Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier xx.xxxx/ACCESS.2019.DOI

Resource Efficient Authentication and Session Key Establishment Procedure for Low-Resource IoT Devices

SARMADULLAH KHAN[1], AHMED IBRAHIM ALZAHRANI[2], OSAMA ALFARRAJ[2], NASSER ALALWAN[2] , ALI H. AL-BAYATTI[1],

[1]School of Computer Science and Informatics, De Montfort University, Leicester, United Kingdom (e-mail: {sarmadullah.khan,alihmohd}@dmu.ac.uk)

[2]Department of Computer Science, Community College, King Saud university, Riyadh, 11437, Saudi Arabia (e-mail:{ahmed,oalfarraj,nalalwan}@ksu.edu.sa)

Corresponding author: Osama Alfarraj(e-mail: oalfarraj@ksu.edu.sa).

ABSTRACT The Internet of Things (IoT) can includes many resource-constrained devices, with most usually needing to securely communicate with their network managers, which are more resource-rich devices in the IoT network. We propose a resource-efficient security scheme that includes authentication of devices with their network managers, authentication between devices on different networks, and an attack-resilient key establishment procedure. Using automated validation with internet security protocols and applications tool-set, we analyse several attack scenarios to determine the security soundness of the proposed solution, and then we evaluate its performance analytically and experimentally. The performance analysis shows that the proposed solution occupies little memory and consumes low energy during the authentication and key generation processes respectively. Moreover, it protects the network from well-known attacks (man-in-the-middle attacks, replay attacks, impersonation attacks, key compromission attacks and denial of service attacks).

INDEX TERMS Internet of things, wireless sensor networks, resource-constrained device, authentication, encryption, key establishment, security, IoT deployment, IoT scalability.

I. INTRODUCTION

The Internet of Things (IoT) consists of billion of devices ranging from the very resource constrained (e.g., sensors) to the more resource-rich (e.g., PCs) that are capable of sensing, communicating, computing, and possibly actuating. Sensors and resource-constrained-devices play an important role in automating systems due to their low cost. For example, a sensor is one physical entity of an IoT network that has the ability to sense, gather surrounding data, and exchange data with other devices within the IoT network.

Based on the nature of IoT network and its compatibility in various applications like an indoor monitoring and localization of people, control and management of heating and water systems, transportation systems [1], [2], issuing disease warnings in the smart cities, monitoring and controlling home appliances from remote locations [3], [4], monitoring health condition of persons in body area networks, analysing relevant information using deep learning techniques [5], [6], [7] and conveying all the relevant information to the physicians [8] and/or central servers [9], the security an privacy of data in such applications is very challenging [10], [11].

It is therefore necessary to assess the authenticity and integrity of the transferred information, and improve the resilience of the resource-constrained devices that enable most IoT systems to various cyber attacks. Potential attackers of IoT systems can take over the sensors either physically (due to their open deployment in public environments) or virtually (by gaining remote access to them). Attackers can manipulate the information to misguide the decision taking authority and damage the system operation [12], or can take over a server or connect fake servers to attract network traffic. In such scenarios, it is very important to achieve mutual authentication of both sensors and servers before the start of actual data exchange. The authentication procedure must be lightweight and resistant to attacks (e.g., man in the middle).

Many lightweight authentication and key establishment mechanisms based on the Elliptic Curve Cryptography (ECC) have been introduced in the literature [4], [8], [12] and an intrusion detection system [13] for IoT enabled v2x network to detect any abnormality. However, they need some central certification authorities to manage keys and have no mechanism to cope with man in the middle attacks, impersonation attacks and Sybil attacks.

A. MAIN CONTRIBUTIONS

The main contributions of this article are:

- a mutual authentication mechanism based on the ECC approach that is capable of protecting the system from the well-known types of cyber attacks;
- reducing dependency on a central certification authority to establish secret keys among devices for each session in the IoT network, which is needed by typical PKI approaches;
- a memory- and processing-efficient secure communication procedure that scales well from resource-constrained devices to very large heterogeneous networks;
- a resource-efficient on-line symmetric key establishment procedure to secure session communications, which is suitable for resource-constrained IoT devices.

The rest of this article is organized as follows. Section II discusses related work. Section III presents the threat model and solution preliminaries. Section IV presents the network model and the proposed solution for authentication and mutual key establishment. Section V presents the security analysis of the proposed solution against well known attacks. The experimental results are discussed in Section VI, and Section VII concludes the article.

II. RELATED WORK

Security of resource constrained embedded devices (e.g., sensors) is a major challenge, especially when used in the IoT context due to their numerous real world applications to automate monitoring and control operations [14]. Generally, IoT networks consist of heterogeneous devices that support different technologies and embedded platforms. This makes communication security even more challenging.

Cryptography and key management are among the main building blocks of communication security. In typical security approaches over the Internet, clients authenticate servers using their digital signatures, while servers authenticate clients using their usernames and associated passwords. These approaches are not suitable for most IoT devices, which lack keyboards and screens, and are even more challenging for resource-con- strained devices (for example, sensors or smart IoT objects). Widely used public key cryptography techniques (e.g., RSA, Diffie–Hellman) are ill-suited because they require significant memory and processing resources, while low-cost embedded devices (sensors) have typically very limited memory, processing, and energy resources. In comparison, public key cryptography based on elliptic curves is less demanding, hence better suited for such devices.

ECC, first proposed by Miller [15], can efficiently secure communications of resource-constrained devices. For instance, NIST Elliptic Curve Digital Signature Algorithm (ECDSA) is a widely used digital signature scheme based on ECC. Liu introduced the TinyECC library [16] for MSP430 and AVR microcontroller families, used by various projects to secure the communication architecture. Other works use standard ECC implementations as well [17]. Wang used the popular Texas Instruments 16-bit MSP430 microcontroller series to implement a software-based ECC over a 160-bit prime field and observed that the execution required 25 million cycles for fixed-base scalar multiplication [18]. Sankar [19] proposed an Elliptic Curve Diffie Hellman (ECDH) based key exchange mechanism where each device of the communication pair multiplies its private key and the other device public key and the result is a symmetric key. In this approach, symmetric key always remains same irrespective of the session and time because the public/private key pair of each device does not change. There is no mechanism to update the symmetric key if it gets compromised.

TABLE 1. Notations used in the proposed approach of authentication and key establishment

Parameter	Size (Bits)	Definition
SK	160	Secret (Private) Key
PK	160	Public Key
a, d, l, r, x	16	Additive factors
C_1, C_2	160	Cipher Texts
TTP	16	Trusted server
D	16	Device
NM	16	Network Manager
OTST	160	One Time Secret Token
e_1, e_2	160	Point on elliptic curve
SID	16	Session ID
ST	16	Secret Token
$H(.)$	160	collision resistant function

Other research addressed the optimization of the most energy-intensive set of arithmetic operations in ECC to reduce its impact on the limited resources of small IoT devices [20]–[23]. For instance, Gura uses an efficient hybrid multiplication technique based on a combination of operand and product scanning methods implemented on 8-bit AVR microprocessors [24].

III. THREAT MODEL AND SOLUTION PRELIMINARIES

The preliminaries include an overview of the threat model and various stages involved in the proposed solution. The notations used in this paper along with the proposed number of bits for various security-related parameters are listed in TABLE 1.

A. THREAT MODEL

In the proposed threat model, we have a probabilistic polynomial time attacker A who can access the communication channel, intercept messages, modify messages and/or inject fake messages. It is assumed that attacker can access the device memory using attack [25] and can create multiple instances of a device, $D = \{D_1, D_2, ..., D_l\}$.

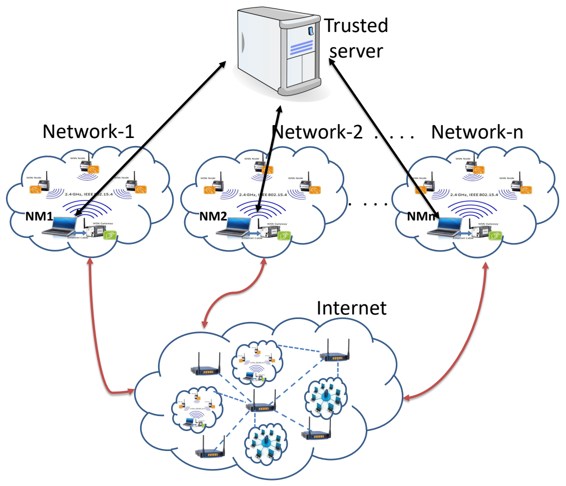


FIGURE 1. Network architecture of the proposed solution and its components

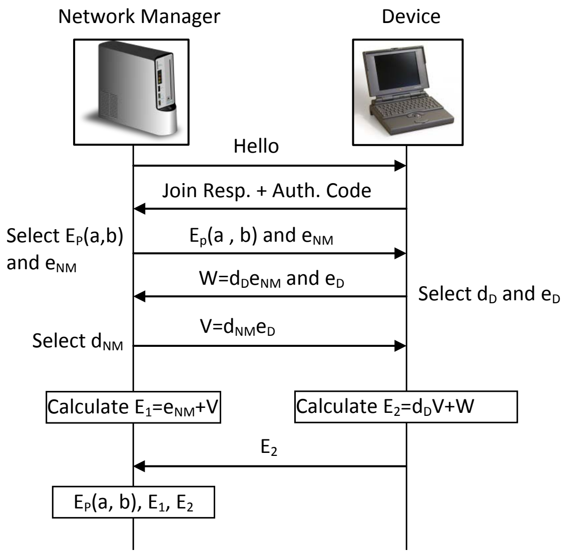


FIGURE 2. Authentication procedure

B. SOLUTION OVERVIEW

The proposed solution consists of the following phases, 1) network setup, 2) device initial authentication and registration, 3) authentication and key establishment, and 4) key freshness and expiration. In the network setup phase, the network manager (NM) is responsible for generating and assigning all the necessary parameters which are used during the initial authentication and registration phase, immediately after deploying the network. During the registration phase, each device generates secret parameters and securely shares them with its NM which are then used during the mutual authentication and key establishment phases with other devices. The proposed solution uses ECC-based point multiplication operations and hash functions. It also uses a session based timestamps mechanism to ensure key freshness and protects network from the replay attacks.

IV. THE PROPOSED SOLUTION

The proposed network architecture and various phases of our proposed solution are discussed in the following subsections.

A. NETWORK ARCHITECTURE

The architecture of the proposed network is shown in FIGURE 1. In the proposed network, we have one or more instances of (1) a device that utilizes the services of (2) network manager that provides services to devices and that can be either a resource constrained device or resourceful device, (3) a trusted server that provides trust certificates to the NMs.

B. SETUP PHASE

The setup phase consists of the following steps:

Step-1: Each device is given a non-singular elliptic curve $E_P(a, b) : y^2 = x^3 + ax + b \pmod P$ over a finite field Z_P, where a and b are constants and satisfy the condition $4a^3 + 27b^2 \neq 0 \pmod P$ and $Z_P = \{0, 1, 2, ..., P - 1\}$

Step-2: Each device selects a group generator G of order n such that $n.G = \vartheta$ where ϑ is the point at zero or infinity and a collision resistant function $H(.)$.

Step-3: Each device of the network is equipped with OTST along with its hash by the network administrator, an ST, a list of other devices OTSTs and a public key of the network administrator.

Step-4: Each device selects randomly e_1 over $E_P(a, b)$ and generates randomly l as its own SK to calculate $e_2 = le_1$. The device announces $(E_P(a, b), e_1, e_2)$ as its PK.

C. REGISTRATION PHASE

Step-1: Once the network is deployed by the network administrator, the network manager starts broadcasting *Hello* messages to devices in its vicinity. Each *Hello* message includes OTST and its hash. The hash is used to check authenticity and integrity of the OTST as attacker can modify it during its transmission.

Step-2: Each device decrypts *Hello* messages that it receives from the NM using the NM's public key, then checks the integrity of the OTST using the hash as follows:

- the device generates the hash of the decrypted OTST and
- compares it with the hash extracted from the message, which was generated by the NM.

Upon successful verification of the OTST, both the end devices register each other. It is important to mention here that the OTST is used only once, for device registration after network deployment. Then the OTST is permanently deleted to prevent replay attacks.

Step-3: After the initial authentication and registration, the devices and the NM proceed with the creation of their authentication materials as follows:

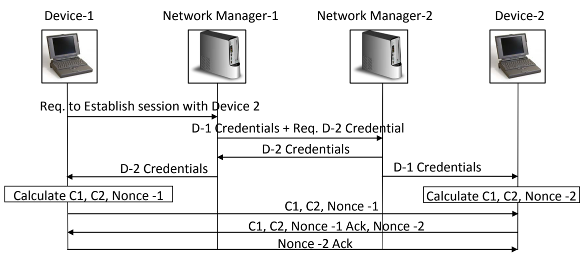

Device-1
Network Manager-1
Network Manager-2
Device-2
Req. to Establish session with Device 2
D-1 Credentials + Req. D-2 Credential
D-2 Credentials
D-2 Credentials
D-1 Credentials
Calculate C1, C2, Nonce -1
Calculate C1, C2, Nonce -2
C1, C2, Nonce -1
C1, C2, Nonce -1 Ack, Nonce -2
Nonce -2 Ack

FIGURE 3. Authentication process between the two devices of different networks through their network managers and a network coordinator

1) the NM selects $E_{\mathrm{P}}(a,b)$ and e_{NM};
2) the NM sends $E_{\mathrm{P}}(a,b)$ and e_{NM} to D;
3) the D selects a random number d_{D} and e_{D} over $E_{\mathrm{P}}(a,b)$ that it has received from the NM;
4) the D calculates W as follows:

$$W = d_{\mathrm{D}} e_{\mathrm{NM}} \tag{1}$$

5) the D sends e_{D} to the NM;
6) the NM selects a device-specific d_{NM};
7) the NM calculates V as follows:

$$V = d_{\mathrm{NM}} e_{\mathrm{D}} \tag{2}$$

and sends V to the D;
8) the D calculates

$$E_2 = d_{\mathrm{D}} V + W \tag{3}$$

and sends E_2 to the NM;
9) the NM calculates E_1 as follows:

$$E_1 = e_{\mathrm{NM}} + V \tag{4}$$

10) the NM records $E_{\mathrm{P}}(a,b)$, E_1 and E_2 as D's authentication materials.

These pre-shared authentication materials avoid the repetition of these calculations each time a communication session needs to be established.

D. AUTHENTICATION AND KEY ESTABLISHMENT PHASE

Any two IoT devices mutually authenticate each other using their generated materials $(E_P(a,b), e_1, e_2)$ that are made public through the NMs.

For example, let us consider that device 1 in FIGURE 3 needs to establish a secure session with device 2, which belongs to a different network. The authentication steps are as follows:

1) device 1 sends to its own (NM_1) a session establishment request with another network device;
2) NM_1 uses the authentication materials of NM_2 (the network manager of device 2) if available, otherwise NM_1 and NM_2 proceed to establish mutual trust between them using the procedure described above;
3) NM_1 sends to NM_2 the authentication materials of device 1 and NM_2 sends to NM_1 the authentication materials of device 2;

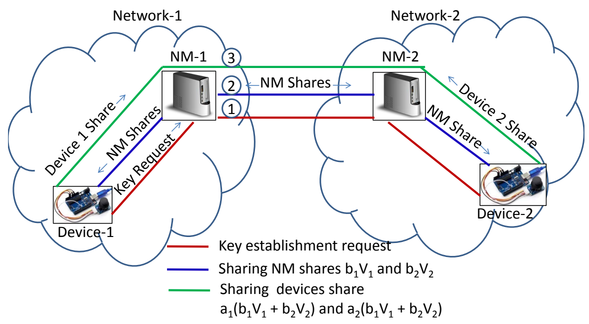

Network-1
Network-2
NM-1
NM-2
3
2
1
NM Shares
Device 1 Share
NM Shares
Key Request
Device 2 Share
NM Share
Device-1
Device-2
Key establishment request
Sharing NM shares b_1V_1 and b_2V_2
Sharing devices share $a_1(b_1V_1 + b_2V_2)$ and $a_2(b_1V_1 + b_2V_2)$

FIGURE 4. Sequence of messages exchanged for symmetric key establishment between two IoT devices

4) NM_1 forwards to device 1 the authentication materials of device 2 and NM_2 forwards to device 2 the authentication materials of device 1;
5) device 1 uses device 2 authentication materials to generate C_1 using r as follows:

$$C_1 = rE_1 \qquad : 1M \tag{5}$$

The cost of calculating C_1 is one scalar multiplication, i.e., $1M$. C_2 is calculated using a random nonce as follows:

$$C_2 = \text{nonce} + rE_2 \quad : 1M + 1A \tag{6}$$

The cost of calculating C_2 is one scalar multiplication operation and one scalar addition operation, i.e., $1M + 1A$.

C_1 and C_2 are the parts of cipher text that contain the encrypted nonce and are generated using the private key of device 1 and the public key of device 2.
6) the two devices exchange their own generated C_1 and C_2 with each other;
7) each of the two devices extracts the nonce from the received material as follows:

$$\text{nonce} = C_2 - d_{\mathrm{N}} C_1 \quad : 1M + 1A \tag{7}$$

8) each device sends back the extracted nonce after encryption to the other device to prove its own authenticity.

After successfully authenticating each other, two IoT devices establish a mutual symmetric key for secure communication. In our proposed solution, the symmetric key is based on four parts:

- share of device 1 $(a_1(b_1V_1 + b_2V_2))$;
- share of network manager of device 1 (b_1V_1);
- share of device 2 $(a_2(b_1V_1 + b_2V_2))$;
- share of network manager of device 2 (b_2V_2).

Dependency on both device and network shares prevents impersonation attacks. In fact, a successful impersonator would need to know the shares, the SK and randomly generated parameters of the D and of the NMs (i.e., a, d, l, r).

FIGURE 4 shows the steps for the establishment of a mutual symmetric key between two devices on different networks, as follows:

- device 1 sends a key establishment request to device 2 through the network managers
- NM_1 selects a random parameter b_1 and generates b_1V_1 and NM_2 selects a random parameter b_2 and generates b_2V_2. These generated parameters are shared with device 1 and device 2, where V_1 and V_2 are defined by (2)
- device 1 selects a random a_1 and responds with its own key share, $a_1(b_1V_1 + b_2V_2)$ while device 1 selects a random a_2 and responds with its own key share, $a_2(b_1V_1 + b_2V_2)$
- both devices use the received key shares to establish a mutual symmetric key as follows:

$$\text{Device 1 Key} = a_1\{a_2(b_1V_1 + b_2V_2)\} \tag{8}$$
$$\text{Device 2 Key} = a_2\{a_1(b_1V_1 + b_2V_2)\} \tag{9}$$

E. KEY FRESHNESS AND EXPIRATION

Network-wide time synchronization is generally costly and internal time reference accuracy can be limited for resource-constrained devices that may enter often long deep sleep modes to save energy. Although there are exceptions to this rule (e.g., the Texas Instruments CC2538 circuit that we used in our experiments, which has a precise low-power reference oscillator on board), good security practice advises against making key freshness or expiration decisions based only on device time.

To address this need, our method generates the secret symmetric session key using the network and device shares, and a random point addition, which are all session-specific. Thus, the key is implicitly invalidated at the end of its session, irrespective of session duration.

V. SECURITY ANALYSIS

Dependable cryptographic systems strongly rely on effective mechanisms for authentication and key management. These mechanisms are even more challenging to implement in the IoT due to its inherently distributed nature and the reduced resources of many IoT devices. Numerous attacks are possible on a network, at different levels. We consider the most vulnerable network attack points and an evil adversary with the capabilities shown in FIGURE 5.

A. AUTHENTICATION PROCEDURE EVALUATION

Here we evaluate the attack model shown in FIGURE 6 during the authentication phase as described in section IV-D. NMs retrieve the devices information from each other using the other NM credentials received from the network coordinator/central TTP server. The attacker's influence in this step is ignored because the communication between the NM and central server is secured using their public private key pair that was established before the network deployment phase (and is considered secure). However, after network deployment, an attacker can influence network operations. FIGURE 6 represents the attack as man in the middle during the authentication phase between the two devices. In this case, the attacker is outside each network but between the

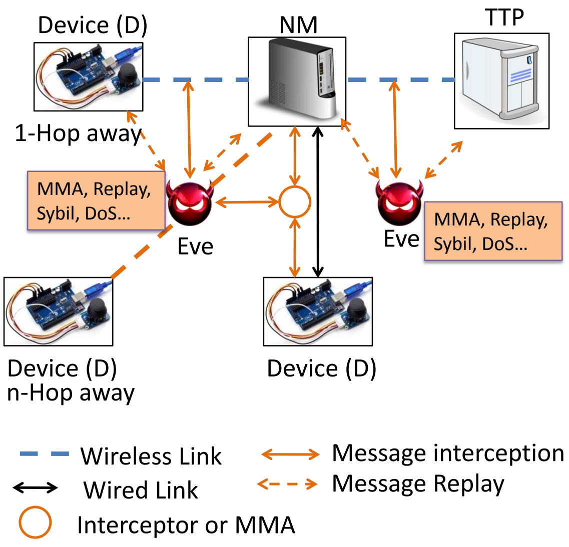


FIGURE 5. Attack models

two networks. During the authentication phase, each end device encrypts the nonce as shown in (5) and (6) using the other device's public parameter retrieved through NMs and decrypts them using (7). Since the attackers are in the middle, they can modify the information to find the actual value of the nonce. We consider N_1 and N_2 as nonce of device 1 and device 2, respectively, as shown in FIGURE 6. Device 1 encrypts the N_1 using device 2 public parameters $(E_P(a,b), E_1, E_2)$ as

$$C_1 = rE_1 \tag{10}$$
$$C_2 = N_1 + rE_2 \tag{11}$$

Device 2 encrypts the N_2 using device 1 public parameters $(E_P(a,b), E_1, E_2)$ as

$$C_1 = rE_1 \tag{12}$$
$$C_2 = N_2 + rE_2 \tag{13}$$

These encrypted nonces pass through the attacker on their way to the destination. The attacker modifies them to try and make the authentication process unsuccessful as

$$C_1^* = C_1 \tag{14}$$
$$C_2^* = N + C_2 \tag{15}$$

Device 1 retrieves the nonce from the received C_1^* and C_2^* as

$$N_2 + N = C_2^* - d_{\mathrm{N}}C_1^* \tag{16}$$

Device 2 retrieves the nonce from the received C_1^* and C_2^* as

$$N_1 + N = C_2^* - d_{\mathrm{N}}C_1^* \tag{17}$$

Once the nonces are decrypted, each device adds its own nonce with the received nonce to calculate the final nonce for authentication and this results in

$$\text{Resultant Nonce} = N_1 + N_2 + N \tag{18}$$

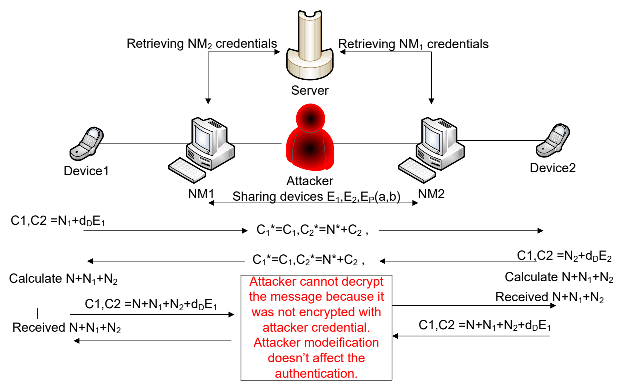

Retrieving NM$_2$ credentials
Retrieving NM$_1$ credentials
Server
Device1
NM1
Attacker
Sharing devices E$_1$,E$_2$,E$_P$(a,b)
NM2
Device2
C1,C2 =N$_1$+d$_D$E$_1$
C$_1$*=C$_1$,C$_2$*=N*+C$_2$,
C$_1$*=C$_1$,C$_2$*=N*+C$_2$,
C1,C2 =N$_2$+d$_D$E$_2$
Calculate N+N$_1$+N$_2$
Calculate N+N$_1$+N$_2$
Attacker cannot decrypt the message because it was not encrypted with attacker credential. Attacker modeification doesn't affect the authentication.
Received N+N$_1$+N$_2$
C1,C2 =N+N$_1$+N$_2$+d$_D$E$_1$
C1,C2 =N+N$_1$+N$_2$+d$_D$E$_1$
Received N+N$_1$+N$_2$

FIGURE 6. Attack scenario during authentication

where N is the nonce added by the attacker. Each device sends the resultant nonce to the other for authentication purposes. This resultant nonce passes through the attacker but the attacker cannot decrypt it as it is encrypted using the destination device public parameters, not with the attacker public parameters.

Each destination end device decrypts the received resultant nonce and compares it with the calculated nonce. If both received and calculated nonce are identical, then the authentication of the two devices is considered successful even in the presence of an attacker as man in the middle.

Moreover, this procedure does not allow the attacker to authenticate itself to the device. This is because, each device encrypts the nonce using the other device's public parameters and the attacker cannot decrypt it using its own private key. The modification also does not affect the authentication. But if the attacker does change the resultant nonce, this will be detected at the end devices because both calculated and received nonces will not be identical and this will warn the end devices about the presence of an attacker.

B. KEY ESTABLISHMENT PROCEDURE EVALUATION

During the key establishment process, as discussed in section IV-D and shown in FIGURE 4, each secret key depends upon the NM shares and device shares. The attacker could come in the middle between the source node and its network manager or between the network managers or between the destination node and its network manager.

As shown in FIGURE 7, we consider an attacker between device 1 and its network manager NM1. Device 1 sends a key establishment request to its NM1. NM1 forwards this request to device 2 through NM2 using its public parameters along with b_1V_1 as:

$$C_1 = d_{NM_1}E_1 \tag{19}$$
$$C_2 = b_1V_1 + d_{NM_1}E_2 \tag{20}$$

In response, NM1 receives b_2V_2 and $a_2(b_1V_1 + b_2V_2)$ that NM2 encrypted using NM1 public parameters as:

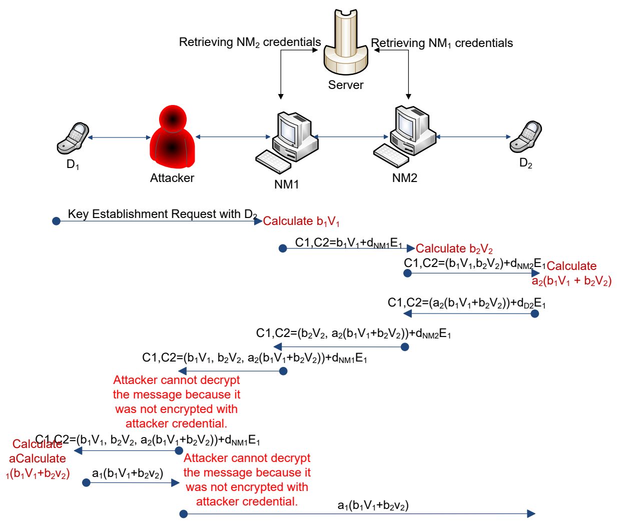

Retrieving NM$_2$ credentials
Retrieving NM$_1$ credentials
Server
D$_1$
Attacker
NM1
NM2
D$_2$
Key Establishment Request with D$_2$
Calculate b$_1$V$_1$
C1,C2=b$_1$V$_1$+d$_{NM1}$E$_1$
Calculate b$_2$V$_2$
C1,C2=(b$_1$V$_1$,b$_2$V$_2$)+d$_{NM2}$E$_1$
Calculate a$_2$(b$_1$V$_1$ + b$_2$V$_2$)
C1,C2=(a$_2$(b$_1$V$_1$+b$_2$V$_2$))+d$_{D2}$E$_1$
C1,C2=(b$_2$V$_2$, a$_2$(b$_1$V$_1$+b$_2$V$_2$))+d$_{NM2}$E$_1$
C1,C2=(b$_1$V$_1$, b$_2$V$_2$, a$_2$(b$_1$V$_1$+b$_2$V$_2$))+d$_{NM1}$E$_1$
Attacker cannot decrypt the message because it was not encrypted with attacker credential.
C1,C2=(b$_1$V$_1$, b$_2$V$_2$, a$_2$(b$_1$V$_1$+b$_2$V$_2$))+d$_{NM1}$E$_1$
Calculate aCalculate $_1$(b$_1$V$_1$+b$_2$v$_2$)
a$_1$(b$_1$V$_1$+b$_2$v$_2$)
Attacker cannot decrypt the message because it was not encrypted with attacker credential.
a$_1$(b$_1$V$_1$+b$_2$v$_2$)

FIGURE 7. Attack scenario during symmetric key establishment process

$$C_1 = d_{NM_2}E_1 \tag{21}$$
$$C_2 = (b_1V_1 || a_2(b_1V_1 + b_2V_2)) + d_{NM_2}E_2 \tag{22}$$

NM1 forwards b_1V_1, b_2V_2 and $a_2(b_1V_1 + b_2V_2)$ to device 1 using its public parameters as

$$C_1 = d_{NM_1}E_1 \tag{23}$$
$$C_2 = (b_1V_1 || b_2V_2 || a_2(b_1V_1 + b_2V_2)) + d_{NM_1}E_2 \tag{24}$$

But device 1 receives this message through the attacker. Since device 1 public parameters are used to encrypt the message, the attacker cannot decrypt it and hence can only forward it unchanged to device 1. Device 1 encrypts $a_1(b_1V_1 + b_2V_2)$ using NM1 public parameters and forwards it to device 2 through NM1 as:

$$C_1 = d_{D_1}E_1 \tag{25}$$
$$C_2 = a_1(b_1V_1 + b_2V_2) + d_{D_1}E_2 \tag{26}$$

Again the attacker cannot decrypt the message and forwards it as it is. The two devices establish the key using (8) and (9).

For an attack to become successful, the attacker needs to be able to decrypt $a_1(b_1V_1 + b_2V_2)$ or $a_2(b_1V_1 + b_2V_2)$ and find the correct value of a_2 or a_1 for the decrypted value respectively, which is not possible in this case.

In the worst case, the attacker modifies the messages from NM1 to device 1 and vice versa as:

$$C_1^* = C_1 \tag{27}$$
$$C_2^* = N^* + C_2 \tag{28}$$

After decryption, device 1 gets

$$b_1V_1 || b_2V_2 || a_2(b_1V_1 + b_2V_2) + N^* \tag{29}$$

and device 2 gets

$$a_1(b_1V_1 + b_2V_2) + N^* \tag{30}$$

Then device 1 generates the key as

$$\text{key} = a_1(a_2(b_1V_1 + b_2V_2) + N^*) \tag{31}$$

And device 2 generates the key as

$$\text{key} = a_2(a_1(b_1V_1 + b_2V_2) + N^*) \tag{32}$$

The generated keys are not the same because factor a_1N^* in the device 1 key does not match factor a_2N^*. This alerts the end devices about the presence of an attacker and also verifies integrity without the need of a hash function. Even if the attacker changes its position and comes in the middle between two NMs, it cannot hide its presence and change the content of key messages. Both attacks are detected by the end devices.

C. DENIAL OF SERVICE ATTACKS (DOS)

These usually attempt to isolate a device from the network by sending fake messages or by replaying processing-heavy messages in order to keep the receiving device always busy. In the proposed network scenario and operation, DoS attacks could be launched during (1) the initialization phase, (2) the addition of a new node, (3) session establishment and (4) during normal operation, by replaying old messages.

The initialization phase is normally more secure and robust against any types of attacks due to the presence of network administrators and continuous monitoring of devices and equipment. Hence the risk of DoS attacks during the initialization phase is typically ignored. Also the addition of a new node should involve network administrators and continuous monitoring; hence DoS attacks are ignored also in this phase either. New session key establishment requests can be used to launch DoS attacks if the attacker knows the previous SID that an authentic device has established with other devices. But in the proposed solution, prior to session establishment with the existing user, the authentication procedure is initiated. Each encryption procedure includes SID. When authentication procedure is initiated, it also includes SID along with the nonce N_1 as

$$C_1 = rE_1 \tag{33}$$

$$C_2 = \text{SID} \mid\mid N_1 + rE_2 \tag{34}$$

The destination node decrypts the message to extract the nonce and SID. If the SID is old, it discards the message and informs the NM about possible DoS attacks or replay attacks. Similarly, if the SID is unknown or not in sequence with previous ID, the node informs the NM about the malicious activity and possible DoS attacks.

The session key depends on the key shares as shown in (8) and (9), where the values of a_1 and a_2 are selected randomly by the end devices during each session key establishment. If the attacker replays the previous session messages, they will not be decrypted by the new session key due to the new value of a_1 and a_2. This alerts end devices about possible DoS attack by an attacker using previous session messages or unknown messages. Hence, successful DoS attacks are not possible in our approach using the session establishment requests or old data packets.

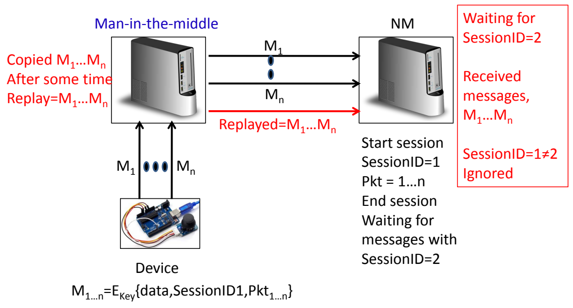


FIGURE 8. Man-in-the-middle forward normal traffic (black lines). It can launch DoS, Replay and MMA attacks (red lines)

D. SYBIL ATTACKS

They are usually implemented by creating and registering on the network multiple fake identities of an authentic device. In the scheme that we propose, this attack is not possible because the scheme authenticates all network devices using public/private key pair. If an attacker creates multiple fake devices with authentic devices ID, these fake devices will only have the authentic devices ID and their public keys. Fake devices will have no idea about the private keys of the authentic devices that are required to decrypt the value of C_1 and C_2.

For example, if an attacker impersonates device 1 as shown in FIGURE 7, it knows the public parameters of device 1, i.e., E_1, E_2 and $E_P(a, b)$, where $E_2 = d_DE_1$. Here, d_D is the private key of device 1 and it is only known to device 1. The private key of the attacker is d_A which is different from d_D. Since the attacker is impersonating device 1 to the network manager/other device, it receives C_1 and C_2 that are encrypted using the authentic device public key. The attacker decrypts it as

$$\text{data} = C_2 - d_AC_1 \tag{35}$$

$$\text{data} = \text{data} + rE_2 - d_ArE_1 \tag{36}$$

$$\text{data} = \text{data} + rd_DE_1 - d_ArE_1. \tag{37}$$

Since the attacker does not know the private key of the authentic device 1, and its private key is different from the device private key, it cannot cancel the term rd_DE_1 using rd_AE_1. This shows that the attacker cannot launch Sybil attacks successfully.

E. MAN-IN-THE-MIDDLE ATTACKS (MMA)

As the name implies, in these attacks the attacker manages to intercept all exchanges among two communicating parties without revealing its real identity to either of them. Hence, in order to succeed, the attacker needs to successfully impersonate itself to each communicating party in the session. The authentication phase during the network initialization phase is normally secure due to the presence of network administrators. Here we consider the authentication and key establishment process for each session establishment after the network initialization phase. As shown in 9, an attacker impersonates itself as NM to a device and vice versa. Each session establishment includes the authentication followed by key establishment. During the session authentication, the attacker receives the session establishment request from the device, that includes the SID and is also encrypted with the NM public parameters. The attacker cannot decrypt the received request as it does not have the NM private key. The attacker generates its own request with a random SID (since it does not know the expected SID), encrypts it with the NM public parameters and sends it to the NM. Upon receiving message, the NM decrypts it and verifies the SID by comparing it with the expected SID. Since it is very rare that a randomly generated SID matches the expected SID, hence the NM discards the message due to the mismatch among SIDs.

This works similarly if the attacker sends fake (i.e. its own) public parameters to the device along with its own generated SID. The device decrypts the message and compares it with its own SID to verify the authenticity of the session and message. The device discards the message and received public parameters upon mismatch in SIDs.

If an attacker performs an impersonation during the key establishment process, it receives the key share requests from the device that is encrypted with the NM public parameters and includes the same SID. Hence the attacker cannot decrypt it to know the SID as it does not have the NM private key. It could only generate its own random SID for fake key share request messages, that would be immediately detected at both ends (NM and device) as described above during the authentication process and shown in FIGURE 9. To successfully launch impersonation attacks, it would need to encrypt and decrypt the information correctly as well as guess the correct value of SID, which is almost impossible.

F. REPLAY ATTACKS

The establishment of a new session with a device starts with the authentication process, followed by the key establishment process. During the authentication phase, each device shares its nonce using (34) that also includes the SID. This SID is unique for each new session and acts as timestamps in our case. If the attacker replays the previous session authentication messages, they will be discarded due to the presence of an old SID that has already expired. Also the attacker cannot know/update/change the SID as it is encrypted by the destination device public key and only the destination node will know it after decrypting it with its private key using (7).

Similarly, each session key depends on the value of a_1, a_2, b_1 and b_2 randomly selected for each session by the devices and NMs respectively as shown in (8) and (9). If the attacker replays the old session messages that were encrypted by the old session key having the old values of a_1, a_2, b_1, b_2, it will not be decrypted by the new session key that has new values of a_1, a_2, b_1, b_2. This shows the robustness of the proposed solution against replays attacks.

G. KEY COMPROMISSION ATTACKS

In these attacks, the attacker manages to obtain the secret key of a node and later uses it for secure communication within the network, as would a legitimate node. In our scheme, each communication session uses a new key, which effectively prevents the attacker from reusing the compromised secret key.

H. ROBUSTNESS

Integrity is one of the key features of security. Hashes are the most common approach to ensure the integrity of the messages. During the key establishment process, integrity of the key is most important and needs to be preserved. In the existing literature, a sender encrypts the symmetric session key with the public key of the destination and sends it. The destination uses its private key to decrypt it as

$$\text{Enc. key} = Enc_{\{K_{RPublic}\}}\{\text{Symmetric Key}\} \tag{38}$$

$$\text{Symmetric Key} = Dec_{\{K_{RPrivate}\}}\{\text{Enc. key}\} \tag{39}$$

where $K_{RPublic}$ and $K_{RPrivate}$ are the public key and private key of the receiver respectively, Enc is the encryption process and Dec is the decryption process. Since the private key is known only by the receiver itself, this approach seems secure, since a message is only decrypted by the correct receiver. But the integrity of content is not guaranteed. This is because, the attacker can also use the receiver public key to encrypt a fake symmetric key and it is impossible for the receiver to know whether the received key is authentic or not because the receiver can successfully decrypt both authentic and fake symmetric keys using its private key.

$$\text{Enc. key} = Enc_{\{K_{RPublic}\}}\{\text{Fake Symmetric Key}\} \tag{40}$$

$$\text{Fake Symmetric Key} = Dec_{\{K_{RPrivate}\}}\{\text{Enc. key}\} \tag{41}$$

To ensure the integrity of the message/key, a hash is used. This hash is created from the message, encrypted with the sender's private key $K_{SPrivate}$ and sent with the message.

$$\text{Enc. Hash} = Enc_{\{K_{SPrivate}\}}\{\text{Hash of message}\} \tag{42}$$

After the receiver gets the packet with the hash, the sender's public key $K_{SPublic}$ is used to decrypt the hash and the receiver's private key is used to decrypt the message. Another hash is created from the received message and compared it with the received decrypted hash to ensure the integrity and authenticity of the message. The message is accepted if both hashes are identical, otherwise it is discarded. Although the attacker can also decrypt the hash using the sender's public key, it cannot decrypt the actual message as the attacker does not know the receiver's private key. This stops the attacker from modifying either the hash or the message.

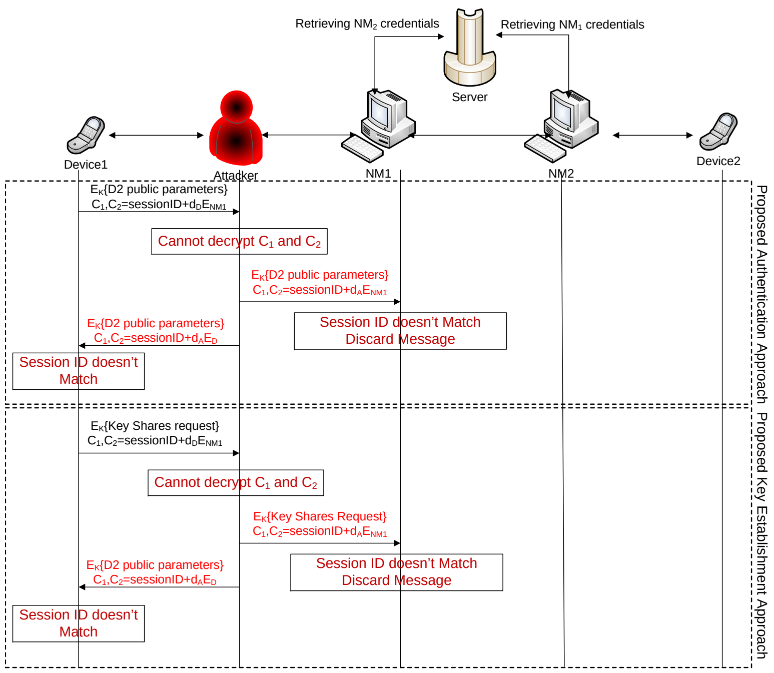

Retrieving NM_2 credentials
Retrieving NM_1 credentials
Server
Device1
Attacker
NM1
NM2
Device2
E_K\{D2 public parameters\} C_1,C_2=sessionID+$d_D E_{NM1}$
Cannot decrypt C_1 and C_2
E_K\{D2 public parameters\} C_1,C_2=sessionID+$d_A E_{NM1}$
Session ID doesn't Match Discard Message
E_K\{D2 public parameters\} C_1,C_2=sessionID+$d_A E_D$
Session ID doesn't Match
E_K\{Key Shares request\} C_1,C_2=sessionID+$d_D E_{NM1}$
Cannot decrypt C_1 and C_2
E_K\{Key Shares Request\} C_1,C_2=sessionID+$d_A E_{NM1}$
Session ID doesn't Match Discard Message
E_K\{D2 public parameters\} C_1,C_2=sessionID+$d_A E_D$
Session ID doesn't Match
Proposed Authentication Approach
Proposed Key Establishment Approach

FIGURE 9. Man-in-the-middle launches an impersonation attacks during the session authentication and key establishment process

These hash functions are useful to ensure the integrity of the messages and verify the sender as well, but they imply an extra overhead. For example, if we consider only a key exchange process, the length of a key is around 160 bits when using ECC. The size of a hash for this key using SHA is 256–512 bits, while using MD5 it is 128 bits. The overall packet size becomes $160 + 256$ bits or $160 + 512$ bits or $160 + 128$ bits, which can be a very significant overhead for the radio packet size and increase the energy consumption of devices.

In hash based approaches, there is also a possibility that two different messages result in same hash. To make it easily understandable, we use a very simple example with a simple hash function. Let us say that we have two messages msg1=45924984 and msg2=68928120 and a hash function with the following feature

$$\text{Hash(message)} = \text{message mod } 65536 \tag{43}$$

The hashes of msg1 and msg2 using this function are

$$\text{Hash(msg1)} = 45924984 \text{ mod } 65536 = 49784 \tag{44}$$

$$\text{Hash(msg2)} = 68928120 \text{ mod } 65536 = 49784 \tag{45}$$

If the attacker replaces the content of msg1 with msg2, the receiver will not be able to detect this change, since both messages generates the same hash.

In our approach, we eliminate the need of hash function and achieve message integrity using only an SID. For each new session, the sender generates a new SID using a specific function and concatenates it with the message. After receiving the message, the destination device decrypts the received message and separates the SID and generates its own SID using the same SID function. If both SIDs match, the destination accepts the message otherwise it discards the message. To show the robustness of the proposed solution, consider the above example again. Let us say that after concatenating the SID to a message, its content becomes equal to msg1 (i.e., 0x2BCC278). During transit over the channel, the attacker changes the content of msg1 to msg2 (i.e., 0x41BC278). The receiver receives the msg2 instead of msg1. The receiver decrypts the received msg2 and extracts the SID (i.e., 0x41BC). The receiver also generates the SID using specific function and output of that function gives 0x2BCC. The receiver compares the received SID (i.e., 0x41BC) with the generated SID (i.e., 0x2BCC) and finds that they are not identical. Hence the receiver discards the message. While this change was not detected in the hash based approach, it is easily detected by the proposed solution. FIGURE 10 shows how the device and the NM check the integrity of messages, both with hashes and without hashes.

The SID is used to ensure the message and session integrity but it is only a session identifier. It is recommended to set the SID length to at least to 16 bits (65536 possible identifiers). This reduces the probability of correct guessing and also imposes less overhead. For example, if we consider 160 bit key size and only 16 bit SID, the packet size would become $160 + 16$ bits, which are much fewer than with the hash based approaches. As mentioned above, reducing the packet size reduces the energy cost of sending a packet and increases the lifetime of resource constrained devices.

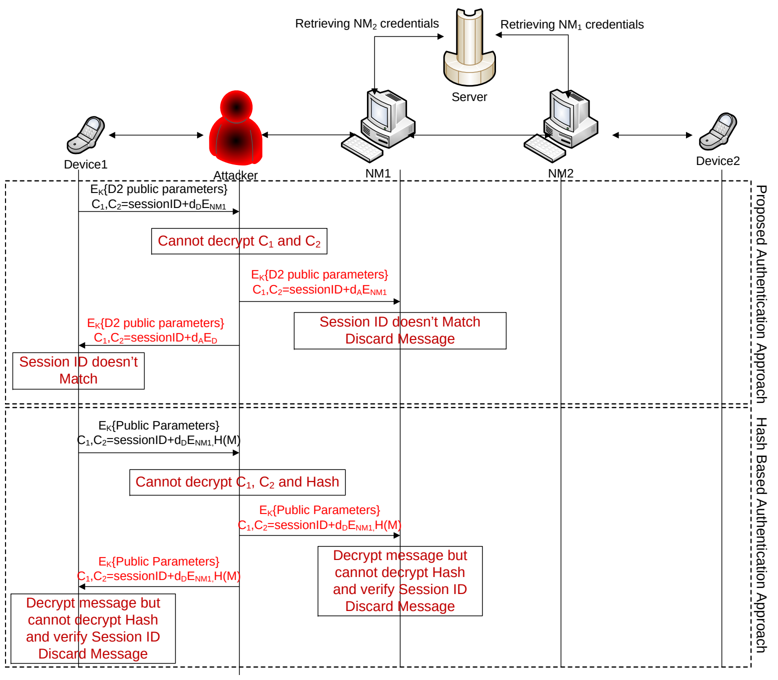

Retrieving NM2 credentials
Retrieving NM1 credentials
Server
Device1
Attacker
NM1
NM2
Device2
Proposed Authentication Approach
Hash Based Authentication Approach
EK{D2 public parameters} C1,C2=sessionID+dDENM1
Cannot decrypt C1 and C2
EK{D2 public parameters} C1,C2=sessionID+dAENM1
Session ID doesn't Match Discard Message
EK{D2 public parameters} C1,C2=sessionID+dAED
Session ID doesn't Match
EK{Public Parameters} C1,C2=sessionID+dDENM1,H(M)
Cannot decrypt C1, C2 and Hash
EK{Public Parameters} C1,C2=sessionID+dDENM1,H(M)
Decrypt message but cannot decrypt Hash and verify Session ID Discard Message
EK{Public Parameters} C1,C2=sessionID+dDENM1,H(M)
Decrypt message but cannot decrypt Hash and verify Session ID Discard Message

FIGURE 10. Integrity check in the presence of impersonation attacks with hash (literature) and without hash (proposed) approaches

I. AVISPA ANALYSIS

We validate the reliability of the proposed key management solution for IoT devices using the Automated Validation of Internet Security Protocols and Applications (AVISPA) tool [26], [27]. AVISPA is an industrial-strength tool for the analysis of large-scale security-sensitive Internet protocols and applications. It provides a High Level Protocol Specification Language (HLPSL) [28] that provides suitable human-readable formal semantics to model the communication patterns of secure protocols. HLPSL descriptions are automatically converted to an Intermediate Format (IF) that is suitable for the formal analysis of the security properties using four AVISPA back-ends: CL-based Model-Checker (CL-AtSe) [29], On-the-Fly Model-Checker (OFMC) [30], TA4SP, and SAT-based Model-Checker (SATMC) [31].

We implement in AVISPA HLPSL the proposed solution as follows:

1) First, we define the role of each device of the network in terms of its public and private parameters and of the existence of a direct inter-device connection.
2) While defining the role of each entity, the messages sent by a device are labelled with a sequential number, in protocol order. For example, if device 1 sends first a message, waits for a reply and then sends a second message, the numbers assigned to device 1 messages are 1 and 3, while the reply from device 2 is assigned number 2.
3) After this, the scope of a communication session is defined in terms of the total number of message exchanges, the number of message exchanges per device, and of the parameters that are exchanged.
4) The next step is to define the role of an intruder that will act as a *man in the middle*. In this quality, the intruder will have knowledge of all public parameters and have access to each exchanged message and its content, since it is within communication range with all the entities of the network.
5) In the next step, the actual verification environment is set up by defining: (1) the public parameters that are exchanged (E(a,b), e_{NM}, e_D, W, V, E_1 and E_2), (2) the total number of devices, (3) the authentication and key generation protocol used by each device, (4) the kind of knowledge that an intruder is assumed to have, and (5) the parameters of the communication session.
6) The last step checks whether the authentication protocols can be breached in a given scenario.

Then we checked our protocol using OFMC and CL-AtSe. OFMC builds part of the infinite tree defined by the protocol analysis problem in a demand-driven way, i.e., on-the-fly, and uses a number of symbolic techniques to represent the state space. CL-AtSe provides a translation from any security protocol specification written as transition relation into a set of constraints that can be effectively used to find attacks on protocols. Both translation and checking are fully automatic and performed internally by CL-AtSe, i.e., no external tool is used. In this approach, each protocol step is modelled by constraints on the adversary knowledge. Both checks succeeded, showing the safety of our approach.

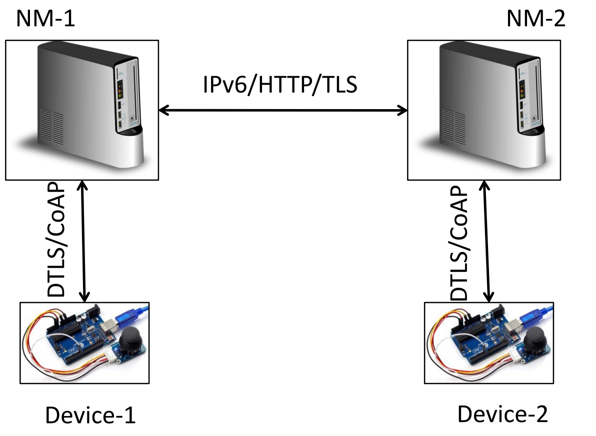


FIGURE 11. Experimental setup for the evaluation of proposed solution

TABLE 2. ROM and RAM occupied by the proposed solution

Function	ROM (text) (Bytes)	RAM (data+bss) (Bytes)
Without code implementation	42016	222+6436
With code implementation	46204	282+6658
Actual code occupation	4188	60+222

VI. EVALUATION

We used the network setup shown in FIGURE 11 to evaluate the performance of the proposed solution in terms of memory, processing and energy consumption.

A. MEMORY COST

To estimate the memory requirements of the proposed solution, we use contikiOS to implement the proposed solution on top of Telosb motes. We observed that without including the code implementing our algorithm that we discussed above, standard contiki library functions occupy 42016 bytes. This memory occupation grows to 46204 bytes by including the code implementing our protocol. This shows a memory requirement of 4188 bytes for the proposed solution implementation on a telosb mote using the parameters described in tab. 1. According to NIST recommendations for the ECC based approach, a 192 bit key size is used. TABLE 2 shows the memory occupied in ROM and RAM, *text* shows the size of the code section in bytes (typically ROM) and *data and bss* show sections that contain variables, stored in RAM.

In the proposed solution, key shares fulfil the requirements of integrity while previous, state-of-the-art approaches used hashes to check message integrity. Hash functions also require additional code memory. Since the proposed solution does not need hash functions to ensure the message integrity, its memory requirement is low and well suitable for the resource constrained devices.

B. ENERGY COST

We used the Contiki Energest module [32] to estimate the total processing and radio time and energy usage. The module uses a real-time clock and can split the reports by system component.

TABLE 3. Energy consumption for authentication and key generation materials in proposed solution

Function	Energy (μJ)
Authentication	32
Key generation	23

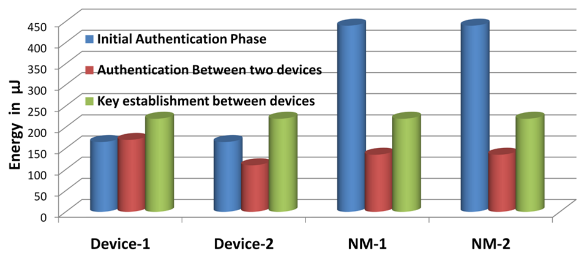


FIGURE 12. Energy consumption of each device during initial authentication phase, authentication between devices and key establishment between devices belonging to two different networks

The total energy of a module is calculated as follows:

$$\text{Energy}_{\text{module}} = VIt_{\text{module}} \tag{46}$$

where $V = 2.1V$ is the supply voltage, $I = 13mA$ is the average supply current and t_{module} is the active time of the module (as reported by the Energest module). We can thus calculate the total energy consumption (by both the processor and the radio transceiver) as the sum of all module energies.

Using the Energest reports we obtained the results shown in TABLE 3. While authentication is a rare event, key generation, which occurs much more often, requires much less energy consumption than with the previous approaches.

FIGURE 12 shows the energy consumption of each device involved in different phases of the proposed solution, including the initial authentication phase shown in FIGURE 2, the authentication between two devices belonging to two different networks shown in FIGURE 3, and the key establishment between the two devices belonging to two different networks shown in FIGURE 4. Since the RF transceiver is a major component of total energy consumption, the difference between the energy consumption of the two devices during mutual authentication is due to the different number and sizes of messages transmitted.

C. PROCESSING EFFORT DURING AUTHENTICATION

In the following we calculate the processing effort during the authentication phase between the two devices belonging to different networks using the following notations:

M	effort for scalar multiplication
A	effort for addition
H	effort for hash function
MAC	effort for message authentication code function

During the authentication process, each device performs the operations given in (5), (6), and (7) that also reflect the

TABLE 4. Comparison of processing cost during the key establishment phase by the end devices using different algorithms

Algorithm	Device 1	Device 2
Yang [33]	4M + 2A + 4H	4M + 2A + 4H
Yoon [34]	4M + 2A + 4H	4M + 2A + 4H
Debiao [35]	3M + 3H + 2MAC	3M + 2H + 2MAC
Goutham [36]	3M + 4H + 2MAC	3M + 3H + 2MAC
Wei [37]	2M + 1A + 4H + 3MAC	2M + 1A + 3H + 2MAC
Proposed	3M + 2A	3M + 2A

kinds of operations involved in authentication. To adhere to literature conventions, we estimate that (5) uses one scalar multiplication operation, and that (6) and (7) use one scalar multiplication and one point addition operations. TABLE 4 summarizes the processing cost of the device authentication phase in some previous approaches. Our proposed solution performs better in terms of computational cost, since it does not include hashing, that consumes even more computational resources than simple encryption. This is because in hashing, first a message digest is calculated using MD5 or SHA and then it is encrypted with the private key of the sender.

D. LATENCY

Processing latency is a key performance aspect for IoT devices. It indirectly limits how many messages per second can be processed by an IoT device. High processing latencies can negatively impact the authentication process and reduce the data rate at which devices exchange messages.

To analyse processing latencies for the proposed solution based on ECC, a software prototype was developed using the pyCryptov2.7a1 libraries. A simple testbed was established consisting of two IoT devices (sender and receiver). A significant portion of processing latency is due to the encryption algorithm and it mainly depends on the message size and device processing power. To this aim, encryption/decryption latencies were measured for a wide range of data sizes. The developed software prototype measures latencies by recording timestamps before and after a message is encrypted and decrypted. FIGURE 13 shows the variation of the encryption latency with message size. We note that the latencies are low even for large messages. The measured latencies for signature calculation are low as shown in FIGURE 14. This is an extra latency that is present in the existing literature and removed in the proposed solution. Due to low encryption latencies, the processing latencies are always less than 20 ms, which can be considered suitable for resource-constrained IoT devices.

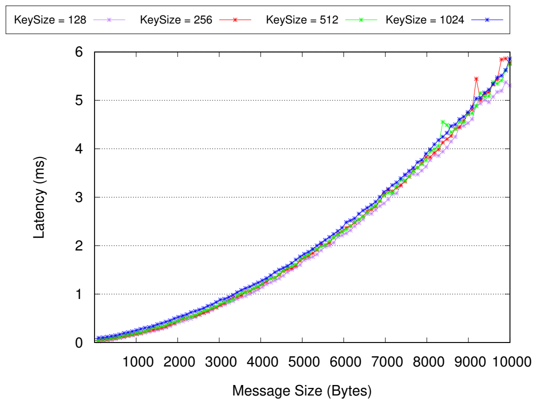


FIGURE 13. ElGamal encryption latencies with increasing message size.

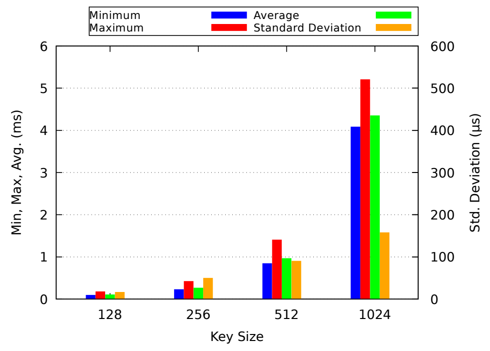


FIGURE 14. Signature latencies averaged over 100 trials.

VII. CONCLUSION

Although DTLS is becoming a security standard for the IoT, its reliance on resource-consuming PKI certification mechanisms limits its usefulness for resource-constrained devices, which are often found in IoT networks.

We propose a scalable mutual authentication and key establishment protocol that does not require a certification authority and is computationally lightweight. We implemented the proposed solution using resource-constrained devices in an IoT network to demonstrate its feasibility and its low energy consumption, which is lower than that of similar state of the art methods. A key performance and energy improvement aspect of our approach is that it does not require expensive hashes to be computed in order to ensure message integrity, but uses a much shorter session ID. We also successfully checked the reliability of the authentication and cryptographic material creation and management using an industrial-strength automated analysis tool for large-scale Internet security-sensitive protocols and applications, namely AVISPA.

In the future, we plan to design similar approaches for other cyber security protocols, such as IPsec/IKE.

ACKNOWLEDGEMENT

The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding this work through Research Group no. RG-1438-070. The authors also extend their special thanks to Luciano Lavagno, Mihai Lazarescue and Rafiullah Khan for their valuable feedback.

REFERENCES

[1] Z. Zhou, J. Feng, B. Gu, B. Ai, S. Mumtaz, J. Rodriguez, and M. Guizani, "When mobile crowd sensing meets uav: Energy-efficient task assignment and route planning," IEEE Transactions on Communications, vol. 66, no. 11, pp. 5526–5538, Nov 2018.

[2] Z. Ning, J. Huang, X. Wang, J. J. P. C. Rodrigues, and L. Guo, "Mobile edge computing-enabled internet of vehicles: Toward energy-efficient scheduling," IEEE Network, vol. 33, no. 5, pp. 198–205, Sep. 2019.

[3] H. Schaffers, N. Komninos, M. Pallot, B. Trousse, M. Nilsson, and A. Oliveira, "Smart cities and the future internet: Towards cooperation frameworks for open innovation," in The Future Internet, J. Domingue, A. Galis, A. Gavras, T. Zahariadis, D. Lambert, F. Cleary, P. Daras, S. Krco, H. Müller, M.-S. Li, H. Schaffers, V. Lotz, F. Alvarez, B. Stiller, S. Karnouskos, S. Avessta, and M. Nilsson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 431–446.

[4] P. K. Dhillon and S. Kalra, "Elliptic curve cryptography for real time embedded systems in iot networks," in 2016 5th International Conference on Wireless Networks and Embedded Systems (WECON), Oct 2016, pp. 1–6.

[5] Z. Zhou, H. Liao, B. Gu, K. M. S. Huq, S. Mumtaz, and J. Rodriguez, "Robust mobile crowd sensing: When deep learning meets edge computing," IEEE Network, vol. 32, no. 4, pp. 54–60, July 2018.

[6] Z. Ning, P. Dong, X. Wang, M. S. Obaidat, X. Hu, L. Guo, Y. Guo, J. Huang, B. Hu, and Y. Li, "When deep reinforcement learning meets 5g vehicular networks: A distributed offloading framework for traffic big data," IEEE Transactions on Industrial Informatics, pp. 1–1, 2019.

[7] H. Meng, D. Chao, and Q. Guo, "Deep reinforcement learning based task offloading algorithm for mobile-edge computing systems," in Proceedings of the 2019 4th International Conference on Mathematics and Artificial Intelligence, ser. ICMAI 2019. New York, NY, USA: ACM, 2019, pp. 90–94. [Online]. Available: http://doi.acm.org/10.1145/3325730.3325732

[8] W. Zhang, D. Lin, H. Zhang, C. Chen, and X. Zhou, "A lightweight anonymous mutual authentication with key agreement protocol on ecc," in 2017 IEEE Trustcom/BigDataSE/ICESS, Aug 2017, pp. 170–176.

[9] Z. Ning, Y. Feng, M. Collotta, X. Kong, X. Wang, L. Guo, X. Hu, and B. Hu, "Deep learning in edge of vehicles: Exploring trirelationship for data transmission," IEEE Transactions on Industrial Informatics, vol. 15, no. 10, pp. 5737–5746, Oct 2019.

[10] J. Wu, M. Dong, K. Ota, J. Li, W. Yang, and M. Wang, "Fog-computing-enabled cognitive network function virtualization for an information-centric future internet," IEEE Communications Magazine, vol. 57, no. 7, pp. 48–54, July 2019.

[11] X. Lin, J. Li, J. Wu, H. Liang, and W. Yang, "Making knowledge tradable in edge-ai enabled iot: A consortium blockchain-based efficient and incentive approach," IEEE Transactions on Industrial Informatics, pp. 1–1, 2019.

[12] N. Li, D. Liu, and S. Nepal, "Lightweight mutual authentication for iot and its applications," IEEE Transactions on Sustainable Computing, vol. 2, no. 4, pp. 359–370, Oct 2017.

[13] H. Liang, J. Wu, S. Mumtaz, J. Li, X. Lin, and M. Wen, "Mbid: Micro-blockchain-based geographical dynamic intrusion detection for v2x," IEEE Communications Magazine, vol. 57, no. 10, pp. 77–83, October 2019.

[14] M. Rezvani, A. Ignjatovic, E. Bertino, and S. Jha, "Secure data aggregation technique for wireless sensor networks in the presence of collusion attacks," IEEE Transactions on Dependable and Secure Computing, vol. 12, no. 1, pp. 98–110, Jan 2015.

[15] V. S. Miller, "Use of elliptic curves in cryptography," in Lecture Notes in Computer Sciences; 218 on Advances in cryptology—CRYPTO 85. New York, NY, USA: Springer-Verlag New York, Inc., 1986, pp. 417–426. [Online]. Available: http://dl.acm.org/citation.cfm?id=18262.25413

[16] A. Liu and P. Ning, "Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks," in 2008 International Conference on Information Processing in Sensor Networks (ipsn 2008), April 2008, pp. 245–256.

[17] H. Wang and Q. Li, "Efficient implementation of public key cryptosystems on mote sensors (short paper)," in Proceedings of the 8th International Conference on Information and Communications Security, ser. ICICS'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 519–528. [Online]. Available: http://dx.doi.org/10.1007/11935308_37

[18] H. Wang, B. Sheng, and Q. Li, "Elliptic curve cryptography based access control in sensor networks," Int. J. Secur. Netw., vol. 1, no. 3/4, pp. 127–137, Dec. 2006. [Online]. Available: http://dx.doi.org/10.1504/IJSN.2006.011772

[19] R. Sankar, T. Subashri, and V. Vaidehi, "Implementation and integration of efficient ecdh key exchanging mechanism in software based voip network," in 2011 International Conference on Recent Trends in Information Technology (ICRTIT), June 2011, pp. 124–128.

[20] C. P. Gouvêa and J. López, "Software implementation of pairing-based cryptography on sensor networks using the msp430 microcontroller," in Proceedings of the 10th International Conference on Cryptology in India: Progress in Cryptology, ser. INDOCRYPT '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 248–262. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-10628-6_17

[21] G. Hinterwälder, C. Paar, and W. P. Burleson, "Privacy preserving payments on computational rfid devices with application in intelligent transportation systems," in Proceedings of the 8th International Conference on Radio Frequency Identification: Security and Privacy Issues, ser. RFIDSec'12. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 109–122. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-36140-1_8

[22] L. Marin, A. J. Jara, and A. F. G. Skarmeta, "Shifting primes: Extension of pseudo-mersenne primes to optimize ecc for msp430-based future internet of things devices," in Availability, Reliability and Security for Business, Enterprise and Health Information Systems, A. M. Tjoa, G. Quirchmayr, I. You, and L. Xu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 205–219.

[23] P. Szczechowiak, L. B. Oliveira, M. Scott, M. Collier, and R. Dahab, "Nanoecc: Testing the limits of elliptic curve cryptography in sensor networks," in Proceedings of the 5th European Conference on Wireless Sensor Networks, ser. EWSN'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 305–320. [Online]. Available: http://dl.acm.org/citation.cfm?id=1786014.1786040

[24] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and rsa on 8-bit cpus," in Cryptographic Hardware and Embedded Systems - CHES 2004, M. Joye and J.-J. Quisquater, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 119–132.

[25] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in Advances in Cryptology — EUROCRYPT 2001, B. Pfitzmann, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 453–474.

[26] L. Viganò, "Automated Security Protocol Analysis With the AVISPA Tool," Electronic Notes in Theoretical Computer Science, vol. 155, pp. 61–86, 2006.

[27] "Automated Validation of Internet Security Protocols and Applications (AVISPA)," http://www.avispa-project.org/, Artificial Intelligence Laboratory, DIST, University of Genova, Italy.

[28] D. von Oheimb, "The high-level protocol specification language HLPSL developed in the EU project AVISPA," in Proceedings of APPSEM 2005 workshop, 2005, pp. 1–17.

[29] M. Turuani, The CL-Atse Protocol Analyser. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 277–286.

[30] D. Basin, S. Mödersheim, and L. Viganò, "OFMC: A symbolic model checker for security protocols," International Journal of Information Security, vol. 4, no. 3, pp. 181–208, 2005.

[31] A. Armando and L. Compagna, SATMC: A SAT-Based Model Checker for Security Protocols. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 730–733.

[32] A. Dunkels, F. Osterlind, N. Tsiftes, and Z. He, "Software-based On-line Energy Estimation for Sensor Nodes," in Proceedings of the 4th Workshop on Embedded Networked Sensors, ser. EmNets '07. New York, NY, USA: ACM, 2007, pp. 28–32.

[33] J.-H. Yang and C.-C. Chang, "An id-based remote mutual authentication with key agreement scheme for mobile devices on elliptic curve cryptosystem," Computers and Security, vol. 28, no. 3, pp. 138 – 143, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404808001120

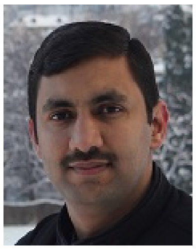
[34] E. J. Yoon and K. Y. Yoo, "Robust id-based remote mutual authentication with key agreement scheme for mobile devices on ecc," in 2009 International Conference on Computational Science and Engineering, vol. 2, Aug 2009, pp. 633–640.

[35] H. Debiao, C. Jianhua, and H. Jin, "An id-based client authentication with key agreement protocol for mobile client–server environment on ecc with provable security," Information Fusion, vol. 13, no. 3, pp. 223 – 230, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253511000029

[36] R. A. Goutham, G.-J. Lee, and K.-Y. Yoo, "An anonymous id-based remote mutual authentication with key agreement protocol on ecc using smart cards," in Proceedings of the 30th Annual ACM Symposium on Applied Computing, ser. SAC '15. New York, NY, USA: ACM, 2015, pp. 169–174. [Online]. Available: http://doi.acm.org/10.1145/2695664.2695666

[37] W. Zhang, D. Lin, H. Zhang, C. Chen, and X. Zhou, "A lightweight

anonymous mutual authentication with key agreement protocol on ecc," in 2017 IEEE Trustcom/BigDataSE/ICESS, Aug 2017, pp. 170–176.



DR SARMADULLAH KHAN (M'12) is a lecturer at De Montfort University, UK. He is currently program leader of MSc cyber technology and its pathways. Dr Khan received his PhD and MSc in Electronics and Telecommunication Engineering from Politecnico di Torino, Italy in 2013 and in 2009 respectively. Dr Khan was assistant professor at CECOS University of IT and Emerging Science, Peshawar Pakistan between 2013 – 2017. Dr Khan is a member of academic board of studies, curriculum revision committee, program management board and school management group. Dr Khan's research focuses on Internet of things security, content centric networks security, cryptographic key establishment and management and wireless sensor networks security. Dr Khan is guest editor of two special issues in the journal of wireless communication and mobile computing.

PLACE PHOTO HERE

DR AHMED IBRAHIM ALZAHRANI is currently an associate professor at computer science department, community college, King Saud University. He acts as the head of the informatics research group, and member of the scientific council - King Saud University.

DR OSAMA ALFARRAJ Osama Alfarraj is an Associate Professor of Computer Sciences at King Saudi University in Riyadh, Saudi Arabia. He has a PhD degree in Information and Communication Technology from Griffith University in 2013. He obtained a Master degree in the same field from Griffith University in 2008. His current research interests include eSystems (eGov, eHealth, ecommerce, etc..), Cloud Computing, Big Data. He also served for 2 years as a consultant and a member of the Saudi National Team for Measuring E-Government in Saudi Arabia.

PLACE PHOTO HERE

DR NASSER ALALWAN is an assistant professor of computer science at computer science department, Community College, King Saud University. His main research interests include database, semantic web, ontology, electronic and mobile services and information technology management.

DR ALI AL-BAYATTI is an Associate professor at De Montfort University. He is the subject leader of Cyber Security at the Cyber Technology Institute. He was awarded his PhD in Computer Science at 2009, and worked with leading organisations such as Deloitte, Airbus, Elektrobit Automotive and Rolls-Royce, among others. Ali's current research is multi-disciplinary, it includes Vehicular Ad hoc Networks, Driver Behaviour, Cyber Security and Smart Technologies that promote collective intelligence. Applications range from promoting comfort to enabling safety in critical scenarios. Ali serves on multiple Editorial Boards and also, is on the Scientific Advisory Boards of multiple institutes in Gulf and Europe. He is also, a visiting professor at multiple institutes and member of the Oman research council. Ali is one of the main factors behind a generated annual income of £1.2 Million at the Cyber Technology Institute.