future internet

MDPI

Article

Development of User-Participatory Crowdsensing System for Improved Privacy Preservation

Mihui Kim * and Junhyeok Yun

School of Computer Engineering & Applied Mathematics, Computer System Institute, Hankyong National University, Anseong 17579, Korea; junhyeok2723@hknu.ac.kr

* Correspondence: mhkim@hknu.ac.kr; Tel.: +82-2-31-670-5167

Received: 25 February 2020; Accepted: 18 March 2020; Published: 20 March 2020

Abstract: Recently, crowdsensing, which can provide various sensing services using consumer mobile devices, is attracting considerable attention. The success of these services depends on active user participation and, thus, a proper incentive mechanism is essential. However, if the sensing information provided by a user includes personal information, and an attacker compromises the service provider, participation will be less active. Accordingly, personal information protection is an important element in crowdsensing services. In this study, we resolve this problem by separating the steps of sensing data processing and the reward payment process. An arbitrary node in a sensing data processing pool consisting of user nodes is selected for sensing data processing, and only the processing results are sent to the service provider server to reward the data providing node. The proposed user-participatory crowdsensing system is implemented on the Kaa Internet of things (IoT) platform to evaluate its performance and demonstrate its feasibility.

Keywords: crowdsensing; privacy preservation; user-participatory; separation of sensing processing

1. Introduction

Crowdsensing is a technique that uses sensing devices, such as smartphones and wearable devices, to receive, process, and utilize sensing data from the public [1]. Unlike existing sensor networks, it is advantageous in that sensing data can be collected from consumer devices, and this implies a low initial investment cost. A crowdsensing service provider can reward data providers with points, momentary goods, and information to induce the public to participate in providing sensing data [2]. However, the payment of financial rewards requires proper identification of the data provider and, thus, sensitive personal information, such as user location, may be exposed to an attacker who compromises the service provider or its server. This may cause the user to adopt a negative attitude toward providing sensing data and, consequently, the service may not operate properly. Therefore, the protection of personal information in crowdsensing services should be considered a necessity [3].

Crowdsensing is being studied and developed in a wide range of applications that require large amounts of data, such as real-time saturation of public transportation (e.g., as buses and subways) [4], vehicle flow information [5], and store operation hours. The provider of such a service, as a central management entity, performs sensing data processing and pays rewards. Therefore, it can access and store sensitive personal information, such as user location and identification information, which is included in the sensing data. This information may be exposed to an attacker, even if the provider is trusted.

In this paper, we propose a user-participatory crowdsensing system in which user nodes in a data processing node pool participate in sensing data processing, which is assigned to randomly selected data processing nodes, whereas the service provider server (SP) only performs reward data

management. That is, by separating personal and identification information included in the sensing data, an attacker who compromises the *SP* cannot access this information. In addition, a large number of nodes process the sensing data in a distributed manner, thereby minimizing the load on the *SP*. To analyze the security and sensing performance of the proposed system, it was implemented on the Kaa Internet of things (IoT) platform (an open-source IoT platform) [6], PostgreSQL (an SQL database management system—DBMS) [7] optimized for structured data management, and MongoDB (a NoSQL DBMS) [8] optimized for unstructured data management. The Kaa IoT platform is based on Constrained Application Protocol (CoAP) [9], a lightweight message protocol, supports security functions, such as 2048-bit key-based RSA (i.e., an asymmetric cryptography algorithm) encryption, as well as 128-bit and 256-bit key-based AES encryption, and is suitable for secure implementation of the proposed system. We use PostgreSQL to handle different types of sensing data and MongoDB to handle formal data, such as user and reward data. Using different DBMSs depending on the data type improves the data processing efficiency.

In Section 2, we discuss crowdsensing, lightweight messaging protocols, the Kaa IoT platform, and related work. In Section 3, we present the architecture and process flow of the proposed system, which is implemented in Section 4. In Section 5, we demonstrate the feasibility of the proposed system by analyzing its security and sensing data processing performance. Section 6 concludes the paper.

2. Literature Review

In this section, we discuss crowdsensing, lightweight messaging protocols, the Kaa IoT platform, and related work.

2.1. Crowdsensing

Figure 1 shows the general structure of a crowdsensing system. It consists of an *SP* and users [10]. The users can be distinguished into data requesters and data providers. (1.1)(1.2) A data requester requests other users to provide sensing data through the *SP*. A data provider is a user who can provide sensing data that meet the request of a data requester. (2) The data provider provides sensing data and (3) receives a reward. The *SP* manages the sensing data requests and the sensing data received from the data provider, transmits the sensing data to the data requester, and handles the reward data. However, the structure in which the *SP* performs sensing data processing and handles reward data poses a great risk of personal information exposure. In addition, if an attacker compromises only one *SP*, the attacker can access sensitive personal information of all users.

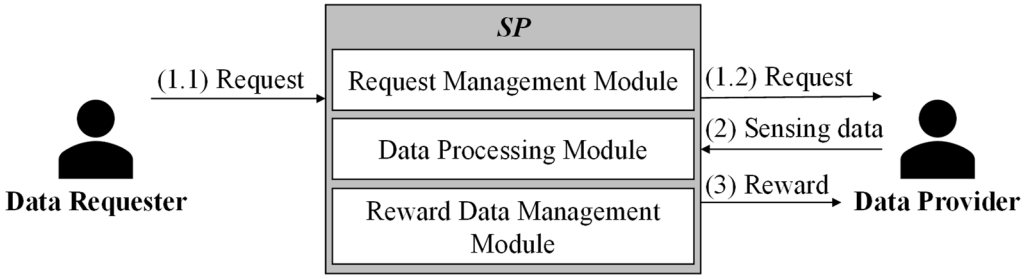

Figure 1. Structure of crowdsensing system.

In this paper, we propose a method for preventing personal information disclosure in conventional crowdsensing systems by separating sensing data processing and reward data management. The *SP* handles the reward data, whereas the processing of the sensing data is delegated to a data processing node managed by a third user other than the data requester and the data provider. This defines a new user type: a data processing node. Each data processing node validates the sensing data and acts as an intermediary for transferring data to the data requester. As a data processing node is randomly selected by the *SP*, it is possible to prevent a malicious user from obtaining personal information of a

user participating in the pool of sensing data processing nodes. The data processing node transmits, to the *SP*, the information about the reward to be paid to the data provider after the validation of the sensing data is completed. Then, the *SP* handles the reward data accordingly. Therefore, unlike in conventional crowdsensing systems, the *SP* cannot access personal information included in the sensing data. In addition, an attacker must attack the *SP* and all data processing nodes to obtain the personal information of a specific user.

2.2. Lightweight Messaging Protocol

Owing to the widespread use of low-power and low-performance devices, including sensors, wearable devices, and IoT sensor devices, systems based on lightweight messaging protocols were extensively studied. These are characterized by small overhead and enable messages to be transmitted to low-power and low-performance devices in low-bandwidth conditions [11,12]. Transmission Control Protocol (TCP)-based Message Queuing Telemetry Transport (MQTT) [13] and User Datagram Protocol (UDP)-based CoAP [9] are the most widely used.

MQTT is a TCP-based lightweight messaging protocol based on the publish/subscribe model. The fixed header length of an MQTT message is two bytes, which is shorter than the fixed header length of the TCP message (20 bytes) and the fixed header length of the UDP message (eight bytes), thereby reducing the length of the entire message. However, the overhead in the handshake process is not suitable for crowdsensing systems, which should establish connection and communicate repeatedly with multiple devices.

CoAP is a UDP-based lightweight messaging protocol. The fixed header length of a CoAP message is four bytes, which is longer than the two bytes of MQTT, but it can be expected to improve performance compared with TCP and UDP. Unlike MQTT, which requires a central broker based on the publish/subscribe model, CoAP allows inter-node communication without a central broker. In addition, the connection establishment overhead is smaller than that of MQTT, which should undergo a three-way handshake process based on TCP. This feature is suitable for crowdsensing systems [14].

In this study, we apply CoAP to the communication between user applications and data processing nodes in the proposed system so that the sensing data can be transmitted and received with high efficiency.

2.3. Kaa IoT Platform

Kaa IoT is an open-source platform for constructing IoT sensor networks [6]. As it is based on CoAP, it reduces communication overhead and supports security functions such as 2014-bit key-based RSA encryption and 256-bit key-based Advanced Encryption Standard (AES) encryption.

Figure 2 shows the architecture of the Kaa IoT platform. It consists of sensor devices, Kaa clusters, and an analytical system for sensing data acquisition. The sensing data are transmitted from a sensor device to a Kaa node. The IoT network manager sets up the sensor device by inserting a Kaa Software Development Kit (SDK) including a sensing data schema, information for the Kaa node to transmit sensing data, and an encryption key for sensing data into the sensor device in advance. The Kaa node stores and manages data received from the sensor device. A Kaa cluster is a cluster consisting of several Kaa nodes. A Kaa cluster manages the nodes constituting the cluster, collects the sensing data stored in each node, and transmits them to the analysis system. The analysis system generates and utilizes information based on the received sensing data.

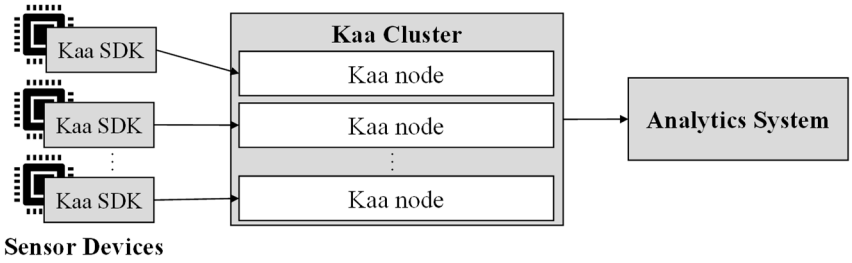

Figure 2. Kaa Internet of things (IoT) platform architecture.

In this study, we use the Kaa IoT platform to implement the basic structure and communication between data processing nodes and the *SP*. A data processing node corresponds to a sensor device in the Kaa IoT architecture. The user application at a data processing node has a built-in Kaa SDK that includes a reward data schema, the address of the *SP*, and so on, and it communicates with the *SP*, which corresponds to a Kaa cluster and the analysis system in the Kaa IoT architecture. The *SP* updates the user database by collecting the reward payment data transmitted by each data processing node through the user application.

2.4. Related Work

The risk of personal information exposure in crowdsensing services discourages users from providing sensing data. This prevents the crowdsensing service from operating properly. Therefore, personal information protection is essential in crowdsensing services, and related research should be based on the possible service types.

Table 1. Classification of crowdsensing services.

Classification Criteria	Classification	
	Vulnerability to Privacy Exposure	Resistance to Privacy Exposure
Request assignment entity	Centralized [15]	Distributed [16]
Request assignment target	Partial [17]	Overall [18]
Participation willingness	Push [19]	Pull [20]
Data providing cycle	Periodic [19]	Participatory [15]
Data preprocessing	Extensive [21]	Non-extensive [22]
Data type	Spatiotemporal [22]	Non-spatiotemporal [23]

We classify crowdsensing services according to the classification criteria in Reference [24], and we present threats and countermeasures against personal information disclosure in each type of service. Table 1 shows the types of crowdsensing services according to various classification criteria. In terms of the request assignment entity, the centralized type is a method in which a central service provider assigns a request to users, and an attacker who compromises the service provider system can access sensitive personal information such as real-time user location. The distributed type is a method via which a user directly participates in a request allocation by a service provider. In contrast to the centralized type, it is difficult for a specific management entity such as a service provider to access sensitive personal information of a user. However, if a service provider or an attacker masquerades as multiple users and participates in a request assignment, it can access sensitive personal information.

In terms of the request assignment target, the partial type is a method in which a request is transmitted only to a user who can provide appropriate sensing data, considering the location and status of the user. Thus, the service provider assigning the request should monitor user location and status in real time. The overall type is a method in which a request is transmitted to all users without considering their location or state, and the service provider does not check sensitive personal information. However, the efficiency of the overall type is low because all requests should be sent to all users.

In terms of participation willingness, the push type requires that the service provider continuously communicates with the user application so that a request is directly sent to the user application. In this process, the service provider can find sensitive personal information such as real-time user location or mobile phone status. The pull type is a method in which the user directly confirms the request information disclosed by the *SP*. Therefore, as the *SP* and the user application need not communicate continuously, the risk of exposure of personal information is relatively low compared with the push type. All pull-type crowdsensing services are partial type.

In terms of the data providing cycle, the persistent type is a method in which sensing data are periodically received without user interference, and the service provider can access real-time user location and status information. The participation type is a method in which sensing data are provided according to user intention when it is determined that the risk of personal information exposure is low or when such an exposure does not become a serious problem.

Some crowdsensing services require extensive data preprocessing, as, for example, in the case of parking lot pictures for smart parking systems [21]. If such preprocessing (e.g., removing the license plate number) is not performed by the sensing data provider, there is a risk of privacy exposure (e.g., by identifying the license plate number and revealing the location of the user). However, services that do not require extensive data processing (e.g., environmental data, such as temperature or vehicle CO_2 emissions) [22] are less vulnerable to privacy exposure. Moreover, crowdsensing services involving spatiotemporal data (e.g., vehicle trajectory) are vulnerable in terms of location or trace privacy [22]. Closed crowdsensing services irrelevant to spatiotemporal characteristics, such as industrial management [23], are resistant to privacy exposure.

In conclusion, distributed, overall, pull, participatory, non-extensive-preprocessing, and non-spatiotemporal crowdsensing services are advantageous compared to centralized, partial, push, periodic, extensive-preprocessing, and spatiotemporal services in terms of privacy protection. It is more difficult for attackers to access sensitive personal information in distributed crowdsensing services than in centralized because this information is processed by multiple users. The overall crowdsensing service sends requests to all users instead of checking user location, status, and so on; thus, it does not experience personal information exposure problems caused by user monitoring. The pull crowdsensing service does not experience personal information exposure problems caused by continuous communication because users check requests directly, and no continuous communication with the service provider is required. In the participatory crowdsensing service, users provide sensing data voluntarily. Thus, if personal information exposure occurs, users can respond immediately by, for example, removing this information or discontinuing participation. We consider these architecture and service types in the design of a crowdsensing system that is resistant to privacy exposure.

The authors in Reference [19] proposed a participatory crowdsensing system using smartphones. The service provider transmits a request to all user applications, and users may respond to the request of their choice. A user can reduce the risk of personal information exposure by providing only selected sensing data to the service provider. However, the central service provider handles all stages of data collection, including request allocation and data processing, so that the personal information of a specific user can be inferred from the data provided by multiple users. In this paper, we propose a system for reducing the risk of personal information exposure in participatory crowdsensing by applying distributed crowdsensing in which users participate in data processing.

The authors in Reference [25] proposed a partial crowdsensing system for road condition monitoring. The application continuously transmits user location information to the service provider. To obtain road condition information at a certain location, the service provider sends a request only to the user at that location; thus, this method is highly efficient. However, as the *SP* continuously collects the location information of all users, the risk of personal information exposure is high. In the present study, to reduce this in a partial crowdsensing system, we use a pull method whereby requests are not allocated to users by the service provider, but rather a user application selects whether to display a request.

We propose a distributed crowdsensing system of pull, partial, and participation type that exhibits relatively low personal information exposure risk, and we implement it using the Kaa IoT platform, CoAP light messaging protocol, and MongoDB.

On the other hand, the proposed mechanism separates the sensing data processing from the reward payment processing in order to preserve the privacy in the data processing by the service provider in the crowdsensing system. Basically, mechanisms to prevent privacy leakage were studied [26,27]. AnonySense [26] is a privacy preservation mechanism through k-anonymity against the system (i.e., it is difficult to link any report to a specific user within a set of k users based on the location or timestamp) and l-anonymity against the applications (i.e., at least l reports are combined together before the aggregate is revealed to the application). The L-diversity system [27] obfuscates data to satisfy L-diversity, which means that a set of the same attribute value must have at least L sensitive information to complement K-anonymity vulnerability. AnonySense assumes that mobile nodes trust the registration authority (RA), and L-diversity also assumes the operation of a centralized system. However, internal attack against RA in AnonySense or the centralized L-diversity system could collapse the privacy preservation. Moreover, it is hard to design the reward management mechanism for the crowdsensing system in the AnonySense and L-diversity system with k- anonymity, and the data quality of both systems could go down due to the aggregation of l reports or the obfuscated data for L-diversity.

Researches on privacy protection in crowdsensing environments were also performed using various technologies [28–30]. The authors in Reference [28] utilized the fog computing technology and, thus, tried to improve the efficiency of blocking privacy leakage by external attackers through fog nodes installed by service provider. Fog nodes collect sensing data and perform data aggregation However, the service provider system is still able to know the exact location of users, and the compromised fog nodes enable the privacy exposure of sensing data. The authors in Reference [29] proposed a crowdsensing system capable of protecting personal information by grouping users using blockchain technology. They tried to secure user anonymity by assigning multiple users to one blockchain node. However, due to the characteristics of the blockchain, there would be a delay in data transmission and reward payment because the transaction speed is lower than that of a general network. The authors in Reference [30] intended to provide anonymity by grouping participants in participant bidding to protect personal information exposed from bids in auction-based crowdsensing systems. Similar to the data processing nodes of our proposed mechanism, the selection of participants through group bidding is passed to the third party and, thus, bid information is protected against the service provider. However, since the third party is fixed and data collected by the service provider is still delivered to the data user, there is a vulnerability to an internal attack on this part. The data processing load is concentrated because the service provider performs data processing. Consequently, many mechanisms were proposed to prevent the exposure of personal information by external attacks. There are not many techniques considering the possibility that a service provider is malicious or compromised. In our proposed mechanism, the service provider does not play any role other than the platform provider, and an external attacker must steal a large number of data processing nodes in order to steal personal information in our proposed system.

3. Proposed System

In this section, we present the structure of a user-participatory crowdsensing system that provides privacy protection by separating sensing data processing and the payment of rewards.

3.1. Structure of Proposed System

Table 2 summarizes the relevant notations. Figure 3 shows the structure of the proposed crowdsensing system. It consists of an SP, a pool of data processing nodes (P), and users (generically denoted by u). Users may be owners of data requester (u_{dr}), data provider (u_{dp}), and data processing nodes (u_{no}). u_{no} may have one or more data processing nodes. Each user u has its own asymmetric

cryptographic key pair $\left(K_{pub}(u),\ K_{pri}(u)\right)$ and uses the public key $(K_{pub}(u))$ as its identifier (ID). The difference between the proposed system and existing systems is that the data processing module is moved from the *SP* to data processing nodes (u_{no}) (Figures 1 and 3). (1.2) The *SP* assigns a request to a data processing node (p_r), which (3) directly forwards the data received from data provider (u_{dp}) to the data requester (u_{dr}). (4.2) The *SP* rewards the data provider (u_{dp}), (4.1) based on the relevant information received from the data processing node.

Table 2. Table of notations.

Notation	Description
$P = \{p_1,\ p_2, \cdots,\ p_r\}$	Pool of data processing nodes
$addr(p_i)$	Address of i-th data processing node
$u_{dr},\ u_{dp}, u_{no}$	Data requester, data provider, and owner of data processing node, respectively
$D = \{d_1,\ d_2, \cdots,\ d_n\}$	Sensing data
$enc(K, d),\ dec(K, d)$	Encrypted/decrypted data d with key K
$i(u)$	Reward amount of user u's incentive amount
$K_{pub}(u),\ K_{pri}(u)$	User u's public key (identifier (ID)) and private key of user u
$K_{ses}(p, u)$	Session key for sensing data transport between data processing node p and user u

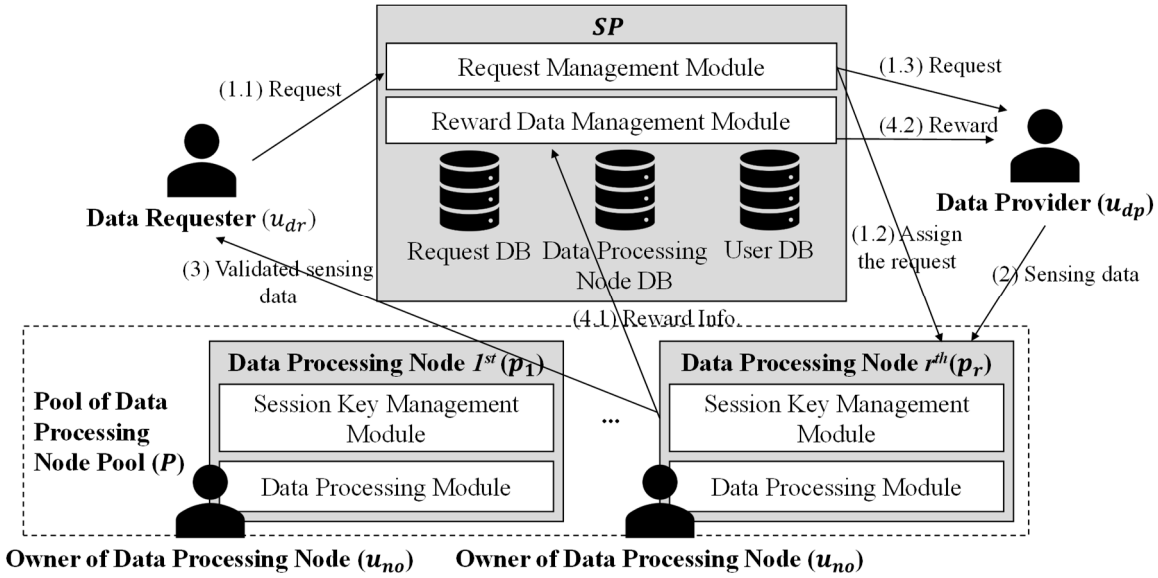

Figure 3. Structure of proposed system.

The *SP* includes a request database, a data processing node database, a user database, a request management module, and a reward data management module. The request database stores the data requester ID $K_{pub}(u)$, the request information for sensing data transmitted by the data requester, and the address $addr(p_i)$ of the data processing node to which the request is assigned. The data processing node database stores the $addr(p_i)$ and $K_{pub}(u_{no})$ of the corresponding data processing node. The user database maps and manages the ID $K_{pub}(u)$ of each user and the reward amount $i(u)$. The request management module forwards the received requests to the appropriate data providers and randomly assigns each request to a node of P. The reward data management module updates the user database using the amount transmitted from the data processing node. The provision request for sensing data includes the description of the requested data, data schema, number of requests, and reward amount.

A pool of data processing nodes is a set of computing nodes owned by users who participate in data processing. The data processing nodes perform sensing data processing for the assigned request. Sensing data processing includes data validation, data aggregation, and transmission to a data requester. The computing nodes that constitute the data processing node pool may vary depending on user participation. The proposed system separates validation management in user identification

for the payment of rewards from sensing data processing and, thus, attackers compromising the *SP* can be prevented from obtaining sensitive personal information included in the sensing data (e.g., location information), together with user identification information. Each data processing node includes a session key issuing module and a sensing data processing module. When the data provider requests participation of the sensing data provision, the session key-issuing module generates a session key $K_{ses}\left(p, u_{dp}\right)$ to be used for sending and receiving the data, encrypts the session key using the ID $(K_{pub}\left(u_{dp}\right))$ of the data provider, and transmits the session key to u_{dp}. The data provider decrypts the encrypted session key using its own private key $(K_{pri}\left(u_{dp}\right))$, encrypts the sensing data using the session key $K_{ses}\left(p, u_{dp}\right)$, and transmits them to the data processing node. The sensing data processing module decrypts the sensing data encrypted using the session key and checks whether they satisfy the data schema defined by the data requester. If it is determined that the data are valid, they are encrypted using the ID $K_{pub}(u_{dr})$ of the data requester and transmitted to the data requester; furthermore, the reward payment data containing the data provider ID and the reward payment amount are transmitted to the *SP*.

3.2. Process Flow of Proposed System

Figure 4 shows the overall process flow at *SP* and p_r of the proposed system. It can be divided into four processes: sensing data request, sensing data provision, sensing data processing, and reward payment.

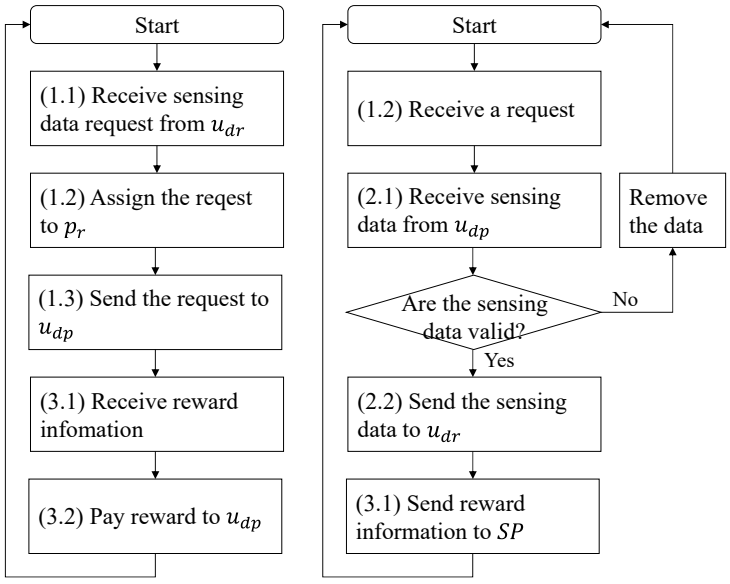

Figure 4. Process flows of proposed system.

3.2.1. Sensing Data Request

Figure 5 shows the process flow from the data requester sending the sensing data request to the *SP* until other users accessing the request. (1.1) The data requester sends the description and schema of the required sensing data, the amount of reward for providing the data, and the data requester ID $K_{pub}(u_{dr})$ to the *SP*. (1.2) The *SP* stores the received request information in the request database and randomly selects one of the data processing nodes that are processing the fewest requests, and it allocates the request. Therefore, it is possible to allocate the requests evenly to all data processing nodes and maximize the processing efficiency. (1.3) The *SP* sends the request information to the assigned data processing node, which uses the data requester ID to transmit the encrypted data after the sensing data are collected. (1.4) The *SP* discloses the request information and the address of the allocated data processing node in the request database so that the user can access the request.

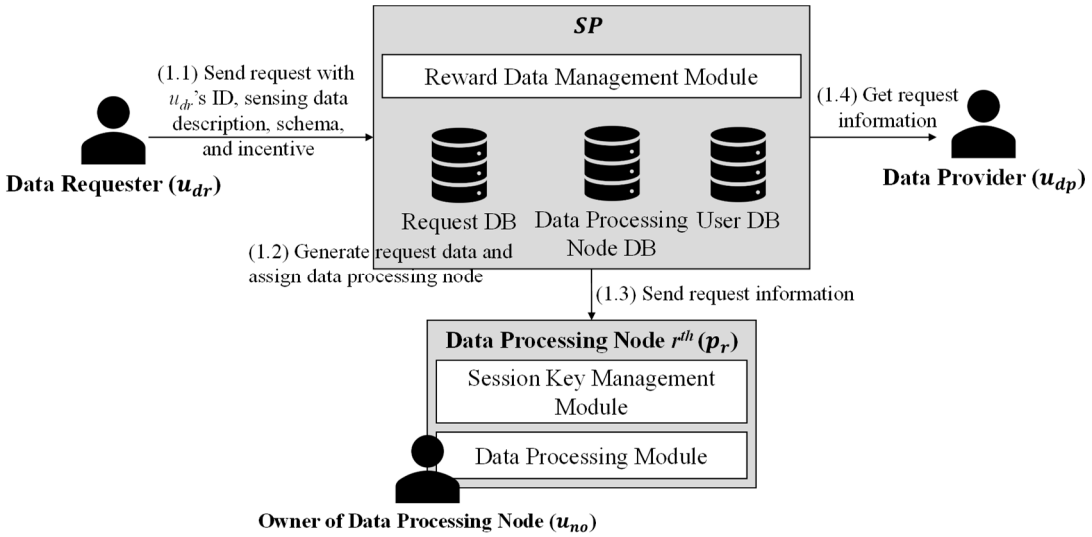

Figure 5. Flow of sensing data request.

By randomly assigning a sensing data provision request, an owner of a malicious data processing node is less likely to be assigned a specific request and, therefore, sensitive personal information included in the corresponding sensing data cannot be obtained.

3.2.2. Sensing Data Provision

Figure 6 shows the process flow of sensing data provision. (2.1) The data provider u_{dp} transmits a data provision request including its ID $K_{pub}(u_{dr})$ to the data processing node. (2.2) The data processing node generates a session key $K_{ses}(p_r, u_{dp})$, encrypts the session key using the data provider ID $K_{pub}(u_{dr})$, and sends it as $enc(K_{pub}(u_{dr}), K_{ses}(p_r, u_{dp}))$ to the data provider. (2.3) The data provider decrypts the received cipher text to obtain the session key, encrypts the collected sensing data d_i with the session key, and sends it as $enc(K_{ses}(p_r, u_{dp}), d_i)$ to the data processing node.

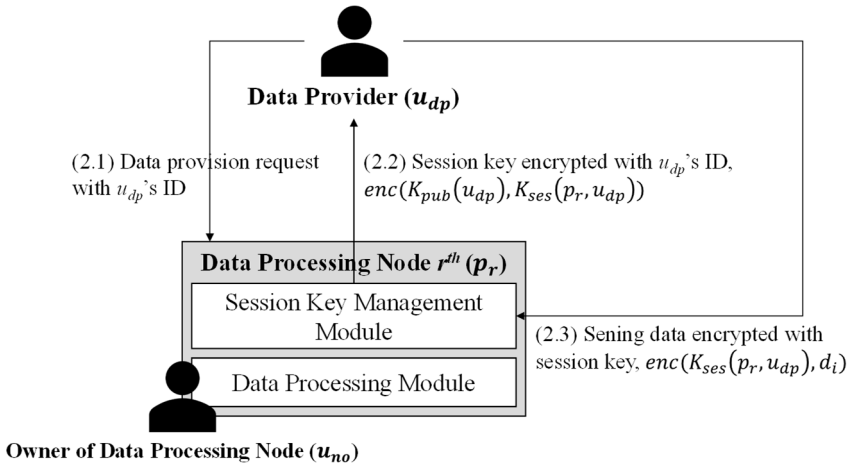

Figure 6. Flow of sensing data provision.

3.2.3. Sensing Data Processing and Reward Payment

Figure 7 shows the process flow in which the data processing node validates the sensing data, provides them to the data requester, and sends the reward information for the data provider to the *SP*. (3.1) The data processing node p_r decrypts the sensing data transmitted by the data provider using the session key $K_{ses}(p_r, u_{dp})$ and verifies whether the data requester satisfies the schema specified by

the data requester. (3.2) If it is determined that the sensing data are valid, the p_r sends to the data requester the sensing data encrypted with data requester ID, $enc(K_{pub}(u_{dp}),\ d_i)$. (3.3) p_r removes the session key $K_{ses}(p_r, u_{dp})$. (4.1) p_r forwards the reward data including the data provider ID $K_{pub}(u_{dp})$ to the *SP*, which updates the reward data on the user database. (4.2) The *SP* finally pays the reward to the data provider u_{dp}.

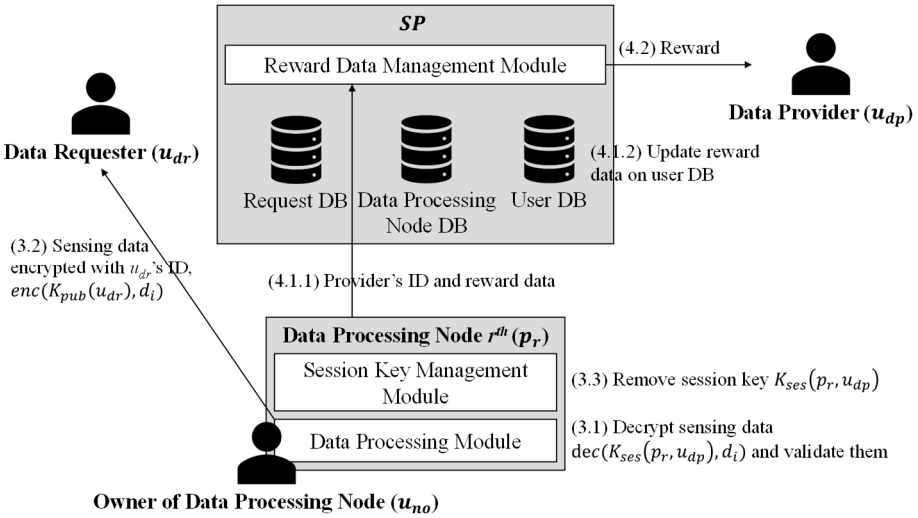

Figure 7. Flow of sensing data processing and reward payment.

4. Implementation of Proposed System

We use the Kaa IoT platform, PostgreSQL, MongoDB, and CoAP to implement the proposed system. Figure 8 shows the process flow between the components of the proposed system. The participating entities are an *SP*, data processing node, data provider, and data requester. Steps 1–4 are requests for providing sensing data. Steps 5–10 correspond to the sensing data provision process, and Steps 11–16 correspond to the sensing data processing and reward payment process.

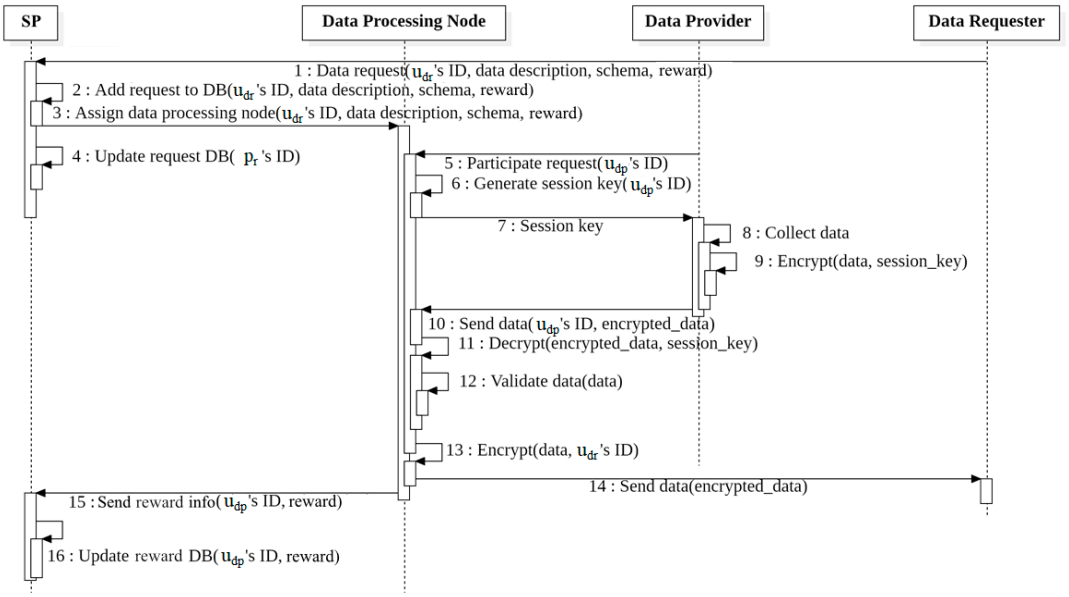

Figure 8. Sequence diagram of proposed system.

4.1. Service Provider Server

The user database, request database, and data processing node pool database were constructed using PostgreSQL. The schema of the user, request, and data processing node pool databases is shown in Tables 3–5, respectively. Communication with the data processing nodes was implemented using the Kaa IoT platform. The Kaa SDK for communication with the *SP* is included in the data processing node registration process. The Kaa SDK contains connection information, encryption algorithms and encryption keys, data schemas, and so on for communication with the *SP*. Reward payment data uses MongoDB, the default endpoint data history database on the Kaa IoT platform. The reward payment data sent by the data processing node are stored in MongoDB on the *SP*, and the reward management module uses these data to update the user database and delete the applied reward data from MongoDB.

Table 3. Schema of user database.

Field Name	Data Type	Filed Content
user_id	String	User ID $K_{pub}(u)$
Reward	Decimal	Reward amount $i(u)$.

Table 4. Schema of data processing node database.

Field Name	Data Type	Filed Content
user_no	Unit	Unique user number
owner_id	String	ID $K_{pub}(u_{no})$ of the owner of the data processing node
node_addr	Unit	Data processing node address $addr(p_i)$
req_count	Unit	Number of assigned requests

Table 5. Schema of request database.

Field Name	Data Type	Filed Content
req_no	Unit	Unique request number
requester_id	String	Data requester ID $K_{pub}(u)$
Description	String	Description for sensing data transmitted by the data requester
Amount	Unit	Amount of requested data
Schema	String	Schema for sensing data
Reward	Decimal	Requested reward
node_no	Unit (foreign key)	Assigned data processing node number

Initially, a user uses an application to generate an RSA-2048 asymmetric cryptographic key pair and sends the public key to the *SP* for registration. Upon receiving the user registration request, the *SP* adds data having a reward field value of 0 to the user database using the public key provided by the user as *user_id*.

The owner of a data processing node sets up a computing node as a data processing node, and then sends its ID and IP address to the *SP* to request data processing node registration. Upon receiving this request, the *SP* provides a data processing node application including the Kaa SDK. Figure 9 shows the interface for creating and issuing the Kaa SDK in the *SP*. The owner of the data processing node installs the provided application on the computing node so that it can operate as a data processing node. The detailed structure and process flow of the data processing node application are described in Section 4.2. After verifying that the new data processing node is functioning properly, the *SP* adds a req_count value of 0, with the ID of the owner of the data processing node as *owner_id* and the data processing node IP address as *node_addr*, to the data processing node database. The *user_no* field value for the new data processing node is set to the smallest value that does not overlap with the *user_no* value of other data processing nodes. The *SP* periodically communicates with the data

processing node and deletes it from the corresponding database if it is determined that the node does not operate normally.

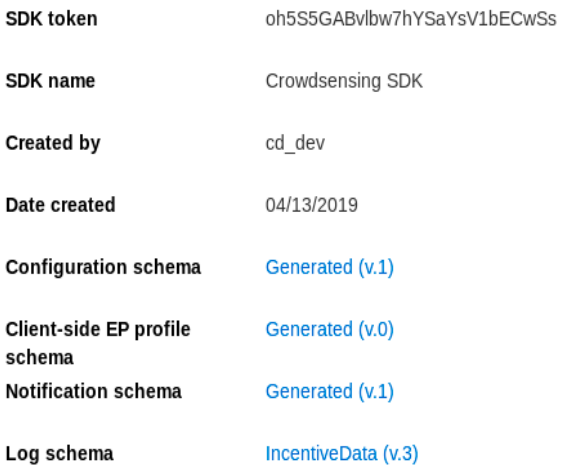

Figure 9. Profile management interface of Kaa SDK.

The data requester sends the description of the requested sensing data, the requested quantity, the data schema, the reward payment amount, and its own ID to the *SP*, and requests the provision of the sensing data. The *SP* adds the data requester ID as *requester_id*, the description of the requested data, the amount of the requested data, and the data schema to the request database. The *req_no* field value of a new request is set to the smallest value that does not overlap with the *req_no* value of other requests. One of the data processing nodes with the smallest *req_count* value is randomly selected and assigned to the request, the *user_no* field value of the request is set to the *user_no* value of the allocated data processing node, and the *req_count* value of the data processing node increases by one. The *SP* transmits the data schema to the data processing node assigned to the request to be used for sensing data validation, and then deletes from the request database the request for which the data processing node sends the completion signal.

4.2. Data Processing Node

The data processing node includes an application provided by the *SP*. Figure 10 shows the structure of such an application. It includes a participant management module, a data processing module, the Kaa SDK, a participant database, and a sensing data management database.

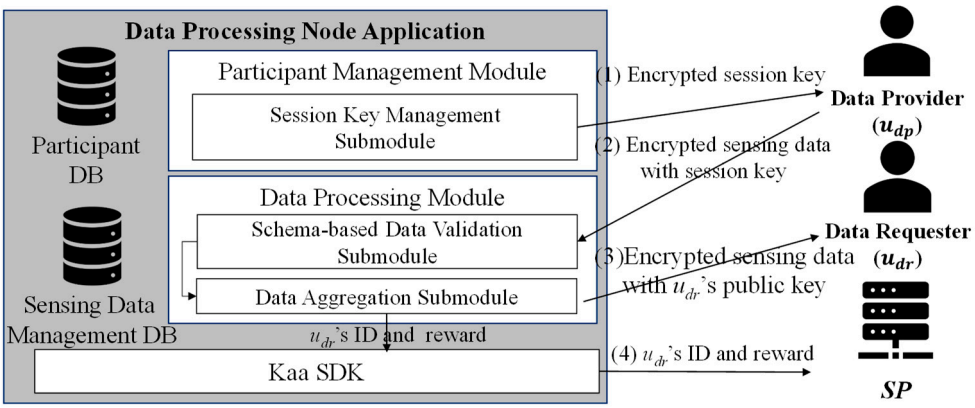

Figure 10. Application structure of data processing node.

As shown in Figure 10, the session key management submodule in the participant management module stores and manages the ID and the session key used by a participant to provide sensing data,

(i.e., data provider). The participant database for storing session keys is implemented in PostgreSQL. The schema of the participant database is shown in Table 6. When the data provider sends a join request to a data processing node, the node generates a session key and stores it in the participant database along with the data provider ID. (1) The session key management submodule securely transfers the session key encrypted with this public to the data provider. (2) When providing the sensing data, the data provider encrypts the data with a session key.

Table 6. Schema of participant database. ID—identifier.

Field Name	Data Type	Filed Content
provider_id	String	ID of user participating as data provider
sess_key	String	Session key used by the data provider

The data processing module consists of a schema-based data validation submodule and a data aggregation submodule. The former receives the sensing data from the data provider using the CoAP protocol. The encrypted sensing data is decrypted using the session key stored in the participant database. The data validation submodule checks whether the sensing data satisfy the schema specified by the data requester. If it is determined that the sensing data are valid, they are transferred to the data aggregation submodule, which collects and appropriately processes them in the sensing data management database. The databased is constructed using MongoDB, considering the characteristics of the crowdsensing system, which can handle various types of sensing data. (3) The data aggregation submodule transmits directly the encrypted sensing data with the ID of the data requester (u_{dr}) (i.e., public key) to the data requester. (4) When the requested sensing data are received, the data aggregation submodule sends the ID of the data provider and the reward payment amount to the *SP*. Finally, the data aggregation submodule deletes all data related to the data provider including the session key, and the reward data are from the sensing data management database. If the data validation submodule determines that the sensing data are not valid, they are discarded.

Through this series of processes, the personal information included in the sensing data and the ID of the data provider are not linked, and, at the time of providing the reward, the *SP* cannot know any information other than the reward payment amount.

4.3. User Application

The user application is used by the data requester to send a request to the *SP*, or by the data provider to retrieve the request and provide sensing data. It also generates and manages RSA-2048 asymmetric cryptographic key pairs required for participating in crowdsensing systems.

The data requester inputs the description, schema, and reward amount of the requested sensing data through the user application, which transmits them with the data requester ID to the *SP*. The data provider retrieves the sensing data provision request through the user application and provides the sensing data. In this process, the user application requests a session key, encrypts and transmits sensing data, and generates a CoAP message.

5. Performance Evaluation

5.1. Secruity Analysis

In this paper, we propose a distributed crowdsensing system with pull, partial, and participatory characteristics. Instead of sending the sensing data provision request directly to the user, the service provider announces the request and the address of the data processing node to which the request is assigned so that the user can access the data processing node voluntarily. Therefore, the service provider or an attacker cannot access real-time personal information of a specific user, and the user can stop providing sensing data as soon as it is suspected that personal information was exposed or provide

unnecessary personal information for sensing information as much as possible [21] (e.g., car number deletion when providing photos in smart parking system).

The proposed system separates the reward data management nodes from the sensing data processing node so that sensitive personal information included in the sensing data may not be exposed to the service provider or an attacker who compromised the *SP*. Sensing data are encrypted and transmitted using a session key shared between the data processing node and the data provider, which transmits its ID together with the header of the sensing data. The data processing node decrypts the sensing data to verify their validity, generates the reward payment data that maps the data provider ID and the compensation amount, and transmits the compensation payment data to the *SP*. As soon as the transmission is completed, the data processing node removes the header of the sensing data, encrypts it with the data requester ID, and stores it in the database. Therefore, after the reward payment data are transmitted to the *SP*, it is not possible to determine the identity of the provider of the specific sensing data. To both obtain the sensing data and identify their provider, the data processing node should be attacked between the decryption of the sensing data and the transmission of the reward payment data. This is a particularly short time, and, as the decrypted sensing data are not stored in the auxiliary memory, a long attack, such as memory reversal, should be carried out. Therefore, it is difficult to attack the data processing node and obtain both the sensing data and the identification information of the data provider within a short time.

In addition, the proposed system is advantageous in the case where personal data can be exposed through multiple series of data analysis from a user (e.g., location trajectory privacy) by distributing sensing data processing subjects to several data processing nodes instead of a central *SP*. In other words, if one *SP* processes all the data, even if the user deletes the information that can expose personal information and send it, there is a risk of exposure of secondary information (e.g., health exposure following periodic hospital visits) by analyzing multiple data. However, since the proposed system is randomly assigned to several nodes for data processing, the risk of personal information exposure by multiple data analysis can be prevented.

The *SP* and a malicious data processing node may collaborate to allocate multiple sensing data providing requests to a data processing node and obtain the sensing data and the identification information of the data provider. However, in this case, it can be detected by the data provider that a specific data processing node is assigned an unusually large number of requests. Therefore, the data provider may refrain from providing the sensing data for the request processed by the corresponding node.

In implementing the proposed system, the user database, request database, and data processing node pool database were constructed using PostgreSQL. The security vulnerabilities of session key management through PostgreSQL are the same as the normal PostgreSQL security vulnerabilities. The session key is securely exchanged with the data processing node using RSA public key encryption when the data provider participates in the operation. Therefore, there are two ways that an attacker can steal the session key from the initial exchange or through the storage attack of the data processing node. Since the safety of the RSA public key cryptography is already proven, a possible attack is to threaten PostgreSQL within the data processing node. The attack can be prevented by applying the solutions (e.g., PostgreSQL anomaly detector [31] and homogeneous encryption implementation for Relational DBMSs such as PostgreSQL [32]) that are DB security solutions to compensate for the security vulnerabilities of PostgreSQL DB. PostgreSQL also provides its own security solutions, such as access control lists (ACLs) [33] and encryption [34], which can be used to improve the key management security of our system.

To ensure the enhanced security of the proposed system, we measure the number of data processing requests allocated to these nodes continuously as the number of malicious or compromised data processing nodes increase, as shown in Figure 11. In this experiment, we perform an experiment to allocate 100 data processing requests. In order for an attacker to obtain personal information from advanced analysis (e.g., the movements or behavior patterns of a particular user), the attacker would

have to steal the continuous record of the user (i.e., multiple series of sensing data). If the central *SP* processes all the sensed data, the attacker can access all the sensed data simply by stealing the authority of *SP*. However, in the proposed system, an attacker must take over all data processing nodes in order to access all the sensing data. Even if an attacker successfully compromises 50% of the total data processing nodes, the number of consecutive sensing data that an attacker can obtain is very low (i.e., less than 10). It can be seen that an attacker must threaten at least 72% of the data processing nodes in order to reliably steal 10 or more consecutive sensing data. Depending on the number of data processing nodes that compose the system, the cost of threatening the data processing nodes would vary, but the cost of attacking the data processing nodes, which account for 72% of the total, is generally expected to be very high. In addition, unlike existing systems where service providers may use malicious sensing data in a malicious way, a service provider in the proposed system acts as a checker for the data processing nodes and, thus, post-processing after the problem occurs is also possible. That is, if the information provided by the data processing node is incorrect, the service provider can restrict the participation of the node.

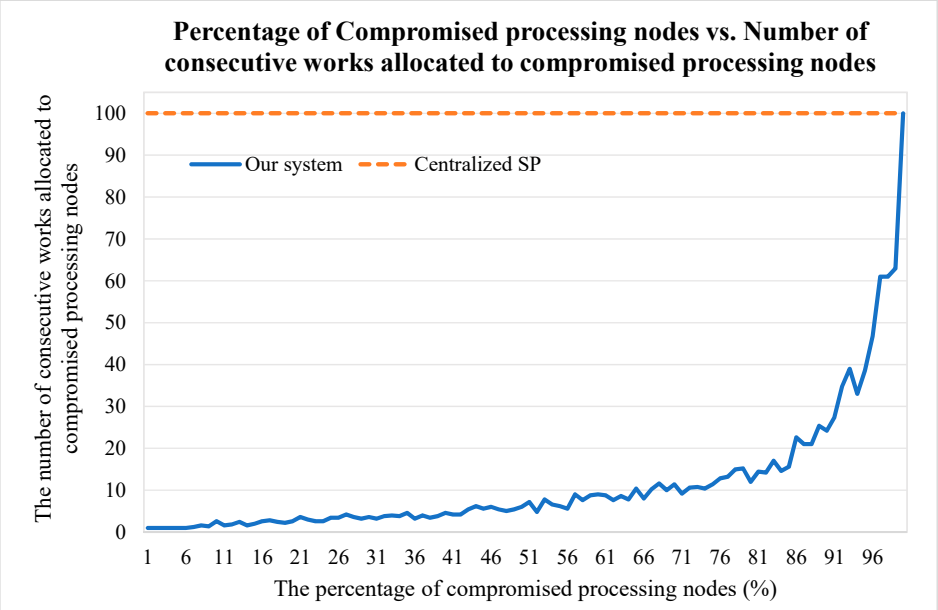

Figure 11. Compromised processing nodes vs. number of consecutive works allocated to compromised processing nodes.

5.2. Processing Performance

To demonstrate the sensing data processing efficiency of the proposed system, the request processing time for the allocated sensing data was measured for varying number of data processing nodes. The request processing time is the time required for completing all the sensing data processing requests from the time when the first request is allocated and opened to the time when the sensing data for the last request is transmitted to the data requester and the reward information is updated (Steps 5–16 in Figure 8). The number of sensing data requests was set to 10, 50, and 100. Ten data providers generated and sent random values as sensing data as soon as they received a new sensing data request. The experiment was performed on the implemented system, as explained in Section 4, and all nodes were connected with a network of virtual machines. The final results were obtained by averaging the results of 10 independent runs of the same experiment.

In order to analyze the efficiency of distributed processing of data processing nodes, we compare the proposed system with the method of performing all data processing in one *SP*. The processing method using the conventional approach is shown in Figure 12. This method is also implemented to

store sensing data, reward information, and user information using PostgreSQL, and the data provider encrypts and transmits the publicly disclosed public key of *SP* instead of exchanging a session key with the service provider. The Kaa framework used for data processing node participation, sensing data transmission, and so on is not used in the central *SP* processing experiment. The request processing time measured in the conventional method is the time required for processing Steps 3–10 in Figure 12.

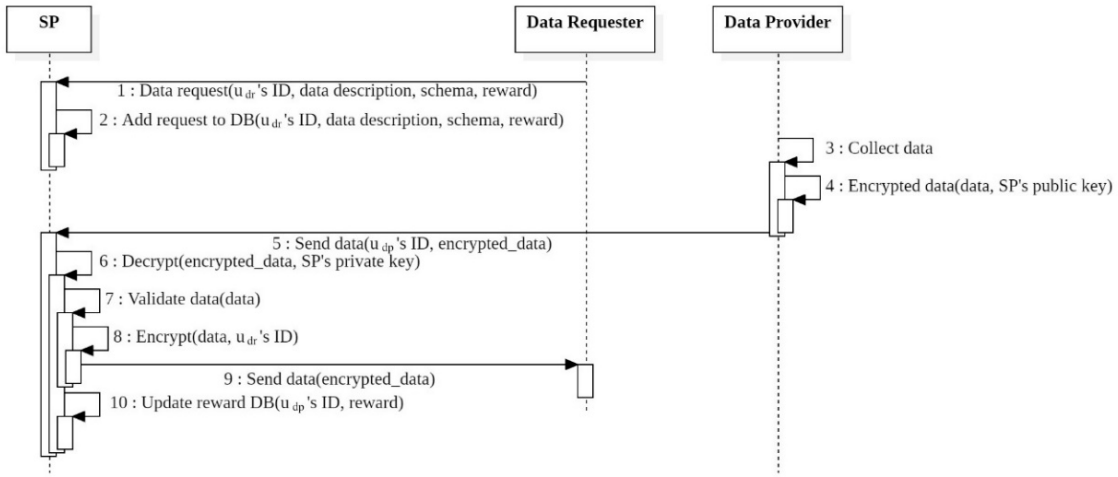

Figure 12. Sequence diagram of traditional method of processing data at the service provider server (*SP*).

Figure 13 shows the request processing time of proposed and traditional methods according to the number of data processing nodes. When the central *SP* processes all the sensing data, it takes 519.858 s to process 100 requests, which is faster than the processing by one data processing node which takes 534.985 s. This difference can be analyzed because the *SP* processes the sensing data, unlike the case in which the data processing node processes the session key exchange and reward information transfer. However, even if only one data processing node is added, the processing speed of the proposed system that distributes and processes the sensing data is faster. In particular, it can be seen that, as the number of data processing nodes increases, the request processing time clearly decreases. The results demonstrate that the presence of a larger number of data processing nodes in the proposed system implies better sensing data processing performance. However, even if a data processing node is configured with a relatively small number of nodes, a significant performance improvement can be expected. That is, distributed processing (i.e., the involvement of several data processing nodes) can significantly reduce processing time. In addition, the proposed system not only has superior security compared to the central *SP* processing method, but also has an advantage in the processing efficiency of sensing data.

Table 7 shows the average time required to process one sensing datum. The participation request and sensing data encryption/decryption time is less than 1 ms and does not affect the total time. Most of the total time is required for session key sharing using asymmetric encryption, for encrypting the sensing data with the requester ID, and for sending the data. Regardless of the total number of requests, the case of a single data processing node involved the longest processing time, as it can be expected that a single data processing node should perform all steps up to issuing a session key for all sensing data and processing time-consuming sensing data. The case of 10 data processing nodes involved the shortest total request processing time. As the number of data processing nodes exceeded five, the request processing time decreased slowly. These results demonstrate that relatively high data processing performance can be achieved with five or more data processing nodes.

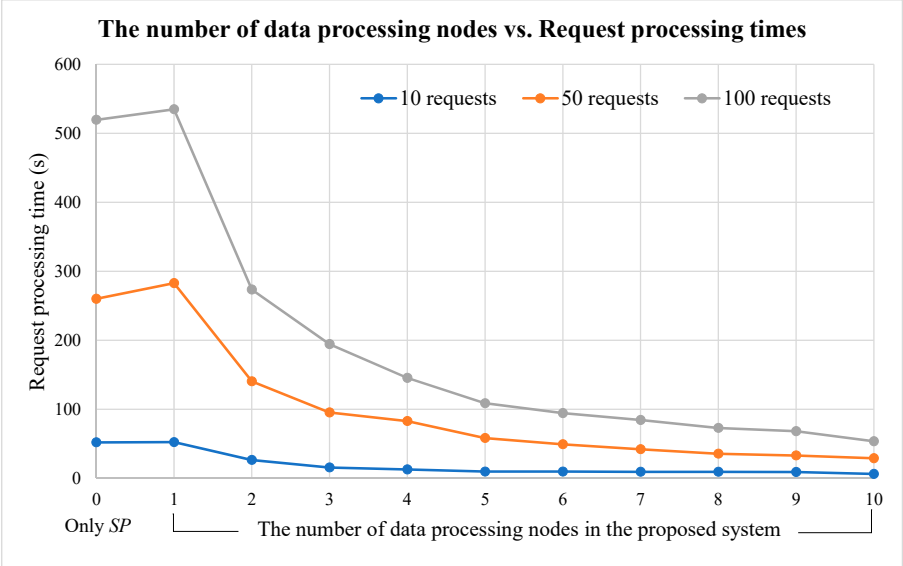

Figure 13. Number of data processing nodes vs. request processing time (only SP processes are requested when the number of data processing nodes is 0).

Table 7. Average time required to provide sensing data.

Step	Time (ms)
Establish connection and send participation request	0.2
Generate and send session key	27
Encrypt data with session key	0.5
Decrypt data with session key	0.5
Encrypt data with requester ID	25
Total	54.2

Accordingly, the crowdsensing service provider may increase the participation of a data processing node by, for example, rewarding the node owner if insufficient processing capacity is expected depending on the number of available data processing nodes.

6. Conclusions

In this paper, we proposed a method for separating reward payment and sensing data processing nodes by delegating data processing to random data processing nodes. This distributed processing method can prevent exposure of sensitive personal information included in sensing data in a crowdsensing system, as well as increase the processing performance. In addition, we demonstrated the feasibility of the proposed system by implementing it and analyzing its processing and security performance.

Author Contributions: M.K. and J.Y. completed this work. M.K. organized the design and development of the proposed system in this work and focused on writing the paper. J.Y. implemented and experimented the prototype of the proposed system. M.K. guided this whole study as the corresponding author. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by a National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2018R1A2B6009620).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Burke, J.; Estrin, D.; Hansen, M.; Parker, A.; Ramanathan, N.; Reddy, S.; Srivastava, M.B. Participatory sensing. In Proceedings of the World-Sensor-Web Workshop Collocated with ACM SenSys, Boulder, CO, USA, 31 October–3 November 2006.
2. Zhang, X.; Yang, Z.; Sun, W.; Liu, Y.; Tang, S.; Xing, K.; Mao, X. Incentives for Mobile Crowdsensing: A Survey. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 54–67. [CrossRef]
3. Vergara-Laurens, I.J.; Jaimes, L.G.; Labrador, M.A. Privacy-Preserving Mechanisms for Crowdsensing: Survey and Research Challenges. *IEEE Internet Things J.* **2017**, *4*, 855–869. [CrossRef]
4. Farkas, K.; Feher, G.; Benczur, A.; Sidlo, C. Crowdsensing based public transport information service in smart cities. *IEEE Commun. Mag.* **2015**, *53*, 158–165. [CrossRef]
5. Pan, B.; Zheng, Y.; Wilkie, D.; Shahabi, C. Crowdsensing of traffic anomalies based on human mobility and social media. In Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems—SIGSPATIAL, Orlando, FL, USA, 5–8 November 2013; pp. 344–353.
6. Kaa IoT Platform. Available online: https://www.kaaproject.org/ (accessed on 18 February 2019).
7. Postgre SQL. Available online: https://www.postgresql.org/ (accessed on 18 February 2019).
8. Mongo DB. Available online: https://www.mongodb.com/ (accessed on 18 February 2019).
9. Shelby, Z.; Hartke, K.; Bormann, C. The Constrained Application Protocol (CoAP). Available online: https://tools.ietf.org/html/rfc7252 (accessed on 13 March 2020).
10. Ganti, R.; Ye, F.; Lei, H. Mobile crowdsensing: Current state and future challenges. *IEEE Commun. Mag.* **2011**, *49*, 32–39. [CrossRef]
11. Caro, N.D.; Colitti, W.; Steenhaut, K.; Mangino, G.; Reali, G. Comparison of two lightweight protocols for smartphone-based sensing. In Proceedings of the 2013 IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux (SCVT), Namur, Belgium, 21 November 2013; pp. 1–6.
12. Tang, K.; Wang, Y.; Liu, H.; Sheng, Y.; Wang, X.; Wei, Z. Design and Implementation of Push Notification System Based on the MQTT Protocol. In Proceedings of the 2013 International Conference on Information Science and Computer Applications (ISCA 2013), Changsha, China, 8–9 November 2013.
13. Hunkeler, U.; Truong, H.L.; Stanford-Clark, A. MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In Proceedings of the 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08), Bangalore, India, 5–10 January 2008; pp. 791–798.
14. Thangavel, D.; Ma, X.; Valera, A.; Tan, H.-X.; Tan, C.K.-Y. Performance evaluation of MQTT and CoAP via a common middleware. In Proceedings of the IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Singapore, 21–24 April 2014; pp. 1–6.
15. Hull, B.; Bychkovsky, V.; Zhang, Y.; Chen, K.; Goraczko, M.; Miu, A.; Shih, E.; Balakrishnan, H.; Madden, S. A distributed mobile sensor computing system. In Proceedings of the 4th International Conference on Embedded Networked Sensor Systems, Boulder, CO, USA, 31 October–3 November 2006; pp. 125–138.
16. Tuncay, G.S.; Benincasa, G.; Helmy, A. Autonomous and distributed recruitment and data collection framework for opportunistic sensing. *ACM SIGMOBILE Mob. Comput. Commun. Rev.* **2013**, *16*, 50–53. [CrossRef]
17. Eisenman, S.B.; Miluzzo, E.; Lane, N.D.; Peterson, R.A.; Ahn, G.S.; Campbell, A.T. Bikenet: A mobile sensing system for cyclist experience mapping. *ACM Trans. Sens. Netw. (TOSN)* **2009**, *6*, 1–39. [CrossRef]
18. Ganti, R.K.; Jayachandran, P.; Abdelzaher, T.F.; Stankovic, J.A. Satire: A software architecture for smart attire. In Proceedings of the 4th International Conference on Mobile Systems, Applications and Services, Uppsala, Sweden, 19–22 June 2006; pp. 110–123.
19. Das, T.; Mohan, P.; Padmanabhan, V.N.; Ramjee, R.; Sharma, A. Prism: Platform for remote sensing using smartphones. In Proceedings of the 8th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys), San Francisco, CA, USA, 15–18 June 2010; pp. 63–76.
20. Shin, M.; Cornelius, C.; Peebles, D.; Kapadia, A.; Kotz, D.; Triandopoulos, N. Anonysense: A system for anonymous opportunistic sensing. *J. Pervasive Mob. Comput.* **2010**, *7*, 16–30. [CrossRef]
21. Yun, J.; Kim, M. Smart Parking System Using Mobile Crowdsensing: Focus on Removing Privacy Information. In Proceedings of the Korea Information Processing Society Conference, Seoul, Korea, 11–12 May 2018; pp. 32–35. [CrossRef]

22. Silva, M.; Signoretti, G.; Oliveira, J.; Silva, I.; Costa, D.G. A Crowdsensing Platform for Monitoring of Vehicular Emissions: A Smart City Perspective. *Future Internet* **2019**, *11*, 13. [CrossRef]
23. Pilloni, V. How Data Will Transform Industrial Processes: Crowdsensing, Crowdsourcing and Big Data as Pillars of Industry 4.0. *Future Internet* **2018**, *10*, 24. [CrossRef]
24. Pournajaf, L.; Garcia-Ulloa, D.A.; Xiong, L.; Sunderam, V. Participant Privacy in Mobile Crowdsensing Task Management: A Survey of Methods and Challenges. *ACM SIGMOD Rec.* **2015**, *44*, 23–34. [CrossRef]
25. Eriksson, J.; Girod, L.; Hull, B.; Newton, R.; Madden, S.; Balakrishnan, H. The pothole patrol: Using a mobile sensor network for road surface monitoring. In Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services—MobiSys, Breckenridge, CO, USA, 17–20 June 2008; pp. 29–39.
26. Kapadia, A.; Tri, N.; Cornelius, C.; Peebles, D.; Kotz, D. Anonysense: Opportunistic and privacy-preserving context collection. In Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services (MobiSys), Breckenridge, CO, USA, 17–20 June 2008; pp. 280–297.
27. Machanavajjhala, A.; Kifer, D.; Gehrke, J.; Venkitasubramaniam, M. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data TKDD* **2017**, *1*, 3-es. [CrossRef]
28. Basudan, S.; Lin, X.; Sankaranarayanan, K. A privacy-preserving vehicular crowdsensing-based road surface condition monitoring system using fog computing. *IEEE Internet Things J.* **2017**, *4*, 772–782. [CrossRef]
29. Wang, J.; Li, M.; He, Y.; Li, H.; Xiao, K.; Wang, C. A blockchain based privacy-preserving incentive mechanism in crowdsensing applications. *IEEE Access* **2018**, *6*, 17545–17556. [CrossRef]
30. Li, T.; Jung, T.; Qiu, Z.; Li, H.; Cao, L.; Wang, Y. Scalable privacy-preserving participant selection for mobile crowdsensing systems: Participant grouping and secure group bidding. *IEEE Trans. Netw. Sci. Eng.* **2018**. [CrossRef]
31. Shebaro, B.; Sallam, A.; Karma, A.; Bertino, E. PostgreSQL anomaly detector. In Proceedings of the 14th Annual Information Security Symposium (CERIAS), West Lafayette, IN, USA, 3–4 April 2013.
32. Popa, R.A.; Zeldovich, N.; Balakrishnan, H. CryptDB: A Practical Encrypted Relational DBMS. Available online: http://people.csail.mit.edu/nickolai/papers/raluca-cryptdb-tr.pdf (accessed on 19 March 2020).
33. Postgre SQL. Access Control. Available online: https://www.postgresql.org/docs/6.5/security13618.htm (accessed on 10 March 2019).
34. Postgre SQL. Encryption Options. Available online: https://www.postgresql.org/docs/8.1/encryption-options.html (accessed on 10 March 2019).