Hide-and-Disclose: On-Site Information Sharing for Privacy-Aware Mobile IoT Communications

Yusuke Fukushima and Ved P. Kafle
Network System Research Institute
National Institute of Information and Communications Technology
4-2-1, Nukui-Kitamachi, Koganei, 184-8795, Japan.
{yfukushima, kafle}@nict.go.jp

Abstract—**In this demonstration, we present a privacy-aware device-to-device communication framework that enables both obfuscation of information source and fine-grade information disclosure control (which we call the Hide-and-Disclose). Instead of using persistently assigned identifiers, introducing randomly generated short-lived identifiers (also known as pseudo-IDs) becomes a popular approach for reducing leakage of sender's location privacy information. To achieve reliable information sharing among mobile IoT devices using pseudo-IDs, issuing pseudo-IDs and information provisioning must be supported by the reliable network service. However, data access by using pseudo-IDs significantly increases database management cost due to the increase in query index size, and frequent invalidation and replication of pseudo-IDs and data associated by the pseudo-IDs. To overcome this problem, Hide-and-Disclosure allows to integrate database operations based on grouping of pseudo-ID and associated data pairs owned by the same IoT device. To further provide fine-grade access control, this framework accepts IoT devices to add rules for data access (i.e., access permission and lifetime) so that IoT devices can disclose own information to the target device. To evaluate the framework, we have implemented the prototype in Linux system.**

I. Introduction

Hiding identifiers, persistently assigned from vendors or having long lifetime, of IoT device is necessary to prevent the revealing of privacy information of the device and its owner (e.g., location and interest) from potential adversaries. Thus, in wireless device-to-device (D2D) communication of mobile IoT devices, introducing randomly generated and short-lived communication identifiers, so-called pseudo-IDs, becomes popular for commercial products of wireless communication systems, such as Bluetooth LE, Wi-Fi Aware, and DSRC. In the conventional mobile IoT applications, pseudo-IDs are mostly used for a stringent message propagation (i.e., car crash avoidance caused by sudden braking) [1] without disclosing the persistent sender's identifier.

However, only using those directional message propagation methods are not possible to realize "autonomic interaction" in privacy-aware mobile IoT communications, which is a key function to contribute the upcoming super-aging society (e.g. construction of extreme eco-systems). As a potential approach to realizing autonomic interaction, pseudo-ID and directory service-based information sharing is proposed [2]. In this method, certifiable information (i.e., public keys and certificates) corresponding to each pseudo-ID is provided from the directory service so that every IoT device can validate the message sender and integrity of the message. This method also provides non-repudiation of message sending, which is important for mutual authentication. However, such a pseudo-ID based infrastructure is not scalable because the index size stored in database system significantly increases by the number of pseudo-IDs. On-demand data replication to cache servers close to mobile IoT devices is also needed since quick data retrieval is important for realizing seamless interaction. In addition, to the best of our knowledge, there is no appropriate access control method to limit the target of information disclosing.

In this demo, we present Hide-and-Disclose, a privacy-aware D2D communication framework enabling both obfuscation of information source and fine-grade access control for information disclosure. By using this framework, IoT devices can share their information with others without using their persistent identifiers.

II. System Overview

In this section, we describe the proposed framework design and prototype implementation.

A. System Architecture

Fig. 1 shows the system architecture of the proposed framework. It consists of a pseudo-ID certificate authority (PCA), Distributed Directory Service (DDS), mobile IoT services, and user terminals (UTs).

PCA: This component is responsible for the generation, validation, and revocation of pseudo-ID and keys. Based on cryptographic algorithm [2], it can provide secure message propagation (i.e., authentication and integrity of sending message, and non-repudiation for message senders). On receiving a request from UT, PCA generates requested number of pseudo-ID and asymmetric key pairs for UT. After the generation process is completed, PCA creates an ID/key map that resolves a public key from the correspondent pseudo-ID. PCA registers the ID/key map to DDS and then allocates the set of pseudo-ID and private key to the UT.

DDS: This component consists of the centralized authorized directory servers and decentralized several cache servers. We assume that DDS has similar functionality design of IoT directory service [3]. In this design, when a UT connects to DDS, the UT is authorized by the main server and the related records are replicated to cache servers nearby the UT. DDS

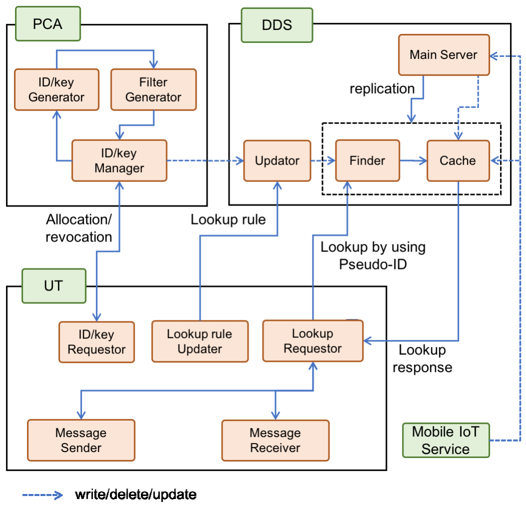

Fig. 1. A system architecture of the propsed on-site information sharing.

can perceives the location of each UT (i.e. mobility context transfer used in cellular networks) to make proactive records replication among cache servers. DDS also accepts various types of data store (i.e. number, text, binary, and so on) to support several IoT services.

Mobile IoT Service: This component is responsible of record insert and delete operations to DDS on behalf of UTs participating in the service. Since records stored in DDS are maintained by the service, data obtained from DDS is fully certified by the service.

UT: This component is a user terminal or mobile IoT device consisting of sensing, computing, and communicating units. The sensing unit obtains quantified information in the surrounding of the user, and computing unit has enough processing performance (e.g., anticipation from sensing data) and power supply. The communication unit is used to communicate with PCA, DDS, and other UTs. For ease of explanation, every UT is authorized by PCA and DDS in advance with using persistently assigned information, such as device ID and security key. For communication between other UTs, they use a special communication API to establish pseudo-ID and key based secure communication.

B. Access Control Mechanism

The motivation to introduce an access control function in this framework is to provide a fine-grade information disclose control for every UT enabling on-site information sharing by D2D communication. For this purpose, introducing a simple access control list (ACL) used in the conventional distributed directory system [4] is not efficient because the ACL size significantly increases by the number of pseudo-IDs and then the ACL becomes very costly to be maintained by DDS. An end-to-end encryption-based access control is another candidate [5]. However, this approach is not easy for updating access rules.

To overcome these problems, the proposed framework introduces a function to perform fast access rule retrieval from group of pseudo-IDs, which is referred to as Finder in this paper. The idea to construct the Finder is threefold.

1) A set of records owned by a UT and access rules is grouped and the set is registered in DDS with the user terminal ID (UTID). Thus, each access rule can be retrieved by using pseudo-ID and UTID. However, since the UTID is a persistently used internal parameter of DDS, using UTID as a lookup ID is not possible for UTs.
2) To retrieve a UTID from the related pseudo-IDs, a hash-based fast information search algorithm, which is called fast hash table [6], is introduced. Although fast hash table algorithm achieve fast lookup even if the number of lookup key becomes large, it still suffers from a lot of insert and delete operations.
3) To further improve Finder performance, a group of pseudo-IDs is integrated into a single bloom filter. Then a set of the bloom filter and lookup rules are inserted in the fast hash table instead of inserting a pseudo-ID one-by-one. Note that those operations are possible in the fast hash table because its data structure is constructed with counting bloom filter.

Fig. 2 illustrates the entire process of the access control mechanism using Finder. When a UT requests pseudo-ID and key allocation to PCA, PCA creates a single M-bit bloom filter from a set of pseudo-IDs in the filter generator function and registers it with a set of pseudo-IDs and public keys to the Finder through the updator function in DDS. In Finder, the registered information, the corresponding lookup rules and UTID are inserted in the fast hash table that consists of M-bit counting bloom filter, where default lookup rule is “blank” that indicating no sharable information. After this process, Finder can obtain public keys from the correspondent pseudo-ID so that UTs can perform mutual authentication using Finder. It is notable that since Finder can quickly find grouped information and UTID from a single pseudo-ID or UTID and remove it and the related records stored in the database at once, Finder can also perform quick replication process.

Once the registration process above is completed, the UT can update lookup rules stored in Finder through the updator function in DDS. In the current design, the following list of sharable data (LSD) and sharable time period (STP) are supported for access control.

- **LSD** can be specified as a list of record key name (e.g., {“car type”, “location”}), or a query rule by using query language of the database (e.g., select car-type, location from UTID). As an extra option, access to a LSD can be restricted by the first requestor ’ s UTID, where the UTID is stored in the optional field of LSD after the first access. Once the field is set, the query rule in the LSD is simply ignored for other UTs.
- **STP** is an optional setting field and can be specified as a time (e.g., 5 seconds) to invalidate the correspondent LSD

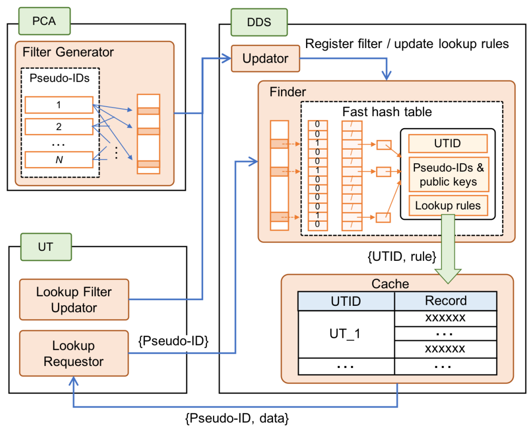

Fig. 2. A pseudo-ID/key store and access control mechanism.

after its first access. If a lookup rule does not have STP field, invalidation of LSD must be performed manually from the UT.

As described above, Finder can provide fine-grade access control of data stored in the database and autonomic invalidation of LSD that will be helpful for autonomic interaction.

III. Prototype Implementation

We implement the prototypes of PCA, DDS, and UT functions (described in Figs. 1 and 2) on Ubuntu 16.04 LTS 64-bit sever and Raspbian wheezy by using C and Python.

To realize secure D2D communication on the basis of pseudo-ID/key pairs, we have implemented Mod-IBS [2]. The server-side function of Mod-IBS is used to realize both pseudo-ID/key generator and manager functions. As a client-side application, *ModSignEncrypt* and *ModSignDecrypt* algorithms are implemented as message sending and receiving functions for D2D communication, respectively. Once a communication session is established by using these functions, the pair of communication initiator and responder pseudo-IDs can be replaced by random IDs during communication for the obfuscation purpose. Since Mod-IBS is constructed with elliptic curve cryptography technique, we use Stanford PBC library version 0.5.14 [7] to implement those functions.

To construct DDS functions, MongoDB version 3.2.20 is used to construct database function. In MongoDB, a database name taken from UTID is created and stored a set of records related to the UTID so that DDS can perform a quick bulk replication of records. Finder function is implemented in C language. In this implementation, Finder consumes 64MB memory space for constructing a counting bloom filter with 2^{24} indexes to achieve reasonable false positive rate for storing 500,000 records.

IV. Demonstration

Fig. 3 shows the demonstration network constructed in a workstation laptop. This network consists of a single PCA, a main data server, three cache servers, and 98 UTs constructed by using virtual isolation methods, such as KVM, docker, and network namespace. To demonstrate with real resource constraint devices, we have also implemented the UT function into two Raspberry Pis connecting to the laptop through wireless communication. ModSignEncrypt and ModSignDecrypt working in Raspberry Pi take about 326ms and 300ms, respectively. Full performance analysis is summarized in [2]. For Finder performance, lookup operation from Raspberry Pi takes about 4.5ms including $6\mu s$ lookup response time and wireless communication delay.

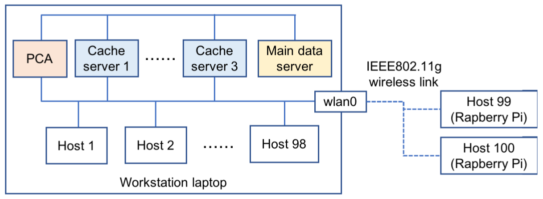

Fig. 3. Demonstration network setup in a workstation laptop.

In this demo, we present information sharing among UTs through the access control function in DDS. DDS stores 100 records for each UT in the main server and replicates them into three cache servers. After PCA completed the pseudo-ID/key allocation, each UT updates access control rule and lifetime to the Finder in the correspondent cache server. Then, each UT selects communication peer and pseudo-ID/key randomly and starts a mutual authentication by using pseudo-ID/key with the peer. After the specified lifetime spent from the first query with pseudo-ID of the target UT, the corresponding pseudo-ID registered in the Finder is invalidated.

V. Conclusion

In this paper, we presented a design and prototype implementation of Hide-and-Disclose, the first proposal for providing a fine-grade control of information disclosure in privacy-aware wireless D2D interaction. By introducing Hide-and-Disclose, each user terminal can specify data access control policy and its lifetime that will be promoting interactions among IoT applications.

References

[1] Certificate Policy for Deployment and Operation of European Cooperative Intelligent Transport Systems, RELEASE 1.1, July 2018.
[2] Y. Fukushima, V. P. Kafle, and H. Harai, “Pseudonym and Key Management Scheme for Supporting Social Smart Applications,” *IEICE Trans. Commun.*, vol. E101-B, no. 8, pp. 1775–1786, Aug. 2018.
[3] V. P. Kafle, Y. Fukushima, P. Martinez-Julia, and H. Harai, “Scalable Directory Service for IoT Applications,” *IEEE Communications Standards Magazine*, vol. 1, no. 3, pp. 58-65, Sept. 2017.
[4] A. Sharma, X. Tie, H. Uppal, A. Venkataramani, D. Westbrook, and A. Yadav, “A global name service for a highly mobile internetwork,” *ACM SIGCOMM’14*, pp.247258, August 2014.
[5] L. Zhou, V. Varadharajan, M. Hitchens, “Achieving Secure Role-Based Access Control on Encrypted Data in Cloud Storage,” *IEEE Trans. Inf. Forensic Secur.*, pp.19471960, vol.8, no.12, October 2013.
[6] H. Song, S. Dharmapurikar, J. Turner, and J. Lockwood, “Fast hash table lookup using extended bloom filter: an aid to network processing,” *ACM SIGCOMM’05*, pp.181-192, August 2005.
[7] Stanford PBC library, https://crypto.stanford.edu/pbc/