Internet of Things 7 (2019) 100061

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Research article

Anonymous mutual IoT interdevice authentication and key agreement scheme based on the ZigBee technique

Mohammed Alshahrani[a,*], Issa Traore[a], Isaac Woungang[b]

[a] Electrical and Computer Engineering Department, University of Victoria, Victoria, Canada
[b] Department of Computer Science, Ryerson University, Toronto, Canada

ARTICLE INFO

Article history:
Received 22 April 2019
Accepted 20 May 2019
Available online 24 May 2019

Keywords:
Internet of Things (IoT)
Smart home
Authentication and access control
BAN
AVISPA

ABSTRACT

Establishing end-to-end device authentication in Internet of Things (IoT) networks is challenging because of the heterogeneous nature of IoT devices. By covering different security properties, various authentication protocols have been introduced to ensure a certain level of security and privacy protection. In this paper, we propose an anonymous device-to-device mutual authentication and key exchange scheme based on the ZigBee technique, designed for a smart home network, an important domain in the IoT. The proposed protocol relies on symmetric encryption and enables IoT devices to authenticate in the network and agree on a shared secret session key when communicating with each other via a trusted intermediary (home controller). To achieve perfect forward secrecy, the session keys are changed frequently after every communication session. The proposed scheme achieves secure anonymous authentication with the unlinkability and untraceability of IoT devices' transactions. The overhead and efficiency of the proposed scheme are analyzed and compared with other related schemes. In addition, the security of the scheme is evaluated using three different methods: informal analysis, formal analysis using the Burrows–Abadi–Needham logic (BAN), and a model check using the automated validation of internet security protocols and applications (AVISPA) toolkit.

1. Introduction

Internet of Things (IoT) technology has gained much attention and popularity over the past two decades. As a concept, the IoT has experienced various developments and advancements that have made it suitable for many uses and fields of operations, with one such prominent field being the IoT smart home infrastructure, which allows devices that are linked through the Internet to operate and monitor the activities of a given household [15].

Despite the growing interest in IoT products and services and the advancements in its underlying technology, IoT devices and networks are exposed to a wide variety of security threats, some of which are well-known and part of the existing attack arsenal against conventional systems, while others involve novel attack vectors specific to IoT technology and that are unknown until they are detected in the network [1,9,13,21]. Authentication breaches represent a significant part of these threats because of the central role that authentication plays in gaining access to the network and web resources and the fact that many IoT devices heavily depend on passwords only for authentication, which in many cases consist of the default

* Corresponding author.
E-mail addresses: malshahr@uvic.ca (M. Alshahrani), itraore@uvic.ca (I. Traore), iwoungan@ryerson.ca (I. Woungang).

https://doi.org/10.1016/j.iot.2019.100061
2542-6605/

passwords supplied by the manufacturers. The reliance of IoT infrastructures on simple authentication schemes, such as password-based authentication, can be explained by the inherent limitations of the resources available in IoT devices. These constraints make the deployment and operation of strong authentication schemes more challenging for IoT infrastructures. Furthermore, authentication necessitates identification by using a unique identity, and past and future transactions linked to the same identity can be traceable, resulting in a privacy breach. In IoT systems, such as a smart home, the traceability of transactions can help to track the lifestyle of the household or infer from the data critical information such as credit history, health history, marital status, and so forth [18]. In this case, it becomes essential for IoT authentication to incorporate anonymity and unlinkability, which are privacy-preserving properties [14] that make the traceability of a homeowner's transactions much harder.

In addition, a large number of IoT devices are made by different manufacturers, and they do not always follow the same protocols; hence, they face interoperability challenges regarding authentication, key exchange, and data exchange. To get these highly heterogeneous schemes to work together on a single network, the devices must use a common language.

To address the aforementioned security challenges, we propose an anonymous interdevice mutual authentication and key exchange scheme for the IoT that ensures anonymity and unlinkability; this authentication scheme is based on the ZigBee technique and has a centralized architecture. Zigbee is one of the most popular wireless technologies used to connect IoT devices and is based on the IEEE 802.15.4 standard and developed by the ZigBee Alliance. ZigBee is a mesh protocol; it builds a network in a hierarchical manner to enable a number of devices to communicate with one another and some communication features, such as authentication and encryption. Our proposed authentication scheme is designed to be used by various devices in the IoT network, and it is particularly streamlined for the IoT smart home network. The proposed protocol is similar to the ZigBee protocol [26] in the sense that each network has a central device (ZigBee coordinator) responsible for starting and maintaining the communications among network nodes (ZigBee nodes). Our scheme also utilizes symmetric encryption between the nodes, which is similar to ZigBee; here, only the controller that possesses the symmetric key of the node can encrypt the messages. The ZigBee security scheme is built on the assumption that devices are preloaded with symmetric keys, and these symmetric keys are securely stored and are never transmitted in an unencrypted fashion [26].

Our proposed Zigbee-based IoT authentication scheme achieves different security properties, anonymity, unlinkability, and conditional traceability, in addition to the important dual properties of confidentiality and integrity.

To demonstrate the ability of the proposed scheme when operating in a resource-constrained environment, we evaluate and compare its performance with existing related schemes. Furthermore, we formally validate the security of our protocol by using the widely accepted Burrows–Abadi–Needham (BAN) logic and the automated validation of internet security protocols and applications (AVISPA) analysis tool. In addition, an informal security analysis of the proposed scheme is discussed.

The remaining parts of the current paper are organized as follows. Section 2 summarizes and discusses the related work. Section 3 briefly discusses the ZigBee network and then introduces the proposed smart home system architecture by presenting the underlying network model and outlining the security design goals and attack model. An anonymous mutual interdevice authentication and key exchange scheme for IoT smart homes is proposed in Section 4. Section 5 evaluates the overhead and performance of the proposed scheme, comparing it with other related schemes. Section 6 validates the security of the proposed scheme through a formal and informal analysis. Finally, the concluding remarks and an outline of future work are presented in Section 7.

2. Related work

Several proposals have been published in the literature regarding the authentication of IoT devices, by taking into consideration the constraints placed on these devices. The constrained devices use networks with a limited bandwidth, limited processing power, limited power supply, and so on. The proposed methods consider at least one of these constraints but may not be suitable for all of them.

Some existing authentication protocols depend on public key encryption, but IoT end devices may not have sufficient resources to support the required operations. Other existing authentication protocols focus on the authentication between the server and IoT devices, ignoring the presence of interdevice authentication (among IoT devices). Some protocols can provide only partial coverage of the crucial security properties required or suffer from unsatisfactory performance.

In the following, we revisit and discuss representative of samples of the existing literature on IoT authentication.

Based on symmetric encryption, Lee et al. [10] proposed a lightweight mutual authentication protocol for IoT systems. In this work, the authors relied on lightweight symmetric encryption and avoided complex encryption schemes, such as asymmetric encryption. Consequently, this protocol is practical for constrained IoT devices, and like our proposed protocol, it supports anonymity, unlinkabilty, and untraceability.

Santoso and Vun [17] presented an IoT authentication scheme for a smart home system. The proposed protocol relies on an AllJoyn framework and capitalizes on Elliptic Curve Cryptography (ECC) for authenticating IoT devices. The scheme assigns the Wi-Fi gateway to initialize system configuration and to authenticate IoT devices. In addition, user access is controlled by a mobile device using an Android application. However, the proposed scheme does not provide full end-to-end security guarantees because some key properties, such as anonymity, unlinkabilty, and untraceability, are not satisfied. In addition, the scheme relies on public key infrastructure, which is not efficient in terms of storage and computation for IoT-constrained devices.

Zhao et al. [25] introduced an asymmetric mutual authentication scheme for an IoT system. In the proposed scheme, authentication occurs between the terminal device and the platform. The scheme is based on hashing and feature extraction. The authors enhanced the IoT security and decreased the computation and communication costs by combining one-way hashing using SHA1 and feature extraction. This combination helped in avoiding any collision attacks. However, the scheme did not support key anonymity, unlinkabilty, and untraceability.

Chu et al. [5] developed an authentication framework that relies on ECC to generate public and private key pairs. The main objective was to satisfy the requirements for heterogeneous IoT environments. In the initialization phase, the elliptic curve public parameters were initialized and computed so that these parameters could be used during the authentication phase. However, this authentication scheme provided only partial coverage of the crucial security properties aforementioned and relied on public key infrastructure.

Gaikwad et al. [7] utilized the three-level secure Kerberos authentication for an IoT smart home system. The proposed scheme employed symmetric algorithms, such as the advanced encryption standard (AES) and hash algorithms, for security. In this work, the authors did not consider IoT-device-to-IoT-device interactions. Furthemore, the proposed scheme does not suitably cover transactions anonymity, unlinkabilty, and untraceability.

Ashibani et al. [4] proposed a dynamic authentication scheme for local and remote access in IoT smart home devices. The proposed framework was based on traditional static credentials and context information. Different types of context information were considered, including physical contexts (location, IP and Bluetooth), systematic contexts (network bandwidth), device contexts (MAC address and display size), personal context (identity, age, and other information, such as name, gender, and height), activity contexts (user's calendar), historical information (access patterns and logs), and application contexts (agent and service). This work considered a particular IoT-device-to-cloud interaction. It did not provide authentication in IoT-device-to-IoT-device interactions, nor did it satisfy the properties of anonymity, unlinkability, and untraceability.

Recently, Mishra et al. [12] introduced a strong authentication protocol using smart card for an IoT-based wireless sensor network. The proposed protocol utilized password hash values and preshared keys to provide authentication between the gateway node and the sensor node. The authors stated that their authentication protocol achieved user anonymity and resisted various attacks. Authentication in IoT-device-to-IoT-device interactions was not supported in this work.

More recently, an anonymous lightweight mutual authentication and automated access control scheme for IoT smart homes using a cumulative keyed hash was introduced by Alshahrani and Traore [2]. The nodes were authenticated using the controller node, establishing a temporary session key in an anonymous and unlinkable manner. Moreover, the access control process was automated during the bootstrapping phase. In addition, the authors capitalized on fog computing architecture to support the identity assurance of IoT devices. This modified fog architecture helped in preventing attacks such as identity theft.

3. Network and attacker models

In this section, we provide an overview of the ZigBee network and introduce the network and attacker models underlying the proposed IoT authentication scheme.

3.1. ZigBee network

A typical Zigbee-based network is comprised of a ZigBee coordinator and ZigBee nodes. The ZigBee coordinator is always the central device and is aware of all the nodes within its network; it is also in charge of managing the information about each node and starting and maintaining the network. The ZigBee coordinator controls each ZigBee node, such as IP camera, door lock, and so forth, in the network. Any communication between any two ZigBee nodes must go through the coordinator before reaching the destination node [8]. Thus, every ZigBee network must have a network coordinator. On the other hand, ZigBee nodes usually only are devices that interact with the physical world. There are several topologies that are supported by ZigBee, including star, mesh, and cluster tree. The star and mesh topologies are the most common ZigBee types [19]. In the star topology, which is depicted by Fig. 1, the coordinator is not only responsible for routing the packets in the network, but also for initiating and maintaining the devices on the network. End devices can communicate only through the coordinator.

3.2. IoT network model

The proposed IoT authentication protocol consists of two kinds of participants: the controller and the devices. The controller denoted as C is always the central device in charge of starting and maintaining the network, and it is aware of all the devices' real identities. The controller is also responsible for forwarding messages between devices and granting a secure channel. Simply put, the controller controls each device denoted as N, such as IP camera, door lock, and so forth, in the IoT network, and any communication between any two devices is subject to the controller's permission, as shown in Fig. 2.

If the communication is permitted, the controller computes a temporary key and distributes it to each IoT device so that it can be used to secure communication between these devices. Therefore, the network is more similar to the mesh network topology, as shown in Fig. 3.

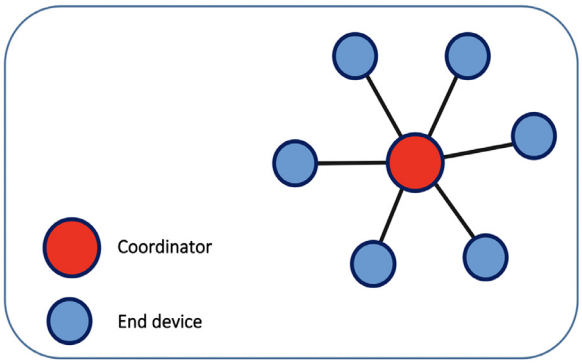

Fig. 1. Star topology.

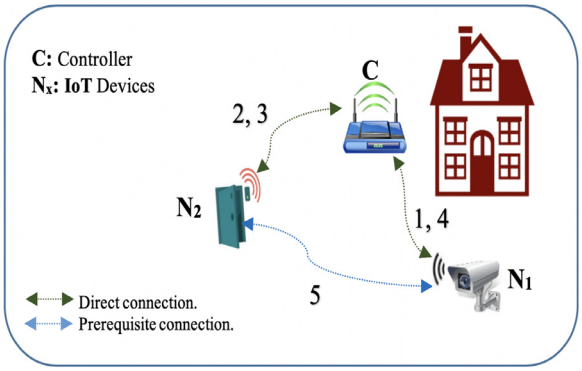

Fig. 2. IoT network model.

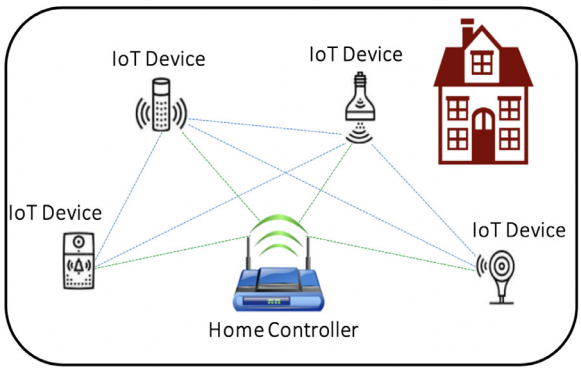

Fig. 3. A mesh network topology.

In our scheme, data security is ensured by encrypting the payload using the AES algorihm with a 128 bits key length (16 Bytes). In addition, a one-way cryptographic hash function with an incremental counter of 128 bits length (16 Bytes as a nonce) is used to validate and ensure the integrity of the data transmitted between the two devices. This is achieved by using the keyed hash message authentication code (HMAC), which is appended to the following message: HMAC = h(CTR,*M*).

Message headers are sent in a clear but authenticated way, while the message payload is encrypted and then authenticated. In this way, we ensure that the device only reads a valid message because the HMAC filters out the invalid ciphertext before decryption, thus protecting against any attacks. In addition, the HMAC does not reveal any information about the plaintext in the message.

The number of IoT devices can vary throughout the lifetime of the network. Moreover, IoT devices could possibly be destroyed or stolen. Moreover, because the addition of new IoT devices to the network is highly likely, this addition must be expected within IoT security protocols.

Our proposed scheme enables the dynamic addition of IoT devices to the network without causing any changes in the present security states of the network. The addition of a dynamic IoT device is solely limited by the controller, which is assumed to be a powerful device because the controller stores separate information for each IoT device in the network.

3.3. Security design goals

To achieve anonymous authentication for a trustworthy IoT, our design should achieve the following security goals:

- *Anonymity*: this is achieved based on both pseudonyms and trust so that devices can communicate anonymously with the controller and with each other without exchanging their real identities.
- *Confidentiality*: this is achieved through symmetric encryption.
- I*ntegrity*: this is achieved by using a keyed HMAC based on a trusted incremental counter.
- *Untraceability*: this ensures that the IoT devices' IDs and messages can only be traced by the trusted controller.
- *Unlinkability*: the IoT device uses a pseudonym when transmitting data. The intruder cannot trace back the real device's identity from the pseudonym. The use of pseudonyms also ensures that when an IoT device uses network resources multiple times, it will be challenging and difficult to link these uses together by intruders. Thus, the communication sessions are not linkable, and the same property applies to both the real device's identity and the pseudonym.
- *Forward/backward security*: this ensures that if the sender's private key is compromised, the past and future messages will remain confidential.

3.4. Attacker model

A better understanding of threats helps us make better decisions about where to deploy defensive techniques. Dolev–Yao's threat model [6] is employed to anticipate any security issues in our IoT network model. The attacker model is based on the following two assumptions:

- Cryptography is secure:
  1. The attacker is not able to decrypt a message without the key.
  2. The attacker is not able to compute HMAC without the key.
  3. The attacker is not able to guess an encryption key or a nonce.
- The attacker has a complete control over the system, so it has the ability to do the following:
  1. The attacker is able to initiate any number of parallel protocol sessions.
  2. The attacker is aware of all the public data of the protocol.
  3. The attacker benefits from all the privileges/keys of bad agents.
  4. The attacker is able to read, store, and block every message in transit.
  5. The attacker is able to create and transmit messages.
  6. The attacker is able to construct and deconstruct messages.
  7. The attacker is able to encrypt/decrypt if the encryption/decryption key is known.

The Dolev–Yao threat model aids in evaluating the security features of the proposed scheme. Hence, the security analysis and simulation of our scheme are provided using this model.

4. Proposed approach

Our proposed scheme is a symmetric encryption-based authentication scheme with key agreement. The symmetric encryption is based on a AES 128/CCM (counter with CBC-MAC) mode. Furthermore, this scheme is payload embedded, that is, the authentication and key exchange process are embedded in the payload. In this way, the handshaking overhead is reduced.

In this section, the various phases in the life cycle of an IoT device from the manufacturing phase to the key exchange phase are laid out. Two common problems in IoT cryptography are further explored, as follows:

1. Device-to-device authentication: where a device determines that one or more devices are indeed who they say they are.
2. Key exchange: where two or more devices agree on a shared key that can later be used to establish an encrypted communications channel.

In our proposed protocol, an IoT device (N) and a controller node (C) are responsible for ensuring secure anonymous mutual authentication and key exchange. In more detail, the proposed protocol framework consists of a predeployment phase, a registration phase, and a protocol execution phase. The protocol execution phase is broken down into two other phases: the device-to-device authentication and key exchange and the device-to-device communication. In the proposed scheme, all IoT entities will be referred to as IoT devices, with the centralized device named the controller. As mentioned, our scheme utilizes symmetric encryption between devices similar to ZigBee, in which only the controller that possesses the symmetric keys of the IoT devices can make a direct connection with these devices and encrypt the messages. The devices encrypt their messages using their symmetric keys when communicating with the controller, and these messages can only

Table 1
Security of IoT network.

Security property ↓ Phases →	Controller (C) ↔ IoT device (N)	IoT device (N) ↔ IoT device (N)
Confidentiality with symmetric encryption	Symmetric key (K_N)	Temporary shared key (TK)
Integrity and Authenticity With keyed-hash Message Authentication Code (HMAC)	Incremental counter (CTR) as a nonce	Nonce (ns)

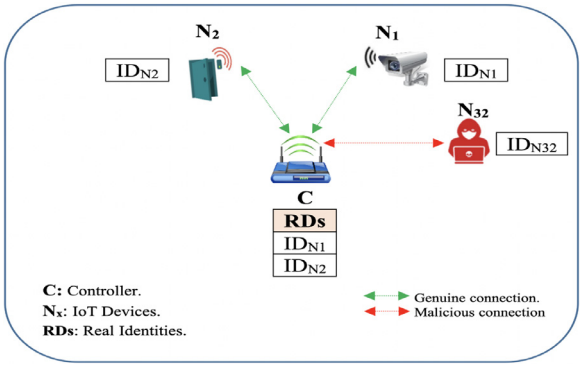

Fig. 4. Identity management.

be read by the controller. Furthermore, communication across the entire network occur from one device to another and are subject to the controller's permission. If the communication is allowed, the controller will generate a temporary session key and share it with the communication participants. For example, an IP camera wishing to send a signal to open a home's door lock will go through the controller so that the controller can act and allow the communication session to be established between the IP camera and the home's door lock. The temporary session keys, which change for every communication session, satisfy the security properties of perfect forward secrecy. The security parameters of the IoT network entities are summarized in Table 1.

Furthermore, during the communication process, IoT devices use dynamic IDs (pseudonyms) instead of using their real IDs. Dynamic IDs offer the anonymity of senders' IDs, receivers' IDs, and the sender-receiver relationship, in addition to device untraceability and end-to-end flow untraceability. The framework phases are explained below.

4.1. Predeployment phase

During the predeployment phase, each IoT device is provided with an identity (ID) and assumed to possess a "fingerprint" generated from an accelerometer [27]. The accelerometer is embedded in the device's inner chips during the device manufacturing process. The main benefit of such an accelerometer is to generate a relevant unique key for the device [24]. The uniqueness of the generated key in the network would be guaranteed based on the diversity of the manufacturing principles and methods. The key will be considered the symmetric key for a device in the network. Moreover, we assume that each IoT device knows the controller's public key.

4.2. Entities registration phase

When the system administrator (SA) receives the IoT devices, he or she will register all devices using their real identities (IDs) in the home controller (C). In this way, C can identify the registered IoT devices and differentiate between them during the authentication process. In addition, SA can capitalize on the access control list to prevent illicit access to and control access between IoT devices using the IoT devices' real identities (IDs). For example, it may prevent device 1 (ID_{N1}) from accessing, say, device 7 (ID_{N7}). We emphasize the fact that the IoT devices' real identities (IDs) are kept secure and never transmitted in plain text because we use only the dynamic IDs during authentication and communication.

Remark 1. The identity ID_N represents the secure permanent real identity for the IoT device, which is never transmitted in plain text. Every IoT device has a unique identity ID and knows the identities of the connected devices that it should communicate with. These permanent real identities facilitate the authentication process between the controller and IoT devices, as shown in Fig 4.

4.3. Authentication phase

The entity authentication process is performed in the system configuration phase. The first step in the authentication process starts with the IoT device and is based on the fact that the controller's public key K_C is known by all network

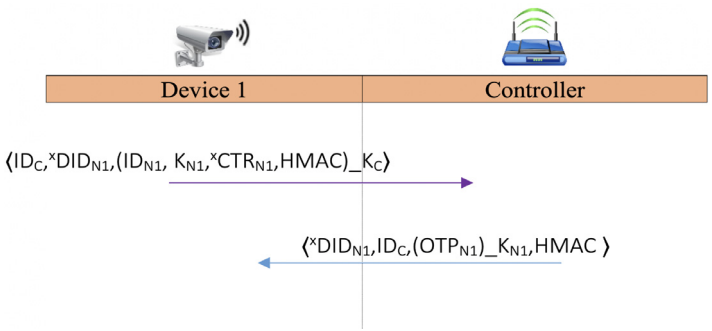

Fig. 5. Entity's authentication process.

devices and each IoT device is initially provided with a real identity *ID* by the manufacturer, as discussed in Section 4.1. This ID facilitates the validation process done by the controller, as explained later in this section. The IoT device will send its symmetric key K_N and the counter CTR_N to the controller using K_C because the key exchange has to be in a secure channel. The counter CTR_N is incremented in every message passing between the IoT device and the controller.

Fig. 5 describes the steps involved in the authentication stage.

The following steps summarize the authentication phase between *N*1 and *C*:

Step A1 $N1 \rightarrow C : \langle ID_{N1}, K_{N1}, {}^{x}CTR_{N1}, HMAC)\_K_C\rangle$. *N* performs as follows.

First, IoT device *N*1 generates two parameters, namely, K_{N1} and ${}^{x}CTR_{N1}$. K_{N1} is the IoT device symmetric key that is used to encrypt data between *C* and *N*1. ${}^{x}CTR_{N1}$ is a counter that is incremented on every message, where x denotes the sequence number corresponding to the previous counter record stored in the database for a given device. Such counter is used to ensure the freshness of the protocol instance, help in the validation process of the integrity of encrypted data, and ensure IoT device anonymity and unlinkability properties. Even if the symmetric key is compromised, the counter ${}^{x}CTR_{N1}$ will withstand and help protect against replay and data integrity attacks.

*N*1 creates a dynamic identity ${}^{x}DID_{N1}$ =h(ID_{N1}, ${}^{x}CTR_{N1}$) that changes for every session. Because ${}^{x}CTR_{N1}$ is different in each protocol instance, ${}^{x}DID_{N1}$ will be unique in each session and can only be tracked by controller C; hence, the untraceability propriety is achieved. In the first authentication message, *N*1 concatenates ID_{N1}, K_{N1} and ${}^{x}CTR_{N1}$ and stores them in M=($ID_{N1} \| K_{N1} \| {}^{x}CTR_{N1}$). Next, *N*1 calculates the HMAC=h(${}^{x}CTR_{N1}$, *M*). In this way, we guarantee that each HMAC is calculated based on a dynamic key so that if one key is compromised, we can still detect the attack. Next, *N* encrypts *M* and HMAC, and sends the message Msg to the controller *C*. However, in all other messages, HMAC is sent unencrypted. Finally, at the end of the step, *N*1 increments ${}^{x}CTR_{N1}$ by 1 (which becomes ${}^{x+1}CTR_{N1}$) and updates its database, as shown in Table 3.

Step A2 $C \rightarrow N1 : \langle {}^{x}DID_{N1}, ID_C, (OTP_{N1})\_K_{N1}, HMAC\rangle$. *C* performs as follows.

Once controller *C* receives the message, it decrypts the message using K_{CP} and looks the ID_{N1} up and validates it. Then, it computes *HMAC =(${}^{x}CTR_{N1}$, M) and ${}^{*x}DID_{N1}$ = h(ID_{N1}, ${}^{x}CTR_{N1}$) and verifies whether *HMAC and ${}^{*x}DID_{N1}$ are equal to the received HMAC and ${}^{x}DID_{N1}$, respectively. *C* stores *N*1 information, namely K_{N1}, ${}^{x}CTR_{N1}$ and ${}^{x}DID_{N1}$, in its database. The controller has a random number generator, which contributes to generating OTP_{N1} (used for the first communication established in the configuration stage) for all devices in the network. C increments ${}^{x}CTR_{N1}$ by 1 (which becomes ${}^{x+1}CTR_{N1}$). Next, *C* encrypts OTP_{N1} using K_{N1} and concatenates the message parameters as M=(${}^{x}DID_{N1} \| ID_C \| (OTP_{N1})\_K_{N1}$). After that, *C* computes HMAC=h(${}^{x+1}CTR_{N1}$,*M*). Then, *C* sends the message Msg to *N*1. *C* increments ${}^{x+1}CTR_{N1}$ again by 1 (which becomes ${}^{x+2}CTR_{N1}$) and computes ${}^{x+1}DID_{N1}$= h(ID_{N1},${}^{x+2}CTR_{N1}$) for the upcoming session. Finally, *C* updates *N*1 information, as shown in 4.

Step A3 *N*1: *N*1 performs as follows.

Upon receipt of the message by *N*1, *N*1 computes *HMAC = h(${}^{x+1}CTR_{N1}$,*M*). Then, it compares the HMAC received in the message with *HMAC. If there is a match, the message's integrity is correctly verified, and the message is considered to be a new message from C because ${}^{x+1}CTR_{N1}$ is fresh. After that, *N*1 decrypts the message using K_{N1} and extracts OTP_{N1}. *N*1 increments ${}^{x+1}CTR_{N1}$ by 1 (which becomes ${}^{x+2}CTR_{N1}$) and computes a new dynamic identity ${}^{x+1}DID_{N1}$= h(ID_{N1},${}^{x+2}CTR_{N1}$) for the upcoming session. Finally, N1 updates its database, as shown in Table 5.

At this stage, the authentication process is done, and the controller will release a secure channel between *N*1 and itself when the device wishes to send or receive data.

Table 2
Notations used in our protocol.

Notation	Description
SA	System Administrator
$N1$	IoT Device 1
$N2$	IoT Device 2
C	Controller
K_{N1}	Device 1 symmetric key
K_{N2}	Device 2 symmetric key
K_{12}	Temporary shared key between $N1$ and $N2$
K_C	Controller's public key
K_{CP}	Controller's private key
OTP_{N1}	One-time password for $N1$
ID_{N1}	Real device 1 identity
ID_{N2}	Real device 2 identity
ID_C	Controller identity
DID_{N1}	Dynamic identity of device 1
DID_{N1}	Dynamic identity of device 2
CTR_{N1}	Counter value between $N1$ and C
CTR_{N2}	Counter value between $N2$ and C
ns	Nonce
PoB	Proof of Belonging
PoC	Proof of Controller
HMAC	keyed-hash message authentication code
M	All message parameters but HMAC
Msg	All message parameters
$N \rightarrow C$:Msg	Device N sends the message Msg to controller C via a public channel

Table 3
Device 1 database.

Device 1 ($N1$)	
Device Identity	ID_{N1}
Dynamic of Device Identity	${}^{x}DID_{N1}$
Device One-time password	–
Counter	${}^{x+1}CTR_{N1}$
Symmetric key	K_{N1}
Controller Key	K_C

Table 4
Controller database.

	$N1$	$N2$
Device Identity	ID_{N1}	
Dynamic of Device Identity	${}^{x}DID_{N1}$	
Device One-time password	OTP_{N1}	
Device symmetric key	K_{N1}	
Counter value	${}^{x+2}CTR_{N1}$	

Table 5
Device 1 Database (updated).

Device 1 ($N1$)	
Device Identity	ID_{N1}
Dynamic of Device Identity	${}^{x+1}DID_{N1}$
Device One-time password	OTP_{N1}
Counter	${}^{x+2}CTR_{N1}$
Symmetric key	K_{N1}
Controller Key	K_C

4.4. Communication phase (Interdevice authentication and data transfer)

In the proposed scheme, communication can occur between the IoT devices and the controller or among the IoT devices. The first scenario was explained in the previous section. Now, we will explain the second scenario about communication among devices. Fig. 6 describes the device communication process. The most important aspect that needs to be addressed before going through the communication scenario is the security of the channel and the exchanged messages. The AES algorithm will be utilized to encrypt the exchanged data among the IoT network devices. As mentioned in the device

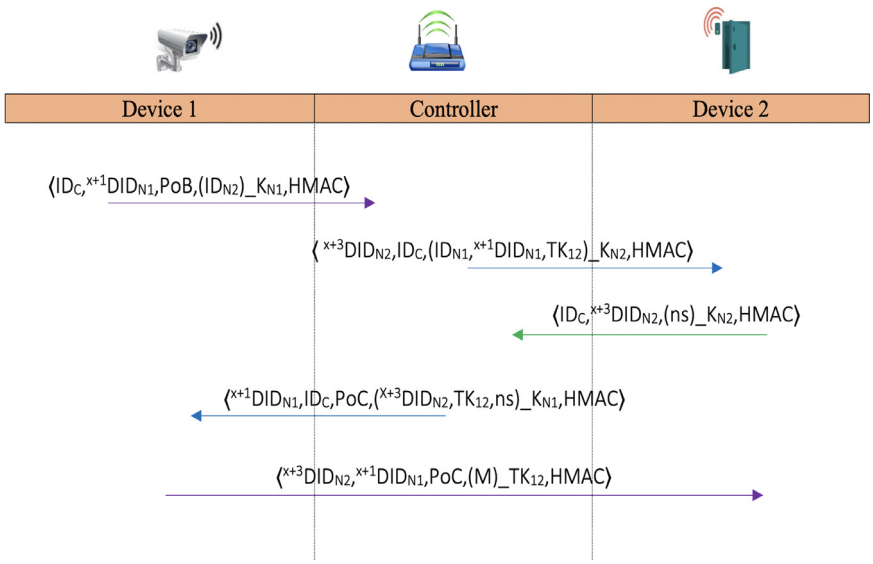

Fig. 6. Entity's authentication and communication between IoT devices.

authentication section, each device has a unique symmetric key, so the messages will be encrypted using the symmetric key when sent to the controller. However, for IoT device $N1$ (sender) to communicate with another IoT device $N2$ (receiver) in the smart home network, the sender initially forwards this communication request to the controller, which will check some secure parameters such as nonce, incremental counter, and HMAC, and if these parameters are valid, it will check its database for the receiver's information and establish a secure communication session between $N1$ and $N2$ using some authentication parameters, such as proof of control (PoC), nonce (Ns), and temporary shared key TK_{12}, as explained in this subsection. PoC is used to prove to the receiving IoT device that the controller is aware of the data request sent by another IoT device because PoC contains secret parameters.

The communication phase occurs after successful completion of the authentication phase. In this phase, N1 wishes to send data to $N2$. To deliver the data from $N1$ to $N2$, the sequence of steps that are taken is summarized as follows:

Step $C1$ N1 $\rightarrow C : \langle ID_C, {}^{x+1}DID_{N1}, PoB, (ID_{N2})\_K_{N1}, HMAC\rangle$. $N1$ performs as follows.

When $N1$ wishes to communicate with another IoT device–say $N2$, where it will be assumed that $N2$ is an authentic device–it connects with the controller and asks for a secure connection with $N2$. $N1$ will load ${}^{x+1}DID_{N1}$ and OTP_{N1} and compute the proof of belonging PoB =h $(OTP_{N1}, {}^{x+2}CTR_{N1})$. PoB is constituted of OTP_{N1} and ${}^{x+2}CTR_{N1}$. Thus, PoB proves to C the authenticity of $N1$ and the freshness of the message during the first data communication. Next, $N1$ loads the destination identity ID_{N2} and encrypts it using K_{N1}. Then, $N1$ concatenates the message parameters as $M = (ID_C \| {}^{x+1}DID_{N1} \| PoB \| (ID_{N2})\_K_{N1})$. After that, N computes HMAC = h$({}^{x+2}CTR_{N1}$,M) and sends the message Msg to C to request a data channel. Finally, $N1$ increments the counter value (which becomes ${}^{x+3}CTR_{N1}$) and updates its database

Step C2 $C \rightarrow N2 : \langle {}^{x+3}DID_{N2}, ID_C, (ID_{N1}, {}^{x+1}DID_{N1}, TK_{12})\_K_{N2}, HMAC\rangle$. $N1$ performs as follows.

Once C obtains the message, it decrypts it using K_{N1}. Next, it computes *PoB=h$(OTP_{N1}, {}^{x+2}CTR_{N1})$ and *HMAC =h$({}^{x+2}CTR_{N1}$,M) and verifies whether *PoB and *HMAC are equal to the received PoB and HMAC, respectively. Otherwise, the message will be dropped and considered part of a replay attack. After that, C decrypts the received message using K_{N1} and extracts ID_{N2}. Then, C increments the counter value (which becomes ${}^{x+3}CTR_{N1}$) and updates its database.

Next, C prepares a message to be sent to $N2$. It first loads ${}^{x+3}DID_{N2}$ for ID_{N2} from its database and then generates the key TK_{12} using a random number generator, which will be used to encrypt the data communication between $N1$ and $N2$. This key will change for each session, ensuring that the data exchange will be safe every time. After that, it encrypts ID_{N1}, ${}^{x+1}DID_{N1}$ and TK_{12} using K_{N2} and concatenates the message parameters as M=$({}^{x+3}DID_{N2} \| ID_C \| (ID_{N1}, {}^{x+1}DID_{N1}, TK_{12})\_K_{N2})$. Then, it computes HMAC =h$({}^{x+4}CTR_{N2}$,M) based on the most recent counter value of N2–denoted as ${}^{x+4}CTR_{N2}$. Then, it sends the message Msg to $N2$. Finally, C increments $N2$'s counter value (which becomes ${}^{x+5}CTR_{N2}$) and updates its database.

Step C3 **N2** $\rightarrow C : \langle ID_C, {}^{x+3}DID_{N2}, (ns)\_K_{N2}, HMAC\rangle$. $N2$ performs as follows.

Upon the receipt of the message by $N2$, $N2$ computes *HMAC =h$({}^{x+4}CTR_{N2}, M)$ and verifies whether *HMAC is equal to the received HMAC. Next, $N2$ decrypts the received message using K_{N2} and extracts ID_{N1}, ${}^{x+1}DID_{N1}$, and TK_{12}.

Then, $N2$ generates a nonce ns that is used to uniquely identify the request and prove the freshness of the established connection between $N1$ and $N2$ later on. This nonce is a random number guaranteed not to be reused in the lifetime of the device generating it. After that, $N2$ encrypts ns using K_{N2} and concatenates the message parameters as $M=(ID_C \| {}^{x+3}DID_{N2} \| (ns)\_K_{N2})$. $N2$ increments ${}^{x+4}CTR_{N2}$ by 1 and (which becomes ${}^{x+5}CTR_{N2}$) and computes HMAC $=$h(${}^{x+5}CTR_{N2}$,M). Next, it sends the message Msg to C. Finally, $N2$ increments the counter value (which becomes ${}^{x+6}CTR_{N2}$) and updates its database.

Step C4 $\mathbf{C} \rightarrow N1 : \langle {}^{x+1}DID_{N1}, ID_C, ({}^{x+3}DID_{N2}, TK_{12}, ns)\_K_{N1}, HMAC \rangle$. $N1$ performs as follows.

When C receives the message from $N2$, it will look ${}^{x+3}DID_{N2}$ up. Next, it will compute *HMAC=h(${}^{x+5}CTR_{N2}$,M) and compare it with the one in the received message. If both are correct, C decrypts the received message using K_{N2} and extracts ns. Then, C increments $N2$'s counter value (which becomes ${}^{x+6}CTR_{N2}$) and updates its database.

Next, C will prepare a message to be sent to $N1$. It will first compute PoC=h($ID_{N2}, ID_{N1}, {}^{x+3}DID_{N2}, {}^{x+1}DID_{N1}, ns, TK_{12}, K_{N2}$), which is used to prove to $N2$ that C is aware of this request because PoC contains secret parameters, such as K_{N2}, which must be known only by C. Next, C loads ${}^{x+3}DID_{N2}$, TK_{12} and ns and encrypts them using K_{N1}. Then, it concatenates the message parameters as $M=({}^{x+1}DID_{N1} \| ID_C \| PoC \| ({}^{x+3}DID_{N2}, TK_{12}, ns)\_K_{N1})$ and computes HMAC=h(${}^{x+3}CTR_{N1}$,M). After that, it sends the message Msg to N1. Finally, C will store $N1$'s and $N2$,s counter values and their dynamic identities in its database.

Step C5 $\mathbf{N1} \rightarrow N2 : \langle {}^{x+3}DID_{N2}, {}^{x+1}DID_{N1}, PoC, (M)\_K_{12}, HMAC \rangle$. $N1$ performs as follows.

Upon receiving the message by $N1$, it will compute *HMAC=h(${}^{x+3}CTR_{N1}$,M) and verify whether *HMAC is equal to the received HMAC. If the verification is successful, $N1$ will decrypt the received message using K_{N1} and extract ${}^{X+3}DID_{N2}$, TK_{12} and ns. Otherwise, C will be rejected. After that, $N1$ will increment the counter value (which becomes ${}^{x+4}CTR_{N1}$) and update its database.

Next, $N1$ will prepare a message to be sent to $N2$. It will first encrypt the intended message using TK_{12} and then concatenate the message as $M=({}^{x+3}DID_{N2} \| {}^{x+1}DID_{N1} \| PoC \| (data)\_TK_{12})$ and compute HMAC=h(ns,M). Then, it sends the message Msg to $N2$. Finally, it will compute ${}^{x+2}DID_{N1}$= h(ID_{N1},${}^{x+4}CTR_{N1}$) for the upcoming session with C, and update its database.

Step C6 **N2**: $N2$ performs as follows.

Upon the receipt of the message, $N2$ computes *PoC = h(ID_{N2}, ID_{N1},${}^{x+3}DID_{N2}$,${}^{x+1}DID_{N1}, ns.TK_{12}, K_{N2}$) and *HMAC=h($ns$,$M$) and verifies whether *PoC and *HMAC are equal to the received PoC and HMAC, respectively. If the abovementioned parameters match, $N2$ will decrypt the received message using TK_{12} and accept data input. Finally, it will compute ${}^{x+4}DID_{N2}$= h(ID_{N2},${}^{x+6}CTR_{N2}$) for the upcoming session and update its database.

This marks the end of the data communication process.

Remark 1. . Notice that TK_{12} is a temporary key generated by the controller and is agreed on by the communicating parties. The temporary keys are utilized by IoT devices to establish direct communication connections among them.

Remark 2. As mentioned in Section 4.2, each IoT device is aware of the devices that it should communicate with. The permanent real identities facilitate the data communication between IoT devices, as shown in Fig. 7.

Remark 3. The key used in HMAC calculations (ns) during the communications among IoT devices is randomly generated by the receiving IoT device and is only valid for the particular communication session.

$$HMAC = h(ns, M)$$

5. Protocol efficiency evaluation

In this section, we analyze the overhead and efficiency of the proposed protocol in terms of its storage, communication overhead, and performance.

5.1. Storage requirements

In our scheme, each IoT node is required to store its real identity ID_N, dynamic identity DID_N, and secret key K_N, incremental counter CTR_N, and the controller identity ID_C. We use SHA-1 as an example of hash function, and the output of SHA-1 is 160 bits. By applying these settings, we obtain $|DID_N| = |CTR_N| =$160 bits, while $|ID_N| = |ID_C| =$ 16 bits, which is assumed short, and $K_N =$128 bits. On the other hand, C is required to store the tuple DID_N, ID_N, K_N CTR_N, in addition to ID_C. By applying these settings, we obtain $|DID_N| = |CTR_N| =$ 160 bits, while $|ID_N| = |ID_C| =$ 16 bits, and $K_N =$128 bits. Hence, the total storage required by each IoT node N is 480 bits, and the total storage required by controller C is $(n \times 464 + 16)$, where n is the number of registered IoT nodes. The storage requirements are summarized in Table 6.

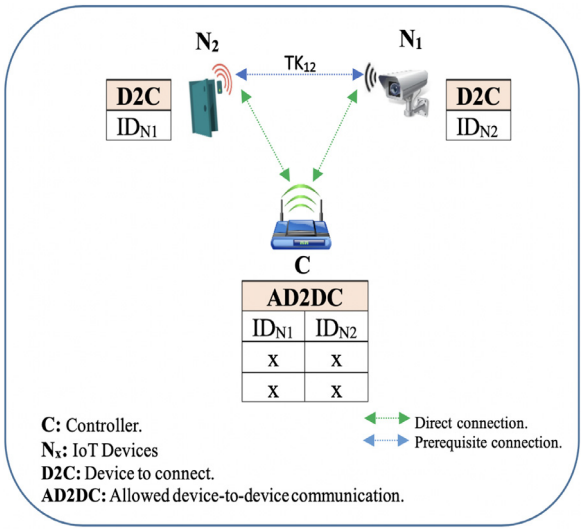

Fig. 7. Data communication through the controller.

Table 6
Storage cost of our scheme.

Node	Storage cost (in bits)
N	480
C	$464n + 16$

Table 7
Communication overheads of our scheme.

Communication between devices	Communication cost
$N \rightarrow C$	640 bits
$C \rightarrow N$	352 bits
$N1 \rightarrow N2$	640 bits

Table 8
Performance analysis of our scheme.

Phases Schemes	Ukil et al. [22]	Risalat et al. [16]	Lohachab [11]	Proposed scheme
Registration	N/A	N/A	$7C_H$	0
Authentication and data exchange $N \rightarrow C/S$	$2C_{XOR} + 4C_E + 4C_D$	$14C_{XOR} + 8C_H + 3C_E + 3C_D$	$5C_H + 3C_E + 3C_D$	$4C_H + 2C_E + 2C_D$
Authentication and data exchange $N \rightarrow N$	N/A	N/A	N/A	$4C_H + 3C_E + 3C_D$
Credential (ID and Secret key) change	N/A	N/A	$C_H + 2C_E + 2C_D$	C_H
Total (Maximum)	$2C_{XOR} + 4C_E + 4C_D$	$14C_{XOR} + 8C_H + 3C_E + 3C_D$	$13C_H + 5C_E + 5C_D$	$9C_H + 5C_E + 5C_D$

5.2. Communication overheads

In the transmission ($N \rightarrow C$), N sends the tuple, DID_N, ID_C, ID_N, K_N, CRT_N and HMAC. Therefore, the size of the transmitted tuple is $(3 \times 160)+(2 \times 16) + 128 = 640$ bits. In the transmission ($C \rightarrow N$), Csends the tuple, ID_C, DID_N, OTP_N and HMAC of size $((2 \times 16) + (2 \times 160)) = 352$ bits. The transmission ($N1 \rightarrow N2$), $N1$ sends the tuple, DID_N1, DID_N2, PoC and HMAC of size $(4 \times 160) = 640$ bits. The communication overheads are summarized in Table 7.

We separately present the communication overhead graph for our scheme in Fig. 8.

5.3. Performance analysis

In this section, we conduct the performance analysis of our proposed scheme. We specifically compute the total cost required by the mathematical operations of the scheme and the number of terms involved in the communication or messages in transit. We also present a comparison between our proposed scheme and other related schemes [11,16,22] in terms of the computational cost of the hash operations (C_H), symmetric key encryption operations (C_E), symmetric key decryption operations (C_D), XOR operations (C_{XOR}), and modular operations (C_{MOD}). Table 8 shows the comparison of the computational

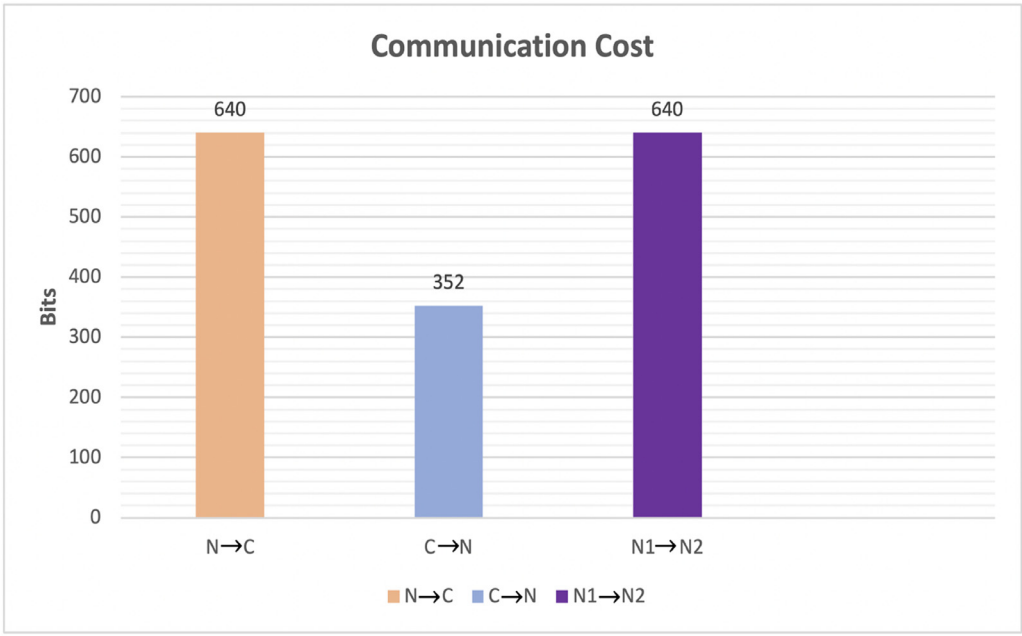

Fig. 8. Communication overheads graph for our scheme.

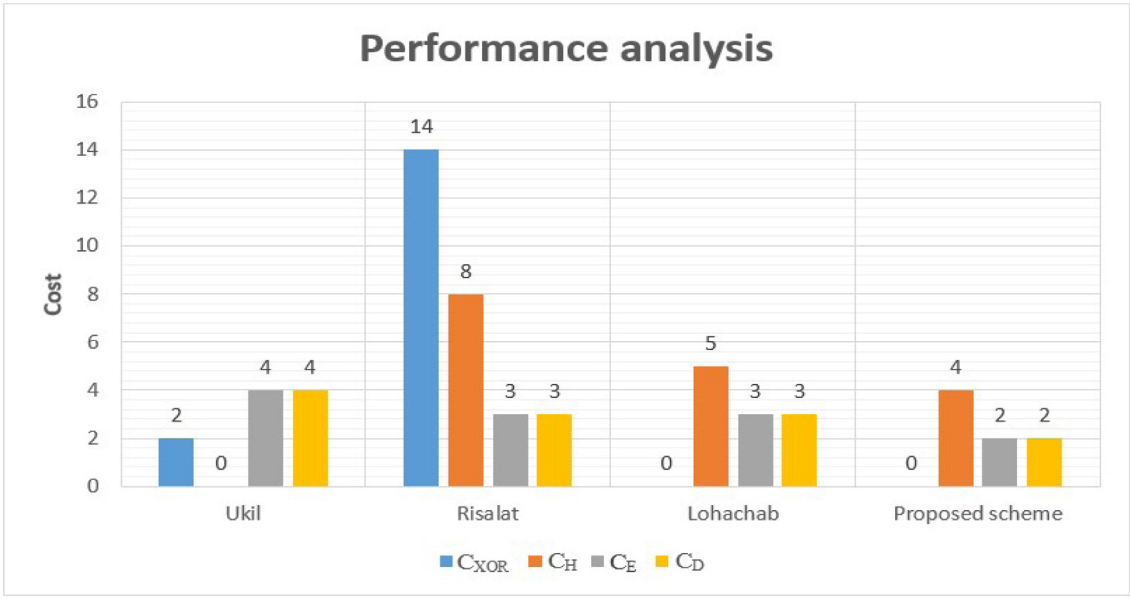

Fig. 9. Comparison graph for computation cost for Authentication and data exchange $N \leftrightarrow C/S$.

cost of our scheme to other related schemes. As shown in Table 8, some of the related schemes are missing some phases; thus, these are marked as N/A, which indicates they are not available.

From Table 8, it can clearly be seen that the number of hash operations in our scheme is less than those in [11] and the same as those in [16]. In addition, our scheme does not involve any exclusive-OR operations as in [16,22]. Furthermore, although our scheme consists of more phases, it involves almost the same total number of symmetric key encryption and decryption operations as the aforementioned related schemes. We also provide a comparison graph for the computational costs for authentication and data exchange $N \leftrightarrow C/S$, as shown in Fig. 9.

6. Protocol security validation

In this section, we validate the security of our proposed protocol using three different approaches: informal security analysis, formal validation using theorem proving based on BAN logic, and formal model checking and simulation using AVISPA.

6.1. Security analysis

In this section, the security of the protocol is explored against various known attacks. We discuss various known attacks and elaborate on how our protocol successfully resists and stops these attacks.

6.1.1. Replay attack

The replay attack is defeated using the incremental counter ^{X}CTR and nonce ns. The counter ^{X}CTR is generated by the IoT device in the first message in the authentication and then used in the HMAC that is sent in every transmitted message, ensuring the message cannot be replaced by the attacker. Both the controller and the IoT device maintain the counter value, increment it after every session, and keep it synchronized; ns is generated by the receiving IoT device, and it is different in each session and never used again. This ensures that the message cannot be replaced by the attacker. Furthermore, the attacker cannot find and reach the temporary shared secret key TK_{12} because she or he is totally unaware of the real identity of IoT device ID_N and the counter ^{X}CTR, which changes for every session. Consequently, the attacker does not know who the message is from or for; thus, any replay attack attempt will fail. Moreover, the HMAC prevents any replay attack attempts. Hence, our protocol protects against the replay attack.

6.1.2. Eavesdropping attack

In the authentication phase, the attacker can easily intercept the message in transit between C and $N1$. However, this kind of attack is prevented because we encrypt the message in transit using symmetric encryption. Moreover, the attacker cannot link the message to a particular device because we use dynamic identities DID_N that change in every session. Moreover, the information that is sent in clear text, such as DID_N and PoC, does not pose any threat because this information is constructed from random parameters that change in every session, such as CTR_N and ns.

6.1.3. Impersonation attack

In this attack, when the intruder eavesdrops the messages transmitted from N to C, she or he can use the intercepted information for malicious actions, such as impersonating the IoT device $N1$ and sending fabricated messages. This attack will not succeed for numerous reasons. First, the dynamic identity of the IoT device DID_N changes for every session. Second, HMAC is protected using the one-way hash function of the message, which includes CTR_N.Thus, our protocol protects against the impersonation attack.

6.1.4. Man-in-the-middle attack

As discussed above in the replay, eavesdropping, and impersonation attacks, the man-in-the-middle attack is defeated because the real identity of the IoT devices ID_N, the counter value CTR_N, the symmetric keys, the nonce ns, and the temporary shared key TK_{12} are unknown to the attacker. Moreover, the HMAC is protected using CTR_N. Therefore, the man-in-the-middle attack is prevented.

6.1.5. Attack against the temporary secret key

The temporary shared key TK_{12} is generated by controller C through random number generation and changes in every session.

6.1.6. Device stolen database attack

If any IoT device is hacked, the other devices will not be affected because the device does not store any secret keys of the other devices in its memory. In addition, these secret keys are different, thus overcoming the issue of having any secret key exposed; if an attacker discloses a secret key on one specific IoT device, the secret keys of the other IoT devices will not be compromised. In addition, the attacker is unable to obtain the temporary shared key because it is only used during a particular session, and then, it is destroyed. Moreover, the IoT device stores the real identities of the other IoT devices that it is allowed to communicate with to ask controller C to open sessions with them, but these identities are useless for the attacker because the attacker does not know how to use them to establish any session.

6.1.7. Forward/backward security

The objective of the forward/backward security property is to ensure that any past or future session keys will not be affected when any temporary session key TK_{12} is exposed. Recall that our temporary shared key is generated using a random number generator, and it changes for every session. Hence, forward/backward security is achieved by our authentication scheme.

6.1.8. Session key Guessing attack

The temporary shared key TK_{12} is generated by controller C using a random number generator. Because TK_{12} depends on randomness, the attacker cannot obtain it from the protocol. The probability of guessing this is so negligible that the attacker will fail to guess the temporary shared key given that TK_{12} changes for every session.

6.1.9. Anonymity unlinkability, and untraceability properties

Consider the tuple DID_{N1}, ID_C, ID_{N1}, K_{N1}, HMAC from N to C. We have $DID_N = \mathrm{h}(ID_N, {}^{X}CTR_N)$, and because ${}^{X}CTR_N$ is a fresh random number chosen by N in each session, the attacker cannot link any two different TID_Ns to the same N and cannot trace a given IoT device to N's messages. The sensitive data, such as ID_{N1}, will always be encrypted. The parameter HMAC is protected using a one-way hash function of the message and the incremental counter CTR_N, which is out of the adversary's reach. Now, consider the tuple ID_C, DID_N, OTP_N, HMAC from C to N; these parameters follow the same principle as discussed above for the tuple from N to C. DID_{N1} remains the same for each session, while CTR_N is fresh in each message; thus, HMAC is different. Therefore, the use of the random values DID_N, HMAC, ${}^{X}CTR_N$ and ns, as well as the hash functions, ensure that all messages transmitted by N or C via public channels are anonymous. Furthermore, for two or more authentication messages that are sent by the same device, the attacker cannot decide whether these authentication messages come from the same device or not; this means that device N cannot be linked to different sessions. Hence, our proposed scheme provides anonymity and unlinkability, and the attacker cannot trace the devices by intercepting messages.

6.2. Formal proof based on BAN logic

In 1989, the BAN logic was introduced by Burrows et al. It is a widely accepted model to describe and analyze authentication protocols. It has been widely employed to verify the protocols' security and provide proof of correctness of the authentication protocols [20]. Hence, we capitalize on it to formally prove that our authentication scheme achieves mutual authentication between an IoT device N and controller C.

We start by presenting a summarized introduction about the important symbols and the rules of BAN logic. Then, we will proceed with our formal proof.

6.2.1. BAN logic overview

Let N (client) and C (server) be participators, and let X and Y denote a parameter, formula or expression. We define the following notations:

- $N \mid \equiv X$: N believes the statement X.
- $\#(X)$: X is fresh.
- $N \mid \Rightarrow X$: N has jurisdiction over the statement X.
- $N \triangleleft X$: N sees the statement X.
- $N \mid \sim X$: N once said the statement X.
- (X, Y): X or Y is one part of the formula (X, Y).
- $\langle X \rangle_Y$: X combined with Y.
- $N \overset{K}{\leftrightarrow} C$: K is a secret parameter shared (or to be shared) between N and C.
- $N \overset{X}{=\!=} C$: X is a secret known only to N and C, and possibly to parties trusted by them.

Furthermore, the following commonly used BAN logic rules are utilized to prove that our authentication scheme ensures secure mutual authentication and key agreement, as follows:

- Message meaning rule: If N sees X encrypted with Y and if N believes Y is a secret key shared with C, then N believes C once said X.

$$\frac{N| \equiv N \overset{Y}{\leftrightarrow} C, N \triangleleft \langle X \rangle_Y}{N| \equiv C| \sim X}$$

- Nonce verification rule: If N believes X is fresh and N believes C once said X, then N believes C believes X.

$$\frac{N| \equiv \#(X), N| \equiv C| \sim X}{N| \equiv C| \equiv X}$$

- Jurisdiction rule: If N believes C has jurisdiction over X and N believes C believes X, then N believes X.

$$\frac{N| \equiv C| \Longrightarrow X, N| \equiv C| \equiv X}{N| \equiv X}$$

- Freshness conjuncatenation rule: if one part of a formula is fresh, then the entire formula must also be fresh; so if N believes X is fresh, then N believes X and Y are fresh.

$$\frac{N| \equiv \#(X)}{N| \equiv \#(X, Y)}$$

- Belief rule: If N believes X and Y, then N believes X.

$$\frac{N| \equiv (X, Y)}{N| \equiv X}$$

- Observation rule: If N sees X and Y, then N sees X.

$$\frac{N \triangleleft (X, Y)}{N \triangleleft X}$$

6.2.2. Goals of the analysis of our authentication scheme

In this section, the main goals of the analysis of our authentication scheme are defined as follows:

- G1: C believes N believes K_N is a secure shared parameter between N and C.

$$C| \equiv N| \equiv (N \overset{K_N}{\leftrightarrow} C)$$

- G2: C believes K_N is a secure shared parameter between N and C.

$$C| \equiv (N \overset{K_N}{\leftrightarrow} C)$$

- G3: N believes C believes OTP is a secure shared parameter between N and C.

$$N| \equiv C| \equiv (N \overset{OTP}{\leftrightarrow} C)$$

- G4: N believes OTP is a secure shared parameter between N and C.

$$N| \equiv (N \overset{OTP}{\leftrightarrow} C)$$

6.2.3. Messages transferred in the authentication

The idealized messages that are exchanged in the authentication phase between an IoT device N and controller C are as follows:

- M1: $\mathrm{N} \rightarrow \mathrm{C}: \langle \mathrm{N} \overset{K_N}{\leftrightarrow} \mathrm{C}, {}^{X}CTR, ID_N \rangle_{\mathrm{N} \overset{ID_N}{\leftrightarrow} \mathrm{C}}$
- M2: $\mathrm{C} \rightarrow \mathrm{N}: \langle \mathrm{N} \overset{K_N}{\leftrightarrow} \mathrm{C}, {}^{X+1}CTR, ID_N, \mathrm{N} \overset{OTP}{\leftrightarrow} \mathrm{C} \rangle_{\mathrm{N} \overset{ID_N}{\leftrightarrow} \mathrm{C}}$

6.2.4. Introductory assumptions

The fundamental assumptions of our authentication scheme are as follows:

- A1: C believes ID_N is a secure shared parameter between N and C.

$$C| \equiv (N \overset{ID_N}{\leftrightarrow} C)$$

- A2: C believes ${}^{X}CTR$ is fresh.

$$CRN| \equiv \#({}^{X}CTR)$$

- A3: C believes N believes ID_N is a secure shared parameter between N and C.

$$C| \equiv N| \equiv (N \overset{ID_N}{\leftrightarrow} C)$$

- A4: N believes ID_N is a secure shared parameter between N and CRN.

$$N| \equiv (N \overset{ID_N}{\leftrightarrow} C)$$

- A5: N believes ${}^{X+1}CTR$ is fresh.

$$N| \equiv \#({}^{X+1}CTR)$$

- A6: N believes C has jurisdiction over OTP, which is a secure shared parameter between N and C.

$$N| \equiv CRN| \Rightarrow (N \overset{OTP}{\leftrightarrow} C)$$

6.2.5. Analysis of our authentication scheme

The analysis of our authentication scheme is shown below to prove that the scheme achieves mutual authentication between N and C.

D1: From assumption A1 and message M1 and by applying the message meaning rule, we derive the following:

$$\frac{C|\equiv(N\overset{ID_N}{\longleftrightarrow}C), C\triangleleft\langle N\overset{K_N}{\leftrightarrow}C, {}^{X}CTR, ID_N\rangle_{N\overset{ID_N}{\longleftrightarrow}C}}{C|\equiv N\sim(N\overset{K_N}{\leftrightarrow}C, {}^{X}CTR, ID_N)}$$

D2: From assumption A2 and by applying the freshness rule, we derive the following:

$$\frac{C|\equiv\#({}^{X}CTR)}{C\|\equiv\#(N\overset{K_N}{\leftrightarrow}C^{X}, CTR, ID_N)}$$

D3: From derivations D1 and D2 and by applying the nonce verification rule, we derive the following:

$$\frac{C|\equiv\#(N\overset{K_N}{\leftrightarrow}C, {}^{X}CTR, ID_N), C|\equiv N|\sim(N\overset{K_N}{\leftrightarrow}C, {}^{X}CTR, ID_N)}{C|\equiv N\equiv(N\overset{K_N}{\leftrightarrow}C, {}^{X}CTR, ID_N)}$$

D4: From derivation D3 and by applying the belief rule, we derive the following:

$$\frac{C|\equiv N\equiv(N\overset{K_N}{\leftrightarrow}C, {}^{X}CTR, ID_N)}{C|\equiv N\equiv(N\overset{K_N}{\leftrightarrow}C)}\ (GoalG1)$$

D5: From assumption A3 and derivation D4 and by applying the jurisdiction rule, we derive the following:

$$\frac{C|\equiv N\Rightarrow(N\overset{K_N}{\leftrightarrow}C), C|\equiv N\equiv(N\overset{K_N}{\leftrightarrow}C)}{C|\equiv(N\overset{K_N}{\leftrightarrow}C)}\ (GoalG2)$$

D6: From assumption A4 and message M2 and by applying the message meaning rule, we derive the following:

$$\frac{N|\equiv(N\overset{ID_N}{\longleftrightarrow}C), C\triangleleft\langle {}^{X+1}CTR, ID_N, N\overset{OTP}{\longleftrightarrow}C\rangle_{N\overset{ID_N}{\longleftrightarrow}C}}{N|\equiv C|\sim({}^{X+1}CTR, ID_N, N\overset{OTP}{\longleftrightarrow}C)}$$

D7: From assumption A5 and by applying the freshness rule, we derive the following:

$$\frac{N|\equiv\#({}^{X+1}CTR)}{N|\equiv\#({}^{X+1}CTR, ID_N, N\overset{OTP}{\longleftrightarrow}C)}$$

D8: From derivations D6 and D7 and by applying the nonce verification rule, we derive the following:

$$\frac{N|\equiv C|\sim({}^{X+1}CTR, ID_N, N\overset{OTP}{\longleftrightarrow}C), N|\equiv\#({}^{X+1}CTR, ID_N, N\overset{OTP}{\longleftrightarrow}C)}{N|\equiv C|\equiv({}^{X+1}CTR, ID_N, N\overset{OTP}{\longleftrightarrow}C}$$

D9: From derivation D8 and by applying the belief rule, we derive the following:

$$\frac{N|\equiv C|\equiv({}^{X+1}CTR, ID_N, N\overset{OTP}{\longleftrightarrow}C)}{N|\equiv C|\equiv(N\overset{OTP}{\longleftrightarrow}C)}\ (GoalG3)$$

D10: From assumption A6 and derivation D9 and by applying the jurisdiction rule, we derive the following:

$$\frac{N|\equiv C\Rightarrow(N\overset{OTP}{\longleftrightarrow}C), N|\equiv C\equiv(N\overset{OTP}{\longleftrightarrow}C)}{N|\equiv(N\overset{OTP}{\longleftrightarrow}C)}\ (GoalG4)$$

Hence, our authentication scheme achieves mutual authentication and key agreement between N and C.

6.3. Simulation based on AVISPA tool

In this section, we show the details and results of the simulation of our protocol when using the AVISPA tool.

6.3.1. Preliminaries

AVISPA, which was introduced by Armando et al. [3], is a security-assessment toolkit based on the Dolev–Yao threat model [6]. The attacker in this model has the capacity to forward, modify, and change messages but cannot delay the computational strength of an algorithm. This toolkit is used to formally validate and assess Internet security protocols and applications. AVISPA is a widely accepted toolkit used in the research community and has been used to evaluate the specifications of several industrial-scale security protocols [23]. AVISPA uses a role-oriented protocol specification language. During the execution of a given protocol, each agent plays a unique role.

The security protocols are defined and described under the AVISPA toolkit using the high-level protocol specification language (HLPSL). HLPSL helps to define which security goals are achieved and is a simple and expressive language that relies on Lamport's work on the temporal logic of actions. HLPSL's main objective is to provide a way to check security properties such as data secrecy and the authentication of messages exchanges between agents. HLPSL describes the security properties in a distinct section called the goal section; the security protocol is verified, stating whether safe or not based on the predefined goals. HLPSL specifications are automatically translated into an equivalent lower language called the intermediate format (IF) language via the HLPSL2IF translator. Furthermore, AVISPA involves four integrated back-end model checkers, which are described as follows:

```
role role_N2 (N2, C,N1: agent,
Ka,Kb,Kd: public_key,IDN2,IDC,IDN1:text,
H:hash_func,
SND, RCV: channel (dy))
played_by N2 def=

local State : nat,
Ctrn2,Ctr1n2,Ctr2n2,OTPn2,Ctr3n2,Ns: text,
Kn2,TK12:symmetric_key,DIDN2,DIDN1:hash(text.text),Mn2:hash
(text.message.text.text.message),HMACn2,HMACn12:hash(message.text),Mc:hash
(message.text.text.text),HMACc:hash(message.text),M1c2:hash
(text.message.message),Mc2:hash(message.text.message),Mn12:hash
(message.message.message),PoC:hash
(text.text.message.message.text.message.message),M:message

init State := 0

transition

0. State = 0 /\ RCV(start) =|>
State'= 2 /\ Ctrn2' := new()/\Kn2':=new()/\DIDN2':=H(IDN2.Ctrn2')/\Mn2':=H
(IDC.DIDN2'.IDN2.Ctrn2'.Kn2') /\HMACn2':=H(Mn2'.Ctrn2')/\SND(IDC.DIDN2'.
{IDN2.Ctrn2'.Kn2'.HMACn2'}_Kb)
/\ secret(Ctrn2',ctrn2,{N2,C}) /\secret(IDN2,sec_idn2,{N2,C,N1})

2. State = 2 /\ RCV(DIDN2.IDC.{OTPn2'}_Kn2.HMACc') =|>
State'= 4 /\Ctr1n2':=new()/\Mc':=H(DIDN2.IDC.OTPn2'.Ctr1n2')/\HMACc':=H(Mc'.Ctr1n2')/
\request(C,N2,n2_c_ctr1n2,Ctr1n2)

4. State = 4 /\ RCV(DIDN2.IDC.{IDN1'.DIDN1'.TK12'}_Kn2.HMACc') =|> State'= 6 /\Mc2':=H
(DIDN2.IDC.{IDN1.DIDN1.TK12'}_Kn2)/\Ctr2n2':=new()/\HMACc':=H(Mc2'.Ctr2n2') /\Ns':=new
()/\secret(Ctr2n2,ctr2n2,{N2,C})/\Ctr3n2':=new()/\M1c2':=H(IDC.DIDN2.{Ns'}_Kn2)/
\HMACn2':=H(M1c2'.Ctr3n2')/\ SND(IDC.DIDN2.{Ns'}_Kn2.HMACn2') /\secret(Ns,ns,{N2,C,N1})/
\secret(TK12,tk12,{N2,C,N1})/\secret(Ctr3n2,ctr3n2,{N2,C})
6. State = 6 /\ RCV(DIDN2.DIDN1'.PoC'.{M'}_TK12.HMACn12') =|> State'= 8 /\ PoC':=H
(IDN2.IDN1.DIDN2.DIDN1.Ns.TK12.Kn2) /\Mn12':=H(DIDN1.DIDN2.PoC'.{M'}_TK12)/\HMACn12':=H
(Mn12'.Ns)/\request(N2,N1,n1_n2_poc,PoC)

end role
```

Fig. 10. Role_N2.

- On-the-fly model checker (OFMC): it explores the state space in a demand-driven way by incorporating various symbolic techniques and algebraic properties.
- Constraint-logic-based attack searcher (CL-AtSe) model checker: it translates any security protocol specification written as a transition relation in intermediate format (IF) language into a set of constraints that effectively helps in discovering the attacks on a given security protocol.
- SAT-based model checker (SATMC): this builds a propositional formula based on a transitional state acquired from the IF specification. The propositional formula describes any violation of the security properties that can be converted into an attack.
- Tree automata-based on automatic approximations for the analysis of security protocols (TA4SP) model checker: this indicates the vulnerability of a protocol or foresees the protocol correctness by using a correct estimation of the intruder's capabilities.

6.3.2. Simulation details

We begin by determining the security goals for the simulation. Our objective is to ensure the secrecy of a number of values, namely TK12, ns, sec_IDN2, CTRn1, CTRn2, and so forth. In addition, we want to be certain that *C* and *N* authenticate each other successfully. Next, we write the HLPSL script for our authentication scheme. We define five roles: (1) role_C, which is played by the controller, (2) role_N1, which is played by the IoT device *N*1, (3) role_N2, which is played by the IoT device *N*2, (4) session, which defines the session role and all its declarations, and (5) environment, which instantiates all agents, variables, and functions and describes the simulation's security goals. In the remainder of this section, we briefly describe each role.

Fig. 10 shows the specification for role_N2 that is played by*N*2. Device *N*2 is aware of all agents in the protocol (*N*2, *C*, and *N*1), the identities IDN1 and IDN2, which should be secured, the controller device identity (IDC), the public key of the controller Kb, the hash function H(.), and the send/receive channels Snd/Rcv. The (dy) notation indicates that the channels are following the Dolev–Yao model. At the first state "state 0," *N*2 receives a start message "Rcv(start)" as a signal to begin the protocol run. *N*2 creates and computes some authentication parameters to authenticate with *C*; it picks a fresh value as a nonce Ctrn2 to be used as an incremental counter that is shared with *C*. Next, it generates a symmetric key Kn2 that will be shared with C and used to secure the communication sessions. The computations for DIDN2, Mn2, and HMACn2 follow the description of our scheme. N2 sends (IDC.DIDN2'.{IDN2.Ctrn2'.Kn2'.HMACn2'}_Kb) to *C* after being encrypted with *C*'s public key, as described in our scheme. For the second transition "State 2," N2 receives the (DIDN2.IDC.{OTPn2'}_Kn2.HMACc')

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role role_C(N2,C,N1: agent,
Ka,Kb,Kd: public_key,IDN2,IDC,IDN1:text,
H:hash_func,
SND, RCV: channel (dy))
played_by C def=

local State : nat,
Ctrn2,Ctr1n2,OTPn2,Ctr2n2,Ctr3n2,Ns,
Ctrn1,Ctr1n1,OTPn1, Ctr2n1,Ctr3n1: text,
Kn2,Kn1,TK12:symmetric_key,DIDN2,DIDN1,PoB:hash(text.text),Mn2,Mn1:hash
(text.message.text.text.message),HMACn2,HMACn1:hash(message.text),Mc:hash
(message.text.text.text),HMACc:hash(message.text),Mc1:hash
(message.text.message),Mc2:hash(message.text.message),M1c2:hash
(text.message.message),M1n1:hash(text.message.message.message),M2n1:hash
(message.text.message.message),PoC:hash(text.text.message.message.text.message.message)

init State := 1

transition

1. State = 1 /\ RCV(IDC.DIDN2'.{IDN2.Ctrn2'.Kn2'.HMACn2'}_Kb) =|>
State':= 3 /\ DIDN2':=H(IDN2.Ctrn2') /\ Mn2':=H(IDC.DIDN2'.IDN2.Ctrn2'.Kn2')/\HMACn2':=H
(Mn2'.Ctrn2') /\Ctr1n2':= new()/\OTPn2':=new() /\ Mc':=H(DIDN2'.IDC.OTPn2'.Ctr1n2')/
\HMACc':=H(Mc'.Ctr1n2') /\SND(DIDN2.IDC.{OTPn2'}_Kn2.HMACc')/\secret(IDN2,sec_idn2,
{N2,C,N1})
/\ secret(Ctr1n2',ctr1n2,{N2,C})
/\ witness(C,N2,n2_c_ctr1n2,Ctr1n2')
3. State = 3 /\ RCV(IDC.DIDN1.{IDN1'.{Ctrn1'}_inv(Kd).Kn1'.HMACn1'}_Kb) =|> State':=5 /\
DIDN1':=H(IDN1'.Ctrn1') /\ Mn1':=H(IDC.DIDN1'.IDN1'.Ctrn1'.Kn1') /\ HMACn1':=H
(Mn1'.Ctrn1') /\secret(Ctrn1,ctrn1,{C,N1})/\ Ctr1n1' := new()/\OTPn1':=new()/\Mc1':=H
(DIDN1'.IDC.{OTPn1'}_Kn1') /\ HMACc':=H(Mc1'.Ctr1n1') /\ SND(DIDN1'.IDC.{OTPn1'}
_Kn1.HMACc')/\secret(Ctr1n1,ctr1n1,{C,N1})/\witness(C,N1,n1_c_ctr1n1,Ctr1n1)

5. State = 5 /\ RCV(IDC.DIDN1'.PoB'.{IDN2'}_Kn1.HMACn1')=|> State':=7 /\Ctr2n1':=new() /
\M1n1':=H(IDC.DIDN1'.PoB'.{IDN2'}_Kn1) /\HMACn1':=H(M1n1'.Ctr2n1')/\secret
(Ctr2n1,ctr2n1,{C,N1})/\request(N1,C,n1_c_pob,PoB)/\TK12':=new()/\Mc2':=H(DIDN2.IDC.
{IDN1.DIDN1.TK12'}_Kn2)/\Ctr2n2':=new()/\HMACc':=H(Mc2'.Ctr2n2')/\ SND(DIDN2.IDC.
{IDN1.DIDN1.TK12'}_Kn2.HMACc')/\secret(TK12,tk12,{N2,C,N1})/\secret(Ctr2n2,ctr2n2,{N2,C})

7. State = 7 /\ RCV(IDC.DIDN2'.{Ns'}_Kn2.HMACn2') =|> State':=9 /\ Ctr3n2':=new() /
\M1c2':=H(IDC.DIDN2'.{Ns'}_Kn2)/\HMACn2':=H(M1c2'.Ctr3n2')/\secret(Ctr3n2,ctr3n2,{N2,C})
/\PoC':=H(IDN2.IDN1.DIDN2.DIDN1.Ns.TK12.Kn2)/\M2n1':=H(DIDN1.IDC.PoC'.{DIDN2.TK12.Ns}
_Kn1)/\Ctr3n1':=new() /\ HMACc':=H(M2n1'.Ctr3n1') /\ SND(DIDN1.IDC.PoC'.{DIDN2.TK12.Ns}
_Kn1.HMACc')/\secret(Ns,ns,{N2,C,N1})/\secret(TK12,tk12,{N2,C,N1})
%%

end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Fig. 11. Role_C.

message from *C*. The computations of DIDN2 and HMACc' follow the description of our scheme. For this transition, if HMACc' appears as expected by *N*2, *N*2 accepts the message and continues processing the message. *N*2 then stores OTPn2' to be used in the first data communication message, as explained in our scheme. For the third transition "State 4," *N*2 receives the (DIDN2.IDC.{IDN1'.DIDN1'.TK12'}_Kn2.HMACc') message from *C*, which is about the communication session between *N*1 and *N*2. At this point, if HMACc' appears as expected by *N*2, *N*2 accepts the message and resumes processing the message. Next, *N*2 picks a fresh value as nonce *Ns* to be used as a secret key in the HMAC between *N*1 and *N*2. *N*2 stores TK12, DIDN1, and *Ns* for the *N*1 and *N*2 session, as explained above in the protocol definition. The computations for M1c2 and HMACn2 follow the description of our scheme. *N*2 sends (IDC.DIDN2.{Ns'}_Kn2.HMACn2' to *C* after being encrypted using Kn2. For the fourth transition "State 6," *N*2 receives the (DIDN2.DIDN1'.PoC'.{M'}_TK12.HMACn12') message from *N*1. Here, the validation of PoC and HMACn12 follow the description of our scheme. If these values are authentic, *N*2 decrypts the message and accepts the data. The "end role" at the end of the *N*2 role denotes the end of the role's definition.

Fig. 11 shows the specification for the role played by *C*. *C* knows all other agents in the protocol (*N*2, *C*, and *N*1), the identities of IDN1 and IDN2, which should be secured, the controller device's identity (IDC), the public key of the controller Kb, the hash function H(.), and the send/receive channels Snd/Rcv. The remaining part of the specification describes the different states of the protocol execution involving Role_C.

Fig. 12 shows the specification of the role played by *N*1. Device *N*1 is aware of all agents in the protocol (i.e., *N*2, *C*, and *N*1), the identities of IDN1 and IDN2, which should be secured, the controller device's identity (IDC), the public key of the controller Kb, the hash function H(.), and the send/receive channels Snd/Rcv. The (dy) notation means that the channels are following the Dolev–Yao model.

Fig. 13 shows the specification of the session role where all three agents' roles are invoked and where all the session's parameters are specified. First, all parameters and their declarations are presented. The predefined constants, which are Kb, IDN2, IDC, and IDN1, are included in this role. Second, each agent is assigned a send channel and a receive channel under the local section. Under the composition section, the role is invoked for each agent, along with its constant values and functions. The *N*2 role is invoked with *N*2, *C*, and *N*1 as agents; Ka, Kb, and Kd as public keys; IDN2, IDC, and IDN1 as predefined constants; *H* as the hash function; and SA and RA as the send and receive channels, respectively, for*N*2. Similarly, the remaining roles are invoked in the same way.

Fig. 14 shows the specification of the environment role. In this role, one or more sessions are instantiated. At the beginning, all constants are instantiated and defined. The constants n2, c, and n1 instantiate the three agents *N*2, *C*, and *N*1,

```
role role_N1 (N2,C, N1: agent,
Ka,Kb,Kd: public_key, IDN2,IDC,IDN1:text,
H:hash_func,
SND, RCV: channel (dy))
played_by N1 def=

local State : nat,
Ctrn1,Ctr1n1,Ctr2n1,Ctr3n1,OTPn1,Ns:text,Kn1,TK12,Kn2:symmetric_key,
DIDN1,DIDN2,PoB:hash(text.text),Mn1:hash
(text.message.text.text.message),HMACn1,HMACn12:hash(message.text),Mc1:hash
(message.text.message),HMACc:hash(message.text),M1n1:hash
(text.message.message.message),M2n1:hash(message.text.message.message),Mn12:hash
(message.message.message),PoC:hash(text.text.message.message.text.message.message),
M:message
init State := 30

transition

30. State = 30 /\ RCV(start) =|>
State':= 32 /\ Ctrn1' := new()/\DIDN1':=H(IDN1.Ctrn1')/\Kn1':= new()/\Mn1':=H
(IDC.DIDN1'.IDN1.Ctrn1'.Kn1') /\HMACn1':=H(Mn1'.Ctrn1') /\SND(IDC.DIDN1.{IDN1.{Ctrn1'}
_inv(Kd).Kn1'.HMACn1'}_Kb)/\secret(Ctrn1',ctrn1,{C,N1})

32. State = 32 /\ RCV(DIDN1.IDC.{OTPn1'}_Kn1.HMACc') =|> State':=34 /\Ctr1n1':=new()/
\Mc1':=H(DIDN1.IDC.{OTPn1'}_Kn1) /\ HMACc':=H(Mc1'.Ctr1n1') /\secret(Ctr1n1,ctr1n1,
{C,N1})/\request(N1,C,n1_c_ctr1n1,Ctr1n1)/\ Ctr2n1':=new() /\ PoB':=H(OTPn1'.Ctr2n1')/\
M1n1':=H(IDC.DIDN1.PoB'.{IDN2}_Kn1) /\HMACn1':=H(M1n1'.Ctr2n1') /\ SND(IDC.DIDN1.PoB'.
{IDN2}_Kn1.HMACn1')/\secret(IDN2,sec_idn2,{N2,C,N1})/\secret(Ctr2n1,ctr2n1,{C,N1})/\
witness(N1,C,n1_c_pob,PoB')

34. State = 34 /\ RCV(DIDN1.IDC.PoC'.{DIDN2'.TK12'.Ns'}_Kn1.HMACc') =|> State':=36 /\
M2n1':=H(DIDN1.IDC.PoC'.{DIDN2'.TK12'.Ns'}_Kn1)/\Ctr3n1':=new() /\ HMACc':=H
(M2n1'.Ctr3n1')/\secret(TK12,tk12,{N2,C,N1})/\M':=new()/\Mn12':=H(DIDN1.DIDN2.PoC.{M'}
_TK12)/\HMACn12':=H(Mn12'.Ns)/\ SND(DIDN2.DIDN1.PoC.{M'}_TK12.HMACn12')/\secret(Ns,ns,
{N2,C,N1})/\witness(N1,N2,n1_n2_poc,PoC')


end role
```

Fig. 12. Role_N1.

```
role session(N2,C,N1: agent,Ka,Kb,Kd: public_key,IDN2,IDC,IDN1:text, H:hash_func) def=

local SA, RA, SB, RB,SD, RD: channel (dy)

composition

role_N2(N2,C,N1,Ka,Kb,Kd,IDN2,IDC,IDN1,H,SA,RA)
/\ role_C(N2,C,N1,Ka,Kb,Kd,IDN2,IDC,IDN1,H,SB,RB)
/\ role_N1(N2,C,N1,Ka,Kb,Kd,IDN2,IDC,IDN1,H,SD,RD)


end role
```

Fig. 13. Role_session.

```
role environment() def=

const n2, c,n1 : agent,
ka,kb,kd, ki : public_key, idn2,idc,idn1:text,
h:hash_func,
ctrn2,ctr1n2,ns,sec_idn2,tk12,ctr2n2,ctr3n2,ctrn1,ctr1n1,ctr2n1,
n2_c_ctr1n2,n1_c_pob,n1_n2_poc,n1_c_ctr1n1: protocol_id

intruder_knowledge = {n2, c,n1,ka,kb,kd, ki, inv(ki)}

composition

session(n2,c,n1,ka,kb,kd,idn2,idc,idn1,h)

end role
```

Fig. 14. Role_environment.

```
goal
secrecy_of ctrn2
secrecy_of ctr1n2
secrecy_of ns
secrecy_of sec_idn2
secrecy_of tk12
secrecy_of ctr2n2
secrecy_of ctr3n2
secrecy_of ctrn1
secrecy_of ctr1n1
secrecy_of ctr2n1
authentication_on n2_c_ctr1n2
authentication_on n1_c_pob
authentication_on n1_n2_poc
authentication_on n1_c_ctr1n1

end goal
```

Fig. 15. Goal.

```
environment()
```

Fig. 16. Environment.

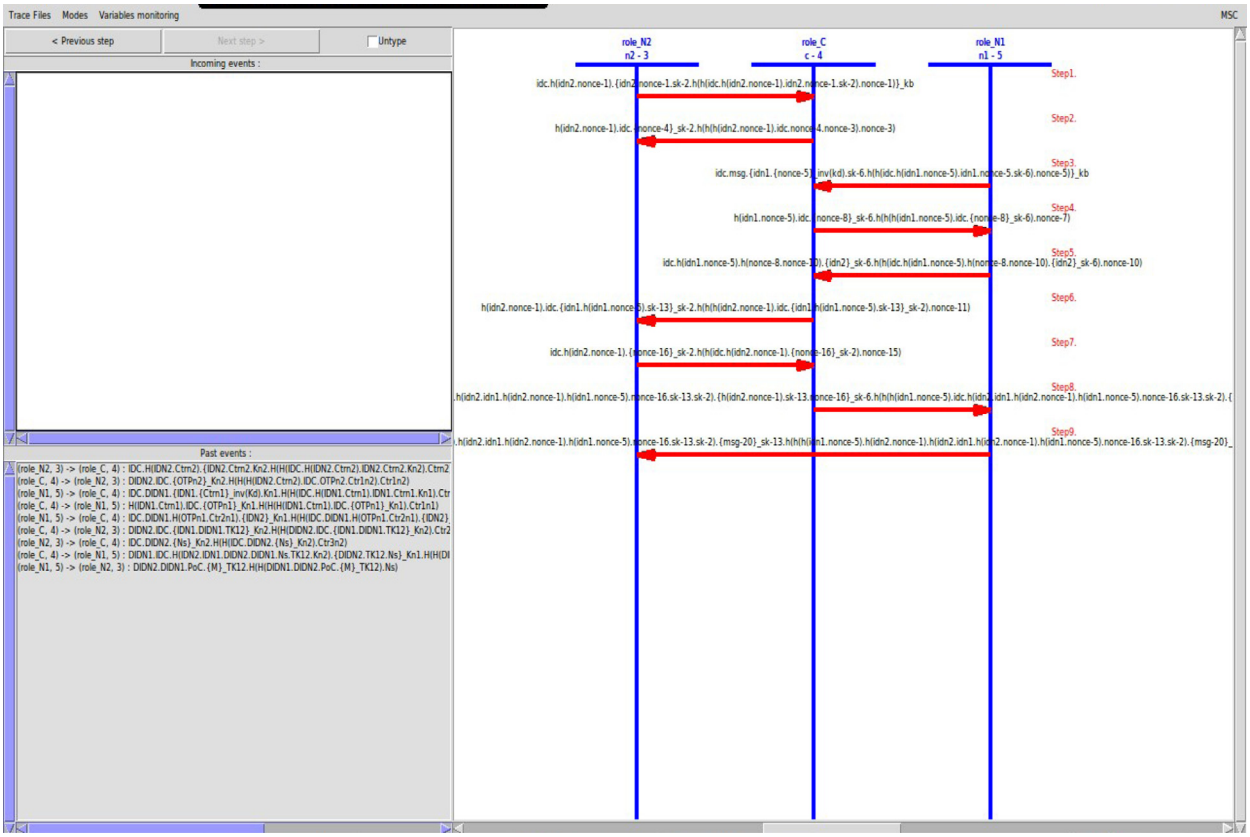

Fig. 17. Snapshot of the protocol simulation in AVISPA.

respectively. Here, ka, kb, and kd instantiate the Ka, Kb, and Kd keys, respectively. Also, idn2, idc, and idn1 instantiate IDN2, IDC, and IDN1, respectively. Next, h instantiates the hash function H. The protocol identifiers–ctrn2, ctr1n2, ns, sec_idn2, tk12, ctr2n2, ctr3n2, ctrn1, ctr1n1, ctr2n1, n2__ctr1n2, n1_c_pob, n1_n2_poc, n1_c_ctr1n–are instantiated and defined as well. In the intruder knowledge section, all the relevant parameters that the intruder is assumed to know are provided prior to the execution. We assume that the intruder knows n2, c, and n1, as well as the hash h. In the composition section, the session is instantiated with the instances (n2,c,n1,ka,kb,kd,idn2,idc,idn1,h).

Fig. 15 shows the simulation goals that are defined under the “goal” keyword using the protocol identifiers declared as “protocol_id.” The simulator is told to check the secrecy of the following parameters: ctrn2, ctr1n2, ns, sec_idn2, tk12,

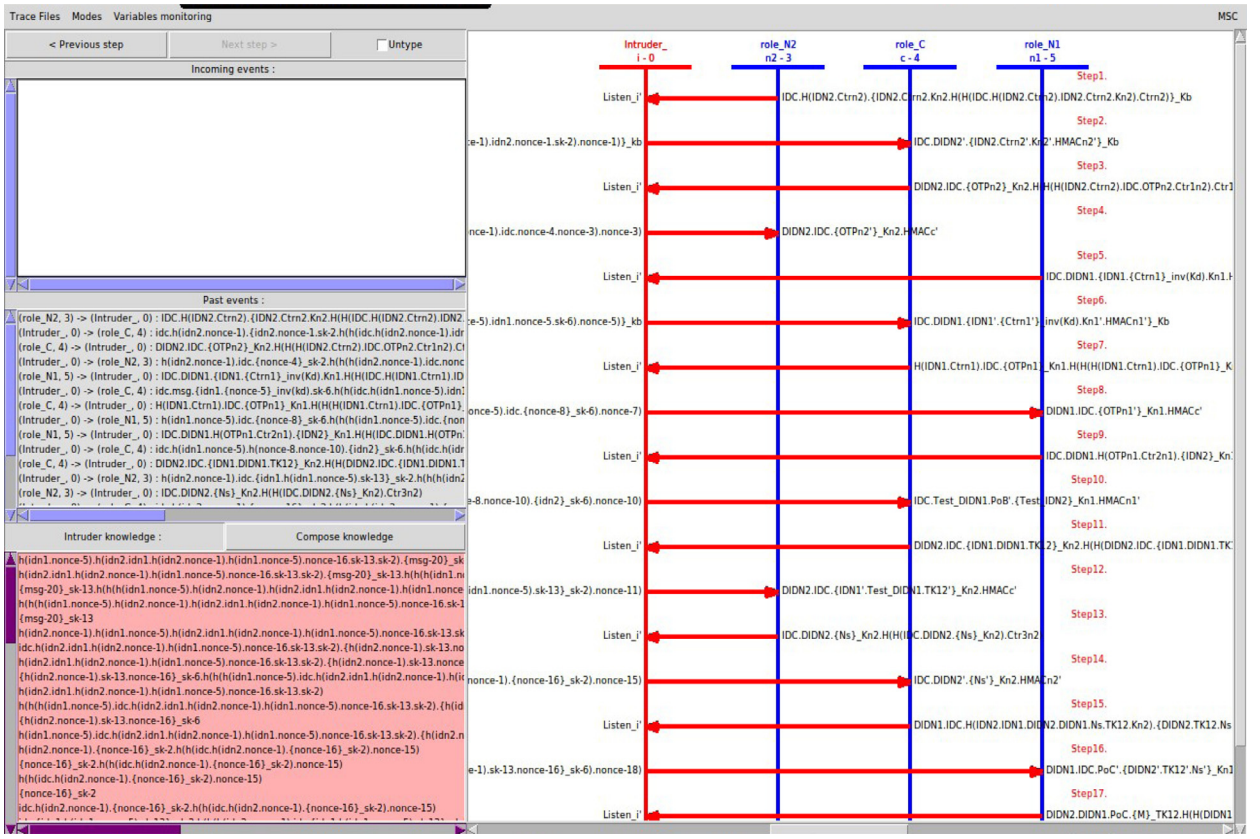

Fig. 18. Snapshot of the protocol full execution using SPAN animator under the intruder's control.

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /home/span/span/testsuite/results/ZB-v-4-11-5.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 6 states
  Reachable  : 2 states
  Translation: 0.13 seconds
  Computation: 0.00 seconds
```

Fig. 19. CL-AtSe summary report.

ctr2n2, ctr3n2, ctrn1, ctr1n1, and ctr2n1. The authentication is performed using "n2_c_ctr1n2," "n1_c_pob," "n1_n2_poc," and "n1_c_ctr1n."

Fig. 16 finally demonstrates the call of the main role "environment".

6.3.3. Simulation results

In this section, the simulation results of our proposed protocol are presented. The results are based on the two widely accepted AVISPA back-end model checkers: OFMC and CL-AtSe. The security protocol animator (SPAN) is employed to

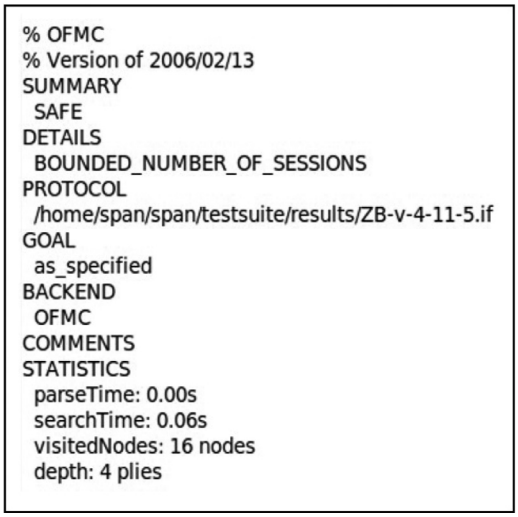

Fig. 20. OFMC summary report.

interactively construct a message sequence chart (MSC) of the protocol execution from the HLPSL specification outlined earlier. Moreover, SPAN automatically initiates attacks on the HLPSL specification using the Dolev-Yao intruder model. The SPAN protocol simulation's MSC corresponding to our HLPSL specification is shown in Fig. 17.

Fig. 18 demonstrates the protocol execution using SPAN animator software [3]. It shows the full execution of our protocol under the intruder's control.

In the AVISPA tool, the security properties, such as authentication, integrity, and secrecy, are defined in a separate section. When SPAN is executed, it verifies if the protocol satisfies the specified properties. SPAN will generate the attack trace if any attack is found, and it will consider the protocol unsafe. Fig. 19 presents the CL-AtSe back-end checker report that assures that our scheme is safe and that is satisfies all the specified security goals.

Fig. 20 shows the OFMC back-end checker report, which states that our scheme is safe, thus meeting the specified security goals.

The other two model checkers-SATMC and TA4SP-reported "NOT SUPPORTED" and produced "INCONCLUSIVE" results. Our protocol includes XOR operations that are not supported by the TA4SP back-end checker.

7. Conclusion

In the current paper, we proposed a mutual authentication scheme for IoT systems that provides confidentiality, integrity, anonymity, unlinkability, and untraceability properties while achieving mutual authentication between IoT devices and controller devices. The security of the proposed scheme was formally verified using the BAN logic. In addition, the security of our scheme was evaluated using the AVISPA toolset, and the results show that our scheme is safe. In addition, we conducted an informal security analysis of our scheme, showing that our scheme resists common attacks. Furthermore, we evaluated the efficiency of the scheme and compared it with other related schemes.

Our plans for future work include the following:

- We will extend the protocol to include a smart access control mechanism so that device access can be restricted within allowable domains to stop any security breaches from occurring in the case where one device is compromised. This will strengthen the safeguarding of smart home devices' sensitive information.
- We will address or mitigate the impact of the key constraints of the proposed scheme, which is its reliance on a too-powerful controller, where the controller knows all the devices' sensitive data. Under these settings, as soon as the controller is compromised, an intruder can gain knowledge of all sensitive data and then can perform any malicious actions.
- We will implement the proposed scheme using OMNeT++, which is designed to simulate and build computer networks and protocols. Then, we will interface the implementation with an attack platform such as Kali Linux to execute various live attacks against the system and evaluate the efficiency under legitimate and attack loads.

Declaration of competing interest

The authors declare that they do not have any conflict of interests.

References

[1] M. Abomhara, G.M. Køien, Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks, J. Cyber Secur. 4 (1) (2015) 65–88.
[2] M. Alshahrani, I. Traore, Secure mutual authentication and automated access control for IoT smart home using cumulative keyed-hash chain, J. Inf. Secur. Appl. 45 (2019) 156–175.
[3] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, S. Mödersheim, The AVISPA tool for the automated validation of internet security protocols and applications, in: Proceedings of the International conference on computer aided verification, Springer, Berlin, Heidelberg, 2005, pp. 281–285.
[4] Y. Ashibani, D. Kauling, Q.H. Mahmoud, A context-aware authentication framework for smart homes, in: Proceedings of the 2017 IEEE Thirtieth Canadian Conference on Electrical and Computer Engineering (CCECE), IEEE, 2017. Pp. 1–5
[5] F. Chu, R. Zhang, R. Ni, W. Dai, An improved identity authentication scheme for internet of things in heterogeneous networking environments, in: Proceedings of the 2013 Sixteenth International Conference on Network-Based Information Systems, IEEE, 2013. Pp. 589–593
[6] D. Dolev, A. Yao, On the security of public key protocols, IEEE Trans. Inf. Theory 29 (2) (1983) 198–208.
[7] P.P. Gaikwad, J.P. Gabhane, S.S. Golait, 3-level secure Kerberos authentication for smart home systems using IoT, in: Proceedings of the 2015 First International Conference on Next Generation Computing Technologies (NGCT), IEEE, 2015, pp. 262–268.
[8] K. Gill, S.H. Yang, F. Yao, X. Lu, A Zigbee-based home automation system, IEEE Trans. Consum. Electron. 55 (2) (2009).
[9] N. Kshetri, Can blockchain strengthen the internet of things? IT Profess. 19 (4) (2017) 68–72.
[10] J.Y. Lee, W.C. Lin, Y.H. Huang, A lightweight authentication protocol for internet of things, in: Proceedings of the 2014 International Symposium on Next-Generation Electronics (ISNE), IEEE, 2014, pp. 1–2.
[11] A. Lohachab, ECC based inter-device authentication and authorization scheme using MQTT for IoT networks, J. Inf. Secur. Appl. 46 (2019) 1–12.
[12] D. Mishra, P. Vijayakumar, V. Sureshkumar, R. Amin, S.H. Islam, P. Gope, Efficient authentication protocol for secure multimedia communications in IoT-enabled wireless sensor networks, Multimed. Tools Appl. 77 (14) (2018) 18295–18325.
[13] L. Pascu, in: The IoT threat landscape and top smart home vulnerabilities in 2018, 2018. Retrieved from https://www.bitdefender.com/files/News/CaseStudies/study/229/Bitdefender-Whitepaper-The-IoT-Threat-Landscape-and-Top-Smart-Home-Vulnerabilities-in-2018.pdf
[14] A. Pfitzmann, M. Köhntopp, Anonymity, unobservability, and pseudonymity – a proposal for terminology, in: Designing Privacy Enhancing Technologies, Springer, Berlin, Heidelberg, 2001, pp. 1–9.
[15] M.A. Razzaq, S.H. Gill, M.A. Qureshi, S. Ullah, Security issues in the internet of things (IoT): a comprehensive study, Int. J. Adva. Comput. Sci. and Appl. (IJACSA) 8 (6) (2017) 383–388.
[16] N.A.M. Risalat, M.T. Hasan, M.S. Hossain, M.M. Rahman, Advanced real time RFID mutual authentication protocol using dynamically updated secret value through encryption and decryption process, in: Proceedings of the 2017 International Conference on Electrical, Computer and Communication Engineering (ECCE), IEEE, 2017, pp. pp.788–793.
[17] F.K. Santoso, N.C. Vun, Securing IoT for smart home system, in: Proceedings of the 2015 IEEE International Symposium on Consumer Electronics (ISCE), IEEE, 2015, pp. 1–2.
[18] N. Saputro, A.I. Yurekli, K. Akkaya, S. Uluagac, Privacy preservation for IoT used in smart buildings, in: F. Hu (Ed.), Security and Privacy in Internet of Things (IoTs): Models, Algorithms, and Implementations. Chapter: 7, CRC Press.
[19] A. Tomar, Introduction to Zigbee Technology, Global Technology Centre, 2011, pp. 1–24.
[20] J. Wen, M. Zhang, X. Li, The study on the application of BAN logic in formal analysis of authentication protocols, in: Proceedings of the Seventh International Conference on Electronic Commerce, ACM, 2005, pp. 744–747.
[21] J. Wurm, K. Hoang, O. Arias, A.R. Sadeghi, Y. Jin, Security analysis on consumer and industrial IoT devices, in: Proceedings of the 2016 Twenty-first Asia and South Pacific Design Automation Conference (ASP-DAC), IEEE, 2016, pp. 519–524.
[22] A. Ukil, S. Bandyopadhyay, A. Bhattacharyya, A. Pal, Lightweight security scheme for vehicle tracking system using coAP, in: Proceedings of the International Workshop on Adaptive Security, ACM, 2013, p. 3.
[23] L. Viganò, Automated security protocol analysis with the AVISPA tool, Electron. Notes Theor. Comput. Sci. 155 (2006) 61–86.
[24] J. Voris, N. Saxena, T. Halevi, Accelerometers and randomness: perfect together, in: Proceedings of the Fourth ACM Conference on Wireless Network Security, ACM, 2011, pp. 115–126.
[25] G. Zhao, X. Si, J. Wang, X. Long, T. Hu, A novel mutual authentication scheme for internet of things, in: Proceedings of the 2011 International Conference on Modelling, Identification and Control (ICMIC), IEEE, 2011, pp. 563–566.
[26] Z. Alliance, in: Zigbee Specification, 2008. San Ramon, United States. ZigBee Document 053474r17
[27] H. Yüzugüzel, J. Niemi, S. Kiranyaz, M. Gabbouj, T. Heinz, Shakeme: key generation from shared motion, in: Proceedings of the 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, IEEE, 2015, pp. 2130–2133.