Research Article

International Journal of Distributed Sensor Networks

International Journal of Distributed Sensor Networks
2016, Vol. 12(12)

DOI: 10.1177/1550147716682037
ijdsn.sagepub.com
SAGE

eDAAAS: Efficient distributed anonymous authentication and access in smart homes

An Braeken[1], Pawani Porambage[2], Milos Stojmenovic[3] and Lambros Lambrinos[4]

Abstract

The smart home field has witnessed rapid developments in recent years. Internet of Things applications for the smart home are very heterogeneous and continuously increasing in number, making user management from a security perspective very challenging. Moreover, the resource-constrained nature of most of the devices implies that any security mechanisms deployed should be lightweight and highly efficient. In this article, we propose an authentication scheme based on symmetric key cryptography, combined with a capability-based access control system, to provide the different stakeholders (residents, recurring guests, or temporary guests) end-to-end secure access to the Internet of Things devices in a smart home, managed by the home owner in an anonymous way. The operations in our scheme only include a small number of communication phases and protect the identities of the entities involved (i.e. stakeholders and end-nodes) from any outside entity. The proposed scheme ensures that even if the stakeholder's device or the Internet of Things device is attacked, the system remains secure.

Keywords

Anonymity, access control systems, lightweight security protocol, authentication, Internet of Things

Date received: 22 June 2016; accepted: 1 November 2016

Academic Editor: Pardeep Kumar

Introduction

With the arrival of the Internet of Things (IoT), anything, anywhere, and anytime connections have become a reality. Consequently, a whole new generation of services and applications can be developed. However, special attention must be paid to the security and privacy aspects from both end-user and device perspectives. Moreover, these measures should be implemented in a highly efficient way due to the power and processing constraints, which characterize many IoT devices.

In many proposed solutions in the literature, node access is managed through a reasonably powerful gateway. However, this approach does not offer end-to-end security and leads to a single point of failure. Making the end-nodes responsible for the access control requires highly efficient access control mechanisms due to the constrained nature of these devices. Therefore, traditional techniques cannot be immediately utilized. Moreover, the standard security mechanisms and protocols for providing authenticated and authorized access need to be adapted toward more lightweight versions in order to make a distributed approach feasible.

[1]Department of Industrial Engineering (INDI), Vrije Universiteit Brussel, Brussels, Belgium
[2]Centre for Wireless Communications (CWC), University of Oulu, Oulu, Finland
[3]Univerzitet Singidunum, Belgrade, Serbia
[4]Cyprus University of Technology, Limassol, Cyprus

Corresponding author:
An Braeken, Department of Industrial Engineering (INDI), Vrije Universiteit Brussel, Nijverheidskaai 170, Brussels 1070, Belgium.
Email: abraeken@gmail.com

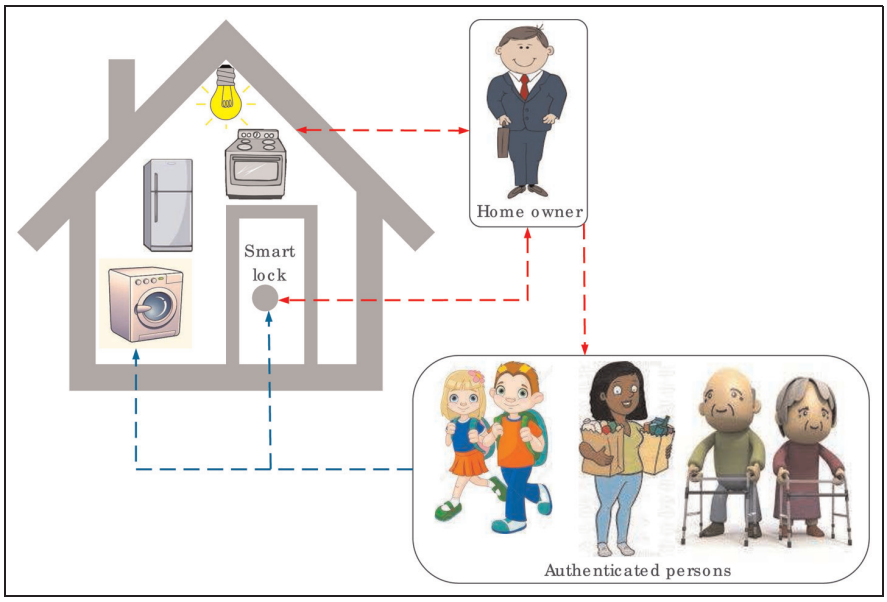

Figure 1. Use-cases of eDAAAS in smart home setting.

Figure 1 illustrates a number of scenarios implemented in the smart home setting. First, the smart locks should grant access to authenticated parties, authorized by the home owner. Next, several electrical devices (e.g. washing machine and dryer) should only be activatable by the residents, and not by temporary or recurring guests. Also the home owner should be able to remotely monitor the operation of some electrical devices (e.g. television and computer).

In this article, we describe a highly efficient authenticated access control mechanism between the stakeholders and the end-nodes, which can be easily coupled with any type of access control model. The proposed solution is entirely based on symmetric key cryptographic operations. In order to provide a good benchmark with previous research, we couple our authentication mechanism with the capability-based access control (CBAC) system in Hernandez-Ramos et al.,[1] but include small changes pertaining to the capability token. In cases where there is a very large network of nodes, one could also easily adopt a capability role-based approach by just changing the content and structure of the token.[2]

Note that our authentication procedure is completely symmetric key based and contains the additional important properties of anonymity and unlinkability of the user and the end-node to any outsider. Thanks to this property, user profiling can be avoided based on the behavior of the accessed nodes. Since the end-nodes cannot derive the real identity of the user, leakage of the user's information is not possible even in instances where the end-node is compromised.

Two-factor authentication is realized in our scheme, based on the password of the user and the possession of the user's device. Even if the user's device is stolen, the stored secret information cannot be abused by the intruder due to its particular construction. All of the other solutions proposed in the literature exploit expensive public key solutions including elliptic curve cryptographic (ECC) operations to obtain authentication and anonymity in the access control system.

The remainder of the article is organized as follows: section "Related work" provides an overview of the related work. Section "System model and assumptions" explains the system model and some assumptions used. In section "eDAAAS system operation," we explain the different phases of our scheme in detail. Sections "Security analysis" and "eDAAAS efficiency assessment," respectively, describe the security analysis and the efficiency of the protocol. Finally, section "Conclusion" presents our conclusions.

Related work

Many publications have been presented on either authentication or access control models in communication systems. Recently, a number of approaches combining both authentication and access control have been proposed in the application domains of wireless sensor networks (WSNs) and in the context of the IoT. We now discuss related work in each of these domains.

Authentication

Paying particular attention to authentication in WSNs to symmetric key based systems, we distinguish two types of approaches. The first approach is based on secret-sharing, where devices exploit symmetric key

threshold schemes (as described in Benenson et al.[3] and Banerjee and Mukhopadhyay[4] for authenticated querying in WSNs, allowing a group of t sensors to collaborate in order to make a decision on the authentication of a query). The second approach follows the domain of smartcard-based authentication protocols, where many interesting proposals have been recently made solely based on symmetric key cryptography. Most of them were vulnerable to man-in-the-middle attacks and impersonation attacks due to faulty constructions[5–11] or consist of a high number of communication rounds.[12–14] In Baruah et al.,[15] a strong and secure authentication algorithm has been proposed, which additionally guarantees the anonymity of the user and consists of only one communication round. The property of user untraceability has been added to this last one in Braeken.[16] Our proposal uses the framework of Braeken[16] for the integration of the access control mechanisms.

Access control

Access control in WSNs can be divided among two major architectural categories: distributed or centralized. In a distributed access control mechanism, the final access decision is made in the end-device and not at the gateway as opposed to the centralized approach.

Centralized access control systems[17,18] have several severe disadvantages. First, they are not able to make decisions based on contextual information related to the end-device itself since the end-nodes can be seen as smart devices. Second, the central gateway, which stores and manages all information of every device, becomes a single point of failure.[1] Moreover, a centralized system layout has a higher degree of network dependency, adding and removing of nodes is more complicated compared to a decentralized system.[19] The two traditional access control systems, role-based (RBAC) and attribute-based (ABAC) access control, are mostly applied in centralized architectures.

Each application requires its own type of granularity for access control. We refer to Maw et al.[20] for a summary and comparison of different access control systems in WSNs and to Adda et al.[21] for some recent access control models. In particular, based on a user study, intuitive access control policies for future smart homes have been presented in Kim et al.[22] and Ur et al.[23]

Authentication and access control

Two approaches for authenticated end-to-end distributed access control using CBAC can be found in Hernandez-Ramos et al.[1] and Mahalle et al.[24] In Hernandez-Ramos et al.,[1] ECC is used to sign the tokens, while in Mahalle et al.[24] a secret shared key is first negotiated using an ECC-based key agreement protocol. The system in Hernandez-Ramos et al.[1] does not perform data authentication, whereas the solution of Mahalle et al.[24] consists of an overload of communication flows. Another distributed access control system[25] uses ABAC in combination with an ECC-based key agreement protocol to enable authentication. None of these schemes[1,24,25] take the property of anonymity into account.

In Hernndez-Ramos et al.,[26] an anonymous end-to-end access system is built in combination with a corresponding type of CBAC access control system, called DCApBAC.[27] In order to establish the anonymity feature, the classical crypto-systems identity-based encryption (IBE) and ciphertext-policy attribute-based encryption (CP-ABE) are used. These systems apply public key–based mechanisms, which require very high-performance capabilities.

In summary, considering the proposals we referred to, none are able to establish authenticated access control with anonymity of the user, using only symmetric key–based operations.

System model and assumptions

We designed a system that performs anonymous authentication for IoT devices based on symmetric key cryptography. Following its purpose and functionality, our system is called efficient distributed anonymous authentication and access control for smart home sensors using symmetric key cryptography (eDAAAS).

Network setting

In our system model design, we distinguish five different entities as follows: the user (U), the user's device (U_d), the gateway (GW), the IoT devices or end-nodes (ENs) in the smart home, and the home owner (O) of the node(s). The user corresponds with the stakeholder (resident, recurrent, or temporary guest) and the home owner is responsible for the management of the authorization of the ENs.

Thus, the owner is able to serve the access control process of the nodes and thus perform the task of the registration center. The role of the gateway is passive, in the sense that it only forwards the messages to the nodes. All the nodes in the network need to be preinstalled with some secure information by the owner. Furthermore, the user first needs to register with the owner in order to receive the capability token and key material related to the particular node or a set of end-nodes. The capability token and secret key material are defined by the owner and stored on the user's device. The user needs to log into their device in order to get access to this security material and to initiate its authentication request.

After successfully logging in, the user can send authenticated and anonymous requests, including a capability-based token, to the nodes in the network without the involvement of the owner. The ENs are able to check the validity of the token and the authentication of the user. If both are positive, a message containing the required information is sent to the user. The actual process to request information by a user to a node only requires one message, followed by only one message for the response in case of successful authentication and access control.

Design goals

Our system, eDAAAS, has the following security features inherently built into its design:

- Data authentication. The source of the information can be corroborated and it is ensured that the information has not been altered by unauthorized or unknown means.
- Anonymity. No outsider is able to derive the identity of the user or the end-node. Note that the system also satisfies the unlinkability property of the behaviors of users and end-nodes with respect to outsiders. Links are still possible at the level of ENs in order to facilitate the detection of malicious behavior.
- Access control. The owner derives the unique token for a particular end-node, which is required in order to gain access to it. This token cannot be altered by any other instance.

We also base our system on a distributed architecture, meaning that end-to-end decisions about the access control and authentication are made at the actual IoT device. Moreover, in order to obtain an efficient system with respect to timing and energy consumption, we limit the operations used in the protocol to hashes, concatenations, and symmetric key encryptions/decryptions. Finally, we make sure that lost user devices, eventually combined with attacked end-nodes, are not able to cause damage to the rest of the system.

In section "Security analysis," we describe how eDAAAS meets the design goals with respect to the above-described security features. Note that due to the particular construction of the key material, no password tables of end-nodes or users need to be stored at the owner. The user's password is easy to change without involvement of any other entity in the system.

Attack model and assumptions

We focus on the communication between users and end-nodes. For the communication among the other entities, we assume the existence of a secret shared key. These keys can be established either by physical contact or by more computer-intensive public key infrastructure (PKI) mechanisms. As these methods are well known, we do not focus on them in this article. Consequently, the following notations are used for the secret shared keys:

- *Between owner and user.* $k_{uo} = H(H(ID_i^r) \parallel K_m)$, where ID_i^r represents the real identity of the user (e.g. by having a direct link with the name) and K_m the master key of the owner. Note that the concatenation of the values m_1 and m_2 is denoted by $m_1 \parallel m_2$.
  This key is stored on the user's device in an encrypted format, $E_K(k_{uo})$, which represents the encryption by means of the secret key K. Consequently, if the user's device is stolen, an attacker is not able to derive this shared secret key from the owner. In order to define the encryption key K, the device generates a random value b, which is also stored on the device. Next, the user enters their username ID_i^r and password PW_i. This information (b, ID_i^r, PW_i) allows the device to compute

$$RPW_i^r = H(PW_i \oplus b)$$
$$ID_i = H(ID_i^r)$$

  and thus derive $K = H(ID_i \parallel RPW_i^r)$. Consequently, the decryption of the stored value $E_K(k_{uo})$, denoted by $D_K(E_K(k_{uo}))$, will lead to the key k_{uo}.
- *Between owner and end-node* V_j. $k_{eo} = H(V_j \parallel K_m)$, where V_j represents the identity of the node and K_m the master key of the owner. Note that the identity V_j is not confidential. It can, for instance, represent a derivation of the uniform resource identifier (URI) of the device.
  This key is stored in the end-node.

We further assume that the owner is a trusted party in the form of a key distribution center (KDC). The goal of the attackers is to obtain illegitimate data or control access to the nodes, to perform service degradation or denial-of-service (DoS) attacks. The attackers may come from inside or outside the network. They are able to eavesdrop on the traffic, inject new messages, replay and change messages, or spoof other identities.

Notations

Table 1 summarizes the most frequently used notations and abbreviations in the article.

eDAAAS system operation

Different phases can be distinguished: (1) the installation phase of end-nodes, (2) the registration of the user

Table 1. Summary of notations and abbreviations.

U, U_d	User and User's device
GW	Gateway
ENs	IoT devices or end-nodes
O	Home owner
K_m, x, y	Master key and two other secrets of O
ID_i^r, PW_i	Real username and password of U
$ID_i = H(ID_i^r)$	Derivated identity of U
V_j	Identity of end-node j
$k_{uo} = H(H(ID_i^r) \| K_m)$	Secret shared key between O and U_d
$k_{eo} = H(V_j \| K_m)$	Secret shared key between O and EN

IoT: Internet of Things.

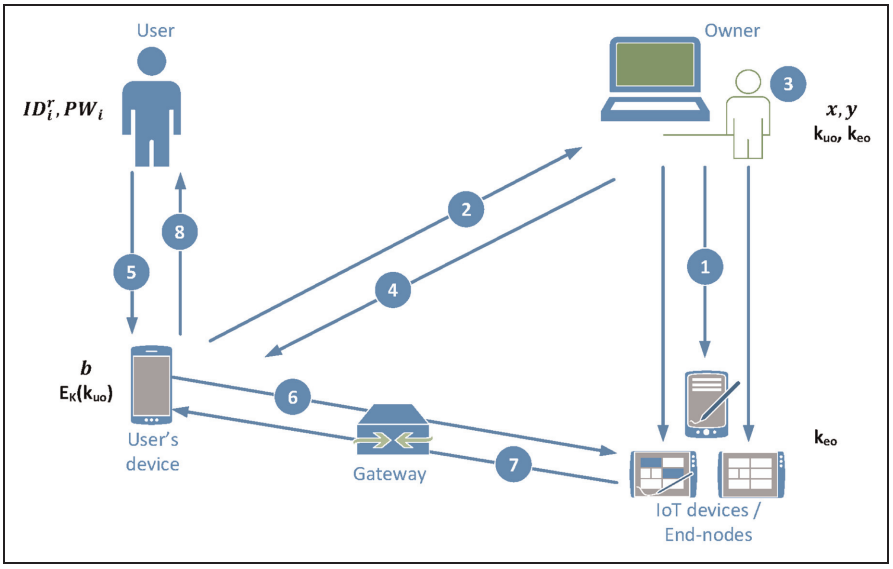

Figure 2. Overview of eDAAAS.

with the owner, (3) the definition of the capability token by the owner, (4) the derivation of secure key material by the owner, (5) the login of the user on their device, (6) the actual request by the user with the IoT device, (7) the answer by the end-node, and (8) information retrieval by the user. Figure 2 gives an overview of the protocol among each entity under the different phases.

The italic parameters in the figure denote the secret values that are randomly determined by the corresponding entities as will be explained in the next phases. The other parameters represent the secret shared keys that are assumed to be established beforehand (as explained in section "Notations"). Besides these eight phases, there are two other phases with only limited use: (9) update of user's password and (10) revocation of the capability token.

We now discuss each phase in detail.

Installation phase of the end-nodes

Let x, y be two secrets chosen by the owner. V_j denotes the identity of the end-node j. The owner shares the parameters $y, V_j, H(x \| y), H(V_j \| H(x))$ with each node in the network using the pre-installed or pre-shared secret key k_{eo} between end-node and owner. Note that y is still a secret parameter for the user.

Registration phase

Here, two actions can be distinguished, activation of the device by the user and submission of the registration request:

- As already mentioned in section "Notations," the user has selected a random value b, its identity ID_i^r, and password PW_i. Only the random value b, together with the encrypted shared secret key with the owner $E_K(k_{uo})$, are stored on its device.
  If the user enters ID_i^r, PW_i on its device, then U_d computes $RPW_i^r = H(PW_i \oplus b)$ and $ID_i = H(ID_i^r)$. Using the information of ID_i and RPW_i^r, the stored encrypted shared key with the owner, k_{uo}, can be decrypted by deriving the corresponding decryption key $K = H(ID_i \| RPW_i^r)$.

- Finally, the registration request consists of the message registration with the owner, requesting access *reqAcc* to a particular device V_j. Note that *reqAcc* contains a time stamp in order to avoid replay attacks

$$ID_i \parallel E_{k_{uo}}(ID_i^r \parallel RPW_i^r \parallel reqAcc)$$

Definition of the capability token

The concept of capability was originally introduced in Dennis and Horn[28] and defined as a tamper-proof token, ticket, or key that gives the possessor permission to access an entity or object in a computer system. In Hernandez-Ramos et al.,[1] a concrete example of a capability is described and implemented. It consisted of the following information:

- Identifier (16 bytes), allowing to distinguish the capability token.
- Issued time (10 bytes), describing the time at which the token was issued. This time follows the time stamp of *reqAcc*.
- Issuer (Variable size), referring to the identity of the owner, the entity that issued the token.
- Subject (56 bytes), giving information on the subject or user to which the token is granted.
- Device (variable size), consisting of an URI to identify the end-node to which the token applies.
- Signature (56 bytes) of the token.
- Access rights (variable size), representing the set of rights (GET, POST, PUT, DELETE) that an issuer has granted to the different resources of the end-node or the functionalities of the end-nodes, taking eventually into account context-aware criteria.
- Not before (10 bytes), indicating the starting time for the validity of the token.
- Not after (10 bytes), representing the end time of the token.

In our system, due to the particular distribution of the key material, we are able to remove the largest entity of the token, being the signature. As we want to enable anonymous requests, the subject should also be removed. However, since we want to establish accountability in case of misuse or abuse of the system, the owner encrypts the subject information using their own secret key. Another, however, much less elegant solution would be to store lists at the owner side, containing all identities of users and tokens.

Installation phase of the users

The secret shared key $k_{uo} = H(ID_i \parallel K_m)$ is computed and after decryption of the received message, the owner checks the identity ID_i^r by verifying if $ID_i = H(ID_i^r)$ of the user and defines the corresponding capability token T_i, following the format described above. A derived form of the password $RPW_i = H(RPW_i^r)$ is used for the computations of the secret key material using its secret parameters x, y

$$A_i = H(y \parallel T_i)$$
$$B_i = H(x \parallel y) \oplus A_i$$
$$C_i = H(RPW_i \parallel ID_i) \oplus H(A_i)$$
$$D_i = H(x) \oplus H(ID_i \parallel RPW_i)$$
$$E_i = RPW_i \oplus H(ID_i \parallel RPW_i)$$

Note that each of these parameters A_i, B_i, C_i, D_i, E_i play a particular role, which will be further exploited in the other phases of the protocol.

The owner sends the message $E_{k_{uo}}(B_i, C_i, D_i, E_i, T_i, H(B_i \parallel C_i \parallel D_i \parallel E_i \parallel T_i))$ to the user's device. After decryption and successful verification of the hash value, the parameters B_i, C_i, D_i, E_i, T_i will be stored on the user's device.

Figure 3 gives a graphical overview of the steps in the registration phase and the installation phase.

User login phase

Here, the user enters its identity ID_i^r and password PW_i into the device. The device first computes $ID_i = H(ID_i^r)$ and $RPW_i = H(RPW_i^r)$. Then, U_d verifies whether $H(ID_i \parallel RPW_i^r) \oplus RPW_i$ equals the stored parameter E_i.

Request phase

This phase is executed only after a successful user login. First, the user selects V_j from its token as the device to contact. Then, the following computations are performed by the device

$$H(x) = D_i \oplus H(ID_i \parallel RPW_i)$$
$$H(A_i) = C_i \oplus H(RPW_i \parallel ID_i)$$

These two steps are required to make the link with information, which is derivable by the end-node. First, from $H(x)$, the value $H(V_j \parallel H(x))$ can be constructed by the user, which is also stored at the end-node. Next, as $H(x \parallel y)$ is stored at the end-node, the end-node can find A_i from B_i due to the construction of B_i. Consequently, the following operations are performed to construct a secret shared key and corresponding cipher text. We denote a random nonce and request by N_i and *Req*, respectively. In order to avoid replay attacks, the *Req* parameter also contains a time stamp

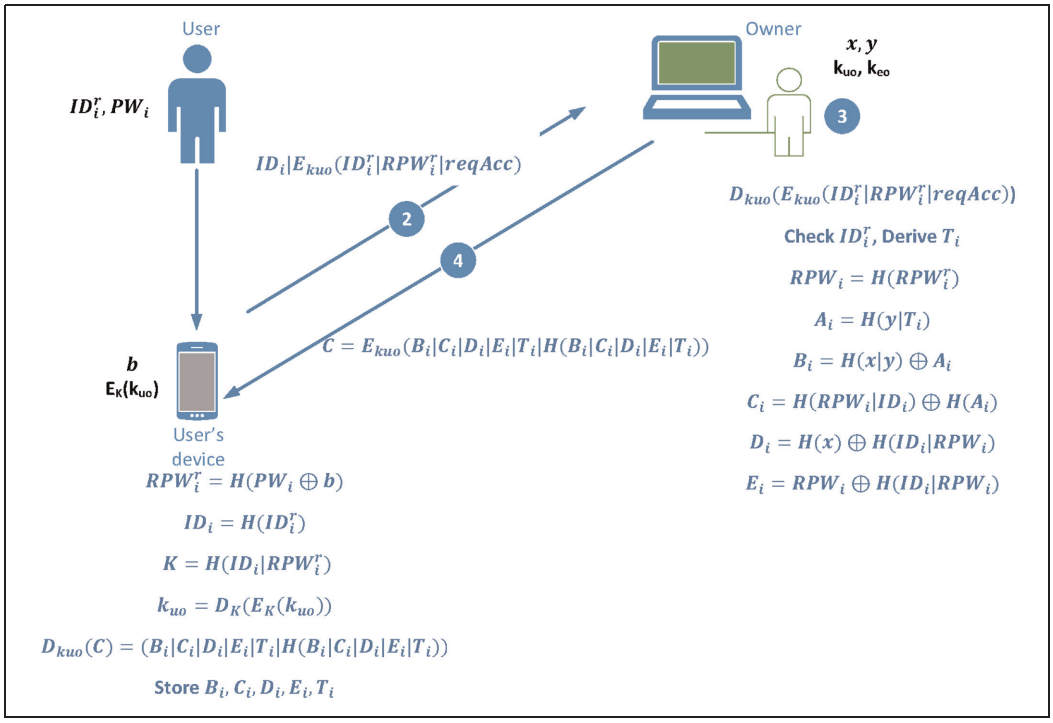

Figure 3. Overview of the steps in the registration phase (2) and installation phase (4).

$$
\begin{aligned}
C_1 &= H(V_j \parallel H(x)) \oplus H(ID_i \parallel N_i) \\
C_2 &= H(A_i) \oplus N_i \\
V_1 &= H(N_i \oplus B_i) \\
CID_i &= B_i \oplus H(H(V_j \parallel (x)) \parallel H(ID_i \parallel N_i)) \\
K &= H(N_i \parallel V_j \parallel H(A_i)) \\
& E_K(T_i \parallel C_2 \parallel Req)
\end{aligned}
$$

The following message is sent to the gateway, which forwards the message further to the corresponding node V_j

$$CID_i \parallel C_1 \parallel C_2 \parallel V_1 \parallel E_K(T_i \parallel C_2 \parallel Req) \qquad (1)$$

Answer phase

Here, V_j executes the following operations using the values which are stored in memory

$$
\begin{aligned}
H(ID_i \parallel N_i) &= H(V_j \parallel H(x)) \oplus C_1 \\
B_i &= CID_i \oplus H(H(V_j \parallel H(x)) \parallel H(ID_i \parallel N_i)) \\
A_i &= B_i \oplus H(x \parallel y) \\
N_i &= H(A_i) \oplus C_2 \\
V_1^* &= H(N_i \oplus B_i)
\end{aligned}
$$

If V_1^* corresponds to the transmitted V_1 value, the user is authenticated. Next, the key is derived by $K = H(N_i \parallel V_j \parallel H(A_i))$, such that the decryption $D_K(T_i \parallel C_2 \parallel Req)$ can be executed. If $A_i = H(y \parallel T_i)$ holds, then the access token origin of T_i is validated. If T_i satisfies the conditions of the end-node regarding timing, services, functionalities, context, and so on, the request can be answered. In this case, the following computations are performed with random value N_j

$$
\begin{aligned}
C_3 &= N_j \oplus H(ID_i \parallel N_i) \\
V_2 &= N_i \oplus H(V_j \parallel B_i \parallel N_j) \\
SK_{ij} &= H(N_i \parallel N_j)
\end{aligned}
$$

Finally, the message $C_3 \parallel V_2 \parallel E_{SK_{ij}}(M)$, with M, the requested info (GET request) or a confirmation (POST, DELETE, PUT request) is sent to the user.

Information retrieval

The user first derives $N_j = C_3 \oplus H(ID_i \parallel N_i)$. Next $N_i \oplus H(V_j \parallel B_i \parallel N_j)$ is calculated and compared with the transmitted value V_2. If positive, mutual authentication is obtained and the shared symmetric key can be derived in order to decrypt the last part of the message.

Figure 4 gives a graphical overview of the steps in the user login, request, and answer phases.

Update of user's password

This can be easily executed by the user, without involvement of the owner or changes to the other nodes in the system. The user needs to compute a new value of the password, leading to an updated version of RPW_i. This

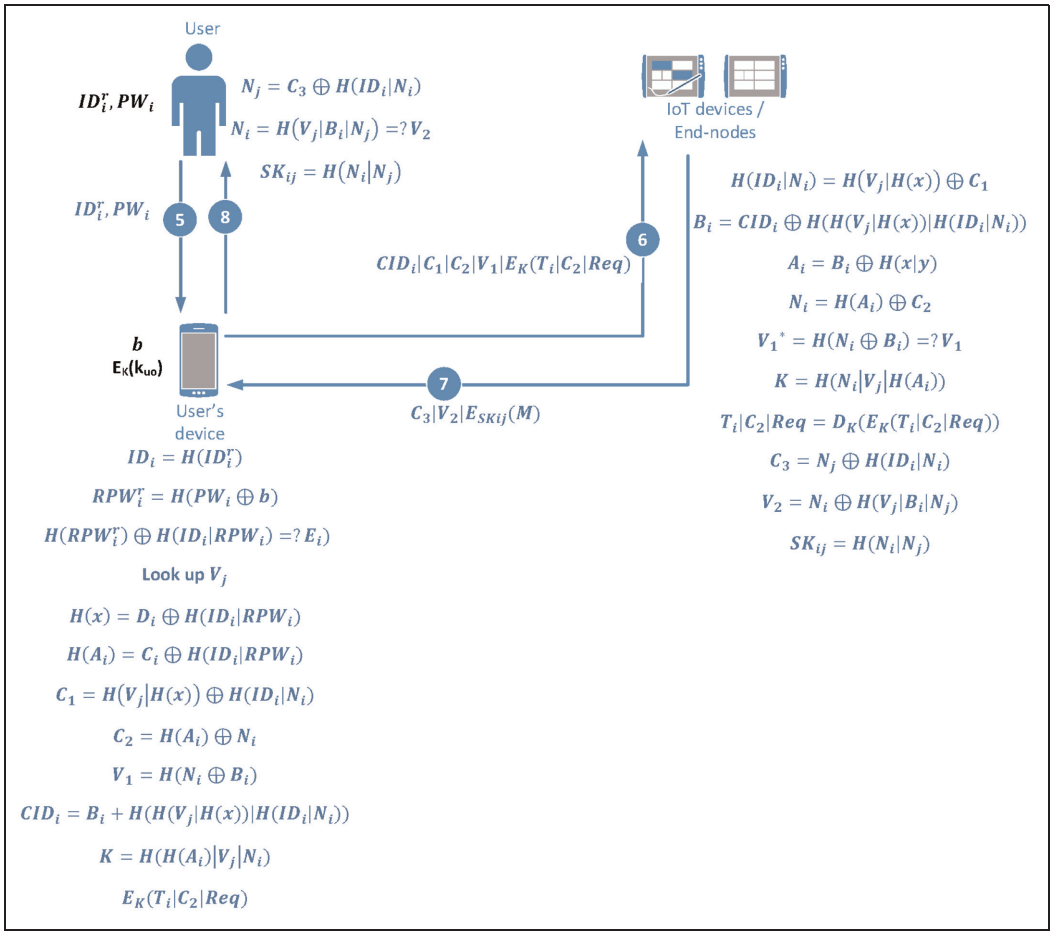

Figure 4. Overview of the steps in user login phase (5), request phase (6), and answer phase (7).

update implies new values of C_i and E_i, which are then stored on the user's device. Note also that the encryption key K of k_{uo} needs to be updated, leading to an update of $E_K(k_{uo})$.

Revocation of the capability token

Suppose the validity of the token's capability expires before the end date of the token. In this case, the identity of the token is sent to the involved nodes in the field using the secret shared key between owner and end-node or a multicast key,[29] keeping track of a black list of identity tokens. As soon as the expiration date is reached, the token can be removed from the list. Note that this implies a trade-off between user friendliness and storage. Consequently, the decision on the end date of a token should be carefully considered and strongly depends on the application, user, or size of the network.

Security analysis

Fulfilling its purpose, eDAAAS protects the system entities from a range of attacks which we describe here in detail.

Replay attacks

Replay attacks in the registration phase are avoided by the inclusion of a time stamp in the *reqAcc* parameter. The issued time in the capability token should follow this time stamp. An attacker is not able to decrypt the registration request or the corresponding answer and thus cannot change this value.

Also a replay attack on the user's request to the IoT end-node is impossible due to the presence of a time stamp in the request message.

Illegitimate data access or control

A user cannot derive a token itself, since the token is included in the computation of the parameter A_i. From A_i, parameter B_i is derived. As a user does not know the secrets x, y, he is unable to find valid constructions for both T_i, A_i, B_i. An adversary cannot get access to a node, even if he is in the possession of the user's device. This follows from the two-factor authentication, where identity and password on the corresponding device are required to further proceed the process.

Accountability

It is important to install a logging mechanism in each node. Each log contains the parameters $T_i \parallel Req$. In the token T_i, there is a field containing the encrypted value of the identity of the user. This field gives no direct information to a certain identity without the knowledge of the secret key of the owner. However, by keeping track of the same fixed value, abnormal behavior leading to service degradation and denial of service attacks can be easily detected. In case of uncertainty, the owner will be contacted to check the validity of the token, to derive the identity, and to update the capability token if needed. Moreover, replay attacks are avoided by using true cryptographic nonces.

In case the owner wants to monitor the users with respect to the behavior of the IoT devices in real time, these logs should be sent to the owner.

Insider attacks

We distinguish the insider attacks by the impact of five different situations, which are dependent on the combination of compromised objects and users:

- Compromised user's device. Here, the adversary is not aware of the user's identity and password. Consequently, the process cannot be continued. Even if the adversary is able to retrieve the stored information on the device, being $B_i, C_i, D_i, E_i, T_i, H(B_i \parallel C_i \parallel D_i \parallel E_i \parallel T_i)$, it is still impossible to formulate a valid request as the identity and password are required.
- Compromised user + device. In this case, the user is able to perform any type of action that is permitted by the token, as long as the attack is not noticed. As explained before, as soon as suspicious behavior is detected, the effective origin of the user can be retrieved. Note that a new user with an associated valid token cannot be created by such a compromised user, since the secret key y is not known.
- Compromised end-node. A compromised end-node can send no or incorrect information. It cannot release critical identity–related information of its users, since the received information is only indirectly associated with the user's identity. A compromised node has insufficient information to create users that can perform valid requests to other nodes, since the other secret key x is not known.
- Compromised device + end-node. Since the identity of the owner is unknown, it is not possible to derive $H(x)$ from D_i. Consequently, no valid requests to other nodes can be performed nor can useful information be retrieved from other requests.
- Compromised user + device + end-node. When the user, its device, and one of the end-nodes are compromised, then the system is completely broken. We can consider the likelihood of such a combination of events happening to be very rare.

HW/SW attacks

The system is based on a security protocol for tamper-resistant smart cards. The same ideas are applied at the side of the user. Even with the knowledge of the parameters $B_i \parallel C_i \parallel D_i \parallel E_i \parallel T_i$, an attacker has no further advantage since the input of the identity and password of the user is required, corresponding to the two-factor authentication feature.

Note that the attacker in this situation can still launch, given the parameter E_i, a dictionary attack on the identity and password of the user. In order to increase security in this case, one could consider to also include the biometric characterization of the user, in addition or instead of the password. This will make the system stronger, but more costly.

Mutual authentication

Mutual authentication between the user and the end-node is obtained since the constructed secret shared key SK_{ij} is built using nonces generated by the user and the end-node. As explained before, only the user and the end-node have the required credentials to generate correct requests and answers for the construction of this key.

Anonymity and unlinkability

Note that the user's request contains the parameter CID_i, which is a dynamic reference, related to the hidden identity B_i of the user. Consequently, no outsider can link the different requests to a particular user or to the same user. This also guarantees the location privacy of the user for any outside attacker.

From the request, the end-node can derive the indirect link B_i with the user's identity. Only the owner is able to retrieve the real identity, since T_i contains the encrypted identity of the user. Note that in contrast to an outsider, the end-node does have the possibility to link the requests to the same user. This feature is needed in order to facilitate abnormal behavior detection.

eDAAAS efficiency assessment

The efficiency of the eDAAAS solution is discussed, taking into account that the end-node is the most

Table 2. Comparison with related work.

System	Year	Access control	Type of operations	Tamper proof	Anonymity	Unlinkability
Ours	2016	CBAC	Symmetric key	Yes	Yes	Yes
Hernandez-Ramos et al.[1]	2013	CBAC	ECDSA	No	No	No
Mahalle et al.[24]	2013	CBAC	ECC	No	No	No
Ye et al.[25]	2014	ABAC	ECC	No	No	No
Hernndez-Ramos et al.[26]	2015	CBAC	CP-ABE (pairing)	No	Yes	No

CBAC: capability-based access control; ABAC: attribute-based access control; ECC: elliptic curve cryptographic; CP-ABE: ciphertext-policy attribute-based encryption; ECDSA: elliptic curve digital signature algorithm.

resource-constrained device. The reception of the user's request and the according response correspond each with a single message phase. Furthermore, since the operations in each phase are limited to xors, hashes, and symmetric encryptions, the computation complexity of the proposed security solution is relatively low.

In order to obtain 80-bit security, the length of the parameters A_i, B_i, C_i, D_i, E_i, CID_i, C_1, C_2, V_1 should be at least 20 bytes. Consequently, the length of the received request (equation (1)) counts 100 bytes plus the variable length of the capability token.

After reception of the message, 5 xors, 5 hashes, and 1 decryption need to be performed. After checking the validity of the token, the response generation requires another 2 hashes, 2 xors, and 1 encryption. The length of the transmitted message corresponds to a maximum of 56 bytes, when limiting the output to 16 bytes. Moreover, the storage requirements are reasonably small. Besides the secret material at each node, $y, V_j, H(x \parallel y), H(V_j \parallel H(x))$, a shared secret key k_{eo} between node and KDC, and an additional black list of token identities should be stored.

In order to elaborate an insight of the computation time of the token at the end-node (EN), we show the timing and energy consumption values with respect to the Libelium Waspmote platform.[30] Therefore, neglecting the timing values for xor operations, we obtain for the 7 hash computations and 1 encryption during the token computation, approximately 369.163 ms as time consumption and 15.7 mJ as energy consumption. (Timing values for SHA-224 hash function and AES-128 CBC PKCS encryption are, respectively, 50.86 and 13.143 ms. Energy consumption is computed by using the formula $U \times I \times t$ based on the execution time (t), the nominal voltage (U), and the current draw in active mode (I) on Waspmote sensors. Energy consumption for SHA-224: 4.2 V $\times$ 10.1 mA $\times$ 50.86 ms. Energy consumption for encryption: 4.2 V $\times$ 10.9 mA $\times$ 13.143 ms.)
Moreover, the minimum token size is calculated to be equal to 73 bytes.

Finally to conclude, we compare the most important characteristics with respect to security features and efficiency for the other existing authentication and distributed access control systems, as discussed in related work in Table 2. Consequently, our proposal is the only symmetric key–based solution in the literature that is able to establish anonymity and unlinkability in an authenticated distributed access control system, leading to a very efficient and highly capable system.

Conclusion

We present the eDAAAS protocol in this article, which is a highly efficient and distributed authentication protocol to access end-nodes in an IoT setting for smart homes, authorized by the home owner. The efficiency is thanks to the fact that only symmetric key–based operations are required. Due to the particular construction of the keying materials, the additional features are also obtained such as anonymity and unlinkability of the user and end-node for any outsider. In addition to that, the authentication mechanism can be easily combined with other access control modes.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work is supported by the Short Term Scientific Mission performed under COST Action IC1303 AAPELE. This work is also partially supported by project TR32054 of the Serbian Ministry of Science and Education.

References

1. Hernandez-Ramos JL, Jara AJ, Marn L, et al. Distributed capability-based access control for the internet of things. *J Internet Serv Inf Secur (JISIS)* 2013; 3: 116.
2. Hasebe K, Mabuchi M and Matsushita A. Capability-based delegation model in RBAC. *In: Proceedings of the 15th ACM symposium on access control models and*

technologies (SACMAT 10), Pittsburgh, PA, 9–11 June 2010, pp.109–118. New York: ACM.
3. Benenson Z, Gedicke N and Raivio O. Realizing robust user authentication in sensor networks. In: *Proceedings of the workshop on real-world wireless sensor networks (REALWSN)*, Stockholm, 20–21 June 2005.
4. Banerjee S and Mukhopadhyay D. Symmetric key based authenticated querying in wireless sensor networks. In: *Proceedings of the first international conference on integrated internet ad hoc and sensor networks (InterSense 06)*, Nice, 30–31 May 2006.
5. Banerjee S, Dutta MP and Bhunia CT. An improved smart card based anonymous multi-server remote user authentication scheme. *Int J Smart Home* 2015; 9(5): 11–22.
6. Li CT and Hwang MS. An efficient biometrics based remote user authentication scheme using smart cards. *J Netw Comput Appl* 2010; 33(1): 1–5.
7. Chuang MC and Chen MC. An anonymous multi-server authenticated key agreement scheme based on trust computing using smart cards and biometrics. *Expert Syst Appl* 2014; 41(4): 1411–1418.
8. Mishra D, Das AK and Mukhopadhyay SA. Secure user anonymity-preserving biometric-based multi-server authenticated key agreement scheme using smart cards. *Expert Syst Appl* 2014; 41(18): 8129–8143.
9. Das AK. Analysis and improvement on an efficient biometric-based remote user authentication scheme using smart cards. *IET Inform Secur* 2011; 5(3): 145–151.
10. An Y. Security analysis and enhancements of an effective biometric-based remote user authentication scheme using smart cards. *J Biomed Biotechnol* 2012; 2012: 1–6.
11. Khan MK and Kumari S. An improved biometrics-based remote user authentication scheme with user anonymity. *J Biomed Biotechnol* 2013; 2013: 1–9.
12. Chang C-C, Lee C-Y and Chiu Y-C. Enhanced authentication scheme with anonymity for roaming service in global mobility networks. *Comput Commun* 2009; 32(4): 611–618.
13. Lee T-F. User authentication scheme with anonymity, unlinkability and untrackability for global mobility networks. *Secur Comm Network* 2013; 6(11): 1404–1413.
14. Chung Y, Choi S and Won D. Anonymous authentication scheme for intercommunication in the internet of things environments. *Int J Distrib Sens N* 2015; 2015: 1–13.
15. Baruah KCH, Banerjee S, Dutta MP, et al. An improved biometric-based multi server authentication scheme using smart card. *Int J Secur Appl* 2015; 9(1): 397–408.
16. Braeken A. Efficient anonym smart card based authentication scheme for multi-server architecture. *Int J Smart Home* 2015; 9(9): 177–184.
17. Jing L, Xiao Y and Chen CLP. Authentication and access control in the internet of things. In: *Proceedings of 32nd international conference on distributed computing systems workshops*, Macau, China, 18–21 June 2012, pp.588–592.
18. Ndibanje B, Lee HJ and Lee SG. Security analysis and improvements of authentication and access control in the internet of things. *Sensors* 2014; 14: 14786–14805.
19. Johansson D, Andersson K and Ahlund C. Supporting user mobility with peer-to-peer-based application mobility in heterogeneous networks. In: *Proceedings of the 38th IEEE conference on local computer networks workshops (LCN Workshops)*, Sydney, NSW, Australia, 21–24 October 2013.
20. Maw HA, Xiao H, Christianson B, et al. Survey of access control models in wireless sensor networks. *J Sensor Actuator Netw* 2014; 3: 150–180.
21. Adda M, Abdelaziz J, Mcheick H, et al. Toward an access control model for IOTCollab. *Procedia Comput Sci* 2015; 52: 428–435.
22. Kim TH, Bauer L, Newsome J, et al. Challenges in access right assignment for secure home networks. In: *Proceedings of the HotSec'10*, Washington, DC, 11–13 August 2010.
23. Ur B, Jung J and Schechter S. The current state of access control for smart devices in homes. In: *Proceedings of the workshop on home usable privacy and security (HUPS)*, Newcastle, 24–26 July 2013.
24. Mahalle PN, Anggorojati B, Prasad NR, et al. Identity authentication and capability based access control (IACAC) for the internet of things. *J Cyber Secur Mob* 2013; 1: 309–348.
25. Ye N, Zhu Y, Wang RC, et al. An efficient authentication and access control scheme for perception layer of internet of things. *Int J Appl Math Inf Sci* 2014; 8(4): 1617–1624.
26. Hernndez-Ramos JL, Bernabe JB, Moreno MV, et al. Preserving smart objects privacy through anonymous and accountable access control for a M2M-enabled internet of things. *Sensors* 2015; 15(7): 15611–15639.
27. Hernndez-Ramos JL, Jara AJ, Marn L, et al. DCapBAC: embedding authorization logic into smart things through ECC optimizations. *Int J Comput Math* 2014; 93: 1–22.
28. Dennis J and Horn EV. Programming semantics for multiprogrammed computations. *Commun ACM* 1966; 9(3): 143–155.
29. Braeken A. SAK: symmetric-key based and authenticated key management for wireless sensor networks, in preparation.
30. Libelium Inc. Waspmote sensor boards, 2016, http://www.libelium.com/products/waspmote/