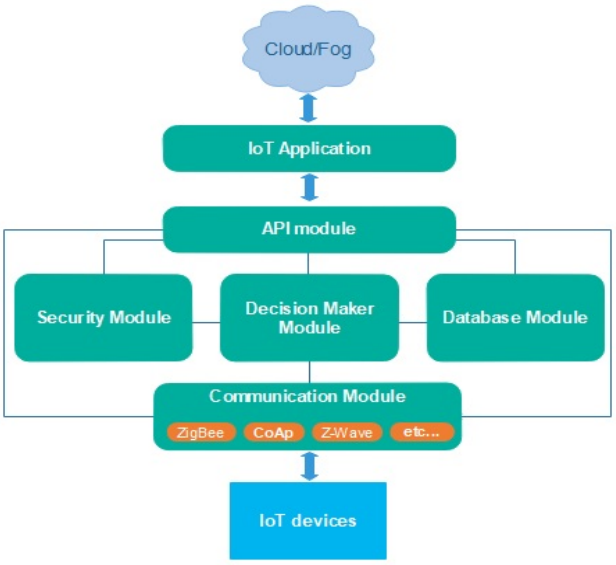

Fig. 1 Proposed generic security middleware architecture

nately, 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.

– *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 preprocess 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.

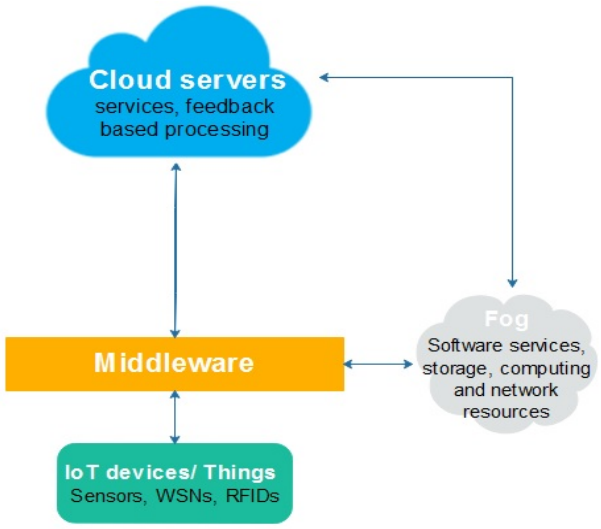

Fig. 2 The physical architecture of our proposal

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.

6.4 Usage scenario example

Four entities are involved in this scenario: the IoT device,

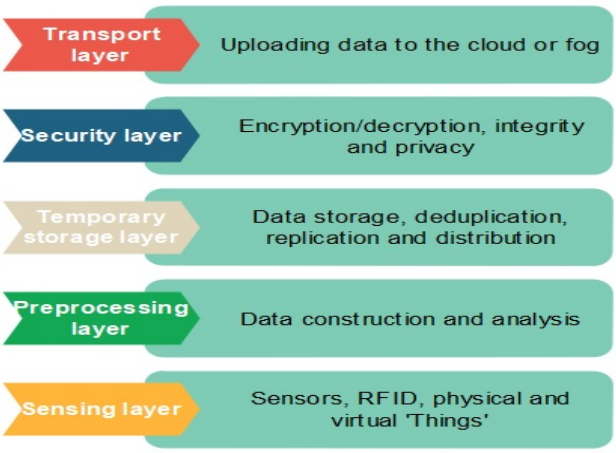

Fig. 3 Our proposed middleware layers

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 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.
- **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

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

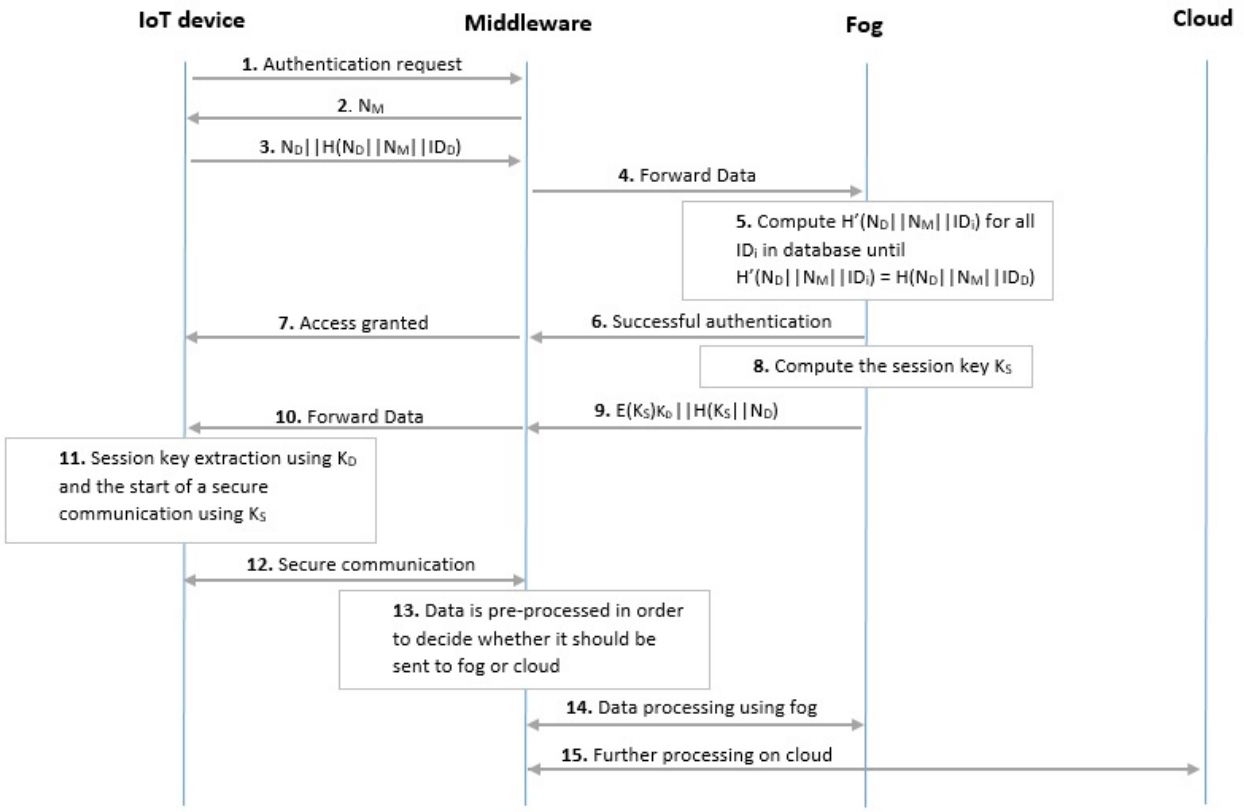

Fig. 4 Authentication and session key agreement

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).
- **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] Alliance, Z.: ZigBee 2007 specification, *Online: http://www. zigbee. org/Specifications/ZigBee/Overview. aspx*, Vol. 45, p. 120 (2007).

[2] Devito, M. and Johannes, J.: A security assessment of Z-Wave devices and replay attack vulnerability, *The SANS Institute* (2016).

[3] Fouladi, B. and Ghanoun, S.: Security evaluation of the Z-Wave wireless protocol, *Black hat USA*, Vol. 1, pp. 1–6 (2013).

[4] Fremantle, P. and Scott, P.: A survey of secure middleware for the Internet of Things, *PeerJ Computer Science*, Vol. 3, p. e114 (2017).

[5] Hoskins, K.: Security Vulnerabilities in Z-Wave Home Automation Protocol (2016).

[6] Jonas, K., Vogl, B. and Rademacher, M.: Security Mechanisms of wireless Building Automation Systems, *Technical Report/Hochschule Bonn-Rhein-Sieg-University of Applied Sciences, Department of Computer Science* (2017).

[7] Razouk, W., Crosby, G. V. and Sekkaki, A.: New security approach for zigbee weaknesses, *Procedia Computer Science*, Vol. 37, pp. 376–381 (2014).

[8] Razouk, W., Sekkaki, A. et al.: A One Round-Trip Ultra-lightweight Security Protocol for Low-Cost RFID Tags, *Journal of Green Engineering*, Vol. 3, No. 3, pp. 261–272 (2013).

[9] Shelby, Z., Hartke, K. and Bormann, C.: The constrained application protocol (CoAP) (2014).

[10] Sicari, S., Rizzardi, A., Grieco, L. A. and Coen-Porisini, A.: Security, privacy and trust in Internet of Things: The road ahead, *Computer Networks*, Vol. 76, pp. 146–164 (2015).

[11] Zhang, T., Zheng, Y., Zheng, R. and Antunes, H.: Securing the Internet of Things, *Fog for 5G and IoT*, pp. 261–283 (2017).