Journal of Computer and System Sciences 86 (2017) 59–69

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

TARD: Temporary Access Rights Delegation for guest network devices

Joonghwan Lee [a], Jae Woo Seo [a], Hoon Ko [b], Hyoungshick Kim [b,*]

[a] *Software R& D Center, Samsung Electronics Co., LTD, 262, Umyeon-dong, Seocho-gu, Seoul, Republic of Korea*
[b] *Department of Electrical and Computer Engineering, College of Information & Communication Engineering, Sungkyunkwan University, Republic of Korea*

A R T I C L E I N F O

Article history:
Received 20 March 2016
Received in revised form 28 May 2016
Accepted 19 July 2016
Available online 8 August 2016

Keywords:
Internet of Things (IoT)
Guest access control
Access rights delegation
Temporary access rights
One-way function
One-way function chain

A B S T R A C T

Many cryptographic protocols were developed to support efficient group membership operations. Although those schemes can be extensively used for controlling temporary access with guest devices which do occur in many real world situations, those schemes incur a significant management overhead. For guest devices, we propose a scheme without heavy key management overheads—Temporary Access Rights Delegation (TARD). The proposed scheme uses a cryptographic token that can be securely constructed by a one-way function chain, used for granting a temporary membership to a guest device. Our analysis shows that the proposed scheme is theoretically secure and outperforms the certificate-based authentication schemes.

1. Introduction

The Internet of Things (IoT) opens up new services and opportunities by integrating various devices with web services and cloud computing platforms. Inherently, network protocols are fundamental building blocks for the IoT, which enables devices to properly discover, configure, and communicate with each other. However, the Internet connectivity could also lead to potential risks of cyberattacks which raise serious privacy and security concerns for users. For example, a user's home devices and/or files stored in the devices could be illegally accessed by unauthorized neighbors via a wireless communication channel unless a proper access control scheme for the home network environment is deployed. Surely, some network management schemes for access control are essential to effectively deny unauthorized access to the IoT devices (and/or their services).

There have been many attempts to provide dynamic key management schemes for group membership operations [1–4]. Those schemes have been mainly developed by taking into consideration solely frequent membership changes in a network. In many real-world applications, however, temporary access is needed for guest devices rather than dynamic group membership in a network. That is, guest devices are only allowed access to the network one time or a few times within a limited time window. In this case, general group management schemes can often be too expensive and complicated—a simpler solution is better than general ones.

For instance, when a visitor wants to use a file sharing service within a home network, the home network's owner can allow the visitor to use the service only during his or her stay at home. Another example is valet parking which is a parking

* Corresponding author.
E-mail address: hyoung@skku.edu (H. Kim).

http://dx.doi.org/10.1016/j.jcss.2016.07.002
0022-0000/© 2016 Elsevier Inc. All rights reserved.

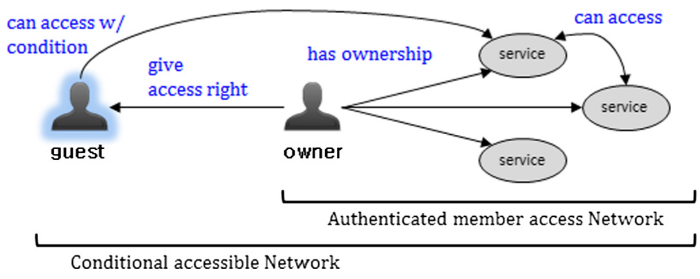


Fig. 1. Authenticated Member Access network consists of a owner and service entities. Conditional Accessible Network, which is a model for temporary access, has an additional guest entity who can be a latent adversary.

service offered by some restaurants, stores, hotels and other places where customers' vehicles are parked by a person called a valet. In such scenarios, an efficient mechanism for managing temporary access is required. In fact, the concept of temporary access is already used in accommodations such as hotels, resorts and hostels protected by an electronic door lock with a digital key that can only be used to open the electronic door lock for a (limited) period of time.

In this paper, we aim to extend the concept of the digital key used for temporary access to generic network applications by introducing a conditional access rights delegation scheme which is significantly more efficient than general group membership schemes (e.g., [5]). Even though several group key protocols can also be used for controlling temporary access to a network, those protocols do not scale up well as the number of group members grow because all group members should be affected whenever the membership changes. The need to update group keys raises an inherent scalability problem in network applications with a lot of group membership operations (i.e., joining or leaving) for guest devices.

We propose a temporary access rights delegation scheme for guest network devices. The key contributions of this paper are as follows:

– We present the Temporary Access Rights Delegation (TARD) scheme to reduce key management overhead for guest network devices. The proposed scheme can be flexibly applied with a generic access control list for more generalized access mechanisms.
– We introduce a formal definition of temporary access security and verify the security properties of the proposed scheme under the assumption that an one-way function exists.
– We particularly demonstrate how to apply the proposed scheme to a real life scenario for home networks to show the feasibility of the proposed scheme.

The rest of the paper is structured as follows. Section 2 introduces the network model and summarizes the cryptographic primitives needed in the implementation of TARD. Section 3 describes security requirements and introduces some background materials. Section 4 describes the proposed scheme in detail. Section 5 explains how the proposed scheme can be applied to a real world application. The security of the proposed scheme is formally verified in Section 6. Related work is covered in Section 7. Finally, our conclusions are in Section 8.

2. Preliminaries

We look at network models whose objective is the secure interaction among network members in IoT network. Then we take a look at the one-way function and the one-way function chain as a basis of our scheme.

2.1. Network model

The network configuration can be differently changed depending on what kind of entities the network consists of. The centralized network comprised of central controller (i.e., admin) and controlled entities (e.g., light bulb, door lock, thermostat, and so on) is an ordinary network configuration in the context of IoT environment such as the home network and building automation system. This type of managed network does not allow non-members to access to it without several procedures of secure association [6]. We present more detailed procedure for secure network management in the centralized network. Along with this, we introduce the conditional accessible network to which our proposed scheme can be applied. The network configuration we consider is shown in Fig. 1.

Authenticated Member Access Network. Since the ordinary centralized network allows only authenticated and authorized member to access to it, we call this type of network Authenticated Member Access Network for clarification. This network has several significant phases with security consideration to maintain network access and continuous management during network lifecycle. Garcia et al. and Internet standard presented simplified network lifecycle including three phases that they were considered in terms of security [1,2]:

Secure Network Association (Phase 1): This phase can be seen as network provisioning steps for the membership establishment. The entity wishing to become a member is required authentication and authorization by the network owner through predefined schemes. The network access is allowed to the entity (i.e., member) thereafter.

Key Management (Phase 2): The key management phase performs a secure key generation, key distribution and key update for the purpose of secure communication between members in the network. These things provide a mean to securely manage several keys that are being used in the network.

The key management includes the definition of key generation scheme, distribution protocol, re-keying protocol and keying material related specification [7]. In those processes, it causes costs for storage, processing and network resource. In particular, this cost might be increased in proportion to the number of members (i.e., Scalability issue).

Secure Communication (Phase 3): The secure communication is intended to provide confidentiality and integrity whenever members interact between them within the network using deployed keying material. In the practical system, proven secure protocols such as TLS are used for secure communication.

Conditional Accessible Network. This paper considers the network configuration which has an added entity who is non-member (e.g., guest device), requires temporary access to the network. In this paper, we call this *Conditional Accessible Network*. This network has different features from the ordinary network (i.e., Authenticated Member Access Network) configured with a central controller and members as principal entities. If a guest wants to get network access right in the Authenticated Member Access Network, he/she needs to process the necessary proceedings, 1–3 phases mentioned before, to be legitimated member since temporary access is not allowed in this network. Here the guest has the limited capability of the network because most of guests are entities who require time bounded access to the network. The conditional accessible network allows a guest to access the network under limited conditions (e.g., limited time and number of accesses) governed by the central controller. It does not require the membership and continuous key management of 1 and 2 phases, thereby the overhead caused by these phases could be minimized.

2.2. Cryptographic primitives

To handle conditional network access, we use a token which is generated by using the one-way function and the one-way function chain. The definitions for them are given below.

Definition 1. A function $f:\{0,1\}^k \rightarrow \{0,1\}^k$ is a one-way function if, for all probabilistic, polynomial time (PPT) algorithms $\mathcal{A}$, there is a negligible probabilistic function $\epsilon(k)$ such that:

$$\Pr\left[x \leftarrow \{0,1\}^k;\ y = f(x);\ x' \leftarrow \mathcal{A}(y) : y = f(x')\right] \leq \epsilon(k).$$

In the real world, a hash function satisfying the pre-image resistance property [8] can be used as the above one-way function (e.g., SHA standard hash functions [9]).

Definition 2. An one-way function chain [10] is a sequence of values, for a one-way function $f:\{0,1\}^k \rightarrow \{0,1\}^k$,

$$\{y_1, y_2, \ldots, y_n\},$$

where $y_1 = f(x)$ for a value x chosen uniformly at random from $\{0,1\}^k$ and $y_i = f(y_{i-1})$.

In the one-way function chain, we call the value x the seed and the value n the chain length. The one-way function chain has been used as an important cryptographic primitive for various applications including payment systems [11], one-time password systems [12], and multicast authentication [13]. We use the one-way function chain to make a secure credential in the form of token.

We also use the collision resistant hash function to prohibit the adversary from forging the credit value on a new condition.

Definition 3. A function $h:\{0,1\}^* \rightarrow \{0,1\}^k$ is the collision resistant hash function if, for any PPT algorithm $\mathcal{A}$, there is a negligible probabilistic function $\epsilon(k)$ such that:

$$\Pr\left[(x, y) \leftarrow \mathcal{A}(h) \wedge x \neq y : h(x) = h(y)\right] \leq \epsilon(k).$$

3. Security requirements

In IoT environments, for the purpose of convenience, an owner sometimes wants to give the access rights of the connected devices to his friends during limited time. The owner wants to give the access right to the clerk in the limited number of times (e.g., pull-in/out), for valet parking in VAMET. Like this, it is required for the owner to delegate the access rights on one's own devices with limited conditions in real world applications. We model these situations in the view of security.

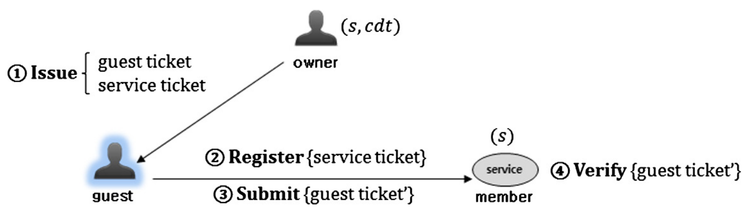


Fig. 2. Relation of entities and their operations. The number of times (n) and expiration time (t) are the primary condition, and both a owner and a service have a shared secret (s).

We regard a network where the owner accesses to a service domain and has a privilege of control. As depicted in Fig. 1, the service domain is simply composed of three entities: owner, service, and guest; in the context of the above IoT environment, an owner is the network owner, service is the connected device, and a guest is the friend.

- owner: an user who accesses to a service with a valid credential, where he/she can give the access right of the service to an outsider in limited conditions.
- service: a service that the owner can use, where its functionality is only provided to someone with a valid credential.
- guest: the outsider who can access to the service with limited conditions, where the access right is given by the owner.

Among these entities, we consider that guest is a latent adversary who wants to access to service regardless of conditions; for example, after the allowed time is expired or the access number of times is burned out, the guest tries to use service. Clearly, guest should not be able to access to service when conditions are not satisfied. We call the security requirement as *conditional accessibility*.

- Conditional accessibility: guest should not be able to access to service when the access rights authorized by owner are disappeared according to the conditions.

We propose a new access right delegation scheme with conditions, which satisfies the above security requirement, conditional accessibility. The security proof is given in Section 6.1.

4. Conditional access right delegation

4.1. Construction

The proposed scheme uses tickets to delegate the access rights of service. Briefly describing these processes as depicted in Fig. 2, owner generates a guest ticket that can be used by guest for authentication and authorization, and a service ticket that can be used by service for registering guest securely. Receiving the service ticket from guest during the register phase, service verifies that the service ticket is generated by owner and not modified (i.e., authentication process). Accessing to service, guest submits the guest ticket to service, where the guest ticket is updated in every access session. Then service verifies the guest ticket based on the service ticket (i.e., authorization process).

In the proposed scheme, we assume that owner and service share at least one secret [14,6]. It means that owner establishes a secure channel with service before delegating the access rights for service. The assumption looks reasonable because service already has a membership from owner. Moreover if the secure channel is not required between owner and service, we can regard that service does not deal with privacy-sensitive resources. In this case, anyone can access to the resource of service and our scheme for the access control is not needed. In addition, we assume that the time of both owner and service is loosely synchronized.

For simple description, we call the proposed scheme TARD, where the conditions are expressed in the field cdt. The condition cdt can include an expiry time, an accessible count, the allowed resources, etc. TARD can define various conditions in the cdt field; the usage of conditions is described in Section 4.2. In TARD, the condition cdt has an expiry time t_e and an accessible count n, certainly for temporary delegation. The proposed scheme TARD is composed of four algorithms: Issue, Register, Submit, and Verify. The parameter ℓ_a is the bit size of the value a.

- $(T_g, T_s) \leftarrow \mathsf{Issue}(s, \mathsf{cdt})$: It outputs a guest ticket T_g and a service ticket T_s from the shared secret $s \in \{0,1\}^{\ell_s}$ between owner and service and conditions cdt that are allowed by owner, where cdt includes at least the expiry time t_e and the accessible count n. It first chooses a random identity $id \in \{0,1\}^{\ell_{id}}$ that is unique and computes the seed of the one-way function chain,

$$y_{seed} = h(s||id||\mathsf{cdt}).$$

  Then, the guest ticket consists of the following values,

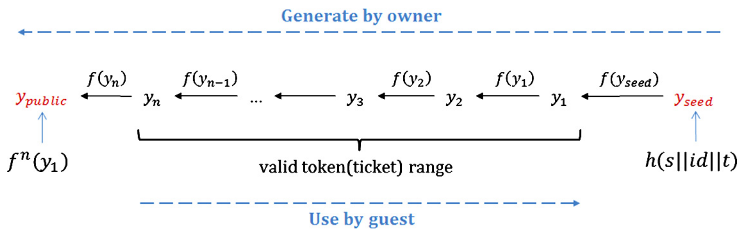


Fig. 3. The usage of one-way function chain: owner computes one-way function chain from y_{seed} to y_{pub}, guest uses the sequence values y_i for $1 \le i \le n$, and service uses y_{seed} and y_{pub} for verification.

$$T_g = (y_1, id),$$

where $y_1 = f(y_{seed})$. Next, the service ticket T_s consists of the following values,

$$T_s = (y_{pub}, id, \mathsf{cdt}),$$

where $y_{pub} = y_{n+1} = f^n(y_1)$. Fig. 3 shows the relations between y_{seed} and y_{pub} in a one-way chain.

- Suc/Fail $\leftarrow$ Register(s, T_s): Given the shared secret s and a service ticket $T_s = (y_{pub}, id, \mathsf{cdt})$, it checks the double registering by comparing id with previous registered ids and then verifies if

$$y_{pub} = f^{n+1}(h(s||id||\mathsf{cdt})),$$

where n is included in the condition cdt and if the expiry time t_e in cdt is further than the current time t; we assume that the current time is acquired by the internal call. If so, it registers the service ticket for the identity id and returns Suc, otherwise, Fail.
- $T'_g \leftarrow$ Submit(T_g): Given $T_g = (y_1, id, n, t)$, it computes the hash value $y_{token} = f^{n-1}(y_1)$ and returns the following guest ticket,

$$T'_g = (y_{token}, id, \mathsf{cdt})$$

where the accessible count n is included in cdt. Finally, it locally updates the mutable condition n in cdt of T_g.

$$T_g = (y_1, id, \mathsf{cdt}')$$

where the accessible count n decreases by one in cdt′, and stores the updated guest token T_g.
- Suc/Fail $\leftarrow$ Verify(T_s, T_g): Given a service ticket $T_s = (y_{pub}, id, \mathsf{cdt})$ and a guest token $T_g = (y_{token}, id, \mathsf{cdt})$, it checks if

$$y_{pub} = f(y_{token})$$

and the conditions cdt (including the expiry time t_e) are satisfied. If so, it decides that the access is authorized and locally updates the service ticket into

$$T_s = (y_{pub} = y_{token}, id, \mathsf{cdt}'),$$

where the mutable condition n decreases by one in cdt′ and returns Suc, otherwise Fail.

In TARD, the computation cost of Submit for computing y_{token} is $O(n)$ at the worst case; it is dependent on the accessible count n. When the value n is much bigger, we can consider the hash chain traversal algorithm, which traverses hash chain with dynamic helper points, to improve the efficiency. When we apply it to our scheme, the computation cost will be decreased to $O(\log n)$ while the space cost increases to $O(\log n)$ [15].

4.2. Extension to generic ACL

We describe the detailed operations of TARD on a generic Access Control List (ACL). In Section 4.1, the scheme implies the accessible count and the expiry time as primary conditions in cdt. It will be obvious that further generic ACL items are anticipated, depending on many of practical scenarios. For instance, TV has resources such as power, channel control and volume. Auto-mobile has door-lock, starting control and infotainment as controllable resources. TARD can simply extend conditions by adding ACL feature to the condition cdt. As described in Fig. 4, the *Access Capability (ACap)* and the *Access Control Entry (ACE) specification* are added to the condition, cdt, during the issuing operation of Issue.

- ACap: a capability description that guest can access to the specific service, where its context is consistent with the user interface (e.g., GUI).

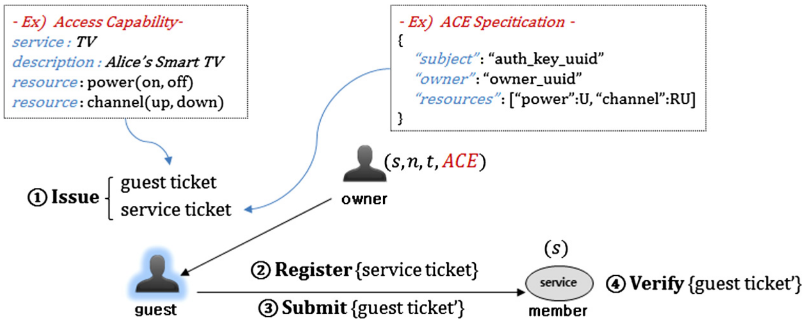

- Ex) Access Capability-
service : TV
description : Alice's Smart TV
resource : power(on, off)
resource : channel(up, down)
- Ex) ACE Specification -
{
"subject": "auth_key_uuid"
"owner": "owner_uuid"
"resources": ["power":U, "channel":RU]
}
(s, n, t, ACE)
owner
① Issue
guest ticket
service ticket
② Register {service ticket}
③ Submit {guest ticket'}
guest
(s)
service
member
④ Verify {guest ticket'}

Fig. 4. A brief example showing ACL extension where Access Capability description is added to *guestticket* and ACE specification is added to *serviceticket*.

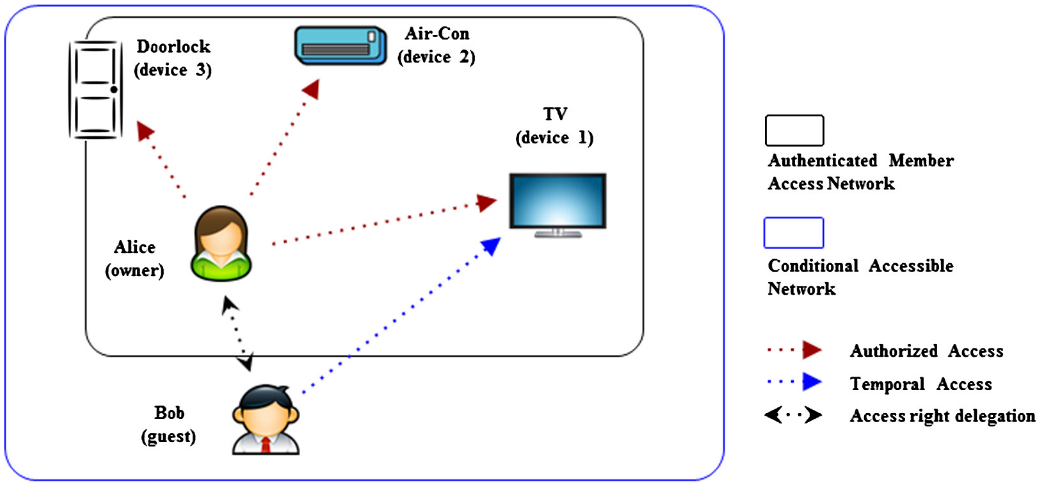


Fig. 5. Network deployment and relationship between entities based on the individual network configuration.

– ACE specification: a data that specifies ACE that is an entry containing information describing the access rights (e.g., CRUD) related to guest, where service manages an ACL that is a list of ACE.

In the registering phase of Register, service stores ACE of cdt in the existing ACL. When guest attempts to require resources of service, service determines whether it is allowed or not.

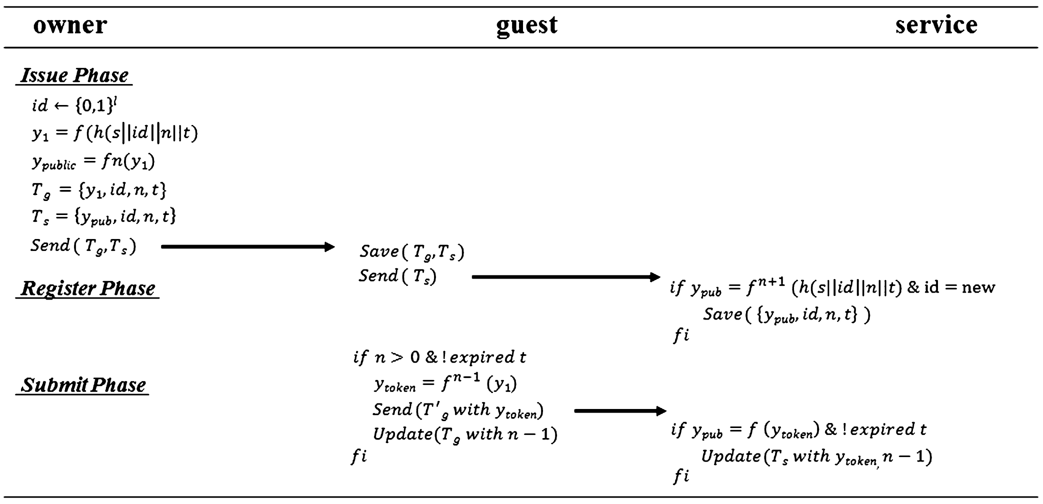
5. Applications scenario

We suggest several application scenarios in Conditional Accessible Network, and then present that our proposed scheme is applied to those scenarios. First we show how our scheme works in the IoT home network.

In the context of IoT, most of entities in this network are consisted of constraint devices which are directed to low power. A representative scenario is aimed at the communication between these devices in smart home network, hence we will show its adaptation in Conditional Accessible Network model. Deployment of Alice's home network and relationship of entities are depicted in Fig. 5. Among the constituent members of home, Alice is owner of the network and has a role of central controller. Other devices such as TV, air conditioner, door lock and light bulb are the member of being controlled by Alice. We suppose that all members (i.e., home devices) are already configured to have a membership of Alice's home. Suppose that one day Bob (i.e., guest), who is a friend of Alice and has not been a member before, comes in Alice's home. If he would like to control a certain home device (e.g., TV) during staying the house, his desire can be simply archived using our proposed scheme.

A protocol that describes such a communication procedure and interactions is illustrated in Fig. 6. Alice selects expiration time t and the number of times n to build limited conditions which restrict availability of keying material (i.e., token). The secret s has been shared between Alice and TV via secure channel which is established previously since we suppose that TV has a membership with Alice through secure network association process (viz. Section 2.1).

At the beginning of Issue process, Alice computes y_{seed} after generating unique *id*, and creates first value y_1 and verifier y_{public} using the one-way function, then generates *guestticket* and *serviceticket*. If anyone wants to transfer them securely, we can consider to share a secret (e.g. PIN) via out-of-band channel because guest scenarios are happened within a close proximity. After both tickets are transferred to Bob, he sends *serviceticket* to TV when he wants to start a service request.



```
owner                                   guest                                   service

Issue Phase
  $id \leftarrow \{0,1\}^l$
  $y_1 = f(h(s||id||n||t)$
  $y_{public} = fn(y_1)$
  $T_g = \{y_1, id, n, t\}$
  $T_s = \{y_{pub}, id, n, t\}$
  $Send(T_g, T_s)$  ──────►  $Save(T_g, T_s)$
                             $Send(T_s)$  ──────►  $if\ y_{pub} = f^{n+1}(h(s||id||n||t)$ & id = new
Register Phase                                         $Save(\{y_{pub}, id, n, t\})$
                                                   $fi$
                             $if\ n > 0$ & $!expired\ t$
Submit Phase                     $y_{token} = f^{n-1}(y_1)$
                                 $Send(T'_g\ with\ y_{token})$  ──────►
                                 $Update(T_g\ with\ n-1)$        $if\ y_{pub} = f(y_{token})$ & $!expired\ t$
                             $fi$                                    $Update(T_s\ with\ y_{token}, n-1)$
                                                                 $fi$
```

Fig. 6. Protocol for token distribution and conditional access using TARD scheme.

At this Register phase, TV checks the integrity (i.e., MAC) as well as the authenticity of *serviceticket* using V_{ticket}. If a *id* of the verified *serviceticket* has not been registered before, TV stores it in the internal storage and adds the *id* to ID_{list} to check upcoming requests. Thereafter, when Bob desires to communicate with TV, he computes one time token y_{token} and sends it to TV. The token y_{token} can be newly generated until exceeding specified conditions for $n > 0$. TV can check the validity of received token using V_{token} with the stored *serviceticket*. If the token is authorized, in turn it is updated *serviceticket* for the next verification. If either condition has been expired, V_{token} operation can detect the token invalidation by the one-way hash chaining characteristic and synchronized time comparison. In case of detecting the invalidity, TV autonomously revokes the *serviceticket* and erases related information from the internal storage without the explicit request of Alice. In the process of natural revocation, TV may keep the revoked *id* until specified expiration time to obviously prevent multiple registration of the same *serviceticket*

As seen in presented protocol, the network access is possible only with initial issued tickets within specified condition (expiration time and number of time), but it does not require additional membership and key management processes by central controller. Besides, it has further advantage in terms of efficiency since the explicit revocation process of burned token is not required once specified conditions are beyond the limit.

6. Analysis

In this section, we show that the proposed scheme satisfies security requirement, conditional accessibility, and performance analysis.

6.1. Security analysis

To prove the security properties of the proposed scheme, we define a new security model (i.e., Definition 4), which is dedicated to our scheme. The definition guarantees that the adversary should not be able to enjoy services out of conditions and fabricate the conditions. The adversary can access to the oracles to simulate Issue, Register and Verify algorithms, and can choose the target condition that she wants.

Definition 4. The proposed scheme, TARD, satisfies the requirement of conditional accessibility if for any PPT algorithm $\mathcal{A}$, the probability

$$\Pr\left[\begin{array}{l|l} s \leftarrow \{0,1\}^{\ell_s},\ \mathsf{cdt}^* \leftarrow \mathcal{A}^{O_{\mathsf{Issue}}, O_{\mathsf{Register}}, O_{\mathsf{Verify}}}(\ell_s), & \\ (T_s, T_g) \leftarrow \mathsf{Issue}(s, \mathsf{cdt}^*), & Suc \leftarrow \mathsf{Register}(s, T'_s) \\ (T'_s, T'_g) \leftarrow \mathcal{A}^{O_{\mathsf{Issue}}, O_{\mathsf{Register}}, O_{\mathsf{Verify}}}(T_s, T_g) & \text{or } \mathsf{Suc} \leftarrow \mathsf{Verify}(T'_s, T'_g) \end{array}\right]$$

is negligible for the parameter ℓ_s, where $T'_s \neq T_s$, T'_g is invalid for the condition cdt and $\mathcal{A}$ maintains its status internally. The issuing, issuing and verifying oracles operate as follows:

- O_{Issue}: For the request on the input values cdt, it returns the service ticket T_s and the guest ticket T_g. The tickets should be valid for the Verify algorithm based on the secret s.
- O_{Register}: For the request of registering on a service ticket T_s, it returns Suc if valid, otherwise Fail. The validity check for the service ticket proceeds on the secret s.

– O_{Verify}: For the request of validity check on a guest ticket T_g, it returns Suc if valid, otherwise Fail. The validity check for the guest ticket proceeds on the validity of the service ticket T_s.

The security is based on one-wayness and collision resistant properties of cryptographic hash functions with random oracle assumption [16,17]. The proposed scheme limits the accessible count by using the one-way function chain [10] and holds the other conditions (which is immutable) by using the keyed-hash technique [18]. Theorems 1 & 2 show that Definition 4 is guaranteed in the proposed scheme.

Theorem 1. *The proposed scheme* TARD *satisfies the conditional accessibility for guest tickets with random oracle under the assumption of the one-way function.*

Proof. To show that guest tickets should not be used out of conditions, we assume that there exists the adversary $\mathcal{A}$ to break the security of the proposed scheme for the guest ticket T_g in Definition 4 with probability ϵ; $\mathcal{A}$ passes the verification process for a guest ticket with fabricated conditions. Additionally, we assume that the service ticket should not be modified or forged by any adversaries, which is guaranteed by Theorem 2. Then, we can construct the simulator $\mathcal{B}$ to break the one-wayness of a one-way (e.g., cryptographic hash) function with probability ϵ by interacting with $\mathcal{A}$. By Definition 1, the output of an one-way function, y, is given to $\mathcal{B}$; the goal of $\mathcal{B}$ is to find its inverse of y for the one-way function.

For the proof, $\mathcal{B}$ chooses random $s \in \{0,1\}^{\ell_s}$ and simulates oracle queries in Definition 4 as follows:

– $O_{\mathsf{Issue}}(\mathsf{cdt})$: $\mathcal{B}$ computes and returns the response by using the issuing algorithm, $(T_g, T_s) \leftarrow \mathsf{Issue}(s, id, t, \mathsf{cdt})$, with the knowledge of the secret s. $\mathcal{B}$ stores the guest and service tickets for simulating response of the verifying oracle, and additionally stores the bit $c \in \{0,1\}$ where 0 means the service ticket is not yet submitted by $\mathcal{A}$, otherwise 1. At this issuing phase, the bit c set to be 0.
– $O_{\mathsf{Register}}(T_s)$: $\mathcal{B}$ checks the validity of the given service ticket T_s by the verifying algorithm, $\mathsf{Verify}(s, T_s)$, with the knowledge of the secret s. If the service ticket is valid, $\mathcal{B}$ set the bit c to 1.
– $O_{\mathsf{Verify}}(T_g)$: $\mathcal{B}$ checks if the corresponding bit c is 1, otherwise $\mathcal{B}$ rejects the oracle queries. If $c = 1$, $\mathcal{B}$ computes the response by the verifying algorithm, $\mathsf{Verify}(T_s, T_g)$; $\mathcal{B}$ consumes the guest ticket one time and updates the internal value, the token y_{token} if passing the token check process, $f(y_{token}) = y_{pub}$.

At the challenge phase, $\mathcal{A}$ submits a challenge condition cdt^* and then $\mathcal{B}$ returns the guest and service tickets,

$$T_g = (y, id), \text{ and } T_s = (y_{pub}, id, \mathsf{cdt}^*),$$

where $y = h(s||id||\mathsf{cdt}^*)$ and $y_{pub} = f^n(y)$. In the above simulation, the random oracle h operates as follows:

– The random oracle, h, returns the output of the one-way function y for the challenge condition cdt^*,

$$y \leftarrow h(s||id||\mathsf{cdt}^*).$$

In the other queries except for the challenge condition, it returns random y_{seed}. $\mathcal{B}$ stores the corresponding input and output values for simulation of h.

Finally, $\mathcal{A}$ outputs the guest ticket T'_g that does not conform to the target condition cdt; as mentioned at the beginning of the proof, the service ticket T_s should not be fabricated by $\mathcal{A}$. Then, $\mathcal{B}$ decides that y_{tk} in the output T'_g is the inverse of y with probability ϵ and submits it for breaking the one-wayness of the one-way function. The case that $\mathcal{A}$ successfully passes the Verify algorithm is only to submit the inverse of y. That is, $\mathcal{A}$ violates the condition of the accessible count. The other conditions are guaranteed by the service ticket T_s.

By contradiction, under the one-wayness assumption (Definition 1), the probability that breaks the security of the proposed scheme for the guest ticket is negligible. □

Theorem 2. *The proposed scheme* TARD *satisfies the conditional accessibility for service tickets under the assumption of the collision resistant hash function.*

Proof. Unlike in Theorem 1, the function h that computes the seed of hash chains is not a random oracle, but a collision resistant hash function described in Definition 3. Based on the collision resistant property, we prove that the service ticket should not be fabricated by an adversary.

For the proof, we assume that there exists the adversary $\mathcal{A}$ to break the security of the proposed scheme for the service ticket T_s in Definition 4 with probability ϵ; $\mathcal{A}$ passes the verification process for a fabricated service ticket. Then, we can construct the simulator $\mathcal{B}$ to find a collision for the function h with probability ϵ. By Definition 4, the attack environment of $\mathcal{A}$ is the same as that of Theorem 1. In addition, the simulation of the issuing, registering and verifying oracles is also the same as that in Theorem 1. The secret s is chosen at random by $\mathcal{B}$.

Table 1
Comparison of TARD and certificate-based authentication for guest join/leave cases. The join operation is performed when a new guest device that requires access to k different devices is added to a network. The leave operation is performed when a registered guest device is revoked.

		Certificate-based authentication	TARD
Join	1 device	1 ACL update	1 ticket generation
	k devices	k ACL updates	1 ticket generation
Leave	1 device	1 ACL update	–
	k devices	k ACL updates	–

At the challenge phase, $\mathcal{A}$ submits a challenge condition cdt^* and $\mathcal{B}$ returns the guest and service tickets (T_g, T_s) to $\mathcal{A}$, where the tickets are computed by the Issue algorithms with the knowledge of the secret s; $y_{seed} = h(s||id||\mathsf{cdt}^*)$.

Finally, $\mathcal{A}$ outputs the service ticket T'_s that has not been issued by $\mathcal{B}$. Then, $\mathcal{B}$ extracts the identity id' and the condition cdt' in the submitted service ticket T'_s and for the collision resistant hash function h, decides that the values $(s||id||\mathsf{cdt}^*)$ and $(s||id'||\mathsf{cdt}')$ have the same hash value with probability ϵ. Without the knowledge of the secret s, the case that the service ticket T'_s passes the verification process of the Verify algorithm is only for $\mathcal{A}$ to find collision for the values id and cdt^*.

By contradiction, under the collision resistant assumption (Definition 3), the probability that breaks the security of the proposed scheme for the service ticket is negligible. □

We demonstrated that TARD satisfies conditional accessibility on the guest and service tickets. However, the proofs do not guarantee that the adversary re-uses the service tickets in malicious intention; Definition 4 does not allow the adversary to output the same service ticket for the challenge. For preventing the re-use of service tickets, the Register algorithm compares id of the service ticket T_s with ids of service tickets saved in local storage of service, where service stores the service tickets until the specified time is expired.

When the service and guest tickets are issued by owner, they should be transmitted through a secure channel supporting confidentiality to prevent for the adversary from acquiring the tickets. In the construction of Section 4.1, TARD does not include any techniques for confidentiality of the issued tickets, but the transmission can be protected by the security protocols on the link layer such as Bluetooth, Wi-Fi, ZigBee and Z-Wave; the security protocols support encryption for the transmission of confidential data.

6.2. Performance analysis

In most real-world security protocols such as TLS/DTLS, IPsec and SSH, public-key based authentication is the most popularly used option for establishing a secure and authenticated channel since an efficient key distribution is required for symmetric-key based authentication [19,20]. For guest device scenarios, we analyze the efficiency of the proposed scheme compared with the certificate-based ACL scheme for existing open standard technologies such as IoTivity of OCF (`http://openconnectivity.org/`) and AllJoyn of AllSeen alliance (`https://allseenalliance.org/`). In the certificate-based ACL, each device sets its access policy and maintains its own ACL to prevent unauthorized access. In an ACL, the identities of a device are bound to each device's certificate signed by a Certificate Authority (CA). Our comparison analysis results are summarized in Table 1.

In the certificate-based authentication, when a new guest device that requires access to k different devices is added to a network, (1) CA issues a certificate (with expiration time) to the device; and (2) an administrator updates the ACLs of those devices to add permissions for the guest device. In TARD, however, it is enough to generate a ticket for a new guest device because the guest device's permissions are specified in the ticket. For the leave operation, the certificate-based authentication still needs to update the ACLs of k devices when the revoked guest device has permissions on those devices. In TARD, however, the guest device's permissions are automatically revoked with the expired ticket. Moreover, unlike TARD, the certificate-based authentication cannot support the count-based temporary access condition.

Hence, our analysis results show the superiority of TARD in the performance for guest join/leave operations compared with the existing certificated-based ACL mechanism.

7. Related work

In general, many existing group membership protocols [1–4] can also be applied to the access control applications for guest devices. As shown in Section 6.2, however, such protocols were mainly designed by taking into consideration frequent membership changes in a group, which could incur significant overhead if the size of group is large.

Only a few access control schemes have designed by considering guest devices. Lee et al. [21] proposed a key sharing mechanism for guest devices. Their main idea is to use a smart card to reduce the burden on users by providing an easy and secure common secret sharing mechanism without typing it for guest devices. However, this approach is not enough to achieve fine-grained access control because all guest devices have a common set of permissions. Unlike this approach, in

TARD, each guest device can have an independent set of permissions, respectively. Moreover, specialized hardware such as smart card is not required for TARD.

Ticket based authentication scheme is not new. Kerberos [22] is an authentication protocol which uses security tokens called *tickets* to authenticate users with a trusted third party. Basically, in Kerberos, the trusted third party could potentially be a single point of failure because all users' secrets are maintained in the trusted third party. However, in TARD, we do not assume that there exists a trusted third-party. The OAuth protocol was also developed to solve the common problem of enabling delegated access to protected resources [23] and popularly used in many web services. Again, unlike TARD, trusted identity providers are required for managing access tokens in the OAuth protocol.

Kim [24] proposed a cryptographic scheme with N-times consumable digital tickets for electronic commerce in order to protect the ticket owner's identification and the quantity of the ticket consumed. The proposed scheme is similar to TARD, but is limited to the access control mechanisms for guest devices.

8. Conclusion

We have presented a conditional access scheme for a guest to have a access right with limited but essential conditions in hardware constrained environments. The scheme uses the token as a role of authenticator, and operations of the scheme are based on lightweight cryptographic primitives such as the collision resistance hash function and the one-way function.

To reduce the security management overhead occurred by the temporary access, our proposed scheme neither requires the membership acquisition of a guest nor needs a explicit revocation process of the token that exceeds bounded conditions. Consequently the conditional access control can be achieved without a significant overhead in terms of the security management. Also as real word scenario, we suggest several applications such as IoT home network, valet parking service and temporary building entrance. We have then shown how proposed scheme works in these applications. At last, we have provided the security proof that the proposed scheme satisfies the conditional accessibility as security requirements.

As a next step we plan to investigate how the presented scheme can be employed based on standardized management protocols and can easily provide support for secure guest access control in hardware constrained environment.

Acknowledgments

This work was supported in part by the NRF Korea (No. 2014R1A1A1003707), the ITRC (IITP-2016-R0992-16-1006), and ICT R&D program (No. B0717-16-0116). Authors would like to thank all the anonymous reviewers for their valuable feedback.

References

[1] O. Garcia-Morchon, S.L. Keoh, S. Kumar, P. Moreno-Sanchez, F. Vidal-Meca, J.H. Ziegeldorf, Securing the ip-based internet of things with hip and dtls, in: Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2013.
[2] T. Hardjono, B. Weis, The multicast group security architecture, Tech. Rep. RFC 3740, RFC Editor, 2004, http://www.rfc-editor.org/rfc/rfc3740.txt.
[3] J.L.H. Ramos, M.V. Moreno, J.B. Bernabé, D.G. Carrillo, A.F. Skarmeta, SAFIR: secure access framework for IoT-enabled services on smart buildings, J. Comput. Syst. Sci. 81 (8) (2015) 1452–1463.
[4] Y. Kim, A. Perrig, G. Tsudik, Simple and fault-tolerant key agreement for dynamic collaborative groups, in: Proceedings of the 7th ACM Conference on Computer and Communications Security.
[5] E. Damiani, J. Jeong, R.J. Howlett, New Directions in Intelligent Interactive Multimedia Systems and Services-2, vol. 2, Springer Science & Business, Media, 2009.
[6] L. Ogiela, M.R. Ogiela, Cognitive systems and bio-inspired computing in homeland security, J. Netw. Comput. Appl. 38 (2014) 34–42.
[7] M.R. Ogiela, U. Ogiela, Linguistic extension for secret sharing (m, n)-threshold schemes, in: International Conference on Security Technology, 2008.
[8] S. Lucks, Design principles for iterated hash functions, IACR Cryptology ePrint Archive.
[9] Secure hash standard, National Institute of Standards and Technology (NIST), FIPS 180-3.
[10] L. Lamport, Password authentification with insecure communication, Commun. ACM 24 (11) (1981) 770–772.
[11] R. Anderson, C. Manifavas, C. Sutherland, Netcard — a practical electronic-cash system, in: Security Protocols Workshop, 1996.
[12] N. Haller, The s/key one-time password system, RFC 1760, Internet Engineering Task Force.
[13] A. Perrig, R. Canetti, D. Song, J.D. Tygar, Efficient and secure source authentication for multicast, in: Network and Distributed System Security Symposium, 2001.
[14] L. Ogiela, M.R. Ogiela, Semantic analysis processes in advanced pattern understanding systems, in: Advanced Computer Science and Information Technology, 2011.
[15] D.H. Yum, J.W. Seo, S. Eom, P.J. Lee, Single-layer fractal hash chain traversal with almost optimal complexity, in: Topics in Cryptology–CT-RSA 2009, Springer, 2009.
[16] M. Bellare, P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, in: Proceedings of the 1st ACM Conference on Computer and Communications Security, 1993.
[17] R. Canetti, O. Goldreich, S. Halevi, The random oracle methodology, revisited, J. ACM 51 (4) (2004) 557–594.
[18] Y. Li, D. Xiao, S. Deng, Keyed hash function based on a dynamic lookup table of functions, Inf. Sci. 214 (2012) 56–75.
[19] C. Boyd, C. Cremers, M. Feltz, K.G. Paterson, B. Poettering, D. Stebila, ASICS: authenticated key exchange security incorporating certification systems, in: Proceedings of the 18th European Conference on Research in Computer Security, 2013.
[20] R. Hummen, H. Shafagh, S. Raza, T. Voig, K. Wehrle, Delegation-based authentication and authorization for the ip-based internet of things, in: IEEE International Conference on Sensing, Communication, and Networking, 2014.
[21] S.M. Lee, H.G. Yook, S.J. Oh, S.H. Han, Guest access: change even your mother into an effective security technician, in: IEEE Consumer Communications and Networking Conference, 2006.
[22] S.P. Miller, B.C. Neuman, J.I. Schiller, J.H. Saltzer, Kerberos authentication and authorization system, in: Project Athena Technical Plan, 1988.

[23] E. Hammer-Lahav, The OAuth 1.0 Protocol, Tech. Rep. RFC 5849, RFC Editor, 2010, http://www.rfc-editor.org/rfc/rfc5849.txt.
[24] H. Kim, N-times consumable digital ticket and its application to content access service, in: 4th IEEE Consumer Communications and Networking Conference, 2007.