Article

A Mechanism for Securing IoT-enabled Applications at the Fog Layer

Nadeem Abbas [1], Muhammad Asim [1], Noshina Tariq [1], Thar Baker [2,*] and Sohail Abbas [3]

[1]   Department of Computer Science, National University of Computer and Emerging Sciences, Islamabad 44000, Pakistan; i161004@nu.edu.pk (N.A.); muhammad.asim@nu.edu.pk (M.A.); noshtariq@yahoo.com (N.T.)

[2]   Department of Computer Science, Liverpool John Moores University, Liverpool L3 3AF, UK

[3]   Department of Computer Science, University of Sharjah, Sharjah 27272, UAE; sabbas@sharjah.ac.ae

[*]   Correspondence: t.baker@ljmu.ac.uk; Tel.: +44(0)-151-231-2371

check for
updates

Abstract: The Internet of Things (IoT) is an emerging paradigm branded by heterogeneous technologies composed of smart ubiquitous objects that are seamlessly connected to the Internet. These objects are deployed as Low power and Lossy Networks (LLN) to provide innovative services in various application domains such as smart cities, smart health, and smart communities. The LLN is a form of a network where the interconnected devices are highly resource-constrained (i.e., power, memory, and processing) and characterized by high loss rates, low data rates, and instability in the communication links. Additionally, IoT devices produce a massive amount of confidential and security-sensitive data. Various cryptographic-based techniques exist that can effectively cope with security attacks but are not suitable for IoT as they incur high consumption of resources (i.e., memory, storage and processing). One way to address this problem is by offloading the additional security-related operations to a more resourceful entity such as a fog-based node. Generally, fog computing enables security and analysis of latency-sensitive data directly at the network's edge. This paper proposes a novel Fog Security Service (FSS) to provide end-to-end security at the fog layer for IoT devices using two well-established cryptographic schemes, identity-based encryption, and identity-based signature. The FSS provides security services such as authentication, confidentiality, and non-repudiation. The proposed architecture would be implemented and evaluated in an OPNET simulator using a single network topology with different traffic loads. The FSS performed better when compared with the APaaS and the legacy method.

Keywords: Internet of Things; fog computing; security; privacy

1. Introduction

The Internet of Things is an emerging paradigm that provides seamless and ubiquitous connectivity to smart devices over the Internet. The IoT is composed of diverse devices, such as Radio-frequency identification (RFID) tags, sensors, and smartphones. These devices can potentially cooperate and collaborate to provide diverse services like, for example, smart cities, smart communities, emergency response, smart metering, home automation, intelligent transportation system, and the Internet of connected vehicles [1,2] Interconnected IoT nodes generate a massive amount of data that are needed to be private and confidential. The data must be stored, processed, and presented in a secure, efficient, and easily interpretable form [3]. According to Cisco anticipation for the year 2020, 50 billion devices would capture the Internet and will reach 500 billion by the year 2025 [1,4]. The Federal Trade Commission's (FTC) report [5] on IoT urged businesses to adopt best practices to address consumer privacy concerns and security risks. It warns that smart devices are involved in

harvesting huge amount of personal information and exposed to a variety of potential security threats, such as unauthorized access and misuse of personal information.

Cloud computing can enable IoT to have the privilege of cost-effective on-demand services for intensive processing and big data storage. However, in the cloud-based IoT applications, there remain unresolved issues such as the requirement of high capacity client access link, variable latency, lack of mobility support, and security and location-awareness [6,7]. In particular, applications such as real-time monitoring, industrial automation (Industry 4.0), sensor and actuators networks, and intelligent transportation systems are too latency-sensitive to be deployed on cloud infrastructures. The fog computing paradigm addresses this challenge by offering computing resources and services to end-users at the edge of the network [8]. Fog computing puts a substantial amount of communication, control, storage, and management at the edge of a network as opposed to establishing dedicated channels to a more centralized remote cloud infrastructure. This approach reduces service latency, improves the Quality of Service (QoS), and provides a superior experience to end-users [7,9]. However, fog computing is a non-trivial extension of cloud computing, and it has been proposed in the IoT. Thus, security and privacy issues are inherited in fog computing. In addition, due to the underlying differences between cloud computing and fog computing, security solutions proposed for cloud computing may not suit fog services available to end-users at the edge of the networks. Moreover, IoT applications are deployed as Low power and Lossy Networks (LLN), such as wireless sensor networks, smart city, and smart health applications. The LLN is a class of networks where the interconnected devices are highly resource constrained (power, memory, processing) and are characterized by high loss rates, low data rates, and instability in the communication links [10]. Due to their limited characteristics, they are not suitable for generating cryptographic keys and for computing complex cryptographic operations. Therefore, addressing security concerns at the fog layer could enable a fog paradigm to provide not only additional computational resources but also adequate level of security to minimize cyber-attacks in the IoT environments.

1.1. Problem Statement

Lee et al. [11] suggested that IoT fog nodes are expected to collect and process personal information originated from millions of IoT devices. While existing security solutions can address some threats, there are other issues that pertain specifically to fog computing environments and which pose unique challenges for security researchers and practitioners [11,12]. For example, authentication is an essential requirement for protecting IoT data both in transit and at rest [13]. Unfortunately, many IoT devices do not have enough memory and Central Processing Unit (CPU) power to perform cryptographic operations that are required by most of today's authentication protocols. For instance, a simple RFID tag consists of a single 16-bit processor, often running at 6-12 MHz in an energy saving mode, with a Random Access Memory (RAM) of 512 bytes and 16 Kbytes of flash for program storage. A micro-controller can spend nearly 0.7μJ on each 16-bit operation at 12 MHz. Thus, if we double the microcontroller (MCU) overhead to accommodate security-related operations on each packet, then approximately 20,000 operations, at 16 bits, are expected to ensure data security, privacy, and validation [14]. Therefore, IoT's focus should be more on their core functionalities rather than spending their valuable resources on generating keys and computing cryptographic operations. One way to address this problem is by offloading the additional security-related operations to a more resourceful entity such as a fog node(s).

1.2. Contributions

In this paper, we present a novel security mechanism which addresses the authentication, confidentiality, and non-repudiation for IoT devices at the fog layer. The security mechanism works as a Fog Security Service (FSS) that helps in the authentication of IoT devices, confidentiality of data generated by IoT devices, and non-repudiation (an assurance that someone cannot deny

their authenticity). The FSS provides end-to-end security between IoT devices and the fog layer. Our contributions are listed below:

- A Fog Security Service (FSS) to provide identity-based authentication, the integrity of data, and non-repudiation of connected nodes.
- A Private Key Generator (PKG) at fog layer to enhance end-to-end security between IoT and fog layer.
- Implementation and evaluation of the proposed FSS to demonstrate the appropriateness of the proposed mechanism.

1.3. Paper Structure

This paper is structured as follows. In Section 2, the fog computing architecture is explained. The security issues of fog-based IoT networks are discussed in Section 3. Section 4 provides a literature survey and a summary of earlier contributions. Section 5 details our proposed fog-based cryptographic solution. Section 6 outlines the implementation and evaluation details of the proposed mechanism. Finally, Section 7 concludes the paper.

2. Fog Computing Architecture

We adopted a more generic IoT-Fog architecture which has been proposed in [1] and in-line with other fog architectures [1,15,16] to provide an insight to better comprehend the abilities and functionalities of the fog layer. Moreover, to the best of our knowledge, there is no standard architecture for fog computing [17]. According to the selected architecture, fog computing can be divided into two categories: cloud-fog-device architecture and fog-device architecture, as shown in Figure 1.
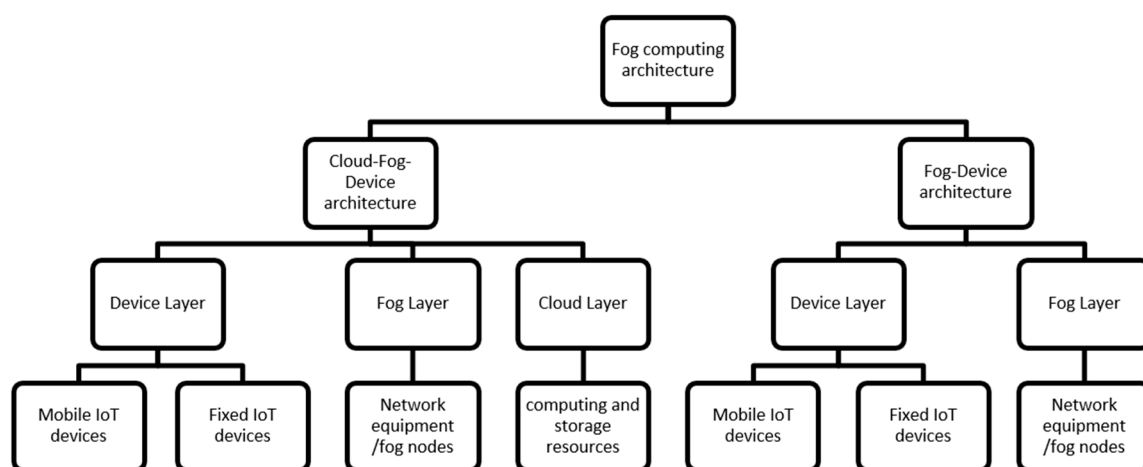

Figure 1. Fog computing architecture.

2.1. Cloud-Fog-Device Architecture

Cloud-Fog-Device architecture has three ordered layers known as device, fog and cloud layers, with inter- and cross- layer communication, as shown in Figure 2. On moving from device to cloud layer, the computing and storage capacities decrease chronologically. To improve flexibility, computation, and storage capacity, virtualization is used on both the cloud and fog layers. The cloud layer consists of computing and permanent storage resources. It also imposes quality-based policies on the fog layer to cater delay-sensitive services. Whereas the fog layer provides data processing and transient storage services near to the network for IoT devices. It also provides several characteristics to IoT devices, such as mobility of the end devices, heterogeneity, location awareness, low latency, and capacity of processing a high number of IoT devices [18]. It incorporates network equipment like, for example, gateways and routers. The device layer may have mobile IoT devices such as smart cars or static IoT

devices with predefined operations such as sensors installed on certain locations. It may consist of different physical objects and mechanical and digital devices. These devices can be anything equipped with storage, communication, processing, and computation capabilities with different capacities, such as mobile phones, RFID tags, laptops, vehicles, fridges, actuators, and ovens. Each thing possesses uniquely identifiable characteristics that are embedded in its computer system to interoperate with existing Internet infrastructure [19].
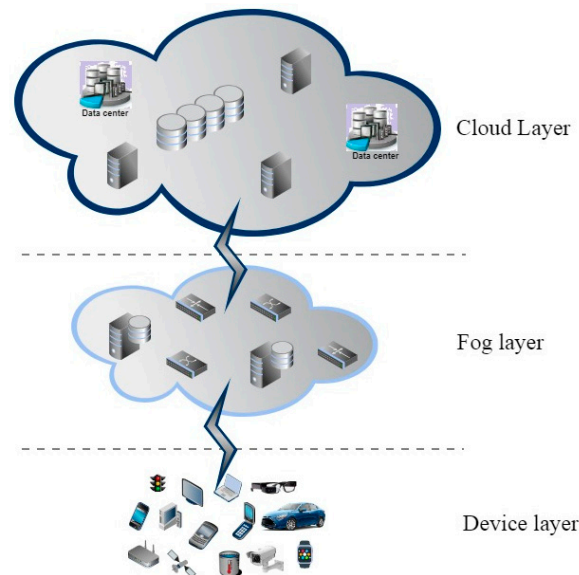

Figure 2. The Cloud-Fog-Device architecture.

2.2. Fog-Device Architecture

In cloud-fog-device architecture, the fog nodes offer transient real-time processing and storage and send summarized data periodically to the cloud layer. However, in fog-device architecture, fog nodes have the highest level, which provides different services without the intervention of the cloud layer. A fog layer can be introduced as a mini-cloud to interoperate with IoT devices without the involvement of cloud servers. However, this entirely depends on the need and requirement of IoT applications, e.g., decentralized vehicular navigation, smart traffic lights, and local content distribution [20]. Our previous paper [8] raised the question of who serves IoT better: Cloud, only; edge, only; or both, together? It then answered this question by identifying and discussing a set of collaborative scenarios with respect to clouds' and edges' duties.

The fog computing layer in both architectures can be leveraged by the security services such as FSS to cope with security attacks which can compromise the confidentiality, integrity, and availability of IoT devices. The FSS can be further extended to provide secure communication between cloud and fog services, but this feature is not the focus of this paper.

3. Security Threats

Local data processing, storage, and analysis make it challenging for external invaders to get access to the user's data and information [21]. However, integration of IoT with a fog layer has posed new security threats and risks [1,22]. Though there are many security mechanisms and services, all are not equally worthy and befitting for fog computing paradigm due to its unique architecture and characteristics. Following are some crucial attributes which must be considered for securing an end-to-end communication and data of IoT devices at the fog layer.

- Authentication: It is a process of validation and verification to prove one's identity as a legitimate user.

- Confidentiality: This property ensures that data is not exposed to unauthorized sources. It ensures adversaries do not gain unauthorized access to data.
- Integrity: Integrity refers to the completeness and accuracy of data. It ensures that data have not been changed and hence received as accurately as sent.
- Availability: It guarantees provisioning of network services and data to authorized users when required.
- Non-repudiation: It refers to ownership of data. It ensures that sender cannot deny having sent the data and the receiver cannot deny having received the data.

There is a great computation and storage overhead involved in existing security protocols such as Secure Sockets Layer (SSL) based communication. For example, it is not worth applying existing Public Key Infrastructure (PKI) based systems on all IoT devices due to their excessive computation and storage. These protocols provide hard security in terms of fixed and predefined large sized keys, therefore causing memory and processing overhead. Therefore, we proposed a fog-based security service for preserving security concerns against authentication, confidentiality, and non-repudiation for fog-based IoT systems. Our scheme is effective against several attacks made on these security attributes. A list of some of those attacks is given in Table 1 along with their description.

Table 1. Security Threats.

Property	Attack	Description
Authentication	Brute Force	The attacker guesses a person's password, user name, secret key (e.g., used for encryption and decryption), and credit card number by each and every possible combination using automated trial and error process [23].
	Insufficient Authentication	Allow invaders to access a website that contains sensitive and important contents. These websites are not directly addressable without the user's necessity to accurately verify their identity [23].
	Man-in-the-Middle	The adversary devises access the networks and insert themselves in between the server and User to gain unauthorized control [24].
	Replay attacks	The identity of IoT devices is spoofed, altered, or replayed in order to intercept or retransmit the data [25].
	Dictionary	All possible words from a dictionary are used to make an attack on authentication data [26].
	Eavesdropping	Malicious invaders capture the packets and read its content by listening the communication channels. If the encryption and decryption algorithms are not used in data, then this attack may be quite effective [25].
	Session Hijacking	Transmission Control Protocol (TCP) session is hijacked to steal session tokens to gain the unauthorized access to a server [27].
	Key and/or Certificate Replication attack	Duplicate keys or certificates of identification proof are used to create ambiguity to disrupt the identification and authentication process [28].
Confidentiality	Packet Capturing (Packet Sniffing)	Attacker captures the data and information packets from Ethernet frames during communication. They can also read the sensitive information such as passwords, usernames, and credit card numbers if the traffic of the network is not encrypted [29,30].
	Wiretapping	One or more edges may get affected in such attacks to compromise the transient data confidentiality. Adversaries wiretap a links to obtain a part of data for decoding a packet [31].

Table 1. *Cont.*

Property	Attack	Description
Non-Repudiation	Repudiation Attack	Either false information is spread, or real event or transaction denial is attempted by the attacker to prove themselves as legitimate participants.
	Masquerading	It is a kind of impersonation attack where adversary may attempt to impersonate the identity of other nodes for communication and transaction processing [32].

4. Related Work

This section provides an overview of existing related work from the literature, proposed in the context of providing security to IoT. Recent work has shown how fog complements and extends cloud computing, emphasizing fog's relevance to IoT and Big Data space. Bonami et al. [33] presented a high-level description of fog software architecture and articulated the different technology components necessary to implement the fog paradigm. Hong et al. [34] proposed the concept of mobile fog, a high-level programming model to support latency-sensitive and large-scale IoT applications that are geospatially distributed. A plethora of research exists in the literature that focuses on the security and privacy issues in fog-based IoT infrastructures [9,12,21]. Alrawais et al. [18] proposed a fog-based mechanism that enhanced the security of IoT devices by distributing certificate revocation among IoT devices. In this scheme, a list of certificates is maintained by a certification authority which is available on a cloud and is updated at the fog layer periodically. However, most of the organizations may not be comfortable with the issuing of security certificates on a cloud due to a lack of trust on the cloud provider. The authors in [35] introduced a cloud service called Authentication Proxy as Service (APaaS) where mobile users are authenticated. The computation/storage resources are outsourced to the cloud in order to perform the expensive authentication process with low latency and limited power consumption. The work in [36] propounded a user authentication and management framework for cloud SaaS application that harnesses the stateless and secure nature of Jason Web Token (JWT) for client authentication and session management. Dsouza et al. [37] proposed a policy-driven security management approach for fog resources including policy analysis and its integration with fog paradigm. However, the approach does not consider the rights inheritance or propagation of rights in the ecosystem [9]. Stojmenovic et al. [21] presented a survey of fog applications and the associated security challenges. In particular, the authors analyzed man-in-the-middle attacks in fog computing where fog devices can be compromised or replaced by fake ones. This paper takes one step further to explore the design and implementation aspects of practical solutions to enhance the security and privacy of cloud-enabled IoT-based systems in the fog level. This will enable the fog paradigm to offer not only a balanced mix of enhanced computation power and connectivity but also provide an adequate level of security to counter cyber-attacks. Bamasag et al. [38] proposed a multicast authentication scheme for frequent message transmission in a short session or time interval. It is based on symmetric cryptography and uses Shamir's secret sharing technique where the secret key is used as an authenticator. However, its key storage cost can be too high for resource-constrained IoT devices. Salman et al. [39] presented an SDN-based identity-based authentication scheme for IoT. The different identity formats used by different communication protocols are translated in a shared identity-based on virtual IPv6 via the SDN controller. The gateways are responsible for authenticating *things* whereas the gateways are authenticated by the central controller. However, the proposed scheme has not been deployed and tested in an SDN-based environment. In addition, no performance analysis for memory and communication overhead was presented [40]. Porambage et al. [41] proposed a two-phase authentication mechanism to facilitate the IoT nodes and the end-users to authenticate each other and initiate a secure connection. However, the communication overhead and the computation cost, during the authentication and key generation phase, are high [42].

In summary, several techniques exist in the literature which focus on the requirement of providing secure communication to the resource-constrained IoT devices. Several existing solutions used cloud for security-related operations such as key generation, certification, and authentication. However, these approaches are not feasible when IoT devices are large in number, and faster response time is crucial to improve service level and increase safety. Moreover, a timely upload of security parameters to the cloud, generated by a huge number of devices, would need unfathomable measures of data transfer capacity. Some techniques require IoT devices to coordinate and cooperate for the exchange of security parameters and key generation related tasks. This puts extra overhead on the resource-constrained IoT devices in terms of memory and power. We, therefore, propose a mechanism that offloads the additional security-related operations to a more resourceful entity (such as a fog node) to reduce the latency and power consumption of resource-constrained IoT devices.

5. Proposed Methodology

This section presents the key features and architecture of the proposed FSS service for securing IoT at the fog layer. The purpose of this section is to discuss the functional and the physical elements of the proposed system and how they accomplish the operational concept of the system [43,44]. The services are deployed at the fog layer to reinforce security features to minimize the risk of mission-critical services being disrupted or taken down by security attacks. The FSS service comprises three main network security features: Authentication, confidentiality, and non-repudiation. To address these security features, we used public key and private key cryptography schemes. The public key cryptographic scheme is used in authentication—in the private key sharing process, specifically. The private key cryptographic scheme is used for communication to provide end-to-end security between IoT devices and the fog layer.

In the proposed mechanism, we assume that the input security parameters (params) are assigned to every IoT device, such as a unique identifier, username, and password. A sender is authenticated by the verifier at the fog layer through the provided params: Denoted by ID*rec*. Furthermore, we used asymmetric encryption for getting symmetric keys from the fog layer after node authentication. We used the Rivest-Shamir-Adleman (RSA) algorithm for the public key encryption. Our FSS service comprises a Verifier, a private-key-generator (PKG), and a hashing algorithm at the fog layer. The Verifier verifies the ID*rec* that comes from the sender for the authentication (e.g., email address, password, and device ID). Besides, the Verifier also maintains a table that consists of ID*rec* and nonce values. During the authentication, nonce values are used to prevent play-back/replay attacks. The PKG is used to generate private keys, which will be used for secure communication between the fog layer and the IoT layer. The steps are given below to elaborate on how the proposed approach works:

(1)　The device, which wants to communicate with the fog layer, provides security params denoted by ID*rec*, which can be any string such as an email address (e.g., abc@gmail.com), a unique identifier (e.g., 3211214687423), or password (e.g., ******).

　　a.　The IoT device that wants to communicate with the fog layer generates a small size secret-key *Ks* (e.g., 128 bit) for the encryption of ID*rec*. A nonce value is also added with that ID*rec*. Then, secret-key is used to encrypt the ID*rec* along with the provided nonce.

　　b.　Encrypt the already generated small size secret-key *Ks* by using the fog layer public key and then, combine both the cipher text (encrypted (ID*rec* + nonce) and encrypted (secret-key)), denoted as *C*.

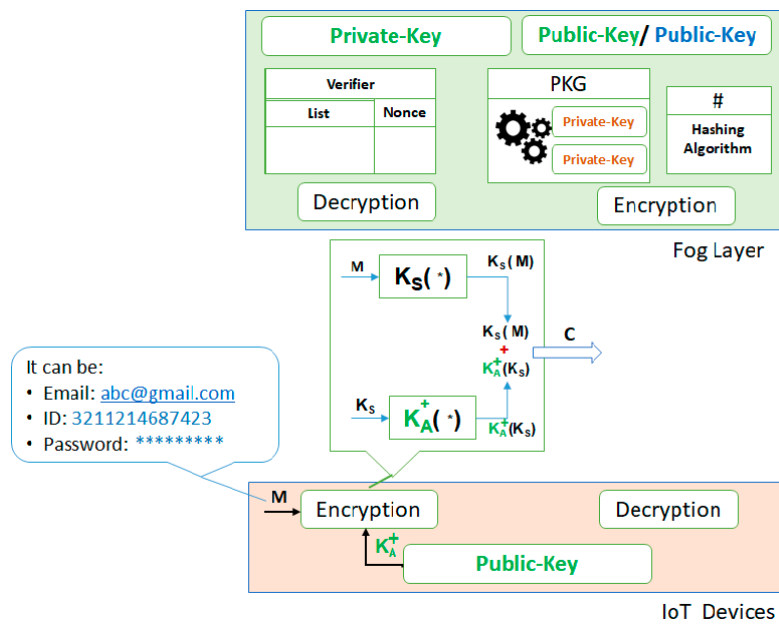　　c.　The cipher text *C* is forwarded to the fog layer for the authentication, as shown in Figure 3.

Figure 3. Cipher text containing both parameters (params) and the secret key.

(2)　Cipher text *C* is received at the fog layer, where a decryption algorithm (fog layer private key) is applied to get the original message, thereby separating both the information received in the cipher text *C* (i.e., ID*rec* + nonce and secret-key *Ks*).

    a.　Apply the fog layer private key on the encrypted secret key *Ks* to get the original secret key (*Ks*).

    b.　Apply the secret key *Ks* on the cipher text (ID*rec* + nonce) to get the original ID*rec* and nonce values, as shown in Figure 4.

    c.　The Verifier at the fog layer authenticates the received ID*rec* against the list of registered devices. If it is matched then the *Verifier* saves its nonce, as shown in Figure 5.

    d.　After authentication, the *Verifier* sends a request to the PKG to generate a private key for that authenticated IoT device.
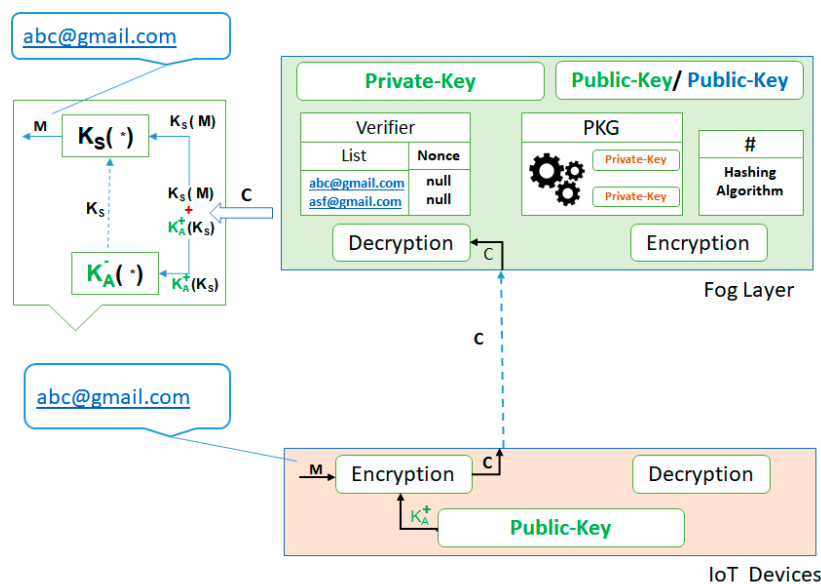

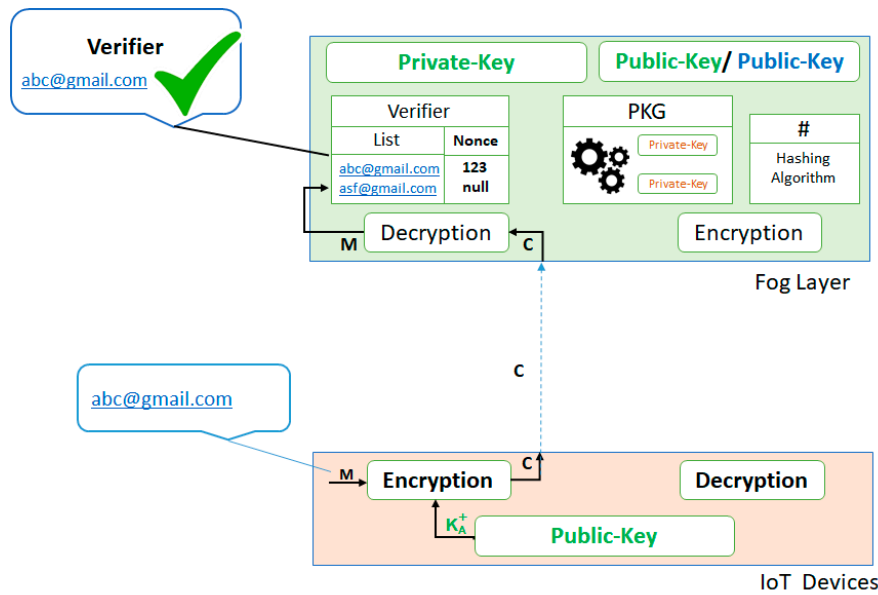**Figure 4.** Getting credentials for authentication using secret-key.

Figure 5. Verification of credentials.

(3) The PKG generates private keys for the authenticated devices and sends those keys to the encryption algorithm for encryption.

(4) The secret key *Ks*, which is received from the IoT device, is used to encrypt the PKG generated private key *PK*, as shown in Figure 6.

(5) In addition, for the non-repudiation requirement, the cipher text is further encrypted with the fog layer's private key, and sends this information back to the authenticated device, as shown in Figure 7. The device can then apply the fog layer's public key to ensure that the message is indeed received from the fog layer.
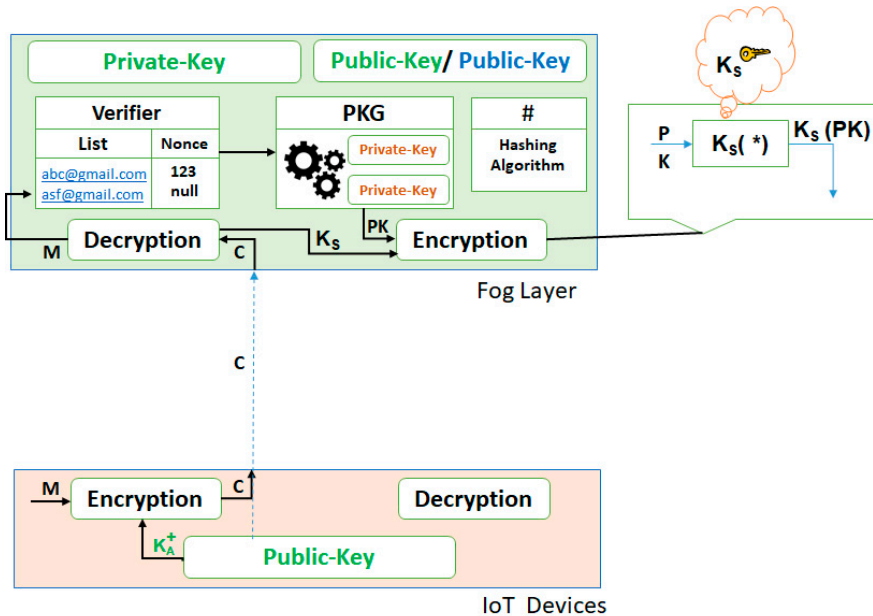

Figure 6. Encryption of the generated private-key with the secret-key.

(6) The authenticated IoT device then applies the decryption process to get the private key *PK* from the received cipher text and:

    a. Applies the secret key (*Ks*) on the encrypted message to get the private key *PK*.

(7) After successfully receiving the private key *PK* in a secure manner, authenticated IoT device can communicate with the fog layer through this private key *PK*.

(8) The private key *PK* is then used for the encryption and decryption on both sides to enable end-to-end security.
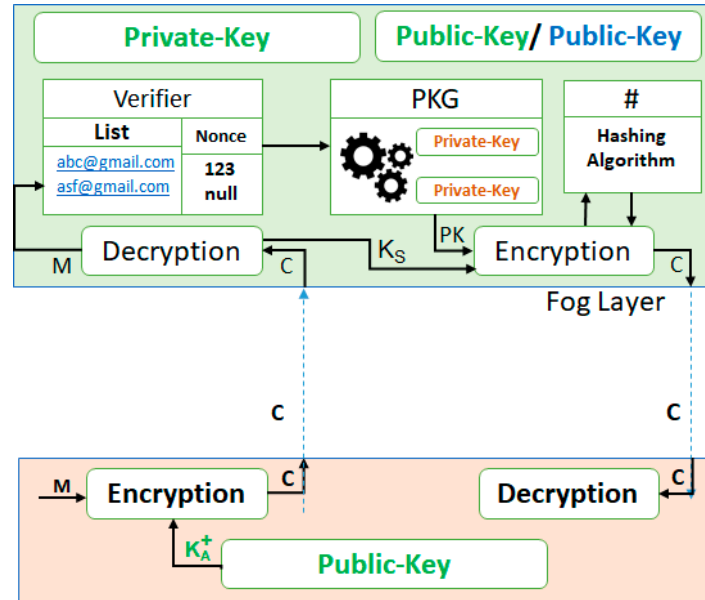

Figure 7. Secure communication using the private-key.

Algorithm 1 details how the authentication and private-key generation algorithm performs in FSS. **Procedure 1**— (Request Authentication) starts in lines 2 and 3 with initiating an IoT request (IoT_r) and Virtual Machine (V_m). In this request, IoT devices generate a small size secret-key (*esk*) for the encryption of the input security parameters. A nonce (*enonce*) value is also added to overcome the replay attacks. The request is then sent to a fog node in Fn, where *Fn = {f1, f2, f3,.., fn}* for authentication as per line 7. The encrypt function (lines 8–15) is responsible for encrypting the outgoing request using the generated *esk* and the public key of the fog server (S_{ip}). **Procedure 2**— (Authentication Response) works in reverse, as shown in (lines 20–31). The decryption is achieved by removing and reversing the elements of the request message, thus only the fog server with the right private key reads the message and archives *esk* correctly.

Algorithm 1: FSS—Authentication and Private-Key Generation Algorithm

---

Input: IoTRequest (IoT$_r$); VirtualMachine (V$_m$); ServerIP (S$_{ip}$)

Parameters: FogNode (f$_n$); EncryptedMsg (e$_m$); EncryptedSecretKey (e$_{sk}$); EncryptedNonce (e$_{nonce}$)

Initialization: IoT$_r$ = ϕ; V$_m$ = ϕ;

1. **Procedure 1:** *Request Authentication (Client side)* **by**

2. IoT_r = *setIoTRequest (username, password);*

3. e_{sk} = *generateSecretKey ( );*

4. e_{nonce} = *setNonceValue( );*

5. S_{ip} = *getServerIP ( );*

6. V_m = *getVirtualMachine ( );*

7. F_n ⟵ sendToFogNode (IoT$_r$, e$_{sk}$, V$_m$);

8. **Function** *Encrypt (* IoT$_r$, e$_{sk}$, V$_m$, S$_{ip}$ *)* **by**

9. e_{sk} ⟶ *encryptMsg(IoT$_r$);*

10. e_m = *initiateMsg (* IoT$_r$, e$_{sk}$, V$_m$, S$_{ip}$ *);*

11. **for** *each element* ∈ *e$_m$* **do**

12. e_m = *addByBytes (e$_m$);*

13. e_m = *shiftRows (e$_m$);*

14. e_m = *mixColumns (e$_m$);*

15. **end**

16. **End**

17. return *e$_m$;*

18. **End**

19. **Procedure 2:** *Response Authentication (Server side)* **by**

20. *get (e$_m$);*

21. **Function** *Decrypt (e$_m$)* **by**

22. e_m = *initiateMsg (IoT$_r$, e$_{sk}$, S$_{ip}$);*

23. **for** *each element* ∈ *e$_m$* **do**

24. e_m = *subByBytes (e$_m$);*

25. e_m = *reverseRows (e$_m$);*

26. e_m ⟶ *(IoT$_r$, e$_{sk}$, e$_{nonce}$);*

27. e_{sk} ⟶ *decryptMsg(IoT$_r$);*

28. IoT_r ⟶ *toAuthenticateIoT (username, password);*

29. Sk_{long} ⟶ *generateSecretKey( );*

30. **end**

31. **End**

32. return *e$_m$;*

33. **End**

---

6. Evaluation Results

Though a complete evaluation of the proposed security mechanism is underway, some preliminary results for the execution of the proposed "FSS" features will be presented in this section.

For implementing the FSS service, we used Visual Studio 2017. We first developed a real client-side environment, which consists of a *login page* and a *webmail page*. The login page comprises different functions (random secret key generation function, private key encryption and decryption function, random nonce generation function, and public key encryption and decryption function). We ran and tested our service on a small fog-based server (a laptop) and IoT devices (four mobile phones). By using this setup, we obtained benchmark values to be used in OPNET-based network simulator to evaluate our proposed method. Table 2 lists the benchmark values obtained during the experiment, such as the execution time of various key generation functions. Table 2 also provides the technical specification of all the devices used.

Table 2. Device specification and benchmark values.

	Name		Description			
1	Computer device		"HP core I3, Ram 6 GB, Hard drive 1TB, Operating System: windows 64 bit			
2	IoT devices		Samsung S4	Huawei P20 Lite	Samsung Galaxy J1	Apple iphone 4
3	Processor	CPU	Quad-core 1.6 GHz Cortex-A15 & quad-core 1.2 GHz Cortex-A7	Octa-core (4 × 2.36 GHz Cortex-A53 + 4 × 1.7 GHz Cortex-A53)	1.2 GHz Dual Core Cortex-A7	1 GHz Apple A4
		Chipset	Exynos 5 Octa 5410 Chipset	HiSilicon Kirin 659	Spreadtrum	Apple A4
		GPU	IT SGX544MP3	Mali-T830 MP2	Mali-400	PowerVR SGX535
		RAM	2 GB	4 GB	512 MB	512 MB
4	Distance between E2E		500 m	500 m	500 m	500 m
4	Connection type		4G LTE	4G LTE	3G	2G
5	"Size of the users login page"		87 KB	87 KB	87 KB	87 KB
6	Manually filling users required credentials		approx. 13 s	approx. 11 s	approx. 17 s	approx. 13 s
7	"Automated filling of credentials and sign in"		~2 s	~2 s	~2 s	~2 s
8	Client-side key generation time (auto)		~0.007 s	~0.0059 s	~0.0191 s	~0.013 s
9	Client-side message encryption (using secret key) time		~0.2 s	~0.17 s	~0.2 s	~0.246 s
10	Client-side secret key encryption (using fog server public key) time		~0.3 s	~0.2 s	~0.373 s	~0.3 s
11	Fog server-side verification time		~0.06 s	~0.06 s	~0.06 s	~0.06 s
12	Fog server PK generation time		~0.04 s	~0.04 s	~0.04 s	~0.04 s
13	Fog server PK encryption (using secret key) time		~0.2 s	~0.2 s	~0.2 s	~0.2 s
14	Fog server PK encryption (using fog private key) time		~0.3 s	~0.3 s	~0.3 s	~0.3 s
15	E2E response time for sharing PK		~2 s	~1.83 s	~3.39 s	~3.15 s
16	Total time for getting PK		~2.5 s	~2.19 s	~3.91 s	~4.083 s

As listed in Table 2, we used four mobile phones with different capacities and resources. We used a Samsung S4, Huawei P20lite, Samsung Galaxy J1 and Apple Iphone4 with medium, high, low, and normal processing resources, respectively, as shown in Figure 8. For the analysis of response

time, we considered different Internet connection speed: 4G, 3G, and 2G. For the Samsung S4 and Huawei P20lite, we used 4G, whereas for the Samsung Galaxy J1 and Apple Iphone4, we used 3G and 2G. We set 500 m (E2E) distance between the fog server and the devices. E2E processing time for sharing the PK between the fog server and the different devices was: ~2 s, ~1.83 s, ~3.39 s, and ~3.15 s. Correspondingly, the total response-time recoded for these devices i=was: ~2.5 s, ~2.19 s, ~3.91 s, and ~4.083 s. However, the response time may have been subject to oscillation based on the available network connection speed. In addition, we calculated the overall performance with regard to response-time based on a complete cycle from the client to the server and then back from the server to the client. The parameters we considered in this calculation were: Client-side key generation, client-side message encryption, client-side key encryption, server-side verification, server-side PK generation, server-side PK encryption using the secret key, and server-side PK encryption using the fog private key.
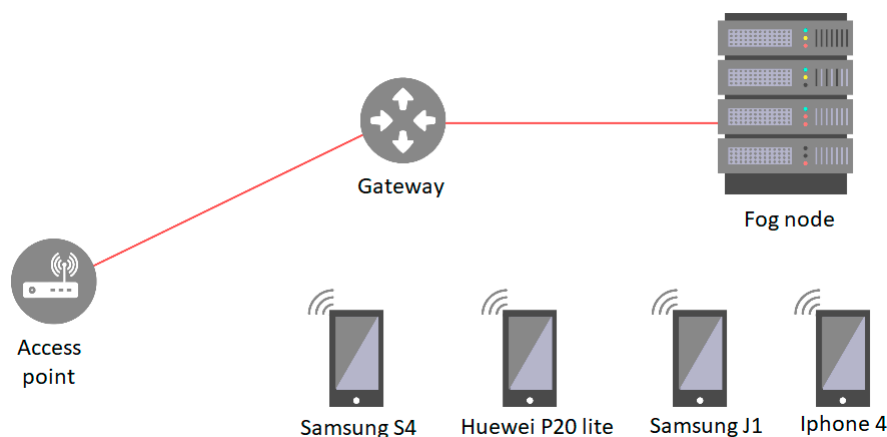

Figure 8. Real environment setup.

In the OPNET simulator, we considered the network topology shown in Figure 8. In this topology, we used a wireless network consisting of wireless nodes, a router, and a gateway. However, one fog server was used to measure the performance of the proposed service. The benchmark values that we obtained from the real environment were used in the simulator and applied to wireless nodes and the fog server.

After applying all benchmark values on the connected nodes and server, we ran and measured the performance of the proposed method with different traffic loads: High-load traffic (24 wired and four wireless devices, as shown in Figure 8), medium-load traffic (four wireless devices) and low-load traffic (one wireless device). The overall performance (in terms of processing time) of FSS in different traffic loads is shown in Figures 9 and 10.

Figure 9 elucidates the overall processing time taken by both the IoT devices and the fog server. The blue bars show the time taken by different IoT devices, whereas the orange bars represent the overall processing time of the fog server. The processing time of the IoT devices depends upon on the following key processes: The client-side key generation time, the client-side message encryption and decryption time using the secret key, the client-side secret key encryption, and decryption time using the fog server's public key, i.e., 1.007 s, 0.7459 s, 1.1652 s, and 1.105 s for the Samsung S4, Huawei P20lite, Samsung Galaxy J1, and Apple Iphone4, respectively. We could see that due to variations in hardware resources of IoT devices, such as CPU, chipset, GPU, and RAM (as shown in Table 2), the processing time varied accordingly. Furthermore, the fog server's processing time of ~1.1 s depended upon on certain parameters: The fog server-side verification time, the fog server PK generation time, the fog server PK encryption and decryption time using secret key, and the fog server PK encryption and decryption time using the fog private key. The fog server processing time was constant, as we used the same server for all the IoT devices.
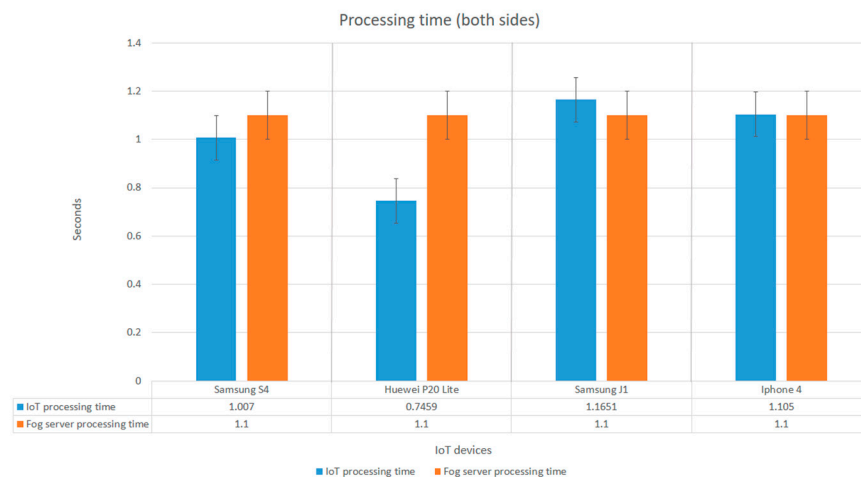
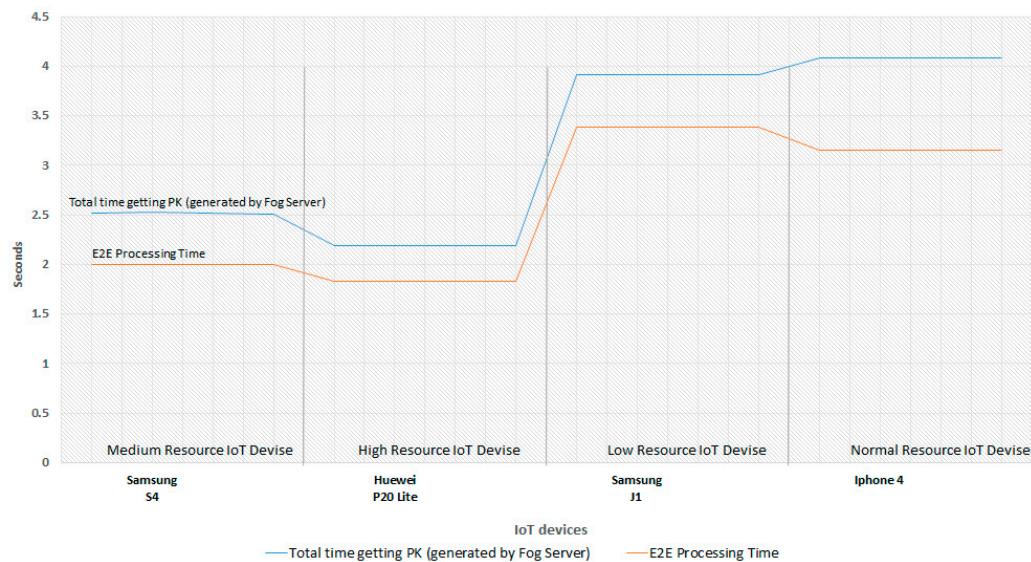**Figure 9.** Overall processing time taken by (Internet of Things) IoT devices and the fog server.

Figure 10. Overall performance in different traffic loads.

Figure 10 demonstrates the correlation between the total time for getting the PK generated by the fog server and E2E processing time in blue and orange lines, respectively. The blue line depicts the total time for getting the PK that starts when a request is sent to the fog server. The fog server node responds with the private key in return. The IoT device then applies the received PK for encryption. Despite that, the E2E time represents the time to access a webpage after a node is authenticated. As shown in Figure 10, E2E time varies for each IoT device relying on their processing capabilities. However, the total time for getting the PK from the fog server fluctuates due to two main reasons: The network connection type used, such as 2G, 3G, or 4G, and the processing capabilities of the IoT devices. For example, E2E time for the Huawei P20lite was the smallest with 1.83 s, but for the Apple Iphone4, it was 3.15 s. Correspondingly, total time for the said devices is 2.19 s and 4.083 s. Where the Huawei P20lite is ranked as a high resource IoT device with 4G network technology, while the Apple Iphone4 is ranked as normal resource IoT device with the 2G network connection type as described earlier. Similarly, if we compare the Samsung J1 (a low resource IoT device with 3G) with the Huawei P20lite (high resource IoT device with 4G), we can see a radical change with a difference of 1.56 s for E2E processing times and a difference of 1.72 s for the total time in getting PK generated by the fog server due to processing and different network connection types.

Figure 11 illustrates the performance comparison among FSS, Authentication Proxy as a Service (APaaS), and the legacy method on invoking a remote service (by a mobile user) to get authenticated.

We refer the readers to 39 for more details on the APaaS service and the legacy method. We used the same parameters for performance evaluation as that of the APaaS service, such as the response time obtained when invoking the authentication service with different alternatives: Cookies enabled, no cookies and no auto-filling, browser-based auto-filling, and E2E time access. The E2E time represents the time to access a webpage after a node is authentication. The benchmark values obtained from the FSS with different alternatives were then compared with the results of the APaaS and the legacy method. As shown in Figure 11, the grey, blue, and orange lines show the performance of the legacy method, APaaS, and FSS. The performance line illustrates the time taken for the authentication, which, without auto-fill, was 21, 15, and 13 seconds for the legacy method, APaaS, and the FSS, respectively. Likewise, time taken for the auto-fill was 13, 4, and 2 seconds. Similarly, the time taken with cookies was approximately 13, 4, and 2 seconds. Lastly, E2E access time for the three techniques was 13, 4, and 3.5 seconds, respectively. Thus, the proposed scheme performed better than the APaaS service and the legacy method in terms of response time when communicating with the fog layer.
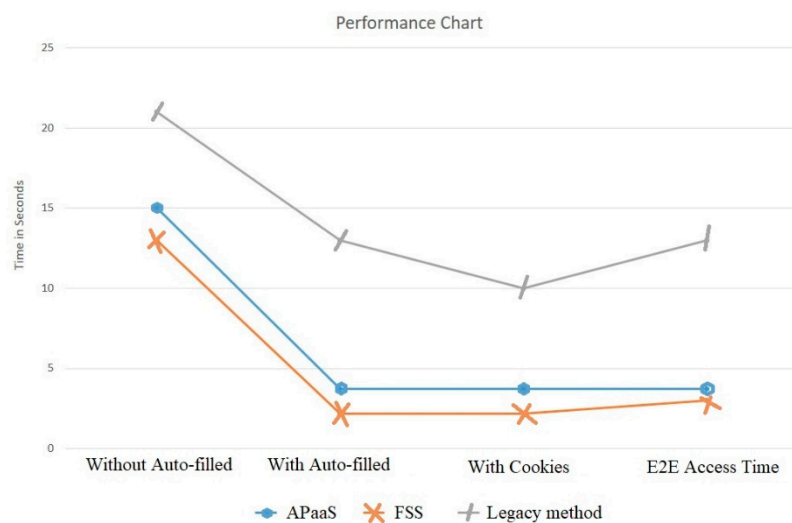

Figure 11. Fog Security Service (FSS) performance vs. Authentication Proxy as a Service (APaaS) and the legacy method.

7. Conclusions

The IoT-based applications are involved in harvesting a huge amount of personal and confidential information and are vulnerable to various security threats. Various cryptographic techniques exist that can effectively deal with different security attacks but are not suitable for resource-constrained IoT devices as they incur high consumption of resources. One way to tackle this issue is by offloading the additional security-related operations to a more resourceful paradigm such as a fog layer. In this paper, we demonstrated how a fog layer can reasonably be made secure to support IoT-based applications. We proposed a new Fog Security Service (FSS), which provides identity-based authentication, the integrity of data, and non-repudiation of connecting nodes via a private key generator (*PKG*) at the fog layer. FSS enhances end-to-end security between the IoT layer and the fog layer. Implementation and evaluation of the proposed FSS have been done, in the form of simulation using OPNET, to demonstrate the appropriateness of the proposed mechanism. First, the benchmark values were obtained in a real environment where IoT devices (with different capacities and resources) wirelessly connected to a fog node were considered. The performance was measured in terms of the response time, using various operations of the proposed method such as random secret key generation function, private key encryption and decryption function, random nonce generation function, and public key encryption and decryption function. These benchmark values were then used in the simulator and applied to nodes and the fog server. We ran and measured the performance of the proposed service with different

traffic loads: High-load traffic, medium-load traffic, and low-load traffic. The overall response time of FSS, in different traffic loads, was better when compared to the APaaS service and the legacy method.

References

1. Ni, J.; Zhang, K.; Lin, X.; Shen, X.S. Securing fog computing for internet of things applications: Challenges and solutions. *IEEE Commun. Surv. Tutor.* **2017**, *20*, 601–628. [CrossRef]
2. Abuarqoub, A.; Abusaimeh, H.; Hammoudeh, M.; Uliyan, D.; Abu-Hashem, M.A.; Murad, S.; Al-Jarrah, M.; Al-Fayez, F. A Survey on Internet of Thing Enabled Smart Campus Applications. In Proceedings of the International Conference on Future Networks and Distributed Systems, (ICFNDS '17), New York, NY, USA, 19–20 July 2017; pp. 1–7.
3. Baker, T.; Asim, M.; Tawfik, H.; Aldawsari, B.; Buyya, R. An energy-aware service composition algorithm for multiple cloud-based IoT applications. *J. Netw. Comput. Appl.* **2017**, *89*, 96–108. [CrossRef]
4. Camhi, J. Former Cisco CEO John Chambers Predicts 500 Billion Connected Devices by 2025. Business Insider, 2015. Available online: https://www.businessinsider.com/former-cisco-ceo-500-billion-connected-devices-by-2025-2015-11?r=US&IR=T&IR=T (accessed on 15 December 2018).
5. Federal Trade Commission. *Internet of Things: Privacy & Security in a Connected World*; Federal Trade Commission: Washington, DC, USA, 2015.
6. Yi, S.; Hao, Z.; Qin, Z.; Li, Q. Fog computing: Platform and applications. In Proceedings of the 2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb), Washington, DC, USA, 12–13 November 2015; pp. 73–78.
7. Mahmud, R.; Kotagiri, R.; Buyya, R. Fog computing: A taxonomy, survey and future directions. In *Internet of Everything*; Springer: Singapore, 2018; pp. 103–130.
8. Maamar, Z.; Baker, T.; Sellami, M.; Asim, M.; Ugljanin, E.; Faci, N. Cloud vs edge: Who serves the Internet-of-Things better? *Internet Technol. Lett.* **2018**, *1*, e66. [CrossRef]
9. Liang, K.; Zhao, L.; Chu, X.; Chen, H.H. An integrated architecture for software defined and virtualized radio access networks with fog computing. *IEEE Netw.* **2017**, *31*, 80–87. [CrossRef]
10. Agustin, J.P.C.; Jacinto, J.H.; Limjoco, W.J.R.; Pedrasa, I. IPv6 routing protocol for low-power and lossy networks implementation in network simulator—3. In Proceedings of the TENCON 2017—2017 IEEE Region 10 Conference, Penang, Malaysia, 5–8 November 2017; pp. 3129–3134.
11. Lee, K.; Kim, D.; Ha, D.; Rajput, U.; Oh, H. On security and privacy issues of fog computing supported Internet of Things environment. In Proceedings of the 6th International Conference on Network of the Future (NOF), Montreal, QC, Canada, 30 September–2 October 2015; pp. 1–3.
12. Yi, S.; Qin, Z.; Li, Q. Security and privacy issues of fog computing: A survey. In Proceedings of the International Conference on Wireless Algorithms, Systems, and Applications, Qufu, China, 10–12 August 2015; pp. 685–695.
13. Atwady, Y.; Hammoudeh, M. A survey on authentication techniques for the internet of things. In Proceedings of the International Conference on Future Networks and Distributed Systems, Cambridge, UK, 19–20 July 2017; p. 8.
14. Trappe, W.; Howard, R.; Moore, R.S. Low-energy security: Limits and opportunities in the internet of things. *IEEE Secur. Priv.* **2015**, *13*, 14–21. [CrossRef]
15. Fan, Q.; Ansari, N. Towards Workload Balancing in Fog Computing Empowered IoT. *IEEE Trans. Netw. Sci. Eng.* **2018**, *22*, 820–823. [CrossRef]

16. Yousefpour, A.; Ishigaki, G.; Gour, R.; Jue, J.P. On reducing iot service delay via fog offloading. *IEEE Internet Things J.* **2018**, *5*, 998–1010. [CrossRef]

17. Naha, R.K.; Garg, S.; Georgakopoulos, D.; Jayaraman, P.P.; Gao, L.; Xiang, Y.; Ranjan, R. Fog Computing: Survey of trends, architectures, requirements, and research directions. *IEEE Access* **2018**, *6*, 47980–48009. [CrossRef]

18. Alrawais, A.; Alhothaily, A.; Hu, C.; Cheng, X. Fog computing for the internet of things: Security and privacy issues. *IEEE Internet Comput.* **2017**, *21*, 34–42. [CrossRef]

19. Ammar, M.; Russello, G.; Crispo, B. Internet of Things: A survey on the security of IoT frameworks. *J. Inf. Secur. Appl.* **2018**, *38*, 8–27. [CrossRef]

20. Mahmood, Z. *Fog Computing: Concepts, Frameworks and Technologies*; Springer International Publishing: New York, NY, USA, 2018.

21. Stojmenovic, I.; Wen, S.; Huang, X.; Luan, H. An overview of fog computing and its security issues. *Concurr. Comput. Pract. Exp.* **2016**, *28*, 2991–3005. [CrossRef]

22. Khan, S.; Parkinson, S.; Qin, Y. Fog computing security: A review of current applications and security solutions. *J. Cloud Comput.* **2017**, *6*, 19. [CrossRef]

23. Mukherjee, M.; Matam, R.; Shu, L.; Maglaras, L.; Ferrag, M.A.; Choudhury, N.; Kumar, V. Security and privacy in fog computing: Challenges. *IEEE Access* **2017**, *5*, 19293–19304. [CrossRef]

24. Naher, N.; Haque, M.M. Authentication of Diffie-Hellman Protocol against Man-in-the-Middle Attack Using Cryptographically Secure CRC. In *Proceedings of the International Ethical Hacking Conference 2018*; Springer: Singapore, 2019; pp. 139–150.

25. Lu, Y.; Da Xu, L. Internet of Things (IoT) cybersecurity research: A review of current research topics. *IEEE Internet Things J.* **2018**. [CrossRef]

26. Pinkas, B.; Sander, T. Securing passwords against dictionary attacks. In Proceedings of the 9th ACM conference on Computer and communications security, Washington, DC, USA, 18–22 November 2002; pp. 161–170.

27. Sathiyaseelan, A.M.; Joseph, V.; Srinivasaraghavan, A. A proposed system for preventing session hijacking with modified one-time cookies. In Proceedings of the Big Data Analytics and Computational Intelligence (ICBDAC), Chirala, India, 23–25 March 2017; pp. 451–454.

28. Mejri, M.N.; Ben-Othman, J.; Hamdi, M. Survey on VANET security challenges and possible cryptographic solutions. *Veh. Commun.* **2014**, *1*, 53–66. [CrossRef]

29. Gupta, B.B.; Arachchilage, N.A.; Psannis, K.E. Defending against phishing attacks: Taxonomy of methods, current issues and future directions. *Telecommun. Syst.* **2018**, *67*, 247–267. [CrossRef]

30. Stojmenovic, I.; Wen, S. The fog computing paradigm: Scenarios and security issues. In Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS), Warsaw, Poland, 7–10 September 2014; pp. 1–8.

31. Du, R.; Zhao, C.; Li, S.; Li, J. A Strategy of Network Coding Against Wiretapping Attack Based on Network Segmentation. In *Proceedings of the Second International Conference on Communications, Signal Processing, and Systems*; Springer: Cham, Switzerland, 2014; pp. 1137–1144.

32. Hamed, T.; Ernst, J.B.; Kremer, S.C. A survey and taxonomy of classifiers of intrusion detection systems. In *Computer and Network Security Essentials*; Springer: Cham, Switzerland, 2018; pp. 21–39.

33. Bonomi, F.; Milito, R.; Natarajan, P.; Zhu, J. Fog computing: A platform for internet of things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*; Springer: Cham, Switzerland, 2014; pp. 169–186.

34. Hong, K.; Lillethun, D.; Ramachandran, U.; Ottenwälder, B.; Koldehofe, B. Mobile fog: A programming model for large-scale applications on the internet of things. In Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing, Hong Kong, China, 16 August 2013; pp. 15–20.

35. Abdo, J.B. Authentication proxy as a service. In Proceedings of the Second International Conference on Fog and Mobile Edge Computing (FMEC), Valencia, Spain, 8–11 May 2017; pp. 45–49.

36. Ethelbert, O.; Moghaddam, F.F.; Wieder, P.; Yahyapour, R. A JSON Token-Based Authentication and Access Management Schema for Cloud SaaS Applications. *arXiv*, 2017; arXiv:1710.08281.

37. Dsouza, C.; Ahn, G.J.; Taguinod, M. Policy-driven security management for fog computing: Preliminary framework and a case study. In Proceedings of the IEEE 15th International Conference on Information Reuse and Integration (IRI), Redwood City, CA, USA, 13–15 August 2014; pp. 16–23.

38. Bamasag, O.; Toumi, K.Y. Efficient multicast authentication in internet of things. In Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea, 19–21 October 2016; pp. 429–435.

39. Salman, O.; Abdallah, S.; Elhajj, I.H.; Chehab, A.; Kayssi, A. Identity-based authentication scheme for the internet of things. In Proceedings of the IEEE Symposium on Computers and Communication (ISCC), Messina, Italy, 27–30 June 2016; pp. 1109–1111.

40. Kalkan, K.; Zeadally, S. Securing internet of things (iot) with software defined networking (sdn). *IEEE Commun. Mag.* **2017**, *56*, 186–192. [CrossRef]

41. Porambage, P.; Schmitt, C.; Kumar, P.; Gurtov, A.; Ylianttila, M. Two-phase authentication protocol for wireless sensor networks in distributed IoT applications. In Proceedings of the Wireless Communications and Networking Conference (WCNC), Istanbul, Turkey, 6–9 April 2014; pp. 2728–2733.

42. Challa, S.; Wazid, M.; Das, A.K.; Kumar, N.; Reddy, A.G.; Yoon, E.J.; Yoo, K.Y. Secure signature-based authenticated key establishment scheme for future IoT applications. *IEEE Access* **2017**, *5*, 3028–3043. [CrossRef]

43. Raz, A.K.; Kenley, C.R.; DeLaurentis, D.A. System architecting and design space characterization. *Syst. Eng.* **2018**, *21*, 227–242. [CrossRef]

44. Parnell, G.S. (Ed.) *Trade-Off Analytics: Creating and Exploring the System Tradespace*; John Wiley & Sons: Hoboken, NJ, USA, 2016.