Secure DNS Name Autoconfiguration for IPv6 Internet-of-Things Devices

Keuntae Lee*, Hyungsuk Kang*, Jaehoon (Paul) Jeong†, Hyoungshick Kim*, and Jung-Soo Park‡
* Department of Computer Science & Engineering, Sungkyunkwan University, Republic of Korea
† Department of Interaction Science, Sungkyunkwan University, Republic of Korea
‡ Electronics and Telecommunications Research Institute (ETRI), Republic of Korea
Email: {rmsxo1321, hskang9, pauljeong, hyoung}@skku.edu, pjs@etri.re.kr

***Abstract*—This paper proposes a Secure Domain Name System (DNS) Name Autoconfiguration called SDNSNA for IPv6 Internet-of-Things (IoT) Devices. The manual configuration of domain names of devices might be a cumbersome burden for users as the number of devices increases. A legacy scheme called DNS Name Autoconfiguration (DNSNA) can be used to register the DNS names of IoT devices automatically into a DNS system. However, DNSNA registers the DNS name of IoT devices without an authentication process. The DNS name registration of unauthenticated IoT devices can cause security problems. In this paper, we propose SDNSNA to support security functionality to let the legacy scheme of DNSNA provide secure DNS naming services to IoT devices. In SDNSNA, the smartphones of valid users can authenticate devices by interacting with an authentication server so that the devices can register their DNS names and the corresponding IPv6 addresses. Therefore, with a Near Field Communication (NFC)-based authentication, SDNSNA can provide users with secure and easy DNS name autoconfiguration of their IoT devices.**

I. Introduction

The Internet of things (IoT) is the network of physical devices, vehicles, building systems, and various embedded devices with electronics, software, sensors, actuators, and network interface cards that enable these objects to collect and exchange data. Recently, the IoT has become one of the hottest research fields. The number of IoT devices is expected to increase by almost 26 billion units by 2020 [1]. By a large number of devices, it is inefficient to manually configure their domain names in the Domain Name System (DNS), which allows DNS servers to translate between domain names and IPv6 addresses.

Lee et al. proposed a DNS Name Autoconfiguration (DNSNA) scheme for IoT devices [2]. Based on this, DNSNA has been designed to provide all kinds of IoT devices with an efficient DNS naming service. However, DNSNA is vulnerable to sniffing attack because all messages are being sent in plain text. Thus, malicious hackers can obtain the user's information through sniffing. If a malicious IoT device can be connected to the user's network through an access point (AP) or an Ethernet network, the pair of the DNS name and IPv6 address of such a device can be registered into a DNS server through DNSNA protocol [2]. This is because DNSNA can register a DNS name of an IoT device without any authentication. If a malicious IoT device registers its DNS name and services into a DNS server through an AP, a hacker can obtain the information of IoT devices and their users through the AP. This unauthenticated DNS name registration may expose all IoT devices to the possible security attacks of hackers. That means hackers can steal the information of IoT devices and observe the behaviors of the users in the IoT networks. Thus, we need a secure DNS naming service for protecting IoT devices from various security attacks.

This paper proposes a Secure DNS Name Autoconfiguration (SDNSNA) for IoT devices. For authentication credential delivery, we provide an user friendly solution which is implemented with near field communication (NFC). The idea of SDNSNA is to use the smartphone-based authentication through the short-range communication between a user's smartphone and IoT devices by NFC. An administrator's smartphone can securely and easily register valid IoT devices into a DNS server via a router, using NFC to transfer cryptographic keys for device authentication. A router and a smartphone obtain a verification key and a signing key, respectively, from an authentication server (AS). A secure channel (e.g., TLS or SSH) between the AS and each of them is established. The smartphone distributes its signing key to an IoT device via NFC communication. With the delivered signing key, the IoT device can authenticate its DNS registration message sent to the router, which will relay it toward the DNS server. The contributions of this paper are as follows:

- First, we suggest the requirements of the secure and efficient DNS naming services for managing IoT devices efficiently.
- Second, we propose an architecture of SDNSNA for the secure and efficient DNS naming services of IoT devices.
- Third, we show the feasibility of SDNSNA through the implementation and test of SDNSNA in a testbed having IoT devices.

The rest of this paper is organized as follows. Section II summarizes related work relevant to IoT DNS naming. Section III describes the design and protocol of our SDNSNA. Section IV explains an experiment in our testbed. In Section V, we conclude this paper along with future work.

II. Related Work

The IoT has been realized and will be popularly used for a variety of Internet services. The number of IoT devices in use has been increasing steadily. The naming system is an essential element to identify each IoT device in any network where several devices have the same manufacturer and device type. IoT devices have numerous connections between other

devices. In this environment, in order to create a variety of applications and services, security is an important factor.

The oneM2M technical specifications for a common M2M service layer along with an M2M device identification scheme standardized an object identifier (OID) [3] and suggested globally unique IDs for use in oneM2M. An M2M node ID consists of a higher arc (M2M node indication ID) and a sequence of four lower arcs (manufacturer ID, model ID, serial ID, and expanded ID), and the arcs are placeholders for the devices identification and description in the hierarchal tree of globally unique IDs. The M2M node indication ID (the higher arc) represents a globally unique identifier for the M2M node. The higher arc can be composed of several sub-arcs, which are variable. The higher arc is assigned and managed by ITU-T/ISO.

DNSNA [2] was designed to operate in a variety of IoT environments having various sensors ranging from low capacity to high capacity. It consists of the following four steps. The first step is DNS name generation. The second is DNS name collection. The third is DNS name registration. The fourth is IoT device list retrieval. DNSNA uses either the router advertisement (RA) DNS option [4] over IPv6 neighbor discovery (ND) [5] or the DNS option over DHCPv6 [6]. Once it obtains a DNS search list [4], the target network device autonomously constructs its DNS name using the DNS search list and its device information, and then registers its DNS name into a DNS server via a router in the same subnet. An IoT device generates a DNS name from the DNS search list in the DNS option(s). The IoT device checks the uniqueness of the the generated DNS name by performing duplicate address detection (DAD) for its IPv6 address corresponding to the DNS name in the subnet where it exists. If there is no reply for this DAD, the IoT device regards the DNS name as unique such that the other nodes in the subnet do not use the DNS name, so it configures the DNS name as its own. Otherwise, it repeats the generation of another DNS name and the execution of the DAD until the request for the DAD is not responded by any other node. A router sends node information (NI) query to collect the pair of the DNS name and the corresponding IPv6 address of each IoT device. The router checks if the collected DNS names of the IoT devices are registered in the DNS server.

NFC is a form of contactless communication to exchange information without any requirement to touch devices together or going through multiple steps to set up a connection. In 2004, NFC forum was established to standardize applications which use NFC. The NFC Data Exchange Format (NDEF) specification defines the format and rules to exchange information [7]. NFC has three modes of operation: peer-to-peer mode, read/write mode, and card emulation mode [7]. The NFC device behaves as a reader for NFC tags in read/write mode, and as a contactless smart card in card emulation mode, which enables a smartphone or NFC tags to replace credit cards, transit cards, access cards, and etc. Unlike other modes for operation, peer-to-peer mode defines commnuication operation between two NFC devices, and it is used in Android Beam technology [8]. In this research, SDNSNA is operated in peer-to-peer mode through Simple NDEF Exchange Protocol (SNEP) to exchange information, which is a stateless protocol for the exchange of a request and a response.

The rapid increase of IoT devices and the concern for security of them has led to conduct research on mutual authentication and session key agreement on IoT environment [9]. The generation of a session key through mutual authentication is demonstrated on [9]. However, our proposed scheme generates asymmetric keys for the authentication of an IoT device through an authentication server (AS). A pair of asymmetric keys consists of a verification key and a signing key. A signing key is sent to an IoT device by a smartphone with the secure connection with the AS over NFC. The IoT device generates a digital signature of an NI reply message having the DNS name and the IPv6 address of the IoT device. A verification key is sent to the router. The router receives and verifies a received signature to check if the signature is valid.

A wireless network may have several standard security schemes (e.g., wired equivalent privacy (WEP), Wi-Fi protected access (WPA), WPA2) [10]. WEP was designed to provide data security which is similar to a security scheme in a traditional wired network. However, WEP was replaced by WPA after several critical weakness was discovered. WPA security protocol was designed to allow authenticated users to access a service with a shared secret key through an AP. WPA is a secure wireless network standard security scheme, but if (s)he knows a shared secret key of an AP, anyone can access the local devices in the wireless LAN of the AP via an AP, so the information from the devices will be exposed. Furthermore, if a device with malicious purposes passes authentication, the security leak in the network having the device may happen. Also, Lee et al. previous scheme [2] does not include an authentication process. Because of that, if the AP allows an IoT device of a malicious user who has malicious purposes, it may get the information circulating on the local network. After the router collects the pair of the DNS name and the IPv6 address of the malicious IoT device, it registers the pair into the DNS server for further management. By this process of DNSNA [2], the malicious device can access the local network to intercept information from other devices in the same subnet. To prevent this security attack from happening, a new secure scheme for DNS name autoconfiguration is necessary.

III. Proposed Secure DNS Name Autoconfiguration

This paper is an enhanced version of DNSNA in [2]. Fig. 1 shows the sequence diagram for SDNSNA. To demonstrate the process of SDNSNA, a smartphone, an AS, an IoT device, a router, and a DNS server are organized as shown in Fig. 1. Assume that a secure channel (e.g., TLS or SSH) is established between all network devices in a subnet. The smartphone requests authentication information to the AS. Then, the AS generates the key pair of a signing key and a verification key, and sends a signing key and a verification key to the smartphone and the router, respectively. As the router receives the verification key from the AS, the router will broadcast an NI query. On the other hand, a smartphone sends the signing key acquired from an AS to an IoT device via NFC. The IoT device gets a signing key and waits for the router to send an

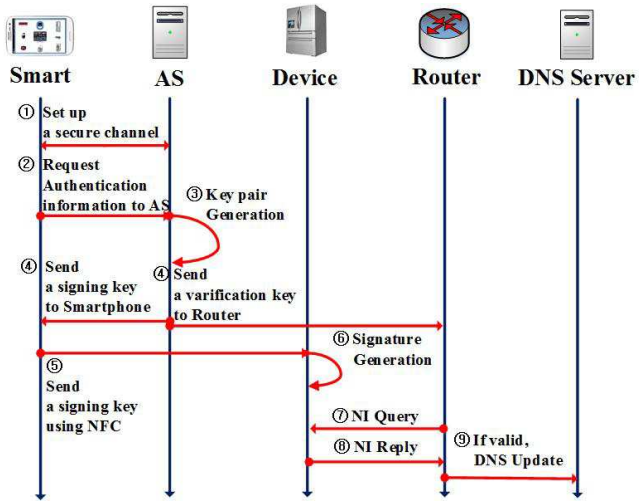

Fig. 1. Sequence Diagram for SDNSNA

NI query. When the router sends an NI query to IoT devices, each IoT device will send an NI reply to the router if it makes the signature using a signing key, which was received from a smartphone via NFC. Then, the router will check the NI reply with the signature. Additionally, the router will check if the received cryptographic nonce and timestamp are valid along with the hashed value of the NI reply or not. If the security information matches, the router will request the DNS server to register the DNS information of the IoT device into its DNS zone file.

A. DNS Naming Service

The ultimate goal of our SDNSNA scheme is to support an IoT DNS naming service that is convenient and secure for users, allowing them to easily distinguish the DNS names of their IoT devices for monitoring and remote-control in their local network or across the Internet. To provide this service, either text-based or graphic-based display can be used. Text-based display shows the list of IoT devices registered into the DNS server, and the user can see the detailed device information by clicking a device listed on the display. Device information can be registered into a service resource record in a DNS zone file for a service. The service name along with the device information rather than the service resource record itself is shown because the user does not need to see the detailed, intricate service resource. Graphic-based display is used for searching and identifying IoT devices in an user-friendly way. For example, if there are two air conditioners in an user's home, the name of the air conditioners may be identical or similar to each other. Instead of differentiating them with distinct IPv6 addresses, graphic-based display can differentiate the DNS names of the air conditioners them with different locations (e.g., bedroom and living room).

B. DNS Name Format

The proposed DNS name format uses a more hierarchical structure using OIDs as an IoT naming service, which is defined in DNSNA [2]. We can apply an oneM2M OID [3] as part of the DNS name format for an IoT device on the Internet. ITU-T and ISO/IEC developed OIDs to assign a globally unique ID to an M2M node. The information contained in the DNS name format in Fig. 2 is described below:

```
unique_id.object_identifier.location.domain_name
```

Fig. 2. DNS Name Format Using Object Identifier

- **unique_id** is an identifier that guarantees the uniqueness of the DNS name in ASCII characters. The identifier can be a sequence number or alphanumeric with readability, such as a product name. For example, a unique_id for a TV could be TV1.
- **object_identifier** is a composite ID that consists of an M2M node indication ID (such as the concatenation of the managing organization, administration, data country code, and M2M node), manufacturer ID, model ID, and serial number ID. For example, 0.2.481.1.100.3030.10011 is an OID [3] where 0 is the managing organization ITU-T, 2 is the administration, 481 is the data country code for Korea 1 is an M2M node, 100 is the node manufacturer, 3030 is the node model, and 10011 is the node serial number.
- **location** is a device's physical location, including the macro location (e.g., living room in an apartment) and micro location (e.g., the center, left-bottom wall, or right-upper corner of the living room), as well as a concatenation of micro and macro locations.
- **domain_name** is a DNS domain name (e.g., skku.edu or home) encoded according to the specification of the representation and domain name use [6].

If the DNS name of an IoT device includes the location information, users can easily identify the physical location of each device. We can consider the physical location in macro and micro terms. If macro location information (such as living room, kitchen, or bedroom in an apartment) is available to an IoT device, a keyword for that location can be used in the DNS name as a subdomain name, such as living_room. Our SDNSNA scheme can use localization [11] [12] [13] in the visual display of indoor IoT devices, such as in an apartment, to estimate the indoor location of each IoT device, allowing users to track the position of their mobile devices (e.g., smartphone, tablet, and robot cleaner). The physical location of each device is defined as a macro location for DNS naming. Using the micro location, an IoT device can be located in the center, wall, or corner of the room specified by the macro location. For example, suppose that a robot cleaner is in the right upper corner of a living room. If the DNS name for the cleaning robot contains the right-upper corner of the living room, a home resident can find it easily. In this paper, we have specified such a detailed location for an IoT device as a micro location subdomain name, such as "right_upper_corner.".

C. Procedure for SDNSNA

This section explains the procedure for our DNS name autoconfiguration scheme, called SDNSNA. SDNSNA is divided into five steps: IoT device authentication, DNS name generation, DNS name collection, DNS name registration, and DNS name retrieval. After these five steps, the DNS name of an IPv6 node (i.e., router and host) is generated according to the type of the IPv6 node (e.g., mobile device or static device) along with its location. Fig. 3 shows a system configuration having the system components for SDNSNA. As shown in Fig. 3, there is a secure channel (e.g., TLS or SSH) that is established between a smartphone and an AS. Also, another secure channel (e.g., TLS or SSH) is established between a router and the AS.

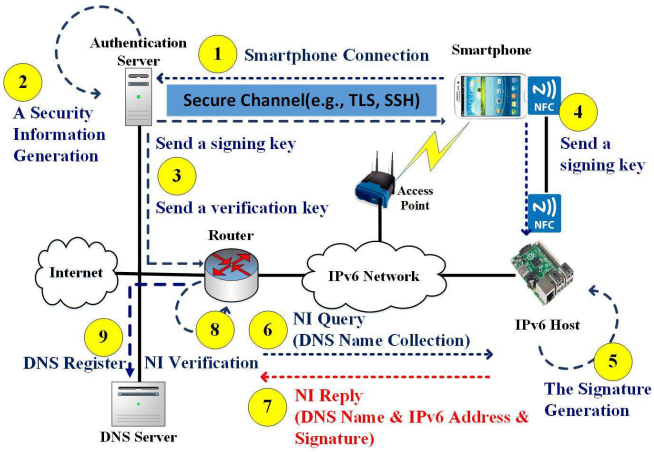

Fig. 3. A System Configuration for SDNSNA

In SDNSNA, the router needs to check if the security information from an NI reply is valid or not. First, for IoT device authentication, SDNSNA uses a smartphone of a user as an administrator in a target network (e.g., owner at a smart home) to obtain a signing key through authentication information request. When an IoT device obtains a signing key from the smartphone via NFC, the IoT device saves a signing key into its local repository, and uses it to generate the digital signature of an NI reply with a flag requiring the authentication for the received NI query.

Second, DNS name generation involves an IoT device to generate its DNS name from each DNS suffix delivered from a DNS option. When an IoT device joins a subnet as an IPv6 host, it receives a DNSSL option [4] through either an RA from a router [5] or a DHCPv6 message from a DHCPv6 server [6]. The DNSSL is a list of DNS suffixes used for the construction of fully qualified DNS names of IPv6 nodes. As an IoT device, an IPv6 host checks the validity of the DNSSL option. If the option is valid, the IPv6 host constructs a DNS name for each DNS suffix and verifies the uniqueness of the constructed DNS names in the subnet. To verify the DNS name, the IPv6 host constructs a global DNS name in accordance with the proposed DNS name format in Fig. 2. The IPv6 host performs DAD to test the uniqueness of the IPv6 address corresponding to the global DNS name in the link (as a tentative IPv6 address) where the host is located [14] [15]. After the successful DAD procedure, it sets the corresponding IPv6 tentative address as its IPv6 unicast address along with the global DNS name. Otherwise, when the host receives a response to the DAD message, it regenerates a new global DNS name with another identifier, which is unique_id, containing another sequence number for a unique ID. It performs another DAD procedure for the new DNS name and repeats this process until it has verified the uniqueness of the generated DNS name.

Third, DNS name collection collects the DNS names of IoT devices via a router after the step of the DNS name generation. An IPv6 node (e.g., router) in the same subnet collects the DNS names of all the IPv6 hosts (e.g., refrigerator and robot cleaner) using the NI protocol [16]. The NI protocol defines a new ICMPv6 code (e.g., code 3) for both the NI query and reply to collect the DNS information of IPv6 IoT devices. The router sends an IPv6 host an NI query with ICMPv6 code 3 to collect the DNS names of IoT devices in the same subnet. The router should transmit the NI query to a link-local multicast address for the IoT DNS name collection. We assume that all the IPv6 hosts join the multicast group for the IoT DNS name collection. Each IPv6 host replies to the NI query with its DNS name and IPv6 address. To prevent message collisions from happening, IPv6 hosts in the same subnet send their NI replies with a random interval between zero and the query response interval in MLDv2 [16]. Thus, the router can collect the DNS names and corresponding IPv6 addresses of all the IoT devices per link of the router.

Fourth, DNS name registration checks the validity of the global DNS name of each host and registers it to a DNS server. The router checks the validity of the global DNS name of each host by sending a DNS query to an authoritative DNS server that may hold the DNS name of each host. If the IPv6 address corresponding to its DNS name is not returned from the DNS server, the router regards the DNS name as unused and registers the DNS information (e.g., the global DNS name and corresponding IPv6 address) into the DNS server through DNS dynamic update [17]. The DNS server manages a zone file as the database of DNS names for IoT devices, and uses it to translate between the DNS names and the IPv6 addresses. Otherwise, a router notifies the IPv6 host of the duplicate DNS name error. In that case, the host should regenerate a new DNS name.

Fifth, DNS name retrieval is the process to retrieve domain names to represent them on an user application. The authoritative DNS server maintains the DNS information about IoT devices in a zone file. After the DNS name registration, a user's smart device (e.g., smartphone or tablet PC) can retrieve the list of IoT devices from the authoritative DNS server and identify them with either text-based or graphic-based user interfaces. With these interfaces, users can easily monitor and remote-control IoT devices in the target network.

D. Authentication

The authentication is carried out with a key pair and a signature. An AS generates asymmetric keys from a digital

signature algorithm (e.g., RSA-based signature schemes). After generating the pair a signing key and a verification key by the request of a smartphone, the AS sends each key to a router and a smartphone, respectively. The smartphone sends the signing key to an IoT device through NFC, and then the IoT device gets the signing key in the end. While an AS forwards the signing key to the smartphone, a verification key is also sent by the AS to the router. The verification key is used for a router to verify if the signature of an NI reply, which is signed NI with a signing key, is valid or not. If the NI reply does not have security information, it will be marked as no security information is included. If it gets a signing key through NFC, an IoT device can generate the signature for its NI reply by using the signing key. An IoT device makes a hashed value from its domain name using a hash function (e.g., SHA-1). Next, the IoT device signs the hashed value with its signing key. Thus, according to this procedure, the IoT device can construct the digital signature of its DNS registration message, that is, an NI reply. If an IoT device receives an NI query, the IoT device sends an NI reply with a signature to a router. When the router receives the NI reply for the NI query, the router verifies the validity of the signature for the NI reply by using a verification key received from the AS. If the signature is valid, the router tries to register the DNS information of the IoT device into the designated DNS server managing the domain of the IoT device's DNS name via DNS dynamic update [17].

IV. Experiment In Testbed

Lee et al. analyzed the performance of DNSNA, comparing with mDNS as a baseline [2]. Therefore, we show the new results of SDNSNA. We implemented SDNSNA to prove the concept of SDNSNA for DNS name services in a smart-home IoT environment. We used a Raspberry Pi as an IoT device in our SDNSNA testbed, a desktop PC as an authoritative DNS server, a desktop PC as an AS, and a smartphone as an authentication device for IoT devices.

```
Server is ready for connection.
Waiting for connection request...
Connection request from /115.145.178.168
Generating key...
The key successfully generated
n : 20593157760313445862684124655560644973 1
Private Key : 84872074915879313434698157768454866787
Public Key : 12761709240567395723
Nonce : 1662
Time : 1055416
The verification key was successfully sent to CS
```

Fig. 4. A Security Information Generation at an AS

We made a smartphone application. When pressing the connect button after inputting the IP address and port number for the AS, a user receives a signing key generated from the AS. The signing key is used to authenticate the authentication of an IoT device. After receiving the signing key, the IoT device can obtain the signing key from the smartphone via its NFC tag if the smartphone allows for the release of the signing key to an adjacent IoT device via NFC.

Fig. 4 shows the screen of the AS. When the smartphone connects to the AS, then the AS generates asymmetric keys (i.e., private key and public key), a nonce, and a time stamp.

```
SDNSNA Client Program Starting...

Timestamp: 2016/07/26 18:51:19
 - param 1 : interface = eth0
 - param 2 : dnsna-id default = model.category.location
 - param 3 : dnsna-id filename = dnsna_local_id.file
 - param 4 : dnsna-name filename = dnsna_name.file
Write New Domain Name to File : 913811.model.category.home
libllcp.llc.link        service 0x1ad28c0 bound to SAP 1
libllcp.llc.link        service 0x1ad28f0 bound to SAP 4
libllcp.mac.link        (PN532 board via UART) Attempting to
activate LLCP Link as target (blocking)
```

Fig. 5. Execution at Raspberry Pi for DNS Name Generation

```
=======================================
* Send NI Query  Len=15032
   To   ff02::1
=======================================
* Type          : 139, Node Information Query
* Code          : 0, Query Subject is IPv6 Address
* Checksum      : 0x0000
* Qtype         : 2, Node Name
* Flags         : 0x0000
* Nonce         : 0x4139418843843a8d
* IPv6 Addr     : ff02::1
=======================================
```

Fig. 6. Node Information Query for DNS Name Collection

```
=======================================
* Recv NI Reply  Len=15057
   From   fe80::ba27:ebff:fe57:279f
=======================================
* Type          : 140, Node Information Reply
* Code          : 0, Reply is Success
* Checksum      : 0x9ce3
* Qtype         : 2, Node Name
* Flags         : 0x0000
* Nonce         : 0x3966b2673d39be39
* TTL           : 0
* Node Name     : 913811.model.category.location.home
* Security Flag : 1
=======================================
```

Fig. 7. Node Information Reply for DNS Name

```
Found zone name : home
The master is : ns.home
Sending update to 115.145.178.190#53
Outgoing update query :
;; ->>HEADER<<- opcode : UPDATE, status : NOERROR, id : 32518
;; flags :; ZONE : 1, PREREQ : 0, UPDATE : 1, ADDITIONAL : 1
;; UPDATE SECTION :
913811.model.category.location.home. 300 IN AAAA  fe80::ba27:ebff:fe57:279f
```

Fig. 8. DNS Dynamic Update

The key pair is made by the RSA algorithm. In this paper, considering the size of the picture in Fig. 4, the key size was 64 bits. Through the AS, the key size can be adjusted. The recommended length of the key should usually be larger than 2048 bits.

Fig. 5 shows the execution of DNS name generation in Raspberry Pi. An IoT device constructs its unique DNS name after executing the given program, and waits for a signing key from NFC module. Since an NI query does not have signature requirement at this stage, it is sent with security flag having zero value. If a signature information is mandated, an NI query requires an NI reply to include signature information so that it can be used to authenticate an IoT device for the registration of its DNS name.

Fig. 6 shows an NI query [16] received by a Raspberry Pi as an IoT device, which is sent by a router for DNS name collection. The router periodically sends an NI query to IoT devices in the subnet attached to one of its network interfaces so that IoT devices can register their autoconfigured DNS names into the DNS server via the router.

Fig. 7 shows an NI reply [16] received by the router, which is sent from the Raspberry Pi for DNS name registration. The router receives an NI reply having the DNS name and IPv6 address of the IoT device along with the signature. For the received the received NI reply, the router will verify if it is valid or not. If the signature in the NI reply is valid, it stores the DNS information of the IoT device into its local repository.

Fig. 8 shows the DNS dynamic update [17] for registering the DNS name and IPv6 address of the IoT device into an authoritative DNS server. We use the Berkeley Internet Name Domain (BIND) software package as the authoritative DNS server for DNS services. The router checks the existence of the global DNS name of the Raspberry Pi by sending a DNS name query to the DNS server. If no corresponding IPv6 address for the queried DNS name is returned from the DNS server, this means that no other IPv6 node is using that DNS name. In that case, the router registers the DNS information (e.g., DNS name and IPv6 address) of the Raspberry Pi into the DNS server. Otherwise, the router cannot register the DNS information, so it needs to notify the corresponding IoT device of the duplicated DNS name, letting the IoT device generate a new DNS name and try again to register it into the DNS server via the router.

Note that our SDNSNA allows for a partial deployment in that only IPv6 hosts understanding SDNSNA protocol can join our SDNSNA service. Thus, SDNSNA-aware IoT devices can coexist with SDNSNA-unaware IoT devices in the Internet.

V. Conclusion

In this paper, we have proposed a Secure Domain Name System (DNS) Name Autoconfiguration (SDNSNA) for the DNS naming services in Internet of things (IoT) environments. Our goal is to provide a confidential and efficient system for DNS name autoconfiguration for IoT devices with minimum human intervention. With SDNSNA, users can easily perform the DNS name registration of their IoT devices with near field communication (NFC) devices and an Authentication Server (AS). We believe that our SDNSNA is a promising approach for secure IoT DNS naming services for the tremendously increasing number of IoT devices. As future work, we will consider the enhancement of our SDNSNA to allow for the usage of other wireless technologies (e.g., Wi-Fi, ZigBee, and Bluetooth) for the authentication of IoT devices via a smartphone in addition to NFC.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2014006438). This work was also partly supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) [10041244, SmartTV 2.0 Software Platform]. Note that Jaehoon (Paul) Jeong is the corresponding author.

References

[1] Gartner, Inc., "Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020," http://www.gartner.com/newsroom/id/2636073.

[2] Sejun Lee, Jaehoon (Paul) Jeong, and Jung-Soo Park, "DNSNA: DNS name autoconfiguration for Internet of Things devices," in *Proceedings of the 18th International Conference on Advanced Communication Technology (ICACT)*, Feb. 2016.

[3] M2M, "Object Identifier based M2M Device Identification Scheme," http://www.onem2m.org.

[4] J. Jeong, S. Park, L. Beloeil, and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration," *IETF RFC 6106*, Nov. 2010.

[5] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)," *IETF RFC 4861*, Sep. 2007.

[6] R. Droms, J. Bound, B. Volz, C. Perkins, and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," *IETF RFC 3315*, Jun. 2003.

[7] Muhammad Qasim Saeed and Colin D. Walter, "Off-line NFC Tag Authentication," in *Proceedings of the 7th International Conference for Internet Technology and Secured Transactions (ICITST)*, Dec. 2012.

[8] NFC Forum, "What are the operating modes of NFC devices?" http://nfc-forum.org/resources/what-are-the-operating-modes-of-nfc-devices/.

[9] Jiye Park, Saemi Shin, and Namhi Kang, "Mutual Authentication and Key Agreement Scheme between Lightweight Devices in Internet of Things," *The Journal of Korean Institute of Communications and Information Sciences*, vol. 38, no. 9, pp. 707–714, 2013.

[10] 802.11i 2004, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements," *IEEE Standard*, 2004.

[11] Kaikai Liu, Xinxin Liu, and Xiaolin Li, "Guoguo: Enabling Fine-grained Indoor Localization via Smartphone," in *Proceedings of the 11th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*. ACM, 2013, pp. 235–248.

[12] Hui-Huang Hsu, Wei-Jan Peng, Timothy K. Shih, Tun-Wen Pai, and Ka Lok Man, "Smartphone Indoor Localization with Accelerometer and Gyroscope," in *Proceedings of the 17th International Conference on Network-Based Information Systems*. IEEE, 2014, pp. 465–469.

[13] Junghyun Jun, Yu Gu, Long Cheng, Jun Sun, Ting Zhu, and Jianwei Niu, "Social-Loc: Improving indoor localization with social sensing," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2013, p. 14.

[14] Jeong Jaehoon, Jungsoo Park, Hyoungjun Kim, and Kishik Park, "Name Service in IPv6 Mobile Ad-hoc Network," in *Proceedings of the International Conference on Information Networking*. Springer, 2003, pp. 692–701.

[15] Jaehoon Paul Jeong, Sejun Lee, and Jung-Soo Park, "DNS Name Autoconfiguration for Internet of Things Devices," *IETF Internet-Draft draft-jeong-its-iot-dns-autoconf*, Mar. 2016.

[16] M. Crawford and B. Haberman, "IPv6 Node Information Queries," *IETF RFC 4620*, Aug. 2006.

[17] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)," *IETF RFC 2136*, Apr. 1997.