# A New Security Middleware Architecture Based on Fog Computing and Cloud To Support IoT Constrained Devices

Wissam Razouk
Kyushu University
Fukuoka, Japan
w.razouk@inf.kyushu-u.ac.jp

Daniele Sgandurra
Royal Holloway, University of London
United Kingdom
daniele.sgandurra@rhul.ac.uk

Kouichi Sakurai
Kyushu University
Fukuoka, Japan
sakurai@csce.kyushu-u.ac.jp

## ABSTRACT

The increase of sensitive data in the current Internet of Things (IoT) raises demands of computation, communication and storage capabilities. Indeed, thanks to RFID tags and wireless sensor networks, anything can be part of IoT. As a result, a large amount of data is generated, which is hard for many IoT devices to handle, as many IoT devices are resource-constrained and cannot use the existing standard security protocols. Cloud computing might seem like a convenient solution, since it offers on-demand access to a shared pool of resources such as processors, storage, applications and services. However this comes as a cost, as unnecessary communications not only burden the core network, but also the data center in the cloud. Therefore, considering suitable approaches such as fog computing and security middleware solutions is crucial.

In this paper, we propose a novel middleware architecture to solve the above issues, and discuss the generic concept of using fog computing along with cloud in order to achieve a higher security level. Our security middleware acts as a smart gateway as it is meant to pre-process data at the edge of the network. Depending on the received information, data might either be processed and stored locally on fog or sent to the cloud for further processing. Moreover, in our scheme, IoT constrained devices communicate through the proposed middleware, which provide access to more computing power and enhanced capability to perform secure communications. We discuss these concepts in detail, and explain how our proposal is effective to cope with some of the most relevant IoT security challenges.

## KEYWORDS

Internet of Things security, ressource-constrained devices, fog computing, cloud.

**ACM Reference Format:**
Wissam Razouk, Daniele Sgandurra, and Kouichi Sakurai. 2017. A New Security Middleware Architecture Based on Fog Computing and Cloud To Support IoT Constrained Devices. In *Proceedings of International Conference on Internet of Things and Machine Learning (IML 2017).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn


*IML 2017, LC, UK*

ACM ISBN 978-1-4503-5243-7…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

The design of large-scale IoT systems is challenging due to the large number of heterogeneous components involved and due to the complex iterations among devices introduced by cooperative and distributed approaches. In many cases, IoT implies that even the smallest device like a sensor or an RFID tag could be connected to the network. This high level of heterogeneity, coupled to the wide scale of IoT systems, is expected to magnify security threats of the current Internet, which is being increasingly used to let humans, machines, and things interact in any combination. Given that IoT security requirements include authentication, data confidentiality and access control, the enforcement of privacy and security policies are crucial, as privacy and trust among users and things are necessary in many cases. Unfortunately, traditional security countermeasures cannot be directly applied to IoT technologies due the limited computing power and the heterogeneous nature of such environment. Indeed, different standards and communication stacks are involved, and the high number of interconnected devices arises scalability issues. As a result, flexible security innovative models and design frameworks need to be created. This can be addressed by deploying a suitable middleware, which sits usually between things and applications to make a reliable platform. Middleware solutions act as a medium for communication among things with different interfaces, architectures and operating systems.
Currently, there are no standarized middleware solution that is suitable for resource-constrained IoT environments, since the available approaches are either customized for the conventional Internet or regular IoT devices. In this paper, we design a novel middleware architecture system for IoT environments. In our scheme we primarily target constrained devices such as low-cost RFID tags and wireless sensor networks. However, our approach can be extended to regular IoT devices.

Our proposed solution includes data confidentiality, authentication and access control mechanisms using the fog computing approach. Our middleware not only pre-process data locally, but also acts as smart gateway to enhance the utilization of network and cloud resources, which improves significantly the system's performances. Contributions of this paper can be summarized as follows:

- We start by an investigation of some of the most relevant IoT security challenges.
- Then, we discuss the difference between fog computing and cloud, and summarize the security benefits of using fog particularly in our case.
- Unlike previous protocols, our protocol relies on combining the benefits of fog computing and cloud to alleviate heavy

cryptographic operations on IoT resource-constrained devices. To the best of our knowledge, this is the first proposed IoT security middleware using fog in order to provide external support to IoT constrained devices.
- We propose a model based on usual technologies such as "the Constrained Application Protocol" (CoAP) and "the Representational State Transfer Application Programming Interface" (REST API) to allow easy implementation and application development.
- Many security weaknesses are already highlighted by the research community when it comes to the most commonly used IoT technologies such as ZigBee, ZWave, EnOcean, KNX, FS20, HomeMatic [6] [7] [3] [2] [5] [8]. For this reason we include a security module in our middleware, which is considered as an optional extra security layer. Therefore, the security option can be turned off in case the application is not sensitive and does not require a high security level.
- Our scheme is lightweight and specially designed to fit the requirements of IoT constrained devices. Thus, no resource-extensive cryptographic operations are needed. However, our approach can be extended to regular powerful IoT devices.

This paper is organized as follows. In Section 2, we present an overview of the related works. Section 3 is about IoT constrained devices security challenges. We discuss in Section 4 the main differences between fog computing and cloud. In Section 5, we explain the benefits of using Fog computing to enhance IoT systems from the security point of view. The proposed middleware security architecture is presented in section 6. Finally, we conclude with a summary of our contributions and future work in Section 8.

## 2 RELATED WORKS

Some IoT devices can support IPv6 communications, unfortunately this not the case for a whole range of IoT constrained devices which do not support the IP protocol within the local area scope. In such a case, gateways and middleware solutions are considered necessary [10] .

In [6], the authors analyze security related to the most commonly used IoT protocols namely ZigBee, EnOcean, ZWave, KNX, FS20, and HomeMatic. Then conclude that none of the technologies provide the necessary security measures needed in IoT environments. Indeed, many other papers in the literature point out their sever security weaknesses [7] [3] [2] [5].

Furthermore, in [4] the authors identify IoT security requirements and analyze the existing IoT middleware solutions in the literature. On December 1st, 2016, 213 related papers were identified. Out of these papers only 55 papers proposed an IoT middleware architecture. Moreover, only 19 out of these 55 papers tackled IoT security issues, and the rest of the papers had no published discussion or architecture for security. The authors go further than that and conclude that none of the remaining examined papers fulfilled all IoT security requirements.

This clearly shows a severe lack of suitable solutions. Therefore, much more research has to be done, and designing innovative IoT security middleware solutions is a pressing need.

This work aims to discuss the most relevant existing IoT security flaws, and propose a novel approach to fix them.

Although prior works have proposed several architectures to address specific issues, however, an IoT security middleware architecture that is based on fog computing and cloud that is able to adapt according to application requirements has not been studied. In this work, we propose a new approach that unifies IoT, fog computing, and cloud paradigms to enhance future IoT environments.

## 3 IOT CONSTRAINED DEVICES SECURITY CHALLENGES

TodayâĂŹs IoT security has many challenges and necessitate new essential changes to the existing security solutions. Indeed, most of the available security approaches are designed for regular internet and are geared toward protecting data centers, enterprise networks and consumer electronics. Moreover, IoT systems are typically made from several resource-constrained devices. Therefore, implementing common protection security solutions such as intrusion detection and malware signature scanning is unfeasible. And although some devices might have sufficient resources, implementing such solutions on a large number of devices might be challenging.

Some of these inconveniences can be fixed, thanks to the possibility of moving resource-intensive security functions from hosts to resource-rich cloud [11]. Unfortunately, this comes at a cost, as cloud based security services can induce significant delays for many systems and applications and require high bandwidth. We explain in this section why these methods are not suitable for IoT environments by discussing some of the core IoT constrained devices security challenges.

### 3.1 Many IoT devices are constrained and difficult to update

IoT devices can be geographically distributed, updating devices might turn out to be demanding and hard to manage. Usually brute-force solutions are the main techniques used as incident response to solve security issues, and systems are required in a lot of cases to be offline in order to replace the compromised files and have the system cleaned up. However, maximal responses such as shutting down a potentially compromised system, re-installing or rebooting its software, or replacing its components or subsystems is not suitable for several IoT systems as this can result in significant business loss and disruption as well. Indeed, unlike some of today's Internet devices such as smartphones, laptop computers, routers and switches, the security hardware and software in industrial manufacturing or control systems cannot always afford upgrading timely when necessary. For example, a nuclear reactor usually runs on 18 months cycles and cannot afford going offline, as this can cause the loss of tens of thousands of dollars [11].

Consequently, regular security systems that require each endpoint or network device to use its built in security mechanisms are not recommended for IoT environments. As an alternative, novel approaches such as Fog computing provide external servers to alleviate the hardware and software on IoT devices, in order to provide higher security level.

### 3.2 Using security firewalls for IoT devices is unpractical and infeasible

Many IoT devices such as sensors, wearable devices, vehicles, drones, etc. are placed in unprotected vulnerable physical environments. Consequently, accessing them through wired or wireless local network is a very feasible task. Moreover, these devices cannot easily implement standard existing security solutions, which rely primarily on firewalled castles and are aimed for perimeter based protection. Therefore, the system has to be placed behind firewalls to be secured, which usually provide intrusion detection and intrusion prevention. In general, these solutions has to be implemented in each one of the devices, and typically require resource intensive security functions in order to offer threat protection for individual hosts.

One example of such case is cars that consist of tens of microcomputers interconnected by several types of in-vehicle networks. Accessing the internal vehicle's network by eavesdropping and false data injection can be easily done through physically attaching low-cost and readily available tools such as dongles on the vehicle. Thus, using firewalled castles is technically not possible, as placing a firewall on every single microcomputer can prove to be unmanageable, complex and costly. As a result, this type of security approaches are not suitable for various IoT devices and systems.

### 3.3 Public key infrastructures are in a lot of cases not suitable for IoT environments

Remote attestation mechanisms have been typically used in order for a device to prove its trustworthiness to a remote verifier. In such case, a device uses its hardware-based root of trust or public key certificates to make claims about properties of its hardware, software or runtime environment. The verifier then validates cryptographically these claims. Unfortunately a large number of resource-constrained devices are not able to implement remote attestations algorithms and protocols because of their intensive processing requirements. Moreover, remote attestation methods are geared toward allowing an individual device to attest for its own trustworthiness. But requesting a large number of devices to execute remote attestation can prove to be challenging and complex in terms of management. Consequently, new techniques are needed to ensure the security of a large number of distributed devices and systems.

## 4 DISTINCTION BETWEEN FOG COMPUTING AND CLOUD

Cloud computing offers to other computers or devices an on-demand access to a shared pool of resources such as processors, storage, servers, networks, applications and services. Cloud computing is also capable of handling a huge amounts of data from IoT systems. But the transfer of massive data to and from cloud comes with many downsides and challenges, part of it is due to the limited bandwidth. Fog computing is a favorable solution to many Cloud of Things issues, as data can be processed near the data source using this approach.

It is important to note that fog complements cloud computing and does not substitute it. In fact, many cases require active cooperation between the "edge" and the "core". Thanks to this approach, the scope of processed data is narrow in space and time at the "edge" and is wide at the "core". In fact, the hierarchical organization of the networking, computing and storage resources is typical in many applications such a smart connected vehicles, connected rail and smart communities.

In meteorology, the word "fog" simply refers to a cloud that is close to the ground. In our case, it refers to extending the cloud to IoT devices in order to process information near the data source. Fog usually sits between the cloud and the underlying network, and it is meant to extend traditional cloud computing paradigm to the edge of the network, and therefore brings the advantages of cloud closer to data source. The fog node receive computation requests and sensed data from various IoT devices and can be implemented in different devices such as edge servers, smart routers, base stations and gateway devices. Thus, fog can be considered as a micro data center (MDC) located at the edge of the network to provide services for IoT systems. Fog computing reduces considerably the data volume that must be exchanged between end devices and cloud as well, as it allows data analytics and knowledge generation to occur at data source.

Fog computing and cloud computing can be differentiated from different perspectives:

– The proximity: Fog is located at the edge of the network, whereas cloud resides within Internet. Which makes fog much closer to the end user.

– The network efficiency: Using fog nodes frees up the core network bandwidth, which offers an enhanced overall network efficiency.

– The distance between client and server nodes: In general, many hops are required in order to connect to cloud, while it is usually possible to communicate with fog through a single hop.

– The latency: Thanks to the proximity of fog to the end devices as compared to the cloud, the latency of data transmission from IoT devices to the offloaded server is significantly reduced. Therefore, fog computing is a better choice for real time IoT applications compared to cloud.

– Mobility support: Unlike cloud which has a limited mobility support, fog computing can be a good option for such cases.

– Specialized content: Fog is more suitable for providing specialized content to IoT devices. As cloud is usually incapable of offering location-based customization of content, services and applications to devices.

## 5 A NEW APROACH FOR SECURING IOT USING FOG COMPUTING

### 5.1 External help for constrained devices

Instead of using expensive security mechanisms in terms of computation and storage requirements, fog computing reduces the security complexity on individual devices and replaces the IoT devices limited capabilities with off-board security services. Therefore, highly scalable, timely, resource-efficient and easy to manage external security services are provided while combining IoT with the fog computing approach. Moreover, as mentioned in previous sections, relying solely on remote attestation mechanisms for all IoT devices is not feasible, and pre-configuring standard security mechanisms on individual IoT devices is impractical. That is why fog systems

are implemented at the edge of the network, as they can assist in achieving transient connection among endpoints. For instance, security keys can be generated on fog to help authenticating IoT devices. For these reasons, fog computing can be considered as a suitable solution for many IoT challenges.

### 5.2 Protection of endpoints

Fog computing can be used as a firewall replacement to protect the network perimeter from attacks coming from outside the security perimeter. Indeed, as discussed in section 3.1, firewalls are not practical for many IoT systems and fog computing might be the only solution for such cases. In addition, other traditional security solutions such as malware detection and prevention can be moved to fog systems [11]. Signature based malware scanning as well as heuristic based malware detection mechanisms can be used thanks to the high storage capacities of Fog nodes and their abilities to communicate with the centralized cloud services. Therefore, any file that is considered suspicious by a certain endpoint can be sent to the Fog node for further analysis.

### 5.3 Proximity-based services

One of the main advantage of using the Fog approach is reducing the distance that information needs to traverse. Using proximity-based authentication challenges can strengthen identity verification and significantly reduce the chances of eavesdropping. Indeed, a lot of IoT devices lack the abilities to perform extensive cryptographic computations, so fog nodes can act as proxies to perform such operations, by providing additional computational resources that help achieve a higher security level within IoT environments.

## 6 OVERVIEW OF OUR SCHEME

Our architecture consists of four distinct components: IoT devices, the middleware, fog and cloud. Security related decisions are taken at various levels depending on their complexity. IoT devices collect data from the physical surroundings, and takes basic security decisions, while the middleware take more advanced decisions on whether data should be processed on fog or cloud . In section 3, we discussed unique IoT security challenges, we propose in this section a new security approach using the above mentioned technologies to help enhance and optimize future IoT performances.

### 6.1 The proposed middleware

– *The Security Module (SM):* Offers the following security properties:

- **Access control:** Ensures that only authorized entities are allowed to connect to the middleware, and denies unknown remote things from connecting to the system. Remote things need to be approved by an administrator or one of the users before they can communicate with other parties in the system. Therefore only authorized "things" can access certain resources or perform a given action such as accessing data or updating an IoT device software.
- **Authentication:** One of the most essential security requirements for IoT devices is authentication. Unfortunately, many IoT devices are resource-constrained in terms of memory and CPU power. Thus, executing the asymmetric cryptographic protocols required for the standard authentication solutions is not feasible. In our scheme, we propose a lightweight authentication protocol and outsource expensive computations and storage to fog trough the middleware.
- **Privacy and data protection:** Ideally, data must be preserved not only at the communication level, but also at the processing level. But information leakage in IoT environments, such as data location and usage, is still an open challenge and is currently attracting the attention of the research community. Indeed, the lack of resources on several types of IoT services limit significantly the techniques that can be used to provide efficient and effective privacy-preserving schemes, and makes IoT systems vulnerable to adversary attacks. In our scheme, we use a lightweight session key agreement to provide data protection. Moreover, data is pre-processed using our middleware, and is either sent to the fog or cloud for further processing and analyzing.
- **Security profiles:** The SM also contains security profiles allowing users or admins to define security disclosure policies. The security policies are used to define whether a given entity is authorized to access data related to "things" that are stored in the database module. Thus, a user can define to whom the information collected from the "things" is disclosed. Moreover, every user possesses a list of authorized entities in the database, and uses security profiles to create policies to define the security level. For example, if a secure channel is required to share the information collected from things with IoT applications or no.

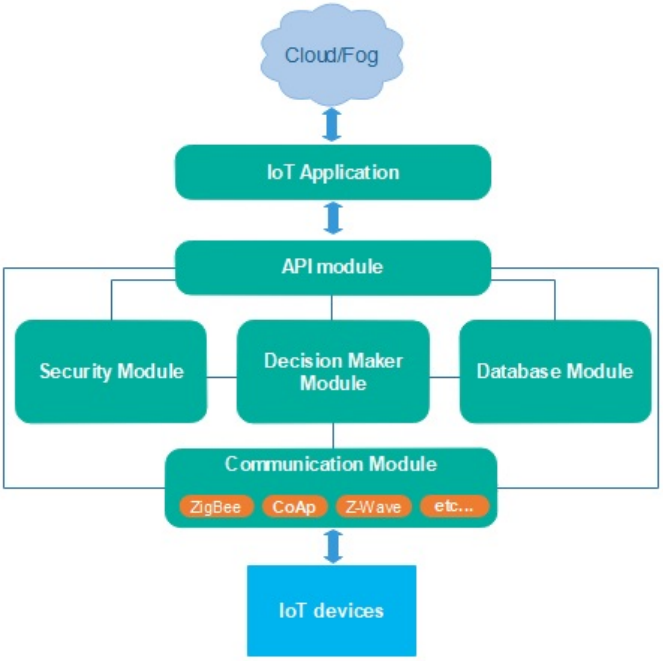


**Figure 1: Proposed generic security middleware architecture**

– *The communication module:* Is in charge of communicating with various IoT devices such as RFID tags and wireless sensors. We base our proposed model on usual technologies to allow easy

integration, development and transparent data exchange with different IoT entities. For this purpose, standard protocols like CoAP[9] and ZigBee[1] are used to comply with the products available on the market. These protocols are specially designed for constrained devises and are already widely deployed.
Unfortunately, as mentioned in section 2, these protocols are not secure and suffer from sever security weaknesses. For this reason, we provide in our scheme an extra optional layer of security to fix these issues. The security option can either be activated if the application is sensitive, or turned off if security is not a necessity.
– *The decision maker module (DMM):* This module has access to a database of authorized things using their IDs. This module not only manages the control policies, but also pre-process data to make decisions whether data should be processed locally at the edge of the network using fog, or send to the core network using cloud. This depends on how quickly data need to be processed. For example, extremely time-sensitive data is processed on the fog node closer to the things generating data, whereas big data analytics on historical data are less time-sensitive, and need the resources and the long term storage of the cloud.
– *The API module:* The application programing interface (API), offers an integrated and improved access interface for IoT applications. Not only it allows IoT applications communication, but also enables them to retrieve information from the database remotely over Internet. For this purpose, we choose the Representational State Transfer (REST) as it is platform and language agnostic. Indeed, REST supports various platforms (Windows, Unix, HTML, Android, iOS, etc.) and does not require the usage of a specific language. Therefore, the API module is compatible with a diverse range of IoT applications regardless of the platform or language they use.
– *The IoT applications:* Since the communication between the middleware and the application layer is not resource-constrained, standard security solutions such as SSL to send requests over HTTP.
– *Cloud and fog computing:* The cloud provides data storage and computing resources for IoT applications. It stores the "big data" of available things and users' profiles, etc. The history data is stored on cloud, whereas the most recent data is usually stored locally on a fog database to support real-time queries. This approach can be used for computing and making decisions based on facts in a quick and reliable way.

## 6.2 Physical Architecture

The proposed framework consists of four entities: the IoT devices, the middleware, fog and cloud. IoT devices can be composed of smart terminals, mobile phones, RFID tags, sensors, etc. The proposed middleware acts as a smart gateway and is meant to be implemented on routers or dedicated servers. It comprises many modules aimed for different tasks as shown in figure 1. The physical architecture of our scheme is shown in figure 2.

## 6.3 Layered Architecture

We present our proposed middleware layered architecture in figure 3.

- *The sensing layer:* In this layer, data is collected from physical environments. In addition, physical and virtual things are also managed and maintained according to different applications.
- *The preprocessing layer:* This layer's purpose is to deal with the collected data in order to analyze it and perform filtering and trimming so that more necessary and meaningful data is generated.
- *The temporary storage layer:* Data might be temporarily stored on the fog resources. Once data is not required to be stored locally anymore, it can be uploaded on the cloud and then removed from the storage media.
- *The security layer:* This layer comes into play when some private data is generated in IoT systems such as data related to patients in smart healthcare, or some location aware data that might be sensitive.
- *The transport layer:* In this layer, preprocessed and secure data is uploaded to the cloud. Therefore, burdening the core is reduced to the minimum allowing cloud to create other relevant services.

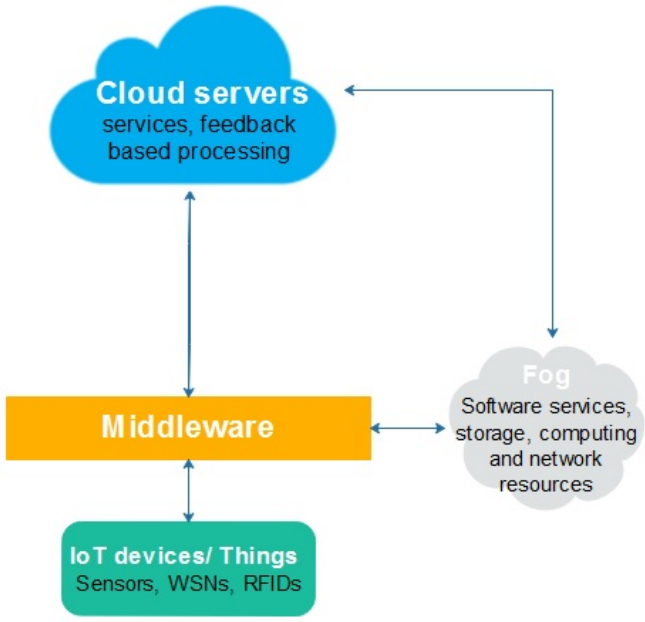


**Figure 2: The physical architecture of our proposal**

## 6.4 Usage scenario example

Four entities are involved in this scenario: the IoT device, the middleware, fog and cloud. Each IoT device stores its identifier $ID_D$ and its secret key $K_D$. The middleware has access to a database where information related to IoT devices are stored (in our case we are interested in the IDs and secret keys related to the IoT devices). The session key is generated by fog and used only one single time. The session key $K_S$ is a one-time-use key generated by fog and shared temporarily between both the initiator and responder during a given communication.
In our approach we use random numbers as a nonce to ensure the freshness of the messages containing the session keys. The nonce

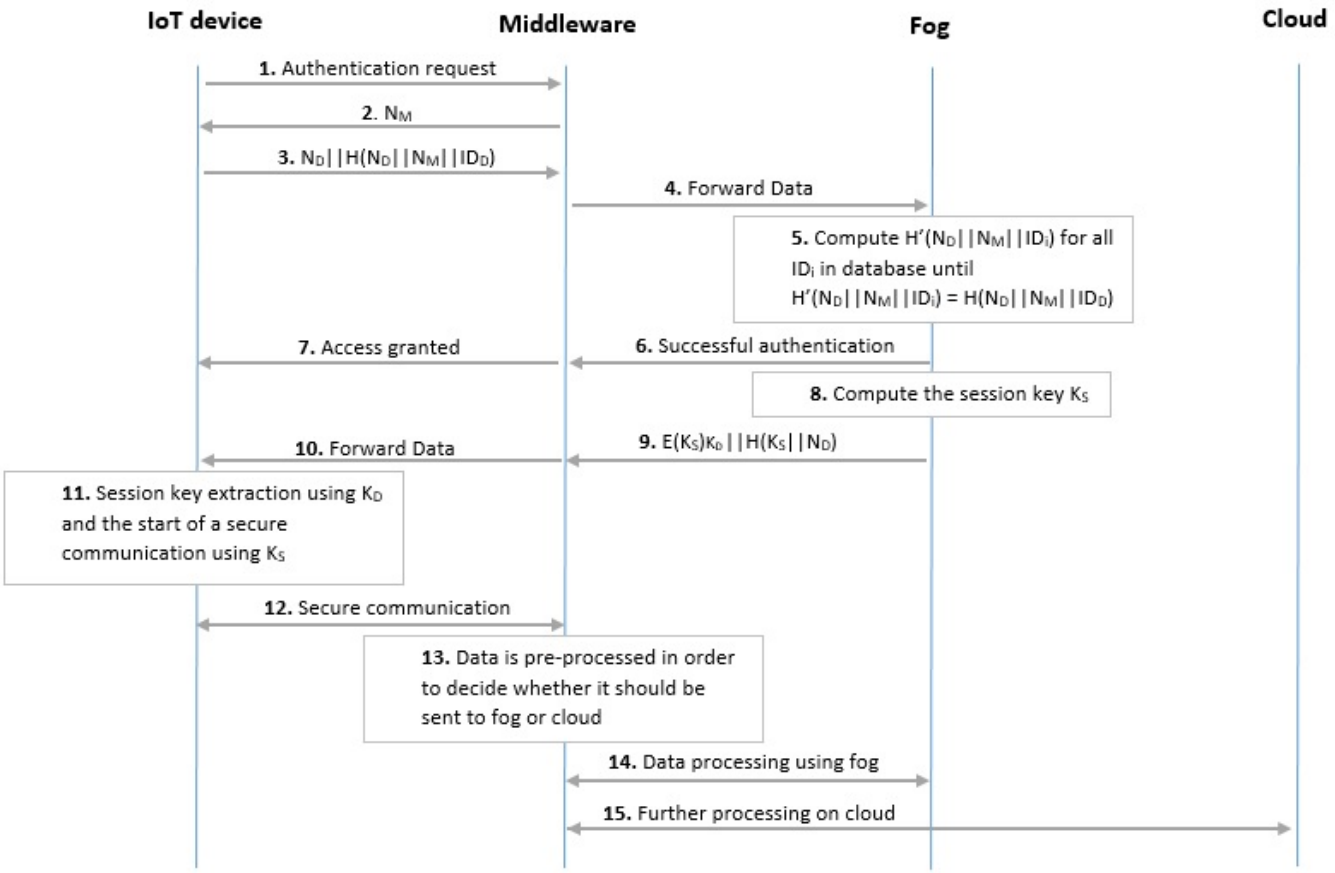


**Figure 4: Authentication and session key agreement**

is updated after each communication to prevent replay attacks and also to protect the IoT devices from traceability. We use the notations in table 1 to describe our solution throughout the paper, and provide in figure 4 an example scenario of a successful authentication and session key agreement to give insight regarding the way the information is processed in our proposed scheme.

- **Step 1.** The IoT device collects information from the physical or virtual environment, and send an authentication request to the middleware.

| Symbole | Description |
|---|---|
| $N_M$ | Random number sent from the middleware |
| $N_D$ | Random number sent from the IoT device |
| $H(X)$ | Hash function applied to X |
| $ID_D$ | ID related to the IoT device |
| $K_S$ | Session key |
| $K_D$ | The IoT device secret Key |
| $\|\|$ | Concatenation |

**Table 1: List of symbols description related to the authentication and key agreement scheme**

- **Step 2.** The middleware sends the random number $N_M$ as a challenge to the IoT device.
- **Step 3.** The IoT device sends $N_D||H(N_D||N_M||ID_D)$ which contains the following items:
  - A random number $N_D$ to protect the system from traceability. Thus, even if an attacker send the same message to the IoT device, the response is going to be different every time and therefore tracking the device by sending continuous requests is not possible. $N_D$ is also used later in Step 9 by the middleware to prevent replay attacks.
  - $N_M$ is included in the hash function to prevent from replay attacks.
  - $ID_D$ is used for authentication and also to retrieve the information related to the IoT device from the database.
- **Step 4&5.** Data is forwarded to fog in order to check if $ID_D$ exists in the database, which would mean that the IoT device is a legitimate device and is part of the system. For this, $H'(N_D||N_M||ID_i)$ is computed for all devices i in the database until H=H' is found (otherwise the authentication is not considered successful).

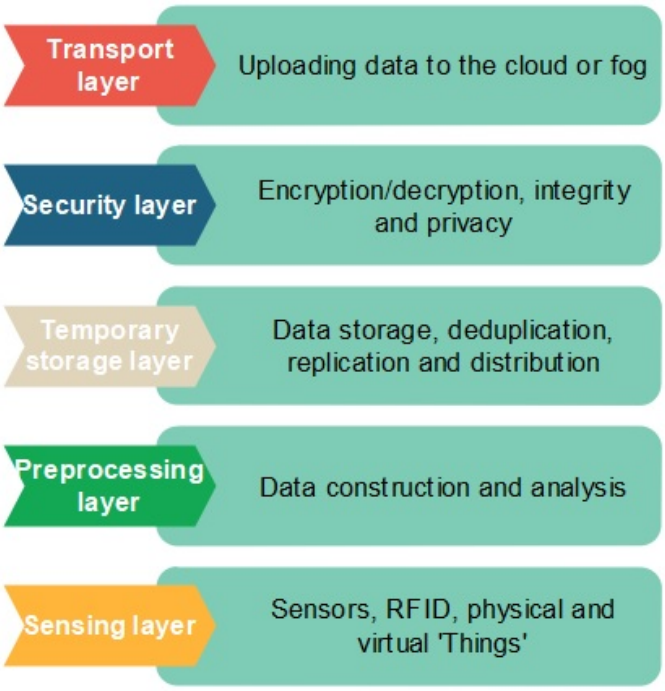


**Figure 3: Our proposed middleware layers**

- **Step 6&7.** The authentication is successful and the access is granted. The following steps are performed only if the security option is turned on in case of a sensitive application.
- **Step 8, 9&10.** The session key $K_S$ is generated on fog, and $E(K_S)_{K_D}||H(K_S||N_D)$ is computed. $K_S$ is encrypted using the IoT device secret key $K_D$. Whereas $N_D$ is included to prevent from replay attacks.
- **Step 11&12.** The session key $K_S$ is extracted, and secure data exchange starts.
- **Step 13.** Data is pre-processed using the decision maker module in order to decide whether it should be sent to fog or cloud.
- **Step 14.** Recent data are stored on fog to support real time queries.
- **Step 15.** History data and big data requiring storage and extensive computing resources are sent to the cloud.

## 7 SECURITY ASSESSMENT

We discuss in this section the security of our proposed scheme. The formal verification can be found in the extended version of this paper.

### 7.1 Data integrity checking

In our scheme we utilize hash functions to provide data integrity checking. This prevent data from being tampered during the transmission. Hash functions can convert an arbitrary length of data into a fixed length of data. Moreover, any change occurred during the transmission procedure will result in a change in the output of the fixed length data. Therefore, the receiver can compare hash values and verify if data has not been tempered. We also use hash data integrity checking in the authentication process to verify data access process. This method effectively prevent data from being tempered by an attacker, and the transmitted messages cannot be arbitrarily modified.

### 7.2 Forward secrecy

Forward security feature guarantees the security of past communications, even if the tag is compromised later. In our proposal, we use one time session keys to secure the communication. In fact, even if current messages are exposed, the one-time session keys used to secure exchanged information are all different and unknown, which prevents from inferring any secrets from previous sessions. Thus, our scheme reduces the chances of using previous sessions to compromise the communication.

### 7.3 Replay attacks protection

Our solution is designed to counter replay attacks. In each session different random numbers are included in the message exchanges to prevent this type of vulnerability. For example, an eavesdropper could try to impersonate the IoT device and replay one of its previous responses in step 3; however, the message would not be validated by the middleware, as the nonce $N_M$ included in the message would not be fresh. Thus, the message would not match the verification and the attack would fail. Therefore our approach resists replay attacks.

### 7.4 Impersonation attack protection

For example, an attacker can try to be authenticated as someone else, and gain access to restricted areas without being authorized to do so. Also, an expensive object can be disguised into a cheap one. In the proposed scheme, even if an adversary wants to deceive the middleware, and pretends to be a legal IoT device, the attack would not be successful, because $ID_D$ is never sent in clear and is therefore unknown to the attacker. As a result, the message in step 3 cannot be forged.

### 7.5 Traceability protection

Malicious traceability allows recognizing and tracing an object or a person in different times and places, and is one of the most difficult problems to solve. Several IoT applications are location based services, especially mobile computing applications. As a result, an adversary can infer the IoT device's location based on the communication patterns. Specially, if the IoT device sends identical responses when queried, which is the case of some IoT devices such as low-cost RFID tags. For instance, an attacker could identify important user's personal belongings in order to steal them, or track an important political person etc. For this reason, we include in each IoT device's responses, a fresh random number for each session, which provide protection against traceability as all responses are distinct for every communication.

### 7.6 Mutual Authentication

This feature is important for many applications. Indeed, the proposed protocol provides mutual authentication and only a legitimate middleware possessing the key $K_D$ can build a valid message in step

9. Similarly, only a genuine IoT device can build a valid message in step 3 as $ID_D$ is unknown from the attacker point of view as it is never sent in clear. Thus, only genuine nodes can derive the session key $K_S$. The exchanged messages involve a hash function that allows data integrity to be checked as well.

## 8 CONCLUSION

The existing traditional security approaches are not suitable to solve IoT challenges and require fundamental changes. This paper outlines some of the most relevant IoT security challenges, and discusses the benefits of using fog within IoT environments to provide a higher security level. In this work, we present a novel security architectural paradigm that harnesses the benefits of IoT, fog, and cloud. Our middleware mediates between the subsystems and the cloud and aims to cope with the highlighted security issues discussed in this paper.

In the future, we would like to implement and analyze it on actual test-beds and real world scenarios to test its feasibility, practicality and performance.

## ACKNOWLEDGMENT

This work has been supported by the Strategic International Research Cooperative Program - Japan Science and Technology Agency (JST). The authors would also like to thank Prof. Anupam Joshi for his valuable comments.

## REFERENCES

[1] ZigBee Alliance. 2007. ZigBee 2007 specification. *Online: http://www. zigbee. org/Specifications/ZigBee/Overview. aspx* 45 (2007), 120.
[2] Mark Devito and Johannes Johannes. 2016. A security assessment of Z-Wave devices and replay attack vulnerability. *The SANS Institute* (2016).
[3] Behrang Fouladi and Sahand Ghanoun. 2013. Security evaluation of the Z-Wave wireless protocol. *Black hat USA* 1 (2013), 1–6.
[4] Paul Fremantle and Philip Scott. 2017. A survey of secure middleware for the Internet of Things. *PeerJ Computer Science* 3 (2017), e114.
[5] Katherine Hoskins. 2016. Security Vulnerabilities in Z-Wave Home Automation Protocol. (2016).
[6] Karl Jonas, Bastian Vogl, and Michael Rademacher. 2017. Security Mechanisms of wireless Building Automation Systems. *Technical Report/Hochschule Bonn-Rhein-Sieg-University of Applied Sciences, Department of Computer Science* (2017).
[7] Wissam Razouk, Garth V Crosby, and Abderrahim Sekkaki. 2014. New security approach for zigbee weaknesses. *Procedia Computer Science* 37 (2014), 376–381.
[8] Wissam Razouk, Abderrahim Sekkaki, et al. 2013. A One Round-Trip Ultra-lightweight Security Protocol for Low-Cost RFID Tags. *Journal of Green Engineering* 3, 3 (2013), 261–272.
[9] Zach Shelby, Klaus Hartke, and Carsten Bormann. 2014. The constrained application protocol (CoAP). (2014).
[10] Sabrina Sicari, Alessandra Rizzardi, Luigi Alfredo Grieco, and Alberto Coen-Porisini. 2015. Security, privacy and trust in Internet of Things: The road ahead. *Computer Networks* 76 (2015), 146–164.
[11] Tao Zhang, Yi Zheng, Raymond Zheng, and Helder Antunes. 2017. Securing the Internet of Things. *Fog for 5G and IoT* (2017), 261–283.