A User-centric Security Solution for Internet of Things and Edge Convergence

DEEPAK PUTHAL, Newcastle University, United Kingdom
LAURENCE T. YANG, St. Francis Xavier University, Canada
SCHAHRAM DUSTDAR, TU Wien, Austria
ZHENYU WEN, Newcastle University, United Kingdom
SONG JUN, Chinese University of Geosciences, China
AAD VAN MOORSEL and RAJIV RANJAN, Newcastle University, United Kingdom

The Internet of Things (IoT) is becoming a backbone of sensing infrastructure to several mission-critical applications such as smart health, disaster management, and smart cities. Due to resource-constrained sensing devices, IoT infrastructures use Edge datacenters (EDCs) for real-time data processing. EDCs can be either static or mobile in nature, and this article considers both of these scenarios. Generally, EDCs communicate with IoT devices in emergency scenarios to evaluate data in real-time. Protecting data communications from malicious activity becomes a key factor, as all the communication flows through insecure channels. In such infrastructures, it is a challenging task for EDCs to ensure the trustworthiness of the data for emergency evaluations. The current communication security pattern of "communication before authentication" leaves a "black hole" for intruders to become part of communication processes without authentication. To overcome this issue and to develop security infrastructures for IoT and distributed Edge datacenters, this article proposes a user-centric security solution. The proposed security solution shifts from a network-centric approach to a user-centric security approach by authenticating users and devices before communication is established. A trusted controller is initialized to authenticate and establishes the secure channel between the devices before they start communication between themselves. The centralized controller draws a perimeter for secure communications within the boundary. Theoretical analysis and experimental evaluation of the proposed security model show that it not only secures the communication infrastructure but also improves the overall network performance.

CCS Concepts: • **Security and privacy → Cryptography**; • **Networks → Network algorithms**;

Additional Key Words and Phrases: Internet of Things, distributed edge networks, perimeter-based security, authentication, secure channel

Author's address: D. Puthal, Z. Wen, A. V. Moorsel, and R. Ranjan, School of Computing, Newcastle University, Newcastle upon Tyne, United Kingdom, NE4 5TG; emails: {Deepak.Puthal, Zhenyu.Wen, aad.vanmoorsel}@newcastle.ac.uk, raj.ranjan@ncl.ac.uk; L. T. Yang, Department of Computer Science, St. Francis Xavier University, Antigonish, NS, B2G 2W5, Canada; email: ltyang@stfx.ca; S. Dustdar, TU Wien, Institute of Information Systems 184/1, Distributed Systems Group Argentinierstrasse 8, A-1040 Vienna, Austria; email: dustdar@dsg.tuwien.ac.at; S. Jun, School of Computer Science, China University of Geosciences, Wuhan 430074, P. R. China; email: songjun@cug.edu.cn.

2378-962X/2020/05-ART32 $15.00
https://doi.org/10.1145/3351882

ACM Reference format:
Deepak Puthal, Laurence T. Yang, Schahram Dustdar, Zhenyu Wen, Song Jun, Aad van Moorsel, and Rajiv Ranjan. 2020. A User-centric Security Solution for Internet of Things and Edge Convergence. *ACM Trans. Cyber-Phys. Syst.* 4, 3, Article 32 (May 2020), 19 pages.
https://doi.org/10.1145/3351882

1 INTRODUCTION

Distributed networks typically compose a network of geo-distributed connected devices with computation and communication capacity. Today, the Internet of Things (IoT) is gaining lots of interest in research, as its applications range from agriculture to battlefield applications [1]. Resource-aware devices are deployed in the sensing area for sensing and lightweight computing. The term "thing" in IoT infrastructure is not only considered in terms of sensors, but also sensing devices in automobiles, bio-chemical sensing devices, and heart-monitoring devices inside of a human body (wearable devices), to name a few. In fact, network devices are capable of collecting and transferring data across the network and are part of an IoT infrastructure.

There are several applications such as smart building management, environment control, e-health monitoring, and so on, where IoT plays a key role in collecting data in real-time for further processing (aggregation, analysis, and visualization) [2, 3]. In several applications, these low-power sensing devices generate critical data to be evaluated in near real-time. Since IoT devices (IoTD) are resource-constrained, i.e., limited computing power, memory, and storage, sensed data are often transmitted to high-power devices or datacenters for evaluation [4]. However, due to resource constraint features of IoTD, Edge computing increasingly plays a critical role. Edge datacenters (EDCs) are often deployed at the network edges (at base stations or vehicles) to process IoTD generated data streams in near real-time. As mentioned in References [3, 5], emerging IoT trends are set to completely change the way businesses, governments, and consumers interact with each other and transact in a data-driven economy.

As EDCs are deployed in unattended network edges, one important task is to build a secure infrastructure for data communications on an end-to-end basis [6], where IoT devices need to securely transmit critical information as well as sensitive information to EDCs for real-time evaluation. If we follow the traditional security approaches on communication networks to secure current infrastructure, then we would communicate first and then authenticate [7]. In this scenario an attacker gets a black hole to enter into the data transmission process before authentication happens. Diffie–Hellman key exchange is considered to be most secure and efficient initial handshaking for secure communications [8]. In the Diffie–Hellman key exchange protocol, both of the devices (i.e., sender and receiver or IoT device and EDC in our network scenario) initiate the handshaking without authenticating themselves [27, 28, 30]. Both sender and receiver also invest computational power to avoid the assumptions of an outside attacker [4].

A future-direction article (Reference [7]) discussed using the software defined perimeter (SDP) to overcome the current security flaws by authenticating first before communication starts. The concept of SDP was initially started by Cloud Security Alliance to securely access cloud resources [9]. We have adapted this concept to develop a novel authentication model for IoT and edge networks. The proposed model consists of three major components: IoTD, EDC, and SDP Controller (SC). The complete architecture and network model is shown in Figure 1. An SDP controller works as a centralized controller and is deployed with a secure module, i.e., a TPM (Trusted Platform Module). TPM is the trusted module of any hardware device that provides a cost-effective computation interface to the intruder. These were software-based encryption algorithms in earlier days achieved by keeping secret keys in the host's disc [4, 10], which provided hardware-based trust by

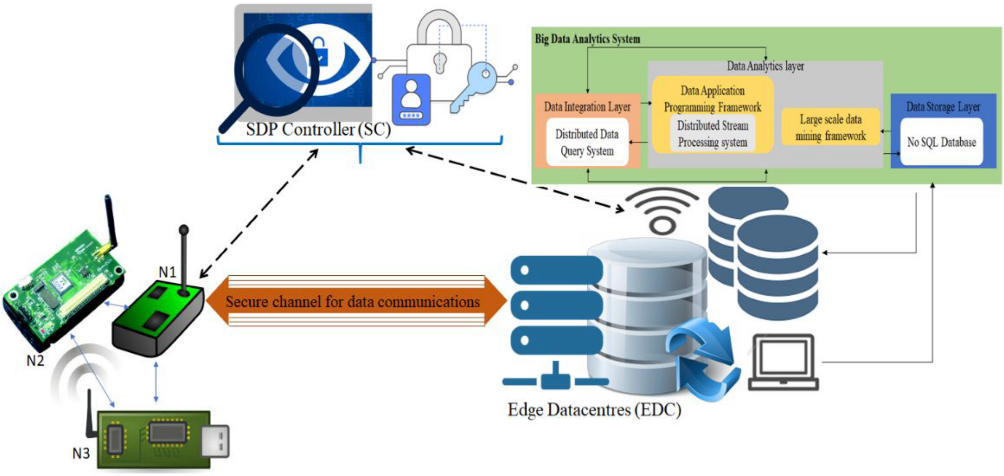

SDP Controller (SC)
Big Data Analytics System
Data Analytics layer
Data Integration Layer
Distributed Data Query System
Data Application Programming Framework
Distributed Stream Processing system
Large scale data mining framework
Data Storage Layer
No SQL Database
N1
N2
N3
Secure channel for data communications
Edge Datacentres (EDC)

Fig. 1. Network model of distributed EDCs and IoT.

keeping secret information such as network device ID, secret key, shared session keys, and so on. In our proposed model, network devices store their ID and associated secret key while initializing the network. According to the properties of the TPM, intruders cannot get the information from it [10].

The authentication process always starts with IoTD by contacting an SC. As the SC works as a fully trusted centralized controller, it authenticates the IoTD and EDC. If everything passes with authentication, then the SC creates a session for handshaking between IoTD and EDC, followed by establishing a secure channel for communications among themselves. This creates an environment that avoids intruders from the first step of communications. This concept is called perimeter-based security. The perimeter security level is determined by the individual user's requirements. According to the user's requirement, the SC initializes the secure channel establishment. The core features of perimeter-based end-to-end security, and the main article contributions, are:

- A novel end-to-end security model for IoT and edge networks.
- A perimeter-based security model avoiding attackers from the first step of communications.
- Theoretical and empirical evaluation of the proposed model to observe the security strength and performance in distributed networks.

The remainder of the article is as follows: Section 2 discusses related work and presents the problem analysis. Section 3 gives the system overview with a network model and adversary model for the proposed security model. Section 4 describes the proposed model with stepwise analysis. The theoretical and experimental evaluations are presented in Section 5 and Section 6, respectively. Section 7 concludes the article with future directions.

2 RELATED WORK AND PROBLEM ANALYSIS

This section presents the background studies of secure IoT infrastructures and analyzes the problems associated with current security solutions.

2.1 Related Work

The integration of IoT and Edge computing is gaining lots of interest, because it combines the technology of sensing, communications, and real-time data processing [11, 31]. Cyber security is a basic requirement for this computing infrastructure. The security issues of IoT and EDC are classified in References [11, 12], where researchers have classified the security threats and potential solutions with respect to individual layers. The related work of secure IoT and EDC are defined as follows:

Dorri et al. [5] outline the core components and functions of a smart home in the context of IoT. Each home is deployed with high-resource devices that are capable of always being online, termed as "miner," i.e., EDC in the context of our proposed model. This EDC handles all communication in or out to the home network. The EDC preserves a private and secure block chain used for communication auditing and controlling. A middleware architecture is designed for end-to-end security infrastructure of cloud-fog communications in Reference [13]. Irregular security adapts to unpredictable network connections, and adaptability is accomplished through security arrangements that are custom-fitted to application needs. Hatri et al. [14] analyzed and proved the major security flaw of the Authenticated Key Exchange (AKE) protocol that renders it insecure against an impersonation attack. Current IoT systems and devices affected by distributed denial-of-service attacks are discussed in Reference [29]. To address these issues, an optimized and scalable security solution is required for IoT and botnets [15]. Miettinen et al. [16] proposed a model named IoT SENTINEL, which is capable of automatically identifying those device types that are connected to IoT or Edge networks. This can also enable rules to protect the system against vulnerable devices entering into the networks. As a result, IoT SENTINEL minimizes damage resulting from being compromised. Subsequently, the authors presented the implementation of IoT SENTINEL protecting the user's network from deployed vulnerable IoT devices [17]. IOT SENTINEL works autonomously by identifying vulnerable devices automatically when they appear to the network or in the communication system. Furthermore, IOT SENTINEL initiates rules for traffic filtering to protect network devices for upcoming threats from vulnerable devices.

Software Defined Security (SDSec) introduces the biggest transformation by providing sustainable centralized security solutions transferring from hardware-based to software-based [18]. This helps to identify the threats and evaluate security without depending on hardware. Al-Ayyoub et al. [18] presented a novel experimental infrastructure for a virtualized real-time testbed environment for SDSec implementation. This experimental setup is built over a Mininet simulator, where the network components and devices are personalized to build an SDSec framework of experimental simulation. Furthermore, they introduced a novel experimental framework named SDDC by building a virtualized real-time testbed infrastructure for Software Defined Datacenter (SDD) systems [19]. Even though the proposed model is built on a Mininet simulator, the network components and the devices are customized to build the experimental setup for SDD.

From the related studies outlined above, it is concluded that there is lack of technology where users authenticate before communication, and also security solutions should be designed based on the network user. There is need for security solutions to draw perimeters dynamically according to the user's demand.

2.2 Problem Analysis

The Transmission Control Protocol/Internet Protocol (TCP/IP) is still being used and considered to be appropriate for communications involving both private and public networks (i.e., the Internet) [7]. IP security (IPsec) is one of the most prominent security protocols out of several solutions implemented over TCP/IP protocol. The IPsec protocol aims to protect network devices and the data flows between the devices. As a result, IPsec secures the network by focusing on the node and data packet authentication and applies encryption techniques over data packets on an end-to-end basis. Through data encryption, IPsec supports network-level data confidentiality, data integrity, and peer authentications [14]. However, TCP/IP model security frameworks are not efficient in providing a strong foundation for security, as these security frameworks allow network devices to start communication before they are authenticated by the handshaking process. As a result, intruders get huge space to be part of the communication system before authentication happens. This motivates us to focus on authentication of devices before a communication link is established.

To overcome this problem, SDP comes into the picture [7, 18], and devices are authenticated first before communication starts. SDP follows perimeter-based security by avoiding attackers becoming part of communications without authentication by drawing a perimeter. As a result, the perimeter stops attackers from being part of communications.

3 SYSTEM OVERVIEW

This section presents the assumptions regarding system design, network model, attack model, and preliminaries to the secure communications in IoT and distributed edges.

3.1 Assumptions

This article has taken some realistic assumptions while defining the security solution for IoT and distributed EDCs. According to the assumptions, IoT devices are registered with SDP controller (SC), i.e., the IoT devices at the time of deployment sense the environment to transmit data packets to a datacenter for further processing. Registration of IoT devices includes the identity (device ID) associated with the device secret key. EDCs follow the same procedure to register their identity with SCs. This sensitive information is stored in the secure module (i.e., TPM) of the SC, to protect from malicious user activities, where an SC is considered as fully trusted and no other devices have unauthorized access to the information from SCs. Both IoT devices and EDCs are deployed in the unattended environment, so it is considered as not to be fully trusted and subject to physical breaches. However, according to our system model, SCs are assumed to be fully trusted and monitor the network continuously. Other device compromises can be detected in a bounded time. Because of the properties of SC, we assume that the SC initializes the session key to establish secure channels between IoT devices and EDC.

3.2 Network Model

The network model for our proposed security framework is shown in Figure 1. It includes the IoT devices (i.e., sensors, mobile devices), SDP controllers, and edge datacenters. Figure 2 is the more realistic and generalized version of Figure 1. The IoT devices consist of end devices with wireless communication networks, whereas EDCs consist of both communication and computing facilities [15]. Generally, all the end devices transmit data to a cloud for further processing. However, all the stream data are evaluated in EDCs for emergency scenarios and real-time decision making. Hence, all these real-time evaluations are based on highly sensitive and emergency scenarios. By following Reference [7] and the CSA report [9], this article considered an SDP controller for establishing a secure channel between IoTD and EDC for further interruption from outside attackers. IoTD always initiates the process for EDC authentication and data communication requests. Then, an SC takes responsibility for secure channel establishment. All the devices such as IoTD, SC, and EDC are connected through wireless communication channels. An SC is considered to be a tamper-proof device, i.e., fully trusted, and stores secret information.

All the IoT devices are equipped with network interfaces, support short-range communications (IEEE 802.5.4), and an SC is equipped with Wi-Fi/GSM access for faster authentication, hence it enables high-data-transfer rates with nominal latency. All the devices are considered to be synchronized and can initiate the data communication process at any time [4].

The network performance is a key concept, and communication overhead plays a key role in network performance. The communication overhead is always computed as a percentage, as in

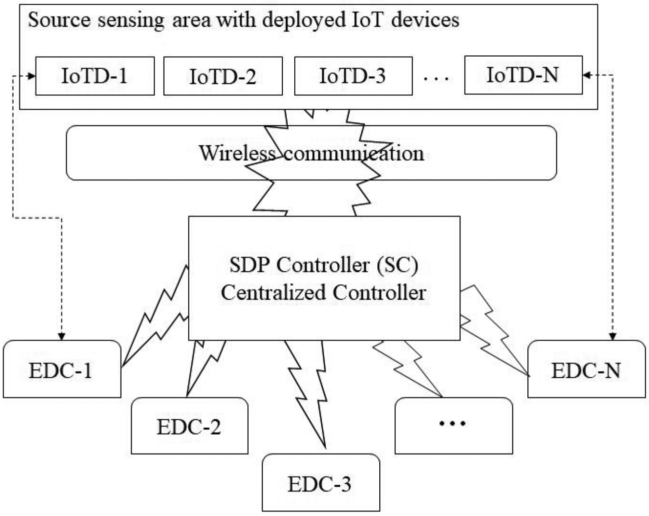


Fig. 2. Network model of distributed EDCs and IoT to develop security framework.

Equation (1) [20]. Communication overhead is computed by considering the number of communications (N_C) and the number of packets transmitted by its node (N_i).

$$CO\,(\%) = \left(\frac{N_C \times\ 74.125}{\sum_1^n N_i \times\ 30}\right) \times 100 \tag{1}$$

3.3 Threat Model

The types of attackers can be broadly divided into two types: internal attacker and external attacker. It is already proved in Reference [21] that external attackers are more powerful compared to internal attackers, because of the way they analyze the transmission medium. The external attackers also have high computational power to analyze data transmission in real-time. As external attackers are not part of the network model, it is impossible for them to decrypt the messages. But they can acquire relevant information during data transmission using computational powers. The external attacks might not be too successful in obtaining the session key to be part of secure data communication between IoTD and EDC, but they may possibly eavesdrop on network or traffic-related information to compromise the user's privacy. Even though external attackers are more powerful, they do not have access to the SC, which reduces the chances for them to get into the data transmission process. However, internal attackers are either compromised IoT devices in the source sensing area or compromised edge devices. Despite low resources, internal attackers' use of methods such as eavesdropping is effective, as they are part of the system and have information related to shared secrets and time to update the keys.

Here, we present potential threats and attacks in our network and data communication modes. An attacker can (a) introduce him/herself as the authenticated node, (b) drop or replay the data packets, (c) inject fake data packets into the streams, (d) impersonate a legitimate IoTD and EDC, (e) compromise the device associated with the network model, or (f) use distributed denial-of-service attack. These are some possible attacks in our model that can be performed individually or by a group of intruders. The proposed security model addresses all the above specified potential attacks.

3.4 Adversary Model

We assume that a large number of IoT devices are deployed in the source area for the monitoring purpose [22] where all of them are fully connected to communicate between themselves. The SC works as a centralized controller for all the IoT devices accessed through the SC to reach the EDC. According to the network model, SC is wary about the identity and associate secret keys of IoTD and EDCs. The SC is treated as fully trusted and protected in our model with the TPM. Attackers have several ways to get into data communications between IOTD and EDC:

- After the secure deployment of IoT devices, an attacker may capture the nodes and program them to behave according to the attacker's command. These types of attacks tend to attack on authentication, black hole, and DDoS.
- Attackers may capture the data packets and utilize resources to break the security key and encryptions. As a result, there is loss of confidentiality and data integrity.
- Attackers can capture the data packets and re-transmit them after some time to confuse the destination EDC during data analysis.

Wireless network nodes can be easily compromised to behave maliciously in the network to drop packets or tamper with data packets as they travel through insecure channels. From high level, this is the most common type of possible attack in wireless infrastructure. Our adversary model introduced many possible potential network attacks causing packet loss or security breaches. An attack on data confidentiality is a passive attack and the most dangerous attack, as it takes your confidential information. If EDC and associated IDS encounter packet loss, then that loss will be investigated and we assume authenticated IoTD responds with the actual packet information and packet sequence upon request from EDC. However, this article also relaxes the constraints and focus on the method to detect the packet loss and attacks on data packets.

4 PROPOSED SECURITY MODEL

The presented security solution has taken some realistic assumptions to define a perimeter-based framework for distributed networks as listed in the last section. There are three major types of devices—IoT devices, EDCs, and SCs—to communicate between themselves. SCs work as a master node to establish secure channels between IoT devices and EDC. IoT devices start the authentication process to establish secure channels with EDC for further communication. The notations used in the model descriptions are listed in Table 1.

4.1 System Setup

All IoT devices initialized their ID and secret key with the SDP Controller (SC). SCs can be considered as the fully trusted and centralized control to the secure infrastructure building. SCs know all the nodes' (i.e., IoT devices and EDCs) IDs and the associated secret key, whereas SCs initiate private public key pairs (K_{PR}, K_{PB}). Trusted Platform Module (TPM) can be used for secure key generation if the SC is not fully secure. All the network devices and EDCs use the public key of SC (K_{PB}) to communicate with SC. Subsequently, all the IoT devices store their identity (IoT-I/J) and secret key (K_{IoT}) at the TPM of the SC. TPM is defined as the secured module of any device, and it is already proved that no one can get the information from the TPM except the same device [4]. Finally, the SC selects the destination node's (IoTD or EDC) secret key for authentication and secure communication.

In an emergency scenario, IoT devices initiate the process to send data to the nearest EDC to evaluate data in real-time. To establish secure communication with the EDC, IoT devices initiate the communications request (RQ) with the SC. The initial *RQ* packet contains the timestamp (T)

Table 1. Notations Used in Model Descriptions

Acronym	Description
SC	SDP Controller
IoTD	IoT Device
EDC	Edge datacenter
ACK	Acknowledgement
NAK	Negative acknowledgement
PRN	Pseudorandom number
E / D	Encryption / Decryption
H ()	Hashing
$\mathrm{K_S}$	Session key
K_{PR}	SC private key
K_{PB}	SC public key
K_S	Session key
K_{IE}	Shared key between IOTD and EDC
K_{IoT}	Secret key of IoT device
CH	Challenge
RQ	Request
RS	Response
T/T'	Timestamp associate with the data packets

for maintaining the data freshness, source ID of the IoT device, and the destination EDC ID. All this information is combined and encrypted with the SC's public key (K_{PB}) to generate the secure packet ($E_{K_{PB}}(T||IoT - I||EDC - I)$) and send it to the SC. Subsequently, the SC uses its own private key (K_{PR}) to decrypt the data packets and check the authenticity of both IoTD and EDC. Initially, the SC checks the data freshness by evaluating the time between the packet generation time and received time ($\Delta T \leq T - T'$), where T is the packet generation time, T' is the packet received time, and ΔT is the estimated time to deliver request packets at the destination SC. Subsequently, the SC checks the data from the TPM to get the information about IoTD and EDC.

$$\begin{array}{l} D_{K_{PR}}(T||IoT - I||EDC - I) \\ \Delta\mathrm{T} \leq \mathrm{T} - \mathrm{T}' \\ \mathrm{COMPARE\ (IoT - I, IoT - I')} \\ \mathrm{COMPARE\ (EDC - I, EDC - I')}, \end{array}$$

where IoT-I/EDC-I are from IOT and IoT-I/EDC-I are from the TPM of the SC.

If the authentication matches, then the SC evaluates the rules associated with IoTD if both devices are authenticated nodes, otherwise it simply drops the packets to overcome a Distributed Denial of Service (DDoS) attack. The overall security model is depicted in Figure 3 and shows the stepwise key exchange establishing the secure channel between IoTD and EDC.

4.2 SDP Controller Evaluation

The integration of IoT and Edge computing is gaining lots of interest, because it combines the technology of sensing, communications, and real-time data processing [11, 31]. Cyber security has become a basic need for this computing infrastructure. The security issues of the IoT and EDC are classified in References [11, 12], where researchers have classified the security threats and potential solutions with respect to individual layers.

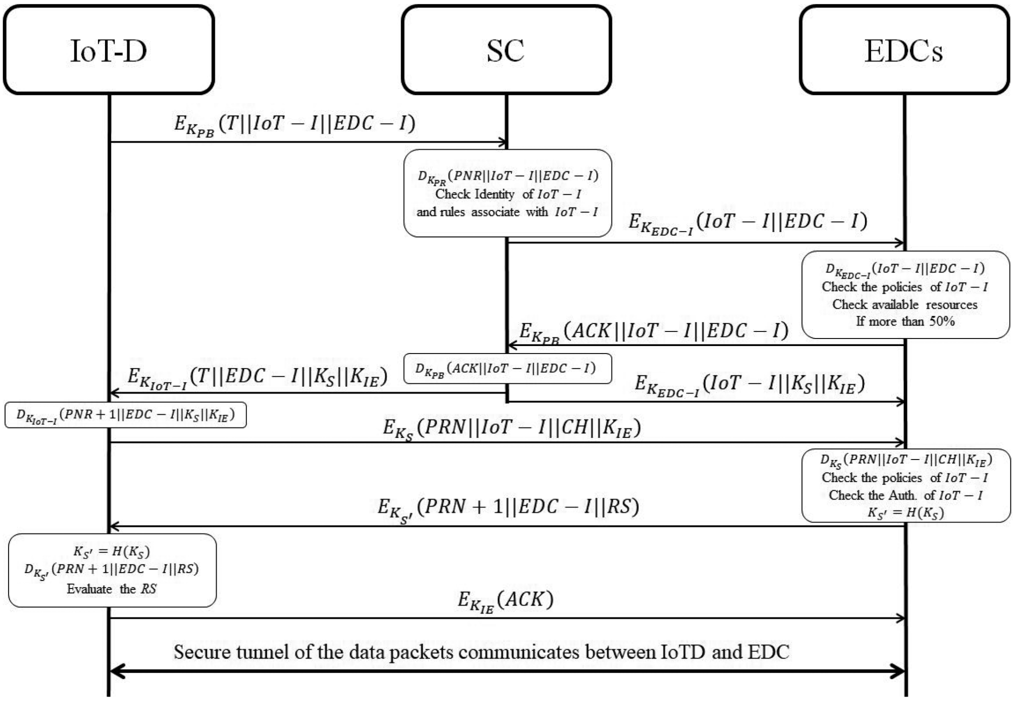

IoT-D
SC
EDCs
$E_{K_{PB}}(T||IoT-I||EDC-I)$
$D_{K_{PR}}(PNR||IoT-I||EDC-I)$
Check Identity of $IoT-I$
and rules associate with $IoT-I$
$E_{K_{EDC-I}}(IoT-I||EDC-I)$
$D_{K_{EDC-I}}(IoT-I||EDC-I)$
Check the policies of $IoT-I$
Check available resources
If more than 50%
$E_{K_{PB}}(ACK||IoT-I||EDC-I)$
$D_{K_{PB}}(ACK||IoT-I||EDC-I)$
$E_{K_{IoT-I}}(T||EDC-I||K_S||K_{IE})$
$E_{K_{EDC-I}}(IoT-I||K_S||K_{IE})$
$D_{K_{IoT-I}}(PNR+1||EDC-I||K_S||K_{IE})$
$E_{K_S}(PRN||IoT-I||CH||K_{IE})$
$D_{K_S}(PRN||IoT-I||CH||K_{IE})$
Check the policies of $IoT-I$
Check the Auth. of $IoT-I$
$K_{S'}=H(K_S)$
$E_{K_{S'}}(PRN+1||EDC-I||RS)$
$K_{S'}=H(K_S)$
$D_{K_{S'}}(PRN+1||EDC-I||RS)$
Evaluate the RS
$E_{K_{IE}}(ACK)$
Secure tunnel of the data packets communicates between IoTD and EDC

Fig. 3. Stepwise key exchange for the perimeter-based security approach.

If the source IoTD satisfies the rules of access of the recipient EDC, then the SC initializes the sending of the communication request information of IoTD and EDC. The SC generates the packet with source and destination ID and encrypts it with EDC's secret key ($E_{K_{EDC-I}}$) and sends it to the recipient EDC. The packet is of form: $E_{K_{EDC-I}}(IoT-I||EDC-I)$. The EDC decrypts the packet using its own secret key (K_{EDC-I}) to find the source IoT device ID (IoT-I). The EDC again checks the associated rules of IoT-I and validates whether EDC sources are eligible for access by the IoTD. If source IoTD satisfies the conditions to get access to EDC resources and current available resources of corresponding EDC is more than 50%, then the EDC considers further steps for secure channel establishment. We have considered the bottom line of the available resources as 50%, because EDC should have enough resources to process the emergency data streams. After all the successful verification, the EDC encrypts the acknowledgement (ACK) with the SC's public key, i.e., with format as $E_{K_{PB}}(ACK||IoT-I||EDC-I)$ and sends it to the SC; otherwise, it sends the NAK packet with the same format ($E_{K_{PB}}(NAK||IoT-I||EDC-I)$). Subsequently, the SC uses its own private key to check the response. The complete procedure to authenticate the IoTD and EDC by the SC is shown in Algorithm 1. After receiving NAK, the SC forwards it to IoTD to notify the unavailability of EDC for that device.

After receiving positive acknowledgement from the EDC, the SC generates two important keys, such as session key (K_S) to initialize the communication between IoTD and EDC, and shared key (K_{IE}) to create a secure communication channel between IoTD and EDC. After creation of these two keys, the SC generates a secure packet for IoTD and EDC to initialize them with the keys. The packet for IoTD looks like $E_{K_{IoT-I}}(T||EDC-I||K_S||K_{IE})$ and packet for EDC looks like $E_{K_{EDC-I}}(IoT-I||K_S||K_{IE})$, where both packets are encrypted with the destination's secret key. The importance of adding timestamps with individual packets is to maintain the data freshness,

ALGORITHM 1: IoTD and EDC authentication by SC

```
Assumptions:
    All the devices have a seed value
    IoTDs, EDCs stored their ID and secret key at the TPM of the SC.
Output:
    SC authenticates the IoTD and EDC for further communication between them.
Procedure:
    IoTD generates a REQ packet
        IoT → SC $\{E_{K_{PB}}\ (T||IoT-I||EDC-I)\}$
        SC evaluates the ID of IoT-I and EDC-I
    If (Authenticated)
        SC → EDC $\{E_{K_{EDC-I}}\ (IoT-I||EDC-I)\}$
        SC ← EDC $\{E_{K_{PB}}\ (ACK)\}$
        IoTD←SC$\{E_{K_{IoT-I}}\ (PNR+1||EDC-I||K_S||K_{IE})\}$
        EDC ← SC $\{E_{K_{EDC-I}}\ (IoT-I||K_S||K_{IE})\}$
    Else
        IoTD ← SC $\{NAK\}$
```

which avoids unnecessary computation for both IOTD and EDC.

$$\text{SC} \rightarrow \text{EDC}\ \{E_{K_{EDC-I}}\ (IoT-I\ ||K_S||\ K_{IE})\}$$

$$\text{SC} \rightarrow \text{IoTD}\ \{E_{K_{IoT-I}}\ (T\ ||EDC-I||\ K_S||K_{IE})\}$$

Both IoTD and EDC use their secret key (K_{IoTD-I} and K_{EDC-I}), respectively, to decrypt the data packets and extract the session key and shared key for further communication. After successful extraction, the IoTD initiates direct communication with the EDC to establish the session.

4.3 Secure Channel Initialization

The IoTD generates a pseudorandom number (PRN) using a very random source such as time of last sensed data, own ID (IoT-I), a challenge (CH) with initial seed value (SEED), and encrypts the session packet with the session key from SC (K_S). The generated packet is of the form $E_{K_S}(PRN||IoT-I||CH||K_{IE})$ and is sent to the EDC. Upon receiving the packet, the EDC uses the session key to decrypt the packet ($D_{K_S}(PRN||IoT-I||CH||K_{IE})$) and get the PRN and challenge. The EDC then solves the challenge (CH) using the seed value and prepares the response (RS). The EDC hashes the shared key $K_{S'} = H(K_s)$ and generates a new key to encrypt the response to IoTD. Finally, the packet is of the form $E_{K_{S'}}(PRN+1||EDC-I||RS)$, including the PRN value increased by 1. When the IoTD receives the packet, it first hashes the session key (K_s), i.e., $K_{S'} = H(K_s)$ and generates a new key to decrypt the incoming packet. After successful decryption ($D_{K_{S'}}(PRN+1||EDC-I||RS)$), check the EDC ID (EDC-I) if it matches, then EDC checks the PRN value and the response to validate the packet generator. If everything looks good, then the IoTD generates an ACK packet and encrypts with the shared secret key for EDC, i.e., $E_{K_{IE}}(ACK)\}$. This packet tends to be the beginning of secure communication between IoTD and EDC. The complete procedure to establish the secure channel between IoTD and EDC is shown in Algorithm 2.

Subsequently, the SC updates the secret key after a certain period, i.e., before there is the possibility of getting the shared key compromised. We took the interval of the rekey process from Reference [20]. After secret shared key generation (Say $K_{IE'}$), the SC uses both IOTD and EDC secret key to initialize them with the new shared secret key.

Now IoTD and EDC have a secure communication channel using shared key (K_{IE}) to transmit the data packets between themselves. The individual steps to establish the secure channel between IoTD and EDC are defined in Algorithm 3.

ALGORITHM 2: Secure channel between IoTD and EDC

```
Assumptions:
    IoTDs and EDCs received the session key and shared key from SC.
Output:
    Establish secure channel between IoTD and EDC
Procedure:
    IoTD generates a REQ packet
    IoT → EDC {$E_{K_S}(PRN||IoT-I||CH||K_{IE})$}
    EDC evaluates CH to generate RS and rules
    If (rules evaluated)
        IoTD ← EDC {$E_{K_{S'}}(PRN+1||EDC-I||RS)$}
        IoTD → EDC {$E_{K_{IE}}(ACK)$}
    Else
        DROP the packet.
```

ALGORITHM 3: Secure Channel Establishment Procedure

```
Description | A user-centric security solution to establish a secure communication channel between IoT
            | device and EDC for data communications without attacker interference.
Input       | SC's private public key pairs, IoTD and EDC registered with SC
Output      | Establish secure channel between IoTD and EDC
Step 1      | Initial setup
IoTD initiates the packet to establish secure communication with EDC
1.1 IoT → SC {$E_{K_{PB}}(T||IoT-I||EDC-I)$}
SC decrypts with $D_{K_{PR}}(T||IoT-I||EDC-I)$ and check
$\Delta T \leq T-T'$ and IoTD ID for authentication
Step 2      | SC Processing
After successful authentication of IoTD
2.1 SC → EDC {$E_{K_{EDC-I}}(IoT-I||EDC-I)$}
2.2 SC ← EDC {$E_{K_{PB}}(ACK)$}
2.3 IoTD ← SC {$E_{K_{IoT-I}}(PNR+1||EDC-I||K_S||K_{IE})$}
2.4 SC → EDC {$E_{K_{EDC-I}}(IoT-I||K_S||K_{IE})$}
Step 3      | Secure channel initialization
3.1 IoTD → EDC {$E_{K_S}(PRN||IoT-I||CH||K_{IE})$}
3.2 IoTD ← EDC {$E_{K_{S'}}(PRN+1||EDC-I||RS)$}
3.3 IoTD → EDC {$E_{K_{IE}}(ACK)$}
Now both IoTD and EDC use key ($K_{IE}$) for both encryption and decryption.
```

5 THEORETICAL ANALYSIS

This section gives theoretical validation of the proposed security method for IoT and distributed EDCs.

5.1 Security Evaluations

This article follows References [4, 20] in defining the attack definitions as follows: Based on the attack definitions and threat models from Section 3, this section proves theorems by analyzing potential threats and ways to protect communication networks.

Definition 1 (Attack on Integrity). An intruder *Mi* is capable of monitoring the data communication between IoTD and EDC and capable of modifying and/or accessing the data packets in transit.

Definition 2 (Attack on Confidentiality). An intruder *Mc* is capable of viewing and accessing the confidential data by reading data packets while in communication between IoTD and EDC.

Definition 3 (Replay Attack). An intruder *Mr* is an unauthorized device in the network, which is capable of intercepting data packets and forwarding them later to EDC.

Definition 4 (Attack on Authentication). A malicious attacker *Ma* can potentially attack on authenticity if it is proficient in network observation, intercepting, and subsequently introduces himself/herself as an authenticated node to be part of communication to EDC.

Theorem 1. *An attacker Ma cannot establish a secure channel with EDC by introducing himself/herself as an authenticated node.*

Proof. In the proposed method, SC has been considered as a fully trusted device with TPM. All the network devices *are* registered with SC by storing their ID and secret keys at TPM of SC. If any unauthenticated devices come into the picture, then SC can directly drop the request packet ($E_{K_{PB}}(T||IoT-I||EDC-I)$). Further, the SC decrypts REQ packets using its own private key (i.e., $D_{K_{PR}}$) to find source identity ($IoT-I$) and check in TPM. If a match is not found, then the SC drops the REQ packet.

More importantly, EDC accepts request only through the SC policy checking and approval. As a result, the attacker Ma is going to *fail* from the first step of authentication. □

Theorem 2. *An intruder Mc cannot access or view the data communications between IoTD and EDC.*

Proof. The session key (K_S) and shared key (K_{IE}) in our method is initialized by the SC and sent to IoTD and EDC by encrypting using their secret keys. Subsequently, IoTD initializes the packet using session key ($E_{K_S}(PRN||IoT-I||CH||K_{IE})$). EDC responds to IoTD by generating a new session key, i.e., $K_{S'}$. The new key is the hash of the previous key ($K_{S'} \leftarrow H(K_S)$). The session key is unknown to the network except IoTD and EDC, so it is hard for *Mi* to get the session key to read the conversion.

After the secure channel is established, IOTD and EDC use the shared key (K_{IE}) for data communications where key K_{IE} is used for strong encryption. By following Reference [20], it confirms that intruder Mc cannot get the data in less than two months' time using the most advanced processor.

Hence, it is proved that data cannot be viewed by Mc, as it does not have the shared secret key to see them. □

Theorem 3. *The intruder Mi cannot break the secure communications initialized by SC to capture the data packets.*

Proof. From Step 1 of Algorithm 1, we conclude that an intruder cannot be authenticated to establish the unauthorized communication with EDC. Consequently, the intruder Mi only has the chance to get access while initial handshaking between IoTD or during data transmission.

Step 2 of Algorithm 1 confirms that intruder Mi cannot get the key to read the data packets. This means that Mi cannot make the logical changes in the data packets. Random changes by applying a brute force method are not helpful, because each packet is associated with a message authentication code (MAC). So, it is impossible for Mi to modify the contents of the packets. □

Theorem 4. *During initial authentication, IoTD can easily identify the reply packets initialized by a replay attacker (Mr) to break the secure communication between IoTD and EDC.*

Proof. The replay attack is also popularly known as a playback attack, where attacker (Mr) intercepts the data packets and forwards them later to the recipient EDC. To avoid this kind of attack, initialized packets include a timestamp ($E_{K_{PB}}(T||IoT-I||EDC-I)$) or $E_{K_{IoT-I}}(T||EDC-I||K_S||K_{IE})$. The initial authentication process follows Algorithm 1. The value T is defined as the packet generation time and T' is the packet received time at destination, which are compared at the destination to detect the playback attack (T'–T ≤ ΔT). If the value of ΔT exceeds some threshold, then it is confirmed that the received packet is replayed by an intruder.

From Algorithm 3, we found that it is hard for intruder Mr to get the encryption key and edit data packets. So, Mr does not have control to modify the value of T; hence, it is proved that the proposed method avoids replay attack. □

Theorem 5. *The proposed method works with low complexity and communication overhead.*

Proof. Today, Diffie–Hellman key exchange is considered to be the most secure and efficient initial handshaking [8]. The Diffie–Hellman method uses lots of computation, i.e., power functions, modular functions and so on, whereas the proposed user-centric perimeter-based security method does not give any responsibility to the IoTD. The SC computes everything in the secure module with less computation, as we are not using public channel for key initialization. The authentication process and security evaluation happen in the secure module (i.e., TPM). Figure 5 shows the efficiency of the proposed model compared with Diffie–Hellman key exchange due to the low complexity. By reducing computational overhead, the proposed security model improved the system scalability by extending IoT device life time. Also, the proposed method not only reduced the complexity overhead but also the communication overhead. □

5.2 Forward Secrecy

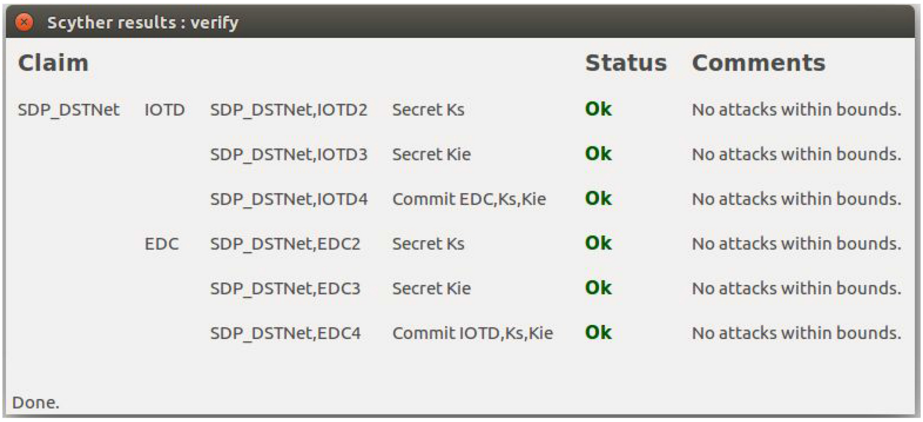
The SC works as a centralized controller and takes responsibility to establish the secure channel to data transfer between the IoTD and EDC. Once the SC authenticates the IoTD and EDC, it generates a session for secure handshaking between IoTD and EDC and secret shared key for secure communication between themselves. The shared key updates after a fixed amount of time (i.e., for time *t* by following Reference [20]). As a result, previous secret shared keys are useless to an adversary, as they are already updated by the SC after a fixed period of time.

6 EXPERIMENT AND EVALUATIONS

To evaluate the network performance and the security strength, the experiment is done in both real-time testbed and simulation environment. The simulation experiment is run in the in-house computational environment on an Intel (R) Core (TM) i5-6300 CPU @ 2.40 GHz 2.50 GHz CPU and 8 GB RAM running on Linux environment (UBUNTU 13.04). First, the security verification is evaluated in a real-time Scyther environment [23]. Second, the IoT device performance and network performance is evaluated after applying proposed security method in COOJA simulator in Contiki OS [24]. Finally, the proposed security solution's compatibility in real-time network scenarios is evaluated in real-time testbed deployment.

6.1 Security Verification

The proposed user-centric security model is experimented with in the simulation environment, Scyther, where Scyther is written in Security Protocol Description Language (.spdl). Scyther aims at evaluating the correctness of the security protocol in an automatic simulation tool by security threat analysis. By following Scyther tool's properties, we defined the roles of IoTD and EDC, as the simulation environment needs the two devices to establish the communication. IoTD and EDC are initialized with the session key and shared key information before running the experiment. In this experiment, IoTD sends the encrypted data packets to EDC for further processing. The EDC also prepares the responses for IoTD after data evaluation happens. Three types of attack



Scyther results : verify

Claim				Status	Comments
SDP_DSTNet	IOTD	SDP_DSTNet,IOTD2	Secret Ks	Ok	No attacks within bounds.
		SDP_DSTNet,IOTD3	Secret Kie	Ok	No attacks within bounds.
		SDP_DSTNet,IOTD4	Commit EDC,Ks,Kie	Ok	No attacks within bounds.
	EDC	SDP_DSTNet,EDC2	Secret Ks	Ok	No attacks within bounds.
		SDP_DSTNet,EDC3	Secret Kie	Ok	No attacks within bounds.
		SDP_DSTNet,EDC4	Commit IOTD,Ks,Kie	Ok	No attacks within bounds.

Done.

Fig. 4. Scyther page for successful security verification.

have been initialized with the proposed security model. First, an adversary changes the network packet while it is in public networks. In the second, an attacker tries to get the information from IoTD to pretend to be an authenticated node and transmit packets to the EDC. Finally, an intruder gets the network packets to analyze and tries to get the information and replay the packets. The experiment is run 100 times with the 10 instance intervals.

Results: As defined above, the simulation run ranges from 0 to 100 instances in 10 instance intervals for network data packet analysis. The data packet confidentiality and integrity is evaluated after successful authentication. As the key initialization process is initialized by the SC, we assume that none of the adversaries have the shared session and secret key information. We are using two different keys, i.e., K_S and K_{IE}, for the packet encryption process, where the key K_S is used for initial handshaking and key K_{IE} for secure communications. While evaluating the proposed security solution, we did not come across any potential network threats between IoTD and EDC to compromise the shared secret key, which concludes that the proposed solution is secure against confidentiality attack and integrity attack. The Scyther simulation result of security verification of the proposed model is shown as in Figure 4.

6.2 Network Performance

The network performance of the proposed security model is evaluated in a COOJA simulator in Contiki OS (i.e., sensor power consumptions and communication overhead) [25]. Two common types of sensor (i.e., Z1 and TmoteSky sensors) have been considered to evaluate network performance and node performance with proposed security solution.

The sensors are low-power WSN modules, which deploy to design and develop the platform for general purpose monitoring of daily life activities. These sensors are designed for maximum backwards compatibility with improved performance. These are flexible with computational and energy power, which are sustainable in terms of rough combination of power supplies, sensors, and connectors. Finally, these sensors support the operating systems that are popular in WSN and IoT communities, such as Contiki [24]. Contiki provides an excellent simulation platform named COOJA, which is compatible with real-time sensor features to evaluate the sensor node and network performance. Z1 sensor is always considered as the best sensor to get the network performance. This is developed with low-power microcontroller MSP430F2617 and structures a powerful 16-bit RISC CPU @16 MHz clock speed, built-in clock factory calibration, 8 KB RAM, and a 92 KB Flash memory. TmoteSky is an ultra-low-power sensor and is developed with the low-power

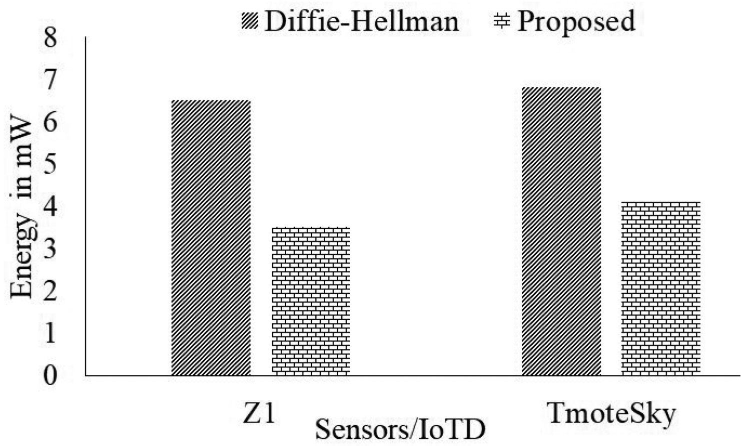


Fig. 5. Energy consumption during key generation.

microcontroller MSP430F1611, with built-in clock factory calibration, 10 KB RAM, and a 48 KB Flash memory. Z1 and TmoteSky sensors are considered for the proposed experimental model: If the proposed model works efficiently with these tiny devices, then they will be fine with other IoT devices.

Starting from request packet generation to establish the secure channel, the power consumption of the sensor devices, i.e., z1 sensor and TmoteSky sensor, has been evaluated in COOJA Simulator. As mentioned in the related works section, the most common key initialization process today is Diffie-Hellman key exchange [8]; so, the sensor node energy consumption for the proposed method is compared with Diffie-Hellman. We found that the proposed security mechanism is supported by these low-power sensors. The performance in terms of energy consumptions during key generation is shown in Figure 5. The performance tends to the normal power consumption behavior while generating or updating the shared secret keys. From this experiment, we conclude that the proposed perimeter-based security solution can also be easily applied to low-power sensor devices.

For network performance in the COOJA simulation environment, a random area is deployed with 101 nodes (i.e., 100 sensors and 1 EDC). The packet size of the simulation is 30 bytes of interval.

We found that the proposed security model also extends the network lifetime considerably compared to another standard security mechanism (i.e., AES 128-bit) with Diffie-Hellman key exchange. Network lifetime of any kind of network is measured by when the first node dies in the network. We have experimented in three phases: without security mechanism, using AES 128-bit technique, and the proposed security mechanism. While running the experiment, we found that the proposed security method extends the network lifetime, as shown in Figure 6. This leads us to conclude that the proposed solution not only secures the data communications but also improves the network performance by extending network lifetime. The communication overhead is computed by following Reference [20]. Subsequently, the communication overhead is computed as a percentage (%) and the performance with respect to the variation of data packets as shown in Figure 7. According to the network properties, the communication overheard is inversely proportional to the number of data packets in the IoT infrastructure as shown in Figure 7.

Finally, the network performance is evaluated for the residual energy, i.e., complete network energy. We have considered the Z1 sensor and initial power as 1 joule for the simulation [26].

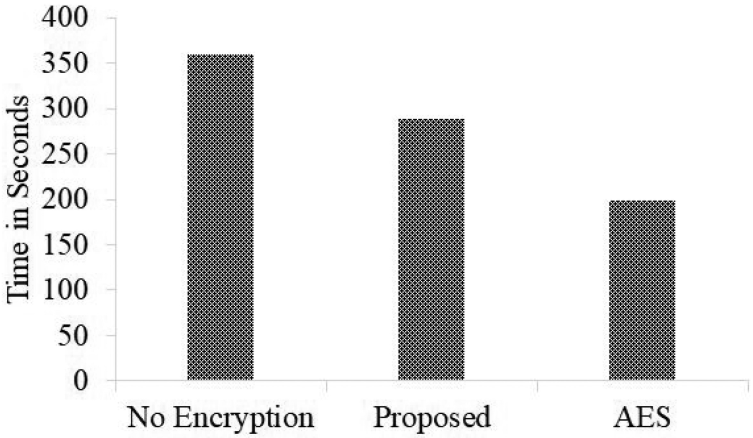


Fig. 6. First node dies in the network.

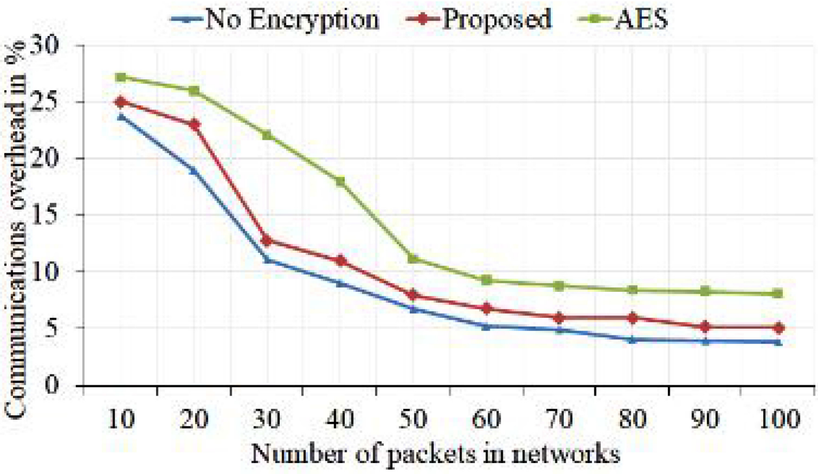


Fig. 7. Communication overhead as a percentage (%).

With this consideration, the experiment runs for 1K seconds and the reading is taken with 100-second interval of time. To conclude and plot the results, we run the simulation five times and take the average. The performance of residual energy is plotted in Figure 8. We have considered the three phases as network lifetime (from last result), i.e., without security mechanism, using AES 128-bit and using the proposed security mechanism. From Figure 8, we found that the experiment result for residual energy performance is much better than traditional AES 128-bit standard security solution and close to data communication without security framework. This concludes the scalability of the proposed security solution. The overall network performance is scalable by reducing communication computational overhead as concluded from Theorem 5. The Theorem 5 claim is validated with the simulation results as shown in Figures 5, 6, and 7. Figure 8 gives the overall network scalability by finding network energy consumption, which is concluded from the first node dying in the network (from Figure 6) and individual node performance depended on the energy consumed during crypto key generation, as shown in Figure 5.

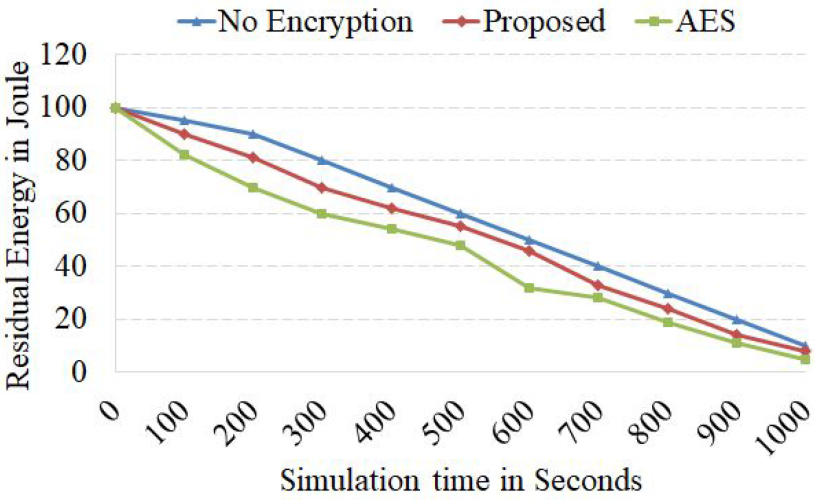


Fig. 8. Residual energy consumption for the network.

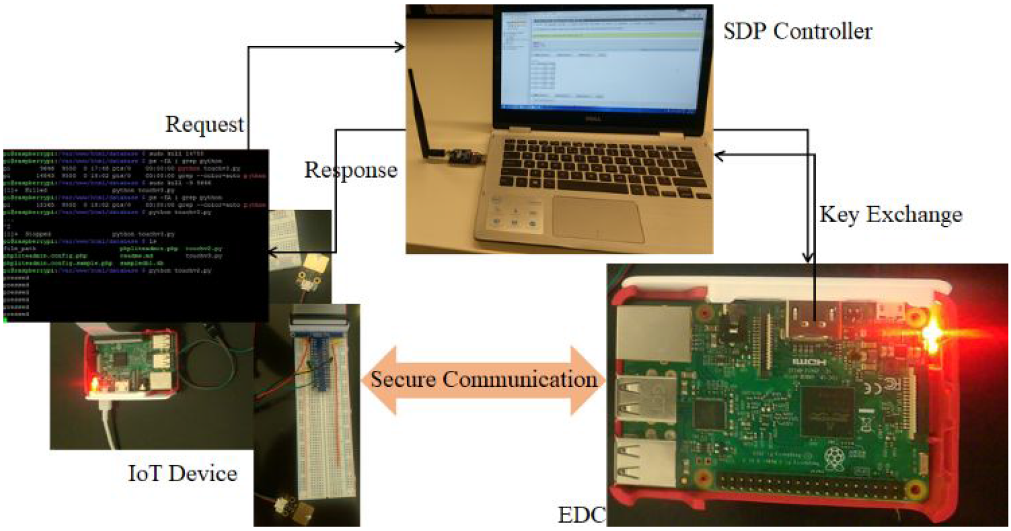


Fig. 9. Real-time testbed scenario for proposed security framework.

6.3 Testbed Evaluation

The proposed security framework was also evaluated in the real-time testbed to see the compatibility. The testbed implementation scenario includes two Raspberry Pis and a computer (laptop) as shown in Figure 9. One Raspberry Pi connected with several sensors and this setup were considered as an IoT device. The other Raspberry Pi works as the EDC and the computer as the SDP Controller. The SDP Controller is a Dell computer with Intel Core i7 processor and 8 GB RAM. The Raspberry Pi configuration is 1.2 GHz 64/32-bit quad-core ARM Cortex-A53 CPU, 1 GB LPDDR2 RAM at 900 MHz, and Broadcom BCM2837 system-on-chip. All the devices are connected with each other through the Internet. Multiple sensors such as temperature, humidity, and touch sensors are connected with Raspberry Pi (IoTD) to gather sensing information and contact to computer (SC) to establish secure channel with the other Raspberry Pi (EDC). SDP-Controller is deployed with MySQL Server to store all the device ID and keys for authentication purpose, whereas the IoTD Raspberry Pi is deployed with SQLite to store the sensed information. Figure 9 shows the complete setup for the testbed deployment.

The Raspberry Pi (IoTD) connects sensors to receive the continuous data streams. The IoTD initiates the authentication process with SC by sharing its identity. Authentication follows the process from Algorithm 1. Subsequently, the SC communicates to the EDC and gets the identification and generates the shared key between IoTD and EDC. Once all the computation is done by SC, it shares the shared secret keys with IoTD and EDC for secure communication between themselves by following Algorithm 2. In overall process, the deployed real-time testbed took 13 seconds starting from IoTD initiation to the secure communication channel initialization between IoTD and EDC. This time is subjective based on the medium of communications and computational power of the SC. The testbed evaluation is mainly focusing on establishing the secure channel between the IoTD and EDC for further communication. The complete testbed setup build follows the Algorithm 3 procedure. The subsequent network performance is considered from simulation results.

From the above theoretical and experimental studies, we conclude that the proposed security framework not only provides secure infrastructure but also improves the communication and computational overhead.

7 CONCLUSION AND FUTURE DIRECTIONS

This article proposed a user-centric security solution for IoT and Edge networks, where the security approach to secure complete systems is shifting from network-centric to user-centric. The proposed security model uses a centralized controller named SDP Controller to authenticate the IoT devices as well as EDCs. IoT devices always initiate the security process to establish secure channels with EDCs for further data processing. The IoTD communicates with the SC and the SC evaluates the IoT authentication and evaluates the associated properties to access a specific EDC. If everything goes well, then the SC generates the shared secret key and initializes the IoTD and EDC for further communication. As a result, the secure channel is established between source and destination after agreement. The performance of the proposed security model is evaluated by theoretical analyses and experimental evaluations. We found that our security solution provides a significant improvement in network performance and prevents potential network attacks.

We will further develop the proposed security solution with symmetric-key encryption for better network lifetime. We are also planning to apply the improved security framework to purely distrusted networks for user-centric security solutions.

REFERENCES

[1] V. Lesser, C. L. Ortiz Jr, and M. Tambe (Eds.). 2012. *Distributed Sensor Networks: A Multiagent Perspective* 9. Springer Science & Business Media 2012.

[2] L. Atzori, A. Iera, and G. Morabito. 2010. The internet of things: A survey. *Comput. Netw.* 54, 15 (2010), 2787–2805.

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. 2013. Internet of things (IoT): A vision, architectural elements, and future directions. *Fut. Gen. Comput. Syst.* 29, 7 (2013), 1645–1660.

[4] D. Puthal, S. Nepal, R. Ranjan, and J. Chen. 2017. DLSeF: A dynamic key-length-based efficient real-time security verification model for big data stream. *ACM Trans. Embed. Comput. Syst.* 16, 2 (2017), 51:1–51:24.

[5] A. Dorri, S. Kanhere, R. Jurdak, and P. Gauravaram. 2017. Blockchain for IoT security and privacy: The case study of a smart home. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops'17)*. 618–623.

[6] R. Roman, P. Najera, and J. Lopez. 2011. Securing the internet of things. *IEEE Comput.* 44, 9 (2011), 51–58.

[7] D. Puthal, S. P. Mohanty, P. Nanda, and U. Choppali. 2017. Building security perimeters to protect network systems against cyber threats. *IEEE Consum. Electron. Mag.* 6, 4 (2017), 24–27.

[8] M. Steiner, G. Tsudik, and M. Waidner. 1996. Diffie-Hellman key distribution extended to group communication. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security*. 31–37.

[9] Cloud Security Alliance (CSA). Retrieved from https://cloudsecurityalliance.org/ group/software-defined-perimeter.

[10] S. Nepal, J. Zic, D. Liu, and J. Jang. 2011. A mobile and portable trusted computing platform. *EURASIP J. Wirel. Commun. Netw.* 1, (2011), 1–19.

[11] D. Puthal, S. Nepal, R. Ranjan, and J. Chen. 2016. Threats to networking cloud and edge datacenters in the internet of things. *IEEE Cloud Comput.* 3, 3 (2016), 64–71.

[12] D. Minoli, K. Sohraby, and J. Kouns. 2017. IoT security (IoTSec) considerations, requirements, and architectures. In *Proceedings of the 14th IEEE Consumer Communications & Networking Conference (CCNC'17)*. 1006–1007.

[13] B. Mukherjee, R. Neupane, and P. Calyam. 2017. End-to-end IoT security middleware for cloud-fog communication. In *Proceedings of the 4th International Conference on Cyber Security and Cloud Computing (CSCloud'17)*. 151–156.

[14] Y. Hatri, A. Otmani, and K. Guenda. 2018. Cryptanalysis of an identity based authenticated key exchange protocol. *Int. J. Commun. Syst.* 31, 3 (2018), 1–8.

[15] E. Bertino and N. Islam. 2017. Botnets and internet of things security. *IEEE Comput.* 50, 2 (2017), 76–79.

[16] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. Sadeghi, and S. Tarkoma. 2017. IoT sentinel: Automated device-type identification for security enforcement in IoT. In *Proceedings of the 37th International Conference on Distributed Computing Systems (ICDCS'17)*. 2177–2184.

[17] M. Miettinen, S. Marchal, I. Hafeez, T. Frassetto, N. Asokan, A. Sadeghi, and Sasu Tarkoma. 2017. IoT sentinel demo: Automated device-type identification for security enforcement in IoT. In *Proceedings of the 37th International Conference on Distributed Computing Systems (ICDCS'17)*, 2511–2514.

[18] M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, and A. Rindos. Sdsecurity: A software defined security experimental framework. In *Proceedings of the International Conference on Communications Workshop*. 1871–1876.

[19] A. Darabseh, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk, and A. Rindos. 2015. SDDC: A software defined datacenter experimental framework. In *Proceedings of the 3rd International Conference on Future Internet of Things and Cloud (FiCloud'15)*. 189–194.

[20] D. Puthal, X. Wu, S. Nepal, R. Ranjan, and J. Chen. 2017. SEEN: A selective encryption method to ensure confidentiality for big sensing data streams. *IEEE Trans. Big Data* 5, 3 (2017). DOI: 10.1109/TBDATA.2017.2702172

[21] D. Huang, S. Misra, M. Verma, and G. Xue. 2011. PACP: An efficient pseudonymous authentication-based conditional privacy protocol for VANETs. *IEEE Trans. Intell. Transport. Syst.* 12, 3 (2011), 736–746.

[22] A. Ghosh and S. Sarkar. 2018. Pricing for profit in internet of things. *IEEE Trans. Netw. Sci. Eng.* 6, 2 (2018). DOI: 10.1109/TNSE.2018.2796592

[23] Scyther. http://www.cs.ox.ac.uk/people/cas.cremers/scyther/. Accessed in February 2018.

[24] Contiki operating system. http://www.contiki-os.org. Accessed in February 2018.

[25] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. 2006. Cross-level sensor network simulation with COOJA. In *Proceedings of the 31st IEEE Conference on Local Computer Networks*. 641–648.

[26] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. 2000. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences.*

[27] H. Liu, H. Ning, Y. Zhang, and L. T. Yang. 2012. Aggregated-proofs based privacy-preserving authentication for V2G networks in the smart grid. *IEEE Trans. Smart Grid* 3, 4 (2012), 1722–1733.

[28] H. Liu et al. 2014. Role-dependent privacy preservation for secure V2G networks in the smart grid. *IEEE Trans. Inf. Forens. Sec.* 9, 2 (2014), 208–220.

[29] F. Kausar, S. Hussain, L. T. Yang, and A. Masood. 2008. Scalable and efficient key management for heterogeneous sensor networks. *J. Supercomput.* 45, 1 (2008), 44–65.

[30] A. Castiglione et al. 2017. Secure group communication schemes for dynamic heterogeneous distributed computing. *Fut. Gen. Comput. Syst.* 74, (2017), 313–324.

[31] A. Castiglione et al. 2017. Supporting dynamic updates in storage clouds with the Akl–Taylor scheme. *Inf. Sci.* 387, (2017), 56–74.

Received January 2019; revised March 2019; accepted July 2019