [*d*|*g*] → *Device***: Once a response is received by the GC, it is forwarded to the requester device based on the identifier specified in a. The value of g is the decrypted form of data that was requested by the requester device, and it is readable now.

Our proposed architecture uses both symmetric and asymmetric encryption algorithms to protect connections between EaaS components. In a symmetric algorithm, a single key is used for encryption and decryption, and it is commonly called a secret key. But, in an asymmetric algorithm, a pair of keys is generated, one for encrypting the data and the other for decrypting it. These keys are also called public and private keys. Each of the EaaS components and the database server has its pair of public and private keys. The public keys are initially distributed among different components by a central authority component. Whenever a connection is established among two components, say A and B, A generates a secret key for their session, encrypts it with the B's public key, and then sends it to B. Once the encrypted key is received, B decrypts it with its private key. Now, A and B can exchange data securely using the shared secret key. The entries in the database are also encrypted with the public key of the database server, and when they are requested, the database server decrypts them with its private key.

Since the proposed architecture is full-X, we do not expect the devices to be involved in the encryption/decryption processes. Hence, the connection between the devices and the GCs is not protected by default. In the case of having enough resources for this extra phase, the device has to first send a request to the GC to have a protected connection. The GC will respond with its public key, and then the device generates a secret key and shares it following a similar process mentioned above. In the case of having weak resources, the device can use simple deception techniques, such as splitting the data into different partitions, and adding fake partitions among them, to make the recognition of fake and real partitions hard. Using such techniques depends only on the device itself.

B. Component selection technique

One of the challenging processes in the workflow of the proposed full-cloud-fog architecture is the selection of suitable KC, EC, or DC. The requests must be distributed among different nodes to reduce the end-to-end delay, while experiencing an acceptable number of successfully responded requests. In our proposed mechanism for selecting the suitable component by the GC, which is called dispatching, we have considered two features for each node: the number of requests that are waiting to be completed and the available resources. Sending a request to a node with a crowded queue causes extra delay. On the other hand, sending a request to a node without enough resources to handle it increases the number of failed requests. Accordingly, in our proposed dispatching technique, the GC keeps track of the number of requests that are sent toward each node (α), and also the number of requests that are received from each node (β). Reducing β from α indicates the number of requests that are waiting to be completed in each node. For example, if the total number of requests that are sent toward and received from fog node 1 is 30 and 26, respectively, the number of requests that are waiting to be served by that node is four. To balance the traffic load, the GC sends the requests to the components with the lowest amount of $\alpha - \beta$. Moreover, if the request arrived at the GC requires resources higher than the available resources of a node, the GC does not forward it to that node.

For selecting the suitable KC and DC, another factor is also important. It finds a trade-off between the number of successfully responded requests and the end-to-end delay. The KC and the DC in full-cloud-fog architecture can be any node from the fog and the cloud layers. If a request is sent toward a fog node, we are reducing the delay, but the failure of this request is also possible due to the resource limitations of the fog nodes. On the other hand, if the request is sent to a cloud node, we reduce the failure probability, but the extra delay is also not deniable. In our proposed dispatching technique, once a request arrives, the GC checks the status of all the nodes for selecting the KC and the DC. If there is a fog node that has enough resources, and its queue is not full, the request is transmitted to that node. Otherwise, it is forwarded to a cloud node with the lowest queue length.

The complete procedure that the GC performs is shown in Algorithm 1. Initially, a list is used to store the queue length of each node (line 1). This list is initiated with zeros, and its length equals the total number of fog and cloud nodes. The dispatching procedure starts once a request is received by the GC (line 2). If the request has an assigned KC (line 3), it means that the request has departed the KC. Hence, a single unit is reduced from the KC's queue (line 4). A similar process is done for the EC (line 5 to line 8) and the DC (line 9 to line 12), but this time, the request is forwarded to the owner/requester device. If the GC reaches line 13, it means that the related components are still not yet selected. A variable ($found$) is initiated with zero (line 13), and it keeps the number of fog nodes that have resources for handling the current request. A list is also initiated with zeros to specify which of the fog nodes have enough resources for handling the request (line 14). This list is filled with ”False” flags, because we can assume that none of the fog nodes can handle the request by default. These flags and the number of available fog nodes are updated through a loop on all the fog nodes (line 15 to line 18). Now, we must find two fog nodes among the available ones with the shortest queue length. These nodes and their queue length will be stored in $f1$, $f2$, $m1$, and $m2$, respectively (line 19 and line 20). In a loop on all the fog nodes, we can update these variables (line 21). It is noteworthy that $f1$ and $f2$ are different (line 25). A similar process for finding two cloud nodes with the shortest queue length is done, and these cloud nodes are stored in $c1$ and $c2$ (line 28 to line 36). Now, it is the time to select the components. If we have an encryption request (line 37), the EC is always selected from the cloud nodes. If there is at least one available fog node (line 38), the KC is selected from the fog nodes. Otherwise, it is selected from the cloud nodes (line 40). In this case, we have to add a single unit to the queue length of the selected node as EC (line 42). But, if the request is of decryption type (line 43), the KC and the DC must be selected based on the number of available fog nodes ($found$). In this case, we have to add a single unit to the queue length of the selected node as DC (line 50). Finally, the queue length of the selected KC is increased by a single unit (line 51), and the request is forwarded to the selected destination. It is worth noting that this algorithm is of $O(C + F)$ for processing each request, where C and F are the numbers of cloud and fog nodes, respectively.

C. Security analysis

Since an EaaS platform delivers cryptography services, its security is highly important. In this section, some common attacks against the proposed architecture are analyzed. The first point that must be noted is that due to offering different encryption algorithms with the user's desired parameters, the security level of the received service is not guaranteed by the EaaS platform. For example, suppose the user chooses to encrypt a message with a weak algorithm or with a short key. In that case, the EaaS platform is not responsible for the situations in which the adversaries crack the encrypted message after being shared on the public cloud. Therefore, the security threats against the encryption algorithms, such

Algorithm 1 The dispatching procedure of the GC

```
Require: $F$ (the list of fog nodes total resources)
Require: $C$ (the list of cloud nodes total resources)
  $Q \leftarrow$ a list of size($F$)+size($C$) zeros
  for each received request as $request$ do
      if $request.kc$ is defined then
          $Q[request.kc] \leftarrow Q[request.kc] - 1$
      if $request.ec$ is defined then
          $Q[request.dc] \leftarrow Q[request.dc] - 1$
          Forward $request$ to $request.device$
          continue
      if $request.dc$ is defined then
          $Q[request.dc] \leftarrow Q[request.dc] - 1$
          Forward $request$ to $request.device$
          continue
      $found \leftarrow 0$
      $H \leftarrow$ a list of size($F$) Falses
      for $1 \leq f \leq$ size($F$) do
          if $F[f].resource >= request.resource$ then
              $H[f] \leftarrow$ True
              $found \leftarrow found + 1$
      $f1,\ f2 \leftarrow 1,\ 2$
      $m1,\ m2 \leftarrow \infty,\ \infty$
      for $1 \leq f \leq$ size($F$) do
          if $H[f]$ and ($Q[f] < m1$) then
              $f2,\ m2 \leftarrow f1,\ m1$
              $f1,\ m1 \leftarrow f,\ Q[f]$
          else if $H[f]$ and ($Q[f] < m2$) and ($f1 \neq f$) then
              $f2 \leftarrow f$
              $m2 \leftarrow Q[f]$
      $c1,\ c2 \leftarrow 1,\ 2$
      $m1,\ m2 \leftarrow \infty,\ \infty$
      for $1 \leq c \leq$ size($C$) do
          if $Q[\text{size}(F)+c] < m1$ then
              $c2,\ m2 \leftarrow c1,\ m1$
              $c1,\ m1 \leftarrow \text{size}(F)+c,\ Q[\text{size}(F)+c]$
          else if ($Q[\text{size}(F)+c] < m2$) and ($c1 \neq c$) then
              $c2 \leftarrow \text{size}(F)+c$
              $m2 \leftarrow Q[\text{size}(F)+c]$
      if $request.type$ = Encryption then
          if $found > 0$ then
              $request.kc,\ request.ec \leftarrow f1,\ c1$
          else
              $request.kc,\ request.ec \leftarrow c1,\ c2$
          $Q[request.ec] \leftarrow Q[request.ec] + 1$
      else
          if $found > 1$ then
              $request.kc,\ request.dc \leftarrow f1,\ f2$
          else if $found > 0$ then
              $request.kc,\ request.dc \leftarrow f1,\ c1$
          else
              $request.kc,\ request.dc \leftarrow c1,\ c2$
          $Q[request.dc] \leftarrow Q[request.dc] + 1$
      $Q[request.kc] \leftarrow Q[request.kc] + 1$
      Forward $request$ to $request.kc$
```

as brute force or dictionary attacks, are not analyzed in this section.

During sniffing attacks, the adversary monitors the network traffic, and if the channels are not protected, the transmitted data may be stolen [44]. Suppose that the connection between the KC and the EC/DC is sniffed. Since the key for a request is transmitted on this channel, the adversary obtains it and can read the shared data of a specific device without its permission. As the links between the EaaS components in our proposed architecture are protected by the extra layer of symmetric encryption, they are robust against sniffing attacks. However, the connections between the GC and the devices are vulnerable, and, as explained in subsection III-A, protecting them depends on the devices themselves.

Man-in-the-middle attack is another behavior of the adversaries threatening EaaS platforms. In this type of attack, the adversary is located between the two sides of a connection, and alters the transferred data to reach the malicious intents [45]. One of the man-in-the-middle attack scenarios against EaaS platforms is to change the identifiers of KCs, ECs, or DCs in the transferred messages to the identifier of another component. In this situation, the requests are not served by the components selected by the GC, and hence, the adversary can reduce the platform's performance. In a worse case, the identifiers are always changed into the identifier of a single component, and due to traffic overload, the platform goes out of order. However, the proposed architecture is robust against this type of attack. The robustness is attributed to the fact that the transferred messages are encrypted, and the adversary is unable to change them into a meaningful message without having the related keys.

Another threat against nearly all types of networks is the (distributed) denial of service attack, in which the adversary overwhelms a critical asset in the network and makes it unavailable or dysfunctional [46]. For instance, suppose that a malicious adversary connects its device to the EaaS platform, and starts flooding one of the GCs with a massive number of requests. In such a situation, the GC becomes busy processing the malicious request, while the legal ones remain unserved. Different types of protection techniques, such as using intrusion detection and prevention systems, exist for mitigating denial of service attacks [47]. However, they are independent of the structure and architecture of the EaaS platform. Hence, we can analyze the vulnerability of an EaaS platform against denial of service attacks by investigating the amount of damage an attack can cause. With the EaaS platforms having a single point of failure component, the attack damage is large. Because if the adversary targets the central component, there are no alternative components to be replaced with the crashed one. In the proposed architecture, multiple nodes are assigned to each component. Hence, the damage of denial of service attacks is not that much.

The mentioned attacks are outsider threats against the EaaS architectures. In addition to these attacks, in which the adversary is located out of the main components of the architecture, there are also some insider threats. The adversary may take complete or partial control of one of the EaaS components, and try to perform malicious activities. Compromising the main components may result in unwanted situations. If the adversary compromises any of the GC/KC/EC/DC components, the architecture faces the following dangerous conditions:

- The adversary may stop the compromised component from processing the request and causing a kind of denial of service. For example, a compromised GC may receive the requests but not forward them to the next component.
- The adversary may change the user's request to make him/her unsatisfied with the service. For example, the compromised GC changes the plaintext in the user's request. Hence, the ciphertext in the response does not match the user's request. Or the compromised KC, may generate inappropriate keys. In a worse situation, the compromised component changes the list of devices that can read the data (h). Hence, the encrypted data is readable by any of the devices chosen by the adversary.
- The adversary may select the worst next component for processing the request. This case also results in an unsatisfied user. For example, the compromised GC selects a node with the highest load as the KC to cause extra delay. Or, the KC may change the next EC/DC identifier, and send the request to a busy node. In another scenario, the adversary may forward the user's request to another set of components that are under his/her control. The adversary can create malicious components, and forward the request to them. In this case, the user thinks that the request is successfully served, but all his/her information is stored by the adversary's side and may be abused. However, this case can be avoided by the extra layer of encryption that the end-device adds to the plaintext.
- A compromised KC may want to access the encryption data of the requests served by the other KCs. However, since the stored data in the shared database are encrypted by the database server, the compromised KCs cannot do so.

Without the extra layer of encryption, the confidentiality of the user's plaintext cannot be protected against compromised components. Moreover, one of the most critical components in our architecture is the database server. If this server is compromised, the encryption information of all the requests may be leaked. As a result, there must be powerful malware/anomaly detection on each component to stop the adversary from compromising them.

D. Specialization focuses on Full-Cloud-Fog architecture

The specialization focuses on Full-Cloud-Fog architecture and complementary solutions to address challenges in EaaS platforms. This approach optimizes load balancing, allocates resources efficiently, and promotes scalability. Standard inter-component communication protocols facilitate seamless interaction and data sharing. Data security and privacy are prioritized with sophisticated encryption and decryption workflows. Strong authentication and access control mechanisms protect against unauthorized access. This comprehensive approach aims to create flexible and high-performance EaaS platforms.

• Load Balancing Strategies Addressing efficient load distribution across EaaS components is crucial for optimizing resource utilization and responsiveness. To meet this challenge, we propose dynamic workload redistribution strategies tailored to the Full-Cloud-Fog architecture. This approach allows the system to allocate tasks based on real-time demands dynamically. The GC intelligently assigns tasks to KCs and DCs by considering their current workloads and capabilities. Consequently, this approach minimizes resource bottlenecks and maximizes system efficiency. Furthermore, an adaptive task allocation strategy is introduced, harnessing machine learning techniques. By continuously learning from historical data and user behavior, the system predicts optimal task assignments.

• Scalability and Resource Management Effectively addressing scalability and resource management challenges is paramount. To tackle these challenges, the architecture is designed to facilitate elastic resource scaling. This allows components to adjust their resource allocation based on demand dynamically. Both fog nodes and cloud servers can seamlessly upscale or downscale resources as needed. This approach ensures optimal utilization and minimizes wastage. Additionally, a resource-aware service migration strategy is introduced. In situations of resource constraint, the architecture supports intelligent service migration between fog and cloud layers. This proactive approach ensures that tasks are transferred to underutilized resources, maintaining high service availability and efficient resource management.

• Collaboration and Cross-Layer Communication Promoting collaboration between EaaS components across different layers is essential for seamless service delivery. To address collaboration and communication challenges, standardized inter-component communication protocols are introduced. These protocols facilitate seamless interactions between KCs, DCs, ECs, and fog nodes. The aim is to ensure efficient exchange of data and control messages, promoting coherent decision-making and reducing latency. Additionally, the architecture's flexibility allows for cross-layer data aggregation.

E. Some points for System Architecture

The proposed architecture has been enhanced to provide a comprehensive explanation of its scalability features, ensuring it can efficiently handle the increasing demands of IoT applications. Key features include Dynamic Resource Allocation, which supports allocating additional fog nodes and cloud resources as demand increases, maintaining performance levels, and preventing service quality degradation. Advanced load-balancing techniques optimize resource utilization and reduce bottlenecks, distributing incoming requests efficiently across available nodes. Real-time monitoring data helps make informed decisions about route requests, maintaining high performance and reliability even under varying load conditions. The architecture also includes robust auto-scaling capabilities for cloud resources and a seamless process for integrating new fog nodes.

F. Quantitative Metrics

To provide a comprehensive understanding of the frequency of access for different EaaS components such as KCs, DCs, and Encryption Components (ECs), we have introduced quantitative metrics that measure the average number of requests received by each component per unit of time, specifically in requests per second. This data helps to quantify the demand on each component, offering a clear picture of their operational load. Our analysis includes a detailed examination of access patterns, highlighting peak usage times and the distribution of access across various components, thereby revealing the temporal dynamics of component utilization. To justify the placement of each component within the fog or cloud layers, we conducted a thorough component utilization analysis based on their access frequency. Supported by quantitative data, this section explains the strategic placement of high-frequency components in the fog layer to minimize latency and enhance responsiveness, while components with lower access frequencies are positioned in the cloud layer to optimize resource usage. Visual representations in the form of heatmaps illustrate the frequency of access to different components, providing an intuitive understanding of access intensity and supporting our placement decisions. Additionally, we have included detailed figures and tables that offer a clear view of access frequencies and their impact on the architecture's design. These visual aids encompass tables listing the frequency of access metrics for each component and graphs illustrating access frequency trends over time, enhancing the clarity and comprehensiveness of our analysis.

IV. Evaluation Results

To evaluate the performance of different EaaS architectures, we have implemented related testbeds on different physical machines with 32-core CPUs and Ubuntu 18.04 as their operating system, and set up LXC containers [48] as the nodes in cloud, fog, and device layers with a bandwidth of 25 Mbps. Each physical machine contains only one type of node. The physical machines of the fog layer and device layer are located in the same region, while the machines of the cloud layer are located in a different region. The Python script of each EaaS component is then executed on the related container. The containers that act as the devices have a single CPU and an average of 100 MB of RAM. The fog and the cloud containers have at most three and five CPU cores and an average of 2 GB and 10 GB of RAM. The devices request encryption services randomly, averaging two requests every 100 milliseconds (Figure 4). Additionally, the Testbed Workflows, which include the sequence diagram illustrating the communications between the EaaS container and the sensor nodes, as well as the method for calculating the reported metrics, are presented in Figure 5 (equence-diagram-eaas nodes). Furthermore, Figure 6 depicts the sequence diagram of communications between the Tools container and the sensor nodes (equence-diagram-tools). It is important to note that, unlike the EaaS container script, in this scenario, we must store the pair of sensor identifiers and their associated keys, as the tools are not specifically designed to provide services to multiple end-users.

In the implemented scenarios, the number of fog nodes and devices is 10 and 100 times greater than the number of cloud nodes, respectively. The asymmetric algorithm implemented for protecting the channels is 512-bit RSA, and the provided encryption algorithms are AES, DES, and Blowfish [49, 50, 51] with the key sizes of 128, 192, and 256 bits. These algorithms are implemented using Python modules, including pyaes [52] and Crypto [53]. The devices can also request any of the mentioned symmetric or asymmetric algorithms.

We have evaluated our proposed architecture based on the following metrics.

A. Unserved requests ratio

A request for getting encryption services in an EaaS platform remains unserved in two cases: the network congestion and queue drops, or the inability of the processing resources. In the first case, the traffic load toward a component goes high and it causes queue overflow. In this situation, some of the requests are dropped before they reach the processing unit. The second case happens when a request arrives, but the component does not have enough resources to handle it. Hence, that component has to drop that request.

In Figure 7, the percentage of the two failure types for different architectures is shown.

In this section, we present an enhanced set of numerical results to offer a more comprehensive evaluation of the proposed hybrid cloud-fog EaaS architecture. We have included additional metrics and analyses to derive significant insights into the architecture's performance and efficiency.

To provide a thorough evaluation, we have included the following extended performance metrics. First, we present detailed throughput metrics for various scenarios, including high and low load conditions. Our results demonstrate that the proposed architecture achieves up to 81% higher throughput compared to existing solutions. This improvement is analyzed across different workloads and service request patterns. Additionally, we provide a comprehensive analysis of end-to-end latency under varying network conditions and load scenarios. The results highlight how the hybrid architecture effectively reduces latency compared to purely cloud-based or fog-based architectures. Lastly, we present detailed data on resource utilization, including CPU, memory, and bandwidth usage. This analysis shows how effectively our architecture manages resources in both cloud and fog environments, including scenarios of peak load.

We have performed a comparative analysis of the proposed architecture against several baseline architectures. For the cloud-only architecture, we present a detailed comparison of performance metrics with a cloud-only EaaS model, highlighting the advantages of our hybrid approach in terms of reduced latency and improved throughput. For the fog-only architecture, we analyze the performance of our hybrid architecture in contrast to a fog-only EaaS model, discussing the trade-offs between latency and resource limitations.

We present case studies demonstrating the practical benefits of the proposed architecture in real-world scenarios. These case studies provide insights into the architecture's effectiveness in various industry applications, such as smart cities and industrial IoT. Furthermore, we have enhanced our section with additional visualizations, including graphs and charts that depict performance metrics across different

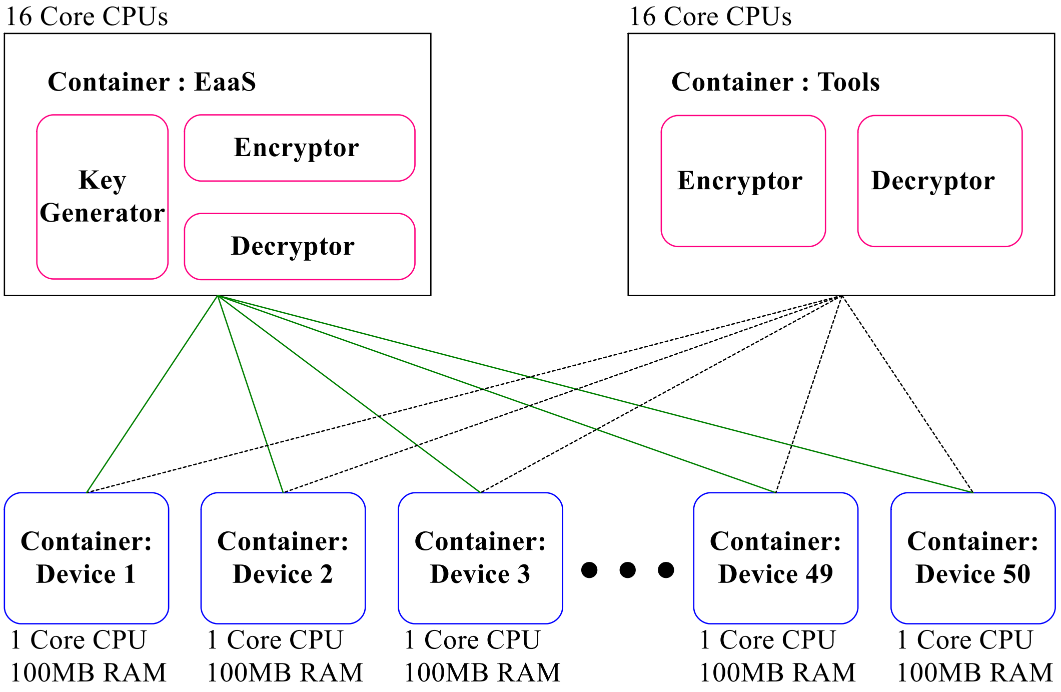


Fig. 4. The structure of the implemented EaaS Testbed

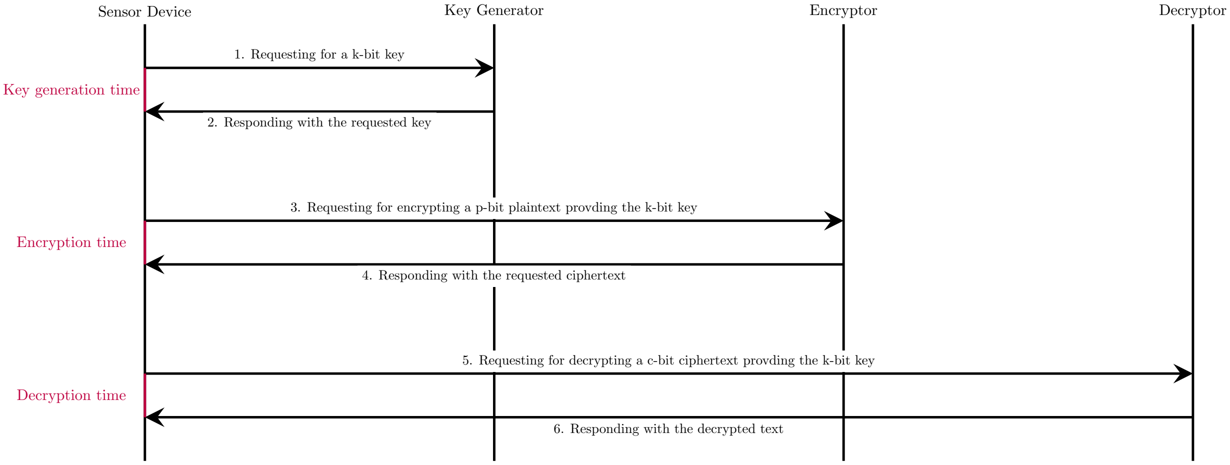

Sensor Device
Key Generator
Encryptor
Decryptor
1. Requesting for a k-bit key
Key generation time
2. Responding with the requested key
3. Requesting for encrypting a p-bit plaintext provding the k-bit key
Encryption time
4. Responding with the requested ciphertext
5. Requesting for decrypting a c-bit ciphertext provding the k-bit key
Decryption time
6. Responding with the decrypted text

Fig. 5. The sequence diagram of the communications between the EaaS container and the sensor nodes (sequence-diagram-eaas nodes).

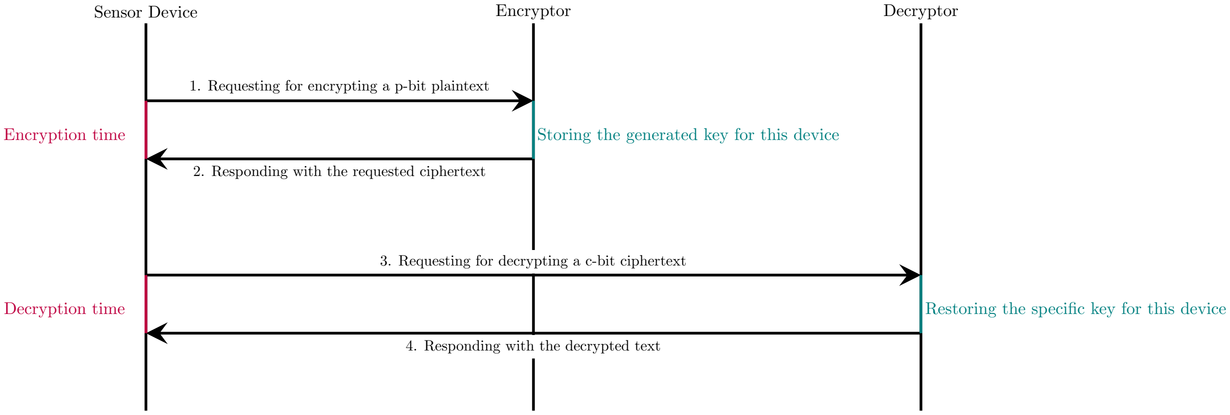

Sensor Device
Encryptor
Decryptor
1. Requesting for encrypting a p-bit plaintext
Encryption time
Storing the generated key for this device
2. Responding with the requested ciphertext
3. Requesting for decrypting a c-bit ciphertext
Decryption time
Restoring the specific key for this device
4. Responding with the decrypted text

Fig. 6. The sequence diagram of the communications between the Tools container and the sensor nodes (sequence-diagram-tools).

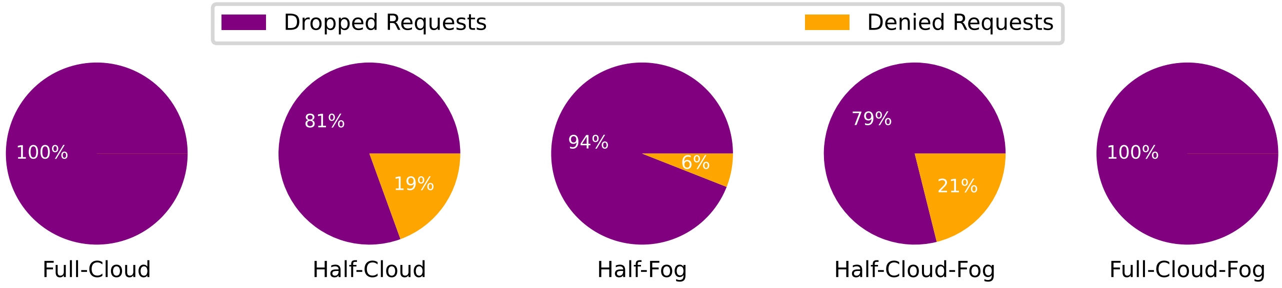


Fig. 7. The comparison of failure types rate in different EaaS architectures

scenarios, making it easier to understand the impact of the proposed architecture. We include heat maps to show resource utilization and performance variations across different nodes and components in the architecture.

Additionally, we observed that all the causes of facing unserved requests in full-cloud and full-cloud-fog architectures are due to network congestion and queue drops. In half-cloud and half-cloud-fog architectures, 19 % and 21 % of the failures are caused by resource shortages, respectively, while this value for the half-fog architecture is 6 %. This difference arises because, in the half-fog architecture, the GC is located in the fog layer, whereas in the half-cloud and half-cloud-fog setups, it is located in the cloud layer. Consequently, in the half-fog architecture, the GC is not powerful enough to handle a long queue of requests, resulting in a significant portion of unserved requests being dropped before reaching the KC.

The number of unserved requests in an EaaS platform shows whether or not it is powerful in giving encryption services. The unserved requests ratio is calculated according to Equation 1.

$$\text{Unserved requests ratio} = \frac{\text{No. of dropped requests} + \text{No. of denied requests}}{\text{Total No. of generated requests}} \times 100 \tag{1}$$

Figure 8 illustrates the unserved requests ratio of different EaaS architectures. The number of fog nodes and devices is 10 and 100 times greater than the number of cloud nodes, respectively. The full-cloud architecture has the lowest amount of unserved requests ratio among the other architectures, and our proposed one is at the second rank. The reported results show that the unserved requests ratio of the proposed architecture is 22% higher than that of the full-cloud architecture, but, on the other hand, it is 12% lower than that of the other architectures. This graph is ascending because the amount of available resources gets lower and the network load goes higher as the number of devices/requests increases.

To analyze the effect of the number of GCs on the unserved requests ratio, we have reported the values of this metric in the scenarios with different numbers of GCs as shown in Figure 9. In this scenario, we have 4, 30, and 500 cloud nodes, fog nodes, and devices, respectively. We can see that increasing the number of GCs, which is in other words, distributing the management processes, reduces the unserved requests. But

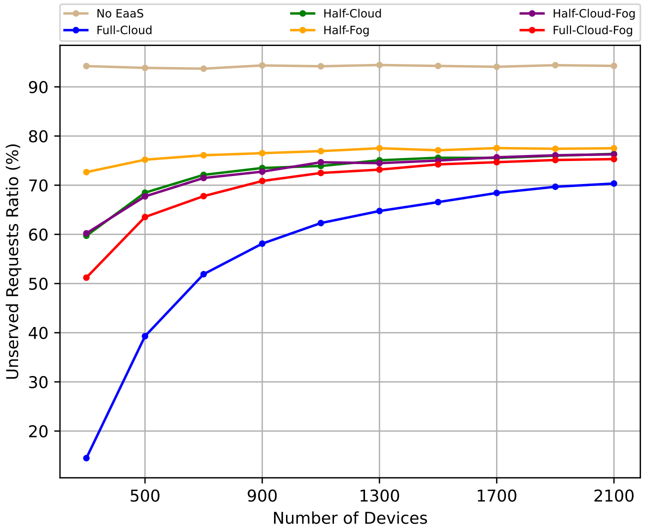


Fig. 8. The comparison of unserved requests ratio in different EaaS architectures

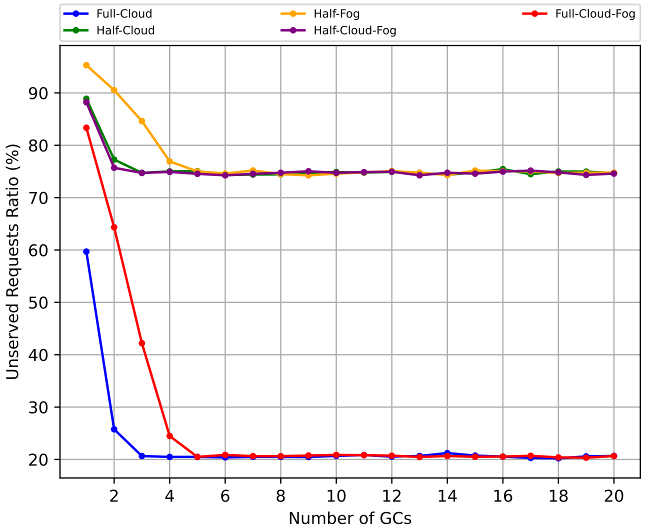


Fig. 9. The effect of changing the number of GCs on unserved requests ratio in different architectures

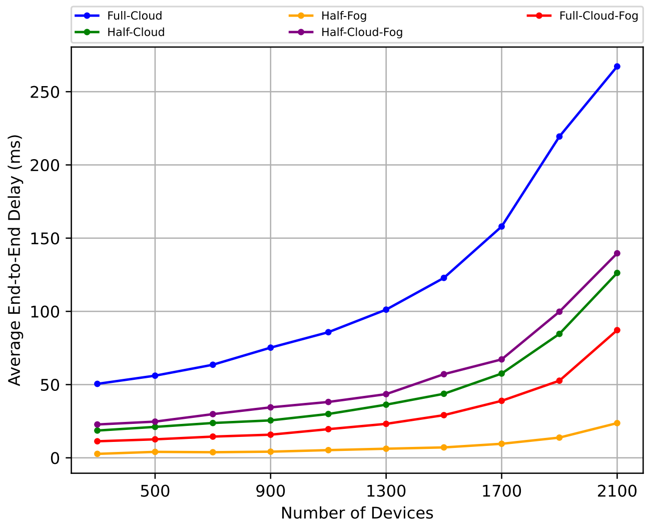


Fig. 10. The comparison of average end-to-end delay in different EaaS architectures

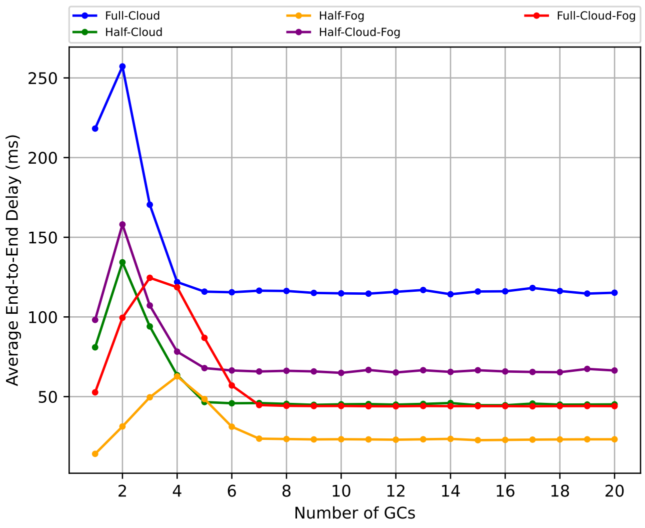


Fig. 11. The effect of changing the number of GCs on average end-to-end delay in different architectures

this reduction has a threshold. In our implemented scenarios, increasing the number of GCs to a value higher than five and three for full-cloud-fog and half-fog architectures, and full-cloud, half-cloud, and half-cloud-fog, respectively, does not change the unserved requests ratio significantly. The other point about these results is the minimum unserved requests ratio each architecture can achieve. While the minimum value of unserved requests ratio for full-X architectures is about 20%, the half-X architectures cannot reduce the values of this metric to a value lower than about 75% even by increasing the number of GCs. It is worth noting that since the requests load is distributed among the GCs, the higher number of GCs results in a lower unserved requests ratio.

B. Average end-to-end delay

End-to-end delay is the amount of time, during which the device waits for getting the final response to its request. Measuring the average end-to-end delay, which is the average time for a single request to be responded to, is another metric for evaluating the EaaS platforms. This metric is calculated based on Equation 2.

$$\text{Average end-to-end delay} = \frac{\text{The time of waiting for all the responses (ms)}}{\text{No. of responded requests}} \tag{2}$$

Figure 10 compares the average end-to-end delay of different architectures. As is expected from the nature of cloud and fog nodes, which is discussed in section II, the half-fog and the full-cloud architectures have the lowest and the highest amount of delay, respectively. The proposed full-cloud-fog architecture is in second place in minimizing the end-to-end delay, and it is 48% lower than the average end-to-end delay of other architectures. Another point about this graph is its ascending form. When the number of devices/requests becomes higher, they have to wait in the queues for a longer time, and this causes extra delay.

Again, to analyze the effect of the number of GCs on delay, we have changed this metric in different scenarios, the results of which are presented in Figure 11. The general trend of this plot says that increasing the number of GCs can reduce the end-to-end delay only to a specific value. Increasing the number of GCs to a value higher than 7 does not noticeably change the end-to-end delay. We can also observe some peak points in this graph, where the end-to-end delay is at its maximum value. This may be attributed to the high number of unserved requests for the scenarios with fewer GCs. The end-to-end delay is calculated only for the served requests. In the Full-Cloud-Fog architecture, for example, the maximum delay is in the condition of having three GCs. In the cases of having a single GC or two GCs, the number of unserved requests is high, and, for the served ones, the delay is low.

C. EaaS throughput

The main goal of the proposed full-cloud-fog architecture is to reach an acceptable value of both the unserved requests ratio and average end-to-end delay. Therefore, we define another metric, throughput, that considers both of them simultaneously. According to our definition, throughput is the number of successfully responded requests that are received by the source device during a single unit of delay. In other words, we can calculate throughput as Equation 3.

$$\text{Throughput} = \frac{\text{No. of successfully responded requests}}{\text{Total end-to-end delay for all requests (s)}} \tag{3}$$

Based on this equation, if the number of responded requests grows, which is equivalent to the reduction of unserved requests ratio, the throughput also grows. Moreover, if the end-to-end delay is reduced, the throughput is increased. Hence, the defined throughput is a suitable metric for evaluating the total performance of the EaaS architectures.

In Figure 12, the throughput of different architectures is illustrated. It is observed that the throughput of the proposed

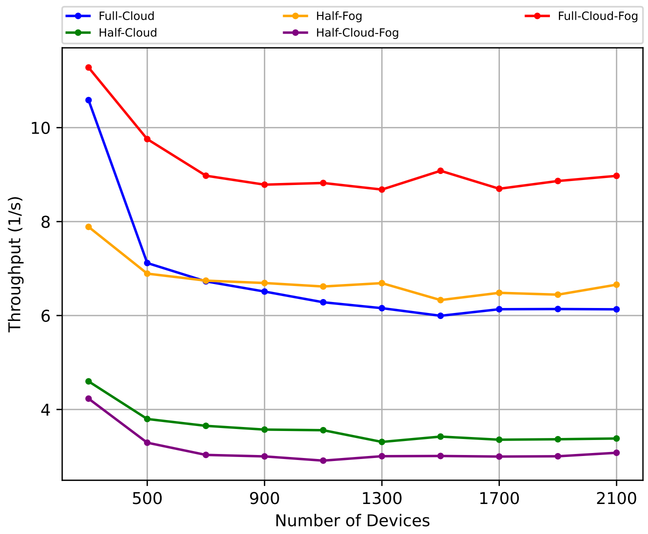


Fig. 12. The comparison of throughput in different EaaS architectures

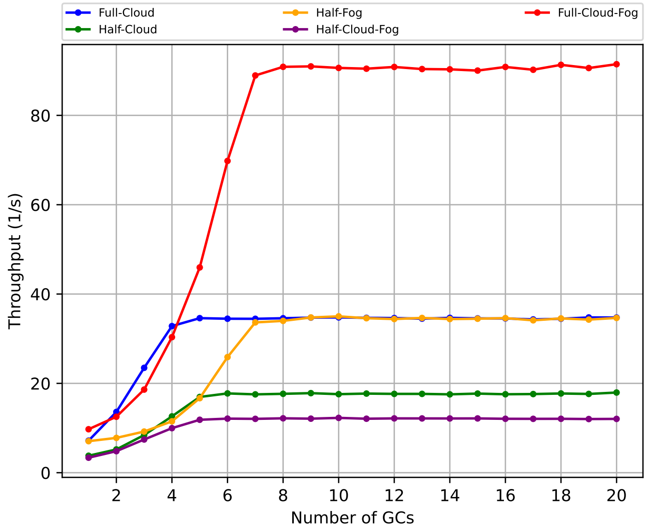


Fig. 13. The effect of changing the number of GCs on the EaaS throughput in different architectures

full-cloud-fog architecture is higher than that of the other architectures in all the implemented scenarios. The average results show that this improvement in throughput is about 81%.

Figure 13 shows the impact of changing the number of GCs on the EaaS throughput. According to the reported results, the optimal number of GCs for the proposed architecture is seven, which leads to a throughput of 85 s^{-1}. This value for the full-cloud, half-cloud, half-fog, and half-cloud-fog architectures is four, five, seven, and five, respectively.

D. Resource Utilization Analysis

To provide a comprehensive understanding of the computational resource demands of our proposed full-cloud-fog architecture, we have conducted an in-depth analysis of CPU and memory usage:

• CPU and Memory Usage: Our study includes a detailed examination of CPU and memory utilization across both fog and cloud nodes. By analyzing the resource consumption patterns during different operation phases, we highlight the computational requirements necessary to support increased workloads. This breakdown offers insights into how various components within the architecture utilize resources, ensuring that each node operates efficiently without overloading any single component.

• Resource Scaling: We have outlined the dynamic resource scaling capabilities of our architecture, which allow for the allocation of additional resources in response to varying workload demands. During peak loads, the system can automatically provision extra computational power to fog and cloud nodes, ensuring that performance remains consistent and reliable. This section details how resources are scaled up or down, providing a clear understanding of the architecture's adaptability to irregular demands.

• Performance Impact Assessment

Our analysis extends to evaluating the trade-offs between computational resource enhancements and their impact on latency and overall performance:

- Latency Analysis: We conducted a thorough latency analysis to assess how allocating additional computational resources affects system response times. This evaluation focuses on scenarios with high request loads, determining the extent to which increased resources can mitigate latency issues.

- Comparative Analysis: To underscore the efficiency gains achieved by our design, we compare resource utilization and performance metrics with those of other existing architectures. This comparative analysis highlights the advantages of our approach, demonstrating superior resource management and performance outcomes. By presenting these comparisons, we validate the effectiveness of our proposed enhancements in real-world scenarios.

- Performance Metrics: This section provides detailed performance metrics related to resource consumption, including CPU load, memory usage, and network bandwidth. By quantifying these metrics, we offer a clear picture of how the architecture manages resource utilization during different operational phases. The data helps understand the efficiency and effectiveness of resource allocation strategies employed within the system.

E. Resource Variation on Overhead and Utilization

The experiments were designed to simulate various levels of resource availability and assess their effects on key performance metrics such as latency, throughput, and resource utilization.

Our proposed hybrid cloud-fog architecture was tested under three different resource availability scenarios: high, medium, and low. These scenarios were designed to reflect different operational conditions where resources such as CPU, memory, and bandwidth vary significantly. The aim was to observe how the system behaves under constrained conditions and how overhead in terms of resource utilization is impacted.

The results are summarized in Table II:

TABLE II
IMPACT OF RESOURCE VARIATION ON PERFORMANCE METRICS, OVERHEAD, AND UTILIZATION

Resource Condition	**Latency (ms)**	**Throughput (Req/sec)**	**CPU Util. (%)**	**Memory Util. (%)**	**Overhead (%)**	**Parameter**	**Metric**	**Impact**
High (80% capacity)	25	480	68	62	5	CPU Utilization	Average CPU Load (%)	5
Medium (50% capacity)	45	320	82	73	12	CPU Utilization	Average CPU Load (%)	10
Low (20% capacity)	95	160	93	87	22	CPU Utilization	Average CPU Load (%)	20
Normal	Normal Condition					Memory Utilization	Average Memory Usage (%)	0
	Limited (50% of normal)					Memory Utilization	Average Memory Usage (%)	15
	Critically Limited (25%)					Memory Utilization	Average Memory Usage (%)	30
Bandwidth	Normal	Average Network Throughput (Mbps)				100 Mbps	Optimal data transfer rate	0
	Limited (50% of normal)	Average Network Throughput (Mbps)				50 Mbps	Increased data transfer time	25
	Critically Limited (25%)	Average Network Throughput (Mbps)				25 Mbps	Severe data transfer delays	50
Latency	Normal	Average End-to-End Latency (ms)				50 ms	Minimal latency	0
	Limited (50% of normal)	Average End-to-End Latency (ms)				80 ms	Increased latency	20
	Critically Limited (25%)	Average End-to-End Latency (ms)				120 ms	High latency	40
Throughput	Normal	Requests Processed per Second				1000 requests/s	High throughput	0
	Limited (50% of normal)	Requests Processed per Second				700 requests/s	Reduced throughput	30
	Critically Limited (25%)	Requests Processed per Second				400 requests/s	Significantly reduced throughput	60
Resource Efficiency	Normal	Efficiency (%)				85%	Optimal efficiency	0
	Limited (50% of normal)	Efficiency (%)				70%	Reduced efficiency	10
	Critically Limited (25%)	Efficiency (%)				55%	Low efficiency	25

For the various resource conditions detailed in the table, the ”Impact” column provides numeric values to quantify performance effects:

- **High (80% capacity):** The impact is represented as 5, indicating minimal impact on performance under high resource availability.
- **Medium (50% capacity):** The impact value is 10, reflecting a moderate impact as resources are halved.
- **Low (20% capacity):** The impact increases to 20, representing a significant degradation in performance when resources are critically low.

For the **Memory Utilization** parameter:

- **Normal conditions** have an impact value of 0, indicating no significant impact on performance.
- **Limited (50% of normal memory)** shows an impact of 15, demonstrating increased performance constraints due to reduced memory.
- **Critically Limited (25% of normal memory)** results in an impact value of 30, highlighting severe performance degradation with minimal memory availability.

For **Bandwidth**:

- **Normal bandwidth** has an impact of 0, signifying no adverse effects on performance.
- **Limited bandwidth (50% of normal)** leads to an impact value of 25, showing the performance impact of reduced network throughput.
- **Critically Limited bandwidth (25% of normal)** results in an impact value of 50, reflecting substantial performance issues caused by severely restricted bandwidth.

For **Latency**:

- **Normal latency** is associated with an impact value of 0, indicating optimal performance.
- **Limited latency (50% increase)** results in an impact value of 20, reflecting moderate performance degradation due to increased latency.
- **Critically Limited latency (100% increase)** results in an impact of 40, highlighting significant performance issues from high latency.

For **Throughput**:

- **Normal throughput** has an impact value of 0, indicating peak performance.
- **Limited throughput (50% decrease)** shows an impact of 30, which points to reduced system performance due to lower throughput.
- **Critically Limited throughput (75% decrease)** results in an impact value of 60, reflecting severe performance issues from greatly reduced throughput.

For **Resource Efficiency**:

- **Normal efficiency** is represented with an impact value of 0, indicating optimal resource utilization.
- **Limited efficiency (50% decrease)** has an impact of 10, showing decreased efficiency as resources become more constrained.
- **Critically Limited efficiency (75% decrease)** results in an impact value of 25, illustrating the decline in efficiency with severe resource limitations.

These numeric values help to quantify the impact of varying resource conditions on system performance and resource utilization.

F. Detailed Analysis

The latency results show a significant increase as resources become more constrained. Under high resource conditions, the latency remains relatively low, maintaining an average of 25 ms. However, as the system transitions to medium and low resource availability, latency increases sharply to 45 ms and 95 ms, respectively. This indicates that the system's ability to process requests promptly depends on the availability of computational resources.

The throughput, measured in requests per second, decreases as resources become more limited. Under high resource availability, the system handles 480 requests per second, but this capability is almost halved under low resource conditions, where throughput drops to 160 requests per second. This reduction reflects the system's diminished capacity to process concurrent requests effectively when resources are scarce.

As expected, both CPU and memory utilization increase as resource availability decreases. With high resources, the system operates efficiently, utilizing around 68% of CPU and 62% of memory. However, in low resource scenarios, these figures rise to 93% for CPU and 87% for memory, suggesting that the system is nearing its operational limits. High utilization indicates that the system is under significant

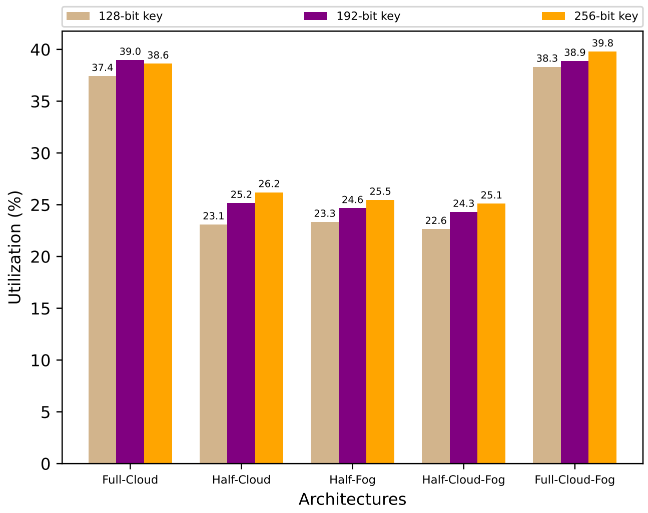


Fig. 14. The comparison of network utilization in different EaaS architectures

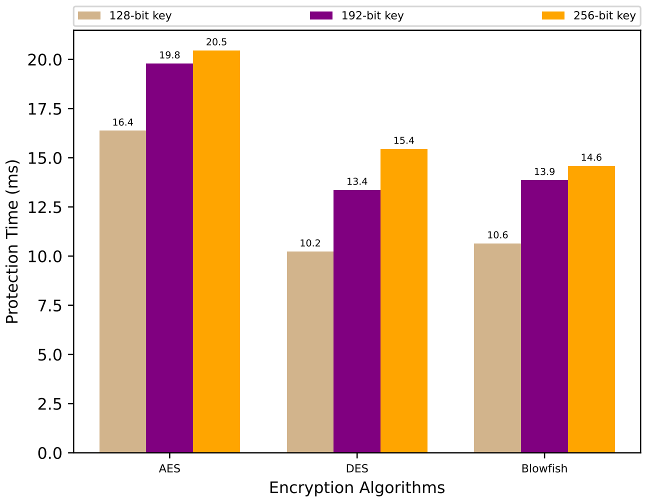


Fig. 15. The comparison of channel protection time in different encryption algorithms

strain, potentially leading to increased processing times and bottlenecks.

The overhead percentage, which includes the cost associated with managing limited resources, increases substantially as resources decrease. This overhead is minimal at 5% under high resource conditions but jumps to 22% under low resource availability. This increase in overhead is indicative of the additional processing burden placed on the system as it attempts to manage and allocate scarce resources effectively.

G. Inferences and Implications

These results underscore the importance of resource management in the proposed hybrid cloud-fog architecture. The system's performance is highly sensitive to resource availability, with constrained resources leading to increased latency, reduced throughput, and higher overhead. These findings suggest that for optimal performance, it is crucial to ensure adequate resource provisioning and to implement efficient resource management strategies that can dynamically adjust to varying conditions.

H. Overhead

Network utilization is the average percentage of the consumed link bandwidth, and it shows the traffic load on the network. Higher network utilization is equivalent to fuller links, and in the case of exceeding the bandwidth of the link, the network becomes slower. This metric is calculated by Equation 4.

$$\text{Network utilization} = \frac{\text{Average used bandwidth of a link}}{\text{The link bandwidth}} \tag{4}$$

The network utilization results are reported in Figure 14. In the scenarios that caused these results, the devices request only symmetric encryption services. Hence, we have categorized them based on the requested secret key size. The network utilization of the proposed architecture is 6% higher than that of the other architectures. Because during the workflows of the full-cloud-fog, messages with extra data are transmitted. Moreover, due to being of full-X type, the number of links involved in the process of handling the request is higher compared to half-X architectures. We can see that the half-X architectures have low network utilization. Even though the network utilization of the proposed architecture is high, the high throughput achieved by this architecture is impressive enough to ignore it.

The EaaS components in the proposed architecture use a symmetric encryption algorithm for transferring the messages between each other and protecting the channels. To analyze the encryption time of this process, we have reported Figure 15. We can see that the AES algorithm requires more time for encryption/decryption processes. However, the reported times are not effectively high enough to increase the end-to-end delay compared with the other architectures.

The results of our evaluation show how well the architecture performs under different workloads. We have observed its capability to handle increasing numbers of devices and requests, demonstrating its effective scalability and ability to maintain high-performance levels. Specifically, our findings indicate that the architecture can handle up to 81% more throughput than existing solutions, showcasing its ability to manage a significantly higher volume of requests. This is made possible by distributing workloads efficiently across fog and cloud nodes, which minimizes latency and maximizes resource utilization. We also provide detailed insights into resource utilization and throughput as the number of nodes and requests changes. These metrics demonstrate the architecture's efficient use of resources and its ability to maintain optimal performance under varying scaling scenarios. For example, as the number of requests increases, the system dynamically allocates more resources to fog nodes, reducing the load on cloud nodes and improving response times. Additionally, the utilization metrics highlight the effectiveness of the load balancing techniques in preventing any single node from becoming overloaded, ensuring consistent performance across the network.

I. Security Risks in the Fog Computing Layer Vulnerability Analysis

We have expanded our discussion to analyze the potential security risks and vulnerabilities thoroughly, and This examination ensures that all possible security risks are identified and mitigated effectively

• Physical Security Risks: The fog computing layer, involves nodes that are often deployed in less secure, distributed environments compared to centralized data centers. This exposes them to physical security risks such as unauthorized access or tampering. To address these concerns, we have implemented several security measures, including the use of tamper-evident hardware, secure enclosures, and stringent physical access controls. Additionally, we have incorporated mechanisms for regular physical inspections and audits to detect and prevent any unauthorized physical access to the fog nodes. These measures ensure that even if a fog node is physically compromised, the impact on the system is minimized through the use of encrypted storage and secure boot processes.

• Network-Based Attacks: Fog nodes communicate extensively with each other and with cloud nodes, making them susceptible to various network-based attacks such as man-in-the-middle, denial-of-service, and eavesdropping. We have implemented encryption protocols (such as TLS) to mitigate these risks to secure all communications between nodes. We have also employed advanced intrusion detection and prevention systems to monitor network traffic for suspicious activities and automatically respond to potential threats. Moreover, using virtual private networks (VPNs) and secure tunneling techniques adds an extra layer of protection for data in transit. These measures collectively enhance the security of network communications, ensuring that data integrity and confidentiality are maintained.

• Data Integrity and Confidentiality: Protecting data integrity and confidentiality is paramount, especially when fog nodes process sensitive information. We have adopted a multi-layered approach to safeguard data at rest and in transit. This includes using advanced encryption standards for data storage and transmission, ensuring that even if data is intercepted or accessed, it remains unreadable without the proper decryption keys. Additionally, we have implemented strict access control policies and role-based access control (RBAC) to limit data access to authorized personnel only. Regular security audits and data integrity checks are performed to detect any unauthorized modifications or breaches. Furthermore, using secure key management practices ensures that encryption keys are stored and managed securely, preventing unauthorized access to sensitive data.

• Security Measures To effectively mitigate the identified risks within the fog computing layer, we have implemented a series of enhanced security measures. These measures are designed to provide robust protection against unauthorized access, data interception, and other potential security threats:

- Secure Communication Protocols: All communications between fog nodes and other components within our architecture utilize advanced encryption protocols. By implementing protocols such as Transport Layer Security and Secure Sockets Layer, we ensure that data transmitted across the network remains confidential and secure. These protocols prevent unauthorized access and data interception, safeguarding the integrity and confidentiality of the information exchanged between nodes. Additionally, the use of end-to-end encryption ensures that data remains encrypted throughout its journey, from the source to the destination, further enhancing the security of the communication channels.

- Access Control: We have established stringent access control mechanisms to prevent unauthorized physical and network access to the fog nodes. This includes the implementation of multi-factor authentication (MFA) to verify the identity of users attempting to access the system. Role-based access control (RBAC) policies are employed to ensure that users have access only to the resources necessary for their roles, minimizing the risk of unauthorized access to sensitive data. Furthermore, regular access audits and monitoring help identify and respond to any anomalies or unauthorized access attempts promptly.

- Intrusion Detection Systems (IDS): To enhance the security of the fog computing layer, we have integrated sophisticated Intrusion Detection Systems (IDS) [54, 55]. These systems continuously monitor network traffic and system activities for signs of suspicious behavior or potential security threats. The IDS employs both signature-based and anomaly-based detection techniques to identify known attack patterns and deviations from normal behavior. In the event of a detected threat, the IDS can trigger automated responses, such as blocking malicious traffic or alerting security personnel, to mitigate the impact of the threat in real-time.

V. Discussion

The architecture of the proposed IoT system is designed to handle the heterogeneity of devices, ensuring optimal performance regardless of their diversity. Its adaptive features, scalability, and flexibility allow it to accommodate various devices with varying capabilities. The architecture also adapts to different connectivity patterns, addressing challenges like intermittent connections and varying bandwidth. It is performance metrics show it can handle different device profiles in terms of latency, throughput, and resource utilization. The architecture also incorporates strategies to enhance resiliency, handle node failures, and ensure fault-tolerance in distributed architectures. Future work will focus on enhancing failover efficiency, scalability testing, and developing adaptive resiliency mechanisms.

• Managing heterogeneity: Our architecture has adaptive features designed to handle the inherent heterogeneity of IoT devices. We have sophisticated dynamic resource allocation mechanisms that adjust based on the capabilities of connected devices, ensuring optimal performance regardless of device diversity. For example, different components and Encryption Components adapt their processing and communication strategies depending on the capabilities of the devices and prevailing network conditions. This compatibility ensures that even devices with limited resources or connectivity can effectively participate in the architecture and maintain system performance and efficiency. By tuning resource allocation and component behavior based on the specific needs and constraints of each device, our architecture can seamlessly integrate a wide range of IoT devices, from high-powered industrial machines to low-power sensors, delivering consistent and robust performance across the board.

• Scalability and flexibility: Our proposed architecture is designed with scalability and flexibility at its core, enabling it to accommodate a diverse range of devices with varying capabilities. The scalable design of the architecture allows it to expand horizontally and incorporate additional modular components as needed to meet growing demands. This modularity ensures that the system can easily integrate new functions or components without significant reconfiguration. In addition, elastic resource management in the fog and cloud layers allows the architecture to dynamically adjust resource allocation in response to the fluctuating demands and diverse needs of different devices. This resilience ensures that resources are used efficiently, avoids bottlenecks, and maintains high performance even under varying load conditions. A combination of modular design and elastic resources enables our architecture to maintain optimal performance and resource utilization, providing a flexible and scalable solution for various IoT environments.

• Connection patterns: Our architecture is designed to adapt to different connectivity patterns, addressing challenges such as intermittent connections and varying bandwidth. Advanced network management techniques optimize network utilization based on IoT device connection patterns, ensuring efficient communication regardless of network conditions. Adaptive data routing strategies maintain effective communication paths and ensure reliable data transmission in various connection scenarios. This architecture increases the reliability and performance of IoT deployments.

• Use Cases and examples: To demonstrate the practical application of our architecture, we include examples of diverse IoT use cases that demonstrate its ability to handle various device capabilities and connectivity patterns. In smart cities, our architecture efficiently manages various sensor and actuator devices, each with different performance profiles, ensuring seamless integration and operation. In industrial IoT settings, the architecture adapts to the different needs of different industrial equipment, from high-powered machinery to low-power monitoring devices. These examples demonstrate the versatility and robustness of the architecture in real-world scenarios and demonstrate its ability to maintain high performance and reliability across a wide range of applications. By providing examples, we demonstrate the practical benefits and adaptability of our architecture in diverse IoT environments.

• Experimental results: Our experimental results provide empirical evidence of how the architecture performs with devices with different capabilities and connectivity patterns. We present performance metrics that demonstrate the architecture's ability to handle different device profiles in terms of latency, throughput, and resource utilization. These benchmarks show that the architecture maintains high performance and efficient resource utilization even when integrating heterogeneous devices. Additionally, we include case studies or scenarios where the architecture has been tested with various devices, providing real-world validation of its compatibility and effectiveness. These experimental results and case studies emphasize the robustness and versatility of the architecture and confirm its suitability for various IoT deployments.

• Fault-tolerance in distributed architectures: In the updated architecture, we have incorporated several strategies to enhance resiliency, handle node failures, and ensure fault-tolerance despite its distributed nature. These strategies include redundancy and replication of critical components. Load balancing mechanisms dynamically distribute workloads to prevent bottlenecks and single points of failure. Continuous health monitoring and automated failure detection systems promptly identify node failures, allowing for quick recovery and minimal downtime. Failover mechanisms reroute requests to operational nodes, maintaining service availability. Checkpointing techniques and distributed consistency protocols ensure data integrity and smooth recovery from failures. Future work will focus on enhancing failover efficiency, scalability testing, and developing adaptive resiliency mechanisms to adjust strategies based on real-time conditions dynamically.

VI. Conclusion and Open issues

In this paper, we introduce a comprehensive full-cloud-fog architecture designed for EaaS platforms. This innovative architecture capitalizes on the benefits of cloud and fog nodes to significantly reduce end-to-end delay while concurrently elevating the success rate of service requests. The fog layer hosts the most frequently accessed EaaS components within the full-cloud-fog framework, thereby reducing delay. Should fog-layer components encounter resource limitations, the architecture seamlessly taps into the resources of cloud nodes. This strategic utilization strategy curtails the occurrence of unaddressed requests. Notably, our evaluated results substantiate the efficacy of the proposed full-cloud-fog architecture, exhibiting an impressive 81% higher throughput than its existing counterparts.

Despite the achievements realized through our proposed EaaS architecture, avenues for further enhancing its performance remain open. We outline several ideas that form the cornerstone of our ongoing endeavors:

- Leveraging a machine learning model to predict optimal components during the dispatching phase can substantially enhance the architecture's throughput. This model can be trained using parameters such as queue length, resource availability, and average throughput of individual components.
- Another learning model can be harnessed to ascertain whether the processing of a resource-intensive request is viable. Rather than addressing a single resource-intensive request, the platform can opt to tackle multiple smaller requests, effectively bolstering both throughput and user satisfaction. Machine learning techniques can inform the decision-making process.

- Introducing a caching memory within the GC proves advantageous in scenarios characterized by frequent requests for decrypting the same shared data. In such cases, the GC can store decrypted data in its cache and subsequently validate access permissions with the KC. The GC's encryption of cache data enhances security. Consequently, the role of the DC component is diminished, leading to a noteworthy reduction in latency while concurrently optimizing DC resources for handling other requests.

Furthermore, we have applied plans to investigate the influence of architecture types on EaaS platforms catering to diverse service categories.

• Security and Privacy as Cornerstones: Amidst the dynamic landscape of IoT, the bedrock of data security and privacy looms large. This paper underscores the compelling need to explore the intricate security mechanisms embedded within EaaS architectures comprehensively. We intend to illuminate how the proposed full-cloud-fog architecture stands resilient against the multifaceted security challenges that characterize IoT environments.

• Navigating Privacy: In an age dominated by interconnectivity, the sanctity of user data privacy becomes sacrosanct. We acknowledge the pressing nature of safeguarding user privacy and adhering to prevailing regulatory frameworks. This survey ardently embraces the responsibility of scrutinizing the efficacy of EaaS platforms, including the purview of our full-cloud-fog architecture, in embracing privacy-centric measures. Our spotlight on subjects like data obfuscation, user-centric consent protocols, and impervious data transit will offer a comprehensive understanding of how these platforms strive to preserve user privacy. Moreover, we assess the intersection of innovative privacy-enhancing technologies—homomorphic encryption and differential privacy—with EaaS platforms [56].

We have emphasized the robust scalability capabilities of our proposed architecture. Additionally, we have outlined future research directions aimed at further enhancing these features:

• Advanced Load Balancing Algorithms: Future work will explore more sophisticated load-balancing algorithms that can predict and adapt to changing workloads more effectively. These algorithms will leverage machine learning techniques to anticipate demand patterns and proactively adjust resource allocation, further optimizing performance and resource utilization.

• Optimizing Dynamic Resource Allocation: We plan to refine the dynamic resource allocation strategies to improve efficiency and responsiveness. This includes developing more granular resource management policies to allocate resources based on specific application requirements and user priorities, ensuring critical tasks receive the necessary resources without delay.

• Exploration of Advanced Technologies: Future research will also explore integrating advanced technologies such as edge AI and blockchain to enhance the architecture's security, efficiency, and scalability.

Acknowledgment

This research work is partially supported by the European Union's Horizon Europe research and innovation program HORIZON-JU-SNS-2022 under the RIGOUROUS project (Grant No. 101095933). The work is also partially supported by the National Natural Science Foundation of China (Grant No. 62372076) and the Sichuan Science and Technology Innovation Talent Program (Grant No. 24CXRC0192).

References

[1] A. Javadpour, F. Ja'fari, T. Taleb, and C. Benzaïd, "Reinforcement learning-based slice isolation against ddos attacks in beyond 5g networks," *IEEE Transactions on Network and Service Management*, 2023.

[2] A. Javadpour, F. Ja'fari, T. Taleb, and C. Benzaïd, "Enhancing 5g network slicing: Slice isolation via actor-critic reinforcement learning with optimal graph features," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE, 2023, pp. 31–37.

[3] R. Yadav, W. Zhang, I. A. Elgendy, G. Dong, M. Shafiq, A. A. Laghari, and S. Prakash, "Smart healthcare: Rl-based task offloading scheme for edge-enable sensor networks," *IEEE Sensors Journal*, vol. 21, no. 22, pp. 24 910–24 918, 2021.

[4] N. G., "How many iot devices are there in 2023?" https://techjury.net/blog/how-many-iot-devices-are-there/#gref, 2023, [Accessed: Feb 2023].

[5] P. B. Hirsch, "Footprints in the cloud: the hidden cost of it infrastructure," *Journal of Business Strategy*, vol. 43, no. 1, pp. 65–68, 2022.

[6] A. Javadpour, F. Ja'fari, T. Taleb, M. Shojafar, and B. Yang, "Scema: an sdn-oriented cost-effective edge-based mtd approach," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 667–682, 2022.

[7] R. Yadav, W. Zhang, O. Kaiwartya, H. Song, and S. Yu, "Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 198–14 211, 2020.

[8] A. Javadpour, F. Ja'fari, T. Taleb, and M. Shojafar, "A cost-effective mtd approach for ddos attacks in software-defined networks," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 4173–4178.

[9] L. Wang, Y. Tian, and J. Xiong, "Achieving reliable and anti-collusive outsourcing computation and verification based on blockchain in 5g-enabled iot," *Digital Communications and Networks*, vol. 8, no. 5, pp. 644–653, 2022.

[10] A. Javadpour, F. Ja'fari, T. Taleb, Y. Zhao, Y. Bin, and C. Benzaïd, "Encryption as a service for iot: Opportunities, challenges and solutions," *IEEE Internet of Things Journal*, pp. 1–1, 2023.

[11] A. Javadpour, F. Ja'fari, T. Taleb, M. Shojafar, and C. Benzaïd, "A comprehensive survey on cyber deception techniques to improve honeypot performance," *Computers & Security*, p. 103792, 2024.

[12] S. V, K. S. Rokhade, and V. N. B, “Email protection in the digital age: Evaluating symmetric cryptographic algorithms,” in *2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, 2024, pp. 1–6.

[13] S. I. Akhtar, A. Rauf, M. F. Amjad, and H. Abbas, “Inter-cloud data security framework, compliance and trust,” 2022.

[14] P. Chinnasamy and P. Deepalakshmi, “Hcac-ehr: hybrid cryptographic access control for secure ehr retrieval in healthcare cloud,” *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–19, 2022.

[15] R. B. Naik and U. Singh, “A review on applications of chaotic maps in pseudo-random number generators and encryption,” *Annals of Data Science*, vol. 11, no. 1, pp. 25–50, 2024.

[16] W. Xue, C. Wang, and J. Wang, “Research on cryptography as a service technique based on commercial cryptography,” in *2022 IEEE 2nd International Conference on Electronic Technology, Communication and Information (ICETCI)*. IEEE, 2022, pp. 260–264.

[17] A. Javadpour, F. Ja’fari, and T. Taleb, “Encryption as a service: A review of architectures and taxonomies,” in *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, 2024, pp. 36–44.

[18] A. Y. Al-Tamimi, M. A. Snober, and Q. A. Al-Haija, “A performance evaluation study to optimize encryption as a service (eaas),” in *Proceedings of Fourth International Conference on Communication, Computing and Electronics Systems: ICCCES 2022*. Springer, 2023, pp. 681–691.

[19] A. Javadpour, F. Ja’fari, T. Taleb, C. benzaid, L. Rosa, P. Tomás, and L. Cordeiro, “Deploying testbed docker-based application for encryption as a service in kubernetes,” *The 32nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2024.

[20] P. K. Deb, A. Mukherjee, and S. Misra, “Ceaas: Constrained encryption-as-a-service in fog-enabled iot,” *IEEE Internet of Things Journal*, 2022.

[21] B. Yang, F. Zhang, and S. U. Khan, “An encryption-as-a-service architecture on cloud native platform,” in *2021 International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2021, pp. 1–7.

[22] T. Z. Benmerar, T. Theodoropoulos, D. Fevereiro, L. Rosa, J. Rodrigues, T. Taleb, P. Barone, G. Giuliani, K. Tserpes, and L. Cordeiro, “Towards establishing intelligent multi-domain edge orchestration for highly distributed immersive services: a virtual touring use case,” *Cluster Computing*, pp. 1–31, 2024.

[23] C. Wang, B. Jia, H. Yu, X. Li, X. Wang, and T. Taleb, “Deep reinforcement learning for dependency-aware microservice deployment in edge computing,” in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 5141–5146.

[24] P. S. JosephNg, Z. C. EricMok, K. Y. Phan, J. Sun, and Z. Wei, “Mitigating social media cybercrime: Revolutionising with aes encryption and generative ai,” *Journal of Advanced Research in Applied Sciences and Engineering Technology*, vol. 46, no. 2, pp. 124–154, 2025.

[25] A. Javadpour, A. K. Sangaiah, F. Ja’Fari, P. Pinto, H. Memarzadeh-Tehran, S. Rezaei, and F. Saghafi, “Toward a secure industrial wireless body area network focusing mac layer protocols: an analytical review,” *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 2028–2038, 2022.

[26] A. Javadpour, F. Ja’fari, T. Taleb, Y. Zhao, Y. Bin, and C. Benzaïd, “Encryption as a service for iot: opportunities, challenges and solutions,” *IEEE Internet of Things Journal*, 2023.

[27] K. Balaji and S. Manikandasaran, “Data security and deduplication framework for securing and deduplicating users’ data in public and private cloud environment,” *Journal of Scientific Research*, vol. 14, no. 1, pp. 153–165, 2022.

[28] D. Yu, Z. Ma, and R. Wang, “Efficient smart grid load balancing via fog and cloud computing,” *Mathematical Problems in Engineering*, vol. 2022, pp. 1–11, 2022.

[29] A. El Bouchti, S. Bahsani, and T. Nahhal, “Encryption as a service for data healthcare cloud security,” in *2016 fifth international conference on future generation communication technologies (FGCT)*. IEEE, 2016, pp. 48–54.

[30] M. Ibtihal, N. Hassan *et al.*, “Homomorphic encryption as a service for outsourced images in mobile cloud computing environment,” in *Cryptography: breakthroughs in research and practice*. IGI Global, 2020, pp. 316–330.

[31] A. Merdan, H. Aslan, and N. Abdelbaki, “Design and implementation of a dockerized, cross platform, multipurpose cryptography as a service framework featuring scalability, extendibility and ease of integration,” in *2022 20th International Conference on Language Engineering (ESOLEC)*, vol. 20. IEEE, 2022, pp. 152–157.

[32] R. Xu and J. B. Joshi, “Enabling attribute based encryption as an internet service,” in *2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)*. IEEE, 2016, pp. 417–425.

[33] S. Tahir, S. Ruj, A. Sajjad, and M. Rajarajan, “Fuzzy keywords enabled ranked searchable encryption scheme for a public cloud environment,” *Computer Communications*, vol. 133, pp. 102–114, 2019.

[34] S. Tahir, L. Steponkus, S. Ruj, M. Rajarajan, and A. Sajjad, “A parallelized disjunctive query based searchable encryption scheme for big data,” *Future Generation Computer Systems*, vol. 109, pp. 583–592, 2020.

[35] A. A. Fairosebanu and A. C. N. Jebaseeli, “Data security in cloud environment using cryptographic mechanism,” *Bulletin of Electrical Engineering and Informatics*, vol. 12, no. 1, pp. 462–471, 2023.

[36] H. Zhang, B. Qin, T. Tu, Z. Guo, F. Gao, and Q. Wen, “An adaptive encryption-as-a-service architecture based on fog computing for real-time substation communications,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 658–668, 2019.

[37] G. Li, J. Yan, L. Chen, J. Wu, Q. Lin, and Y. Zhang, “Energy consumption optimization with a delay thresh-

old in cloud-fog cooperation computing," *IEEE access*, vol. 7, pp. 159 688–159 697, 2019.

[38] D. Darwish, "Emerging trends in cloud computing analytics, scalability, and service models," 2024.

[39] H. Rahmani, E. Sundararajan, Z. M. Ali, and A. M. Zin, "Encryption as a service (eaas) as a solution for cryptography in cloud," *Procedia Technology*, vol. 11, pp. 1202–1210, 2013.

[40] M. Gd and M. Bist, "Eaas-erp as a service," *Journal of Information Operations Management*, vol. 3, no. 1, pp. 141–145, 2012.

[41] A. El Bouchti, S. Bahsani, and T. Nahhal, "Encryption as a service for data healthcare cloud security," in *2016 fifth international conference on future generation communication technologies (FGCT)*. IEEE, 2016, pp. 48–54.

[42] D. Unal, A. Al-Ali, F. O. Catak, and M. Hammoudeh, "A secure and efficient internet of things cloud encryption scheme with forensics investigation compatibility based on identity-based encryption," *Future Generation Computer Systems*, vol. 125, pp. 433–445, 2021.

[43] R. Q. Shaddad, A. A. Saeed, R. Q. Naji, and A. M. Baalawi, "Hybrid traffic dispersion and network densification scheme for 5g millimeterwave wireless networks," in *2019 First International Conference of Intelligent Computing and Engineering (ICOICE)*. IEEE, 2019, pp. 1–6.

[44] D. Glăvan, C. Răcuciu, R. Moinescu, and S. Eftimie, "Sniffing attacks on computer networks," *Scientific Bulletin" Mircea cel Batran" Naval Academy*, vol. 23, no. 1, pp. 202A–207, 2020.

[45] N. Sivasankari and S. Kamalakkannan, "Detection and prevention of man-in-the-middle attack in iot network using regression modeling," *Advances in Engineering Software*, vol. 169, p. 103126, 2022.

[46] E. Dervisevic, F. Lauterbach, P. Burdiak, J. Rozhon, M. Slívová, M. Plakalovic, M. Hamza, P. Fazio, M. Voznak, and M. Mehic, "Simulations of denial of service attacks in quantum key distribution networks," in *2022 XXVIII International Conference on Information, Communication and Automation Technologies (ICAT)*. IEEE, 2022, pp. 1–5.

[47] L. Govindaraj, B. Sundan, and A. Thangasamy, "An intrusion detection and prevention system for ddos attacks using a 2-player bayesian game theoretic approach," in *2021 4th International Conference on Computing and Communications Technologies (ICCCT)*. IEEE, 2021, pp. 319–324.

[48] A. R. Putri, R. Munadi, and R. M. Negara, "Performance analysis of multi services on container docker, lxc, and lxd," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 5, pp. 2008–2011, 2020.

[49] M. N. Alenezi, H. Alabdulrazzaq, and N. Q. Mohammad, "Symmetric encryption algorithms: Review and evaluation study," *International Journal of Communication Networks and Information Security*, vol. 12, no. 2, pp. 256–272, 2020.

[50] S. Al Mamun, M. A. Mahmood, and M. A. Amin, "Ensuring security of encrypted information by hybrid aes and rsa algorithm with third-party confirmation," in *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 2021, pp. 337–343.

[51] S. Banik, A. Bogdanov, and F. Regazzoni, "Atomic-aes: A compact implementation of the aes encryption/decryption core," in *Progress in Cryptology–INDOCRYPT 2016: 17th International Conference on Cryptology in India, Kolkata, India, December 11-14, 2016, Proceedings 17*. Springer, 2016, pp. 173–190.

[52] "Pure-python implementation of the aes block-cipher and common modes of operation," https://pypi.org/project/pyaes/, 2017, [Accessed: Feb 2023].

[53] "Crypto.cipher package," https://pycryptodome.readthedocs.io/en/latest/src/cipher/cipher.html, 2022, [Accessed: Feb 2023].

[54] A. Javadpour, P. Pinto, F. Ja'fari, and W. Zhang, "Dmaidps: A distributed multi-agent intrusion detection and prevention system for cloud iot environments," *Cluster Computing*, vol. 26, no. 1, pp. 367–384, 2023.

[55] A. Javadpour, S. K. Abharian, and G. Wang, "Feature selection and intrusion detection in cloud environment based on machine learning algorithms," in *2017 IEEE international symposium on parallel and distributed processing with applications and 2017 IEEE international conference on ubiquitous computing and communications (ISPA/IUCC)*. IEEE, 2017, pp. 1417–1421.

[56] S. K. Poorazad, C. Benzaid, and T. Taleb, "A novel buffered federated learning framework for privacy-driven anomaly detection in IIoT," in *Proceedings of the IEEE Global Communications Conference (Globecom)*, Cape Town, South Africa, Dec. 2024.

Amir Javadpour holds a Ph.D. in Mathematics/Cybersecurity from Guangzhou University, China. His academic journey is distinguished by numerous publications in highly-ranked journals and prestigious conferences. These works span a diverse range of topics, reflecting his deep expertise in Cybersecurity, Cloud Computing, Software-Defined Networking (SDN), Big Data, Intrusion Detection Systems (IDS), the Internet of Things (IoT), Moving Target Defense (MTD), Machine Learning (ML), Reinforcement Learning, and optimization algorithms.

Beyond his publications, he has made substantial contributions as a reviewer and author for leading academic venues, including IEEE Transactions on Cloud Computing, IEEE Transactions on Network Science and Engineering, and ACM Transactions on Internet Technology. His reviewing efforts extend to various reputable journals under Springer and Elsevier. Additionally, he serves as a dedicated Technical Program Committee (TPC) member for several international conferences. Dr. Javadpour is actively in international collaborations, particularly with European consortiums on funded projects such as Inspire-5Gplus (https://www.inspire-5gplus.eu/) and Rigourous (https://rigourous.eu/). These partnerships have resulted in significant contributions to the field, with his work being featured in top-tier journals and conferences, including Globecom, IEEE Transactions on Industrial Informatics (TII), IEEE Transactions on Information Forensics and Security (TIFS), IEEE Transactions on Network and Service Management (TNSM), and ACM Transactions on Sensor Networks (TOSN). In addition to his research and publication efforts, he is deeply committed to mentoring and supervising Master's and Doctoral students. His extensive experience in this area has equipped him with the skills and confidence necessary to lead a research group and conduct independent, high-impact research.

Forough Ja'fari is a Senior Researcher in cybersecurity and computer science. She received her Bachelor's degree from Sharif University of Technology and her Master's degree in Computer Network Engineering from Yazd University, Iran. She is a visiting scholar researcher at Guangzhou University, China. Cloud computing, software-defined Networking (SDN), cyber deception, Intrusion Detection Systems (IDS), Internet of Things (IoT), Moving Target Defence (MTD), and Machine Learning are some of her research interests. She is currently a Guest Editor (GE) of Cluster Computing (CLUS) Journal and a reviewer for several journals and conferences.

Tarik Taleb Prof. Tarik Taleb is currently a Chair Professor at Ruhr University Bochum, Bochum, Germany. Prior to that, he was a full professor at the Centre for Wireless Communications (CWC) – Networks and Systems Unit, Faculty of Information Technology and Electrical Engineering, The University of Oulu. Between Oct. 2014 and Dec. 2021, he was a Professor at the School of Electrical Engineering, Aalto University, Finland. Prior to that, he was working as Senior Researcher and 3GPP Standards Expert at NEC Europe Ltd, Heidelberg, Germany. He was then leading the NEC Europe Labs Team working on R&D projects on carrier cloud platforms. Before joining NEC and till Mar. 2009, he worked as assistant professor at the Graduate School of Information Sciences, Tohoku University, Japan, in a lab fully funded by KDDI, the second largest mobile operator in Japan. From Oct. 2005 till Mar. 2006, he worked as a research fellow at the Intelligent Cosmos Research Institute, Sendai, Japan. He received his B. E degree in Information Engineering with distinction, M.Sc. and Ph.D. degrees in Information Sciences from Tohoku Univ., in 2001, 2003, and 2005, respectively. Prof. Taleb's research interests lie in the field of telco cloud, network softwarization and network slicing, AI-based software defined security, immersive communications, mobile multimedia streaming, next generation mobile networking. Prof. Taleb was also directly engaged in the development and standardization of the Evolved Packet System as a member of 3GPP's System Architecture working group 2. Prof. Taleb served on the IEEE Communications Society Standardization Program Development Board. As an attempt to bridge the gap between academia and industry, Prof. Taleb founded the "IEEE Workshop on Telecommunications Standards: from Research to Standards", a successful event that got awarded the "best workshop award" by IEEE Communication Society (ComSoC). Based on the success of this workshop, Prof. Taleb also founded and served as the steering committee chair of the IEEE Conf. on Standards for Communications and Networking. Prof. Taleb served as the general chair of the 2019 edition of the IEEE Wireless Communications and Networking Conference (WCNC'19) held in Marrakech, Morocco. He was the guest editor in chief of the IEEE JSAC Series on Network Softwarization and Enablers. He was on the editorial board of the IEEE Transactions on Wireless Communications, IEEE Wireless Communications Magazine, IEEE Journal on Internet of Things, IEEE Transactions on Vehicular Technology, IEEE Communications Surveys & Tutorials, and a number of Wiley journals. Till Dec. 2016, he served as chair of the Wireless Communications Technical Committee, the largest in IEEE ComSoC. He also served as Vice Chair of the Satellite and Space Communications Technical Committee of IEEE ComSoc (2006 - 2010). Prof. Taleb is the recipient of the 2021 IEEE ComSoc Wireless Communications Technical Committee Recognition Award (Dec. 2021), the 2017 IEEE ComSoc Communications Software Technical Achievement Award (Dec. 2017) for his outstanding contributions to network softwarization. He is also the (co-) recipient of the 2017 IEEE Communications Society Fred W. Ellersick Prize (May 2017), the 2009 IEEE ComSoc Asia-Pacific Best Young Researcher award (Jun. 2009), the 2008 TELECOM System Technology Award from the Telecommunications Advancement Foundation (Mar. 2008), the 2007 Funai Foundation Science Promotion Award (Apr. 2007), the 2006 IEEE Computer Society Japan Chapter Young Author Award (Dec. 2006), the Niwa Yasujirou Memorial Award (Feb. 2005), and the Young Researcher's Encouragement Award from the Japan chapter of the IEEE Vehicular Technology Society (VTS) (Oct. 2003). Some of Prof. Taleb's research work have been also awarded best paper awards at prestigious IEEE-flagged conferences.

Bin Yang received his PhD in systems information science from Future University Hakodate, Japan, in 2015. He is a Professor at the School of Computer and Information Engineering, Chuzhou University, China, and a Senior Researcher with MOSA!C Lab, Finland. His research interests include unmanned aerial vehicle networks, cyber security and the Internet of Things. (Email: yangbinchi@gmail.com)

Chafika Benzaïd is currently a senior research fellow at University of Oulu, Finland. Between Nov. 2018 and Dec. 2021, she was senior researcher at Aalto University. Before that, she worked as an associate professor at University of Sciences and Technology Houari Boumediene (USTHB). She holds Engineer, Magister and "Doctorat ès Sciences" degrees from USTHB. Her research interests lie in the field of 5G/6G, SDN, Network Security, AI Security, and AI/ML for zero-touch security management. She is an ACM professional member.

Zhao Yue born in 1983, received the B.S. degree from the North China Institute of Science and Technology, Langfang, China, in 2006, and the Ph.D. degree from Southwest Jiaotong University, Chengdu, China, in 2012. He is currently a professorate senior engineer with the National Key Laboratory of Security Communication, Chengdu, China. He is also the master student supervisor of School of Computing and Artificial Intelligence of Southwest Jiaotong University, School of Computer and Software Engineering of Xihua University, and School of Computer and Information Engineering of Chuzhou University. His research interests include wireless networks and information security. (Email: yuezhao@foxmail.com)