

Blockchain-based Data Management System Exploiting Consistent Hashing for Data Privacy in Urban Computing

Gabin Heo

*Division of Artificial Intelligence and Software
Ewha Womans University
Seoul, Korea
gjrqls@ewhain.net*

Inshil Doh

*Department of Cyber Security
Ewha Womans University
Seoul, Korea
isdoh1@ewha.ac.kr*

Abstract—With the advancement of artificial intelligence, the importance of data has been increasing, leading to a growing demand for intelligent systems that can utilize various types of data in real time. Although data sharing is essential for such systems, traditional centralized data sharing architectures face several limitations, including a Single Point of Failure (SPoF), issues related to data ownership and trust, and privacy violations caused by large-scale data leaks. To address these issues, various studies have explored the integration of additional security mechanisms. However, in systems that require real-time processing, these mechanisms often result in performance degradation. Therefore, a lightweight data protection method is needed to ensure privacy while maintaining performance. In this paper, we propose a blockchain-based data management system designed to enable secure data sharing. The proposed system classifies data into general data and sensitive data, applying a stepwise privacy-preserving approach. General data is randomly distributed through similarity-based shuffling among base stations within the same shard, preventing any specific group from inferring users' behavior patterns. In contrast, sensitive data is stored across shards in a distributed manner without shuffling to minimize pattern exposure during communication. Furthermore, the system leverages consistent hashing to efficiently manage data, reducing the overhead caused by node additions or removals during system scaling. Through this design, our system demonstrates that it is possible to achieve a balance between security, privacy protection, and performance in a real-time data sharing environment.

Index Terms—Blockchain, Data management, Privacy, Consistent hashing, IoT

I. INTRODUCTION

Urban computing is the process of collecting and analyzing data generated by devices within a city to address major urban challenges [1]. Various IoT devices involved in urban computing produce diverse types of data through their embedded sensors. As artificial intelligence technology advances, the data collected within cities is increasingly being used as training data for other devices, making its importance greater than ever. In particular, for intelligent systems, each node shares data from various devices to build diverse training datasets. However, this process raises significant privacy concerns, making research on data privacy protection more important than ever.

Data sharing systems must ensure that multiple participants can exchange data safely and efficiently. However, traditional data sharing methods rely on centralized data storage for integrated management, which leads to issues such as Single Point of Failure (SPoF) and availability problems. In addition, there are concerns related to data ownership and data tampering. These issues can be mitigated by utilizing blockchain technology. Blockchain can be used to track data ownership and protect against tampering. However, as data sharing becomes more widespread, privacy issues become increasingly severe. For example, in traffic management or location-based services, repeatedly recorded location data can be used to infer a user's movement paths or activity range. This can reveal sensitive personal patterns, such as their home address, workplace, or frequently visited places. In addition, by analyzing information such as the time, frequency, and parties of transactions, it is possible to infer a user's behavior and relationships, a process known as transaction pattern analysis. An attacker can track transaction patterns over an extended period to identify the activity characteristics of specific nodes or users and combine this information with location data to perform more precise profiling. These issues not only violate personal privacy but also discourage users from sharing data, thereby reducing participation in the system and ultimately degrading the quality of urban computing services. Therefore, in blockchain-based data sharing systems, a new privacy-focused design is essential to protect such data.

Therefore, this paper proposes a method for managing shared data by distinguishing between sensitive data and general data to enhance privacy protection. General data is shuffled and stored based on similarity with base stations from different regions within a shard, preventing the identification of specific users. In contrast, sensitive data is distributed across multiple shards to make it more difficult for attackers to perform pattern analysis. Furthermore, our approach leverages consistent hashing for data distribution, adjusting the number of virtual nodes according to each server's load level and the required degree of privacy, thereby significantly improving the overall efficiency of the system.

The remainder of this paper is organized as follows. Section 2 describes the related work, including blockchain, consistent hashing, and isolation forest. Section 3 presents the proposed system in detail. Section 4 provides the evaluation and discussion, and finally, Section 5 concludes the paper and discusses future research directions.

II. BACKGROUND AND RELATED WORK

A. Blockchain

Blockchain is a distributed ledger technology that ensures integrity and transparency by maintaining a consistent ledger across all nodes [2]. However, because every node is required to store and validate identical data, the network faces scalability limitations, causing the transaction processing throughput to decrease significantly as the network grows. To address this issue, recent research has focused on blockchain sharding systems [3]. Blockchain sharding is a technique that divides the network into multiple shards, with each shard independently processing transactions and maintaining its own state, thereby enabling parallel processing. This approach allows the overall network throughput to scale linearly while managing data on a regional basis, achieving both privacy protection and efficient data processing. In urban computing, huge amounts of data are generated across domains like transportation, environment, and energy, and are often organized based on geographical regions. This characteristic makes them highly compatible with the structure of a sharding system. This approach enables improved transaction processing performance, localized data management, and privacy protection all. In particular, for data sharing, data users must securely obtain information from trusted providers, while data providers need a clear mechanism to track when and by whom their data is accessed.

B. Consistent Hashing

Consistent hashing is a technique used in distributed systems to evenly and reliably distribute data [4]. In traditional hashing methods, when the number of nodes changes, most of the data must be redistributed, which creates a significant challenge. To address this issue, consistent hashing introduced a hash ring structure. After mapping all keys and nodes to the same hash space, each key is assigned to the closest node in the ring clockwise. Even when a new node is added or an existing node is removed, only a portion of the keys adjacent to that node need to be redistributed, thereby reducing the overall system load. With this characteristic, consistent hashing can provide scalability and availability. However, virtual nodes (VNs) are introduced to prevent uneven data distribution caused by data clustering on specific nodes. By mapping a single physical node to multiple positions on the ring, it prevents data from concentrating on specific nodes and enables a more balanced distribution of the load. However, fixing the number of VNs fails to account for differences in node performance and dynamic workload variations, potentially resulting in bottlenecks or inefficient use of system resources. To address these limitations, various research efforts have been continuously conducted.

C. Isolation Forest

Distinguishing between sensitive data and general data is crucial for protecting data privacy. Traditional methods for identifying sensitive data rely on keyword-based rules or predefined attribute lists. However, these mechanisms face challenges in quickly adapting to new types of sensitive data as they emerge and suffer from limited scalability. To address these issues, the use of unsupervised learning-based anomaly detection techniques is gaining attention. Isolation Forest (iForest), in particular, is being utilized in a variety of fields due to its high scalability and suitability for large-scale training data [5]. The iForest algorithm is based on the idea that anomalies can be more easily separated from normal data points in the feature space. It operates in three main steps: generating random trees, measuring path lengths, and calculating anomaly scores. The core formula is as follows:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \quad (1)$$

$$E[h(x)] = \frac{1}{t} \sum_{i=1}^t h_i(x) \quad (2)$$

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n} \quad (3)$$

Here, $E(h(x))$ represents the average path length of x , $c(n)$ is the normalization constant for a dataset of size n , and H_i denotes the i -th harmonic number. Since sensitive data differs in structure or occurs less frequently compared to general data, it tends to have higher anomaly scores in the iForest.

III. BLOCKCHAIN-BASED DATA MANAGEMENT SYSTEM

This section discusses the proposed system. Considering urban computing environments, our goal is to preserve data locality, protect each node's data privacy, and minimize the linkage between data and real-world identities. Table I presents the description of the symbols.

TABLE I
DESCRIPTION OF NOTATION.

Symbol	Description
S	Number of shards
N	Number of BS belonging to one shard
K	Number of DNs per round
M	Number of similarity-based datasets
V	Total number of VN nodes
P_i	i -th pseudonym
SCD	Similarity combined dataset
$SubData$	General sub data
α	Weighting constant for load value
β	Weighting constant for privacy value
τ	System threshold for data division
VN	Virtual node

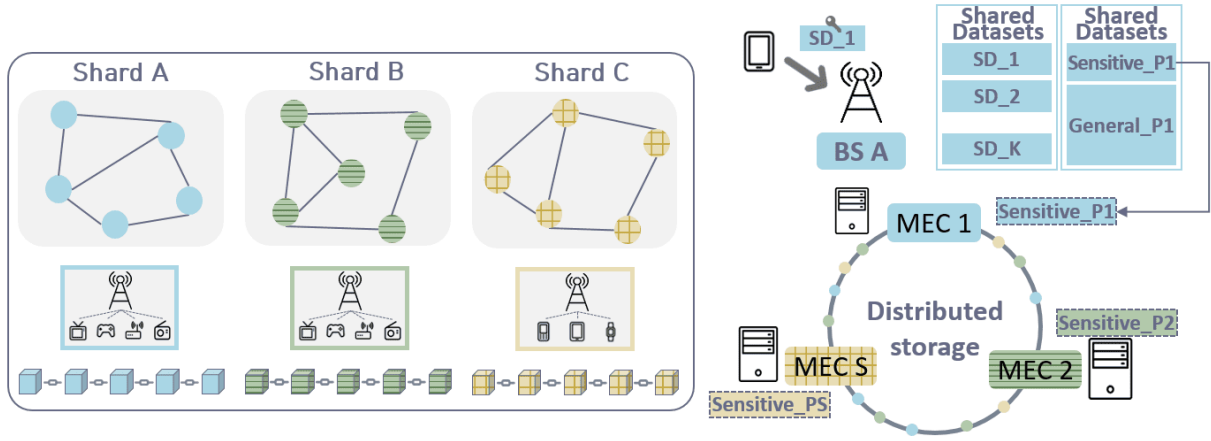


Fig. 1. The overall structure of the proposed system.

A. System overview

Figure 1 shows the overall architecture of the proposed system. The base stations (BSs) are divided into shards based on geographical conditions, and each shard's BSs form a blockchain. In this design, the blockchain records data provision and usage information as transactions to establish trust between data owners and providers. The proposed system comprises three main entities.

- **IoT device (Device node, DN):** DN collects and shares data in urban computing. It communicates with BS in each region and provides a portion of the collected data to the system.
- **Base station (BS):** BS communicates with DNs within the region and manages the shared data received from them. The BSs in the same region form a blockchain shard, where transactions record data provision and usage information, which are then combined into blocks.
- **Multi-access edge computing (MEC):** Each shard's MEC server manages the data within the shard and performs distributed storage across shards using consistent hashing, overcoming the limitations of centralized systems.

Within each geographically divided shard, there are multiple BSs, and each BS communicates with various DNs. Each DN collects sensing data in the urban computing and provides a portion of this data to the system as shared data. The BS generates transactions for the data provided by DNs and records them on the blockchain. It then applies privacy-preserving processing to the shared dataset and stores the processed data on the MEC server. A detailed description of each step is provided in the next section.

B. Data Management Scheme for Data Privacy

Figure 2 shows the data management process of the proposed system. The process consists of three stages: Prepare for shuffle, General dataset process, and Sensitive dataset process.

1) **Prepare for shuffle:** In this phase, each BS collects and classifies aggregated data from multiple DNs into sensitive data and general data. First, each BS collects data from DNs

to create a shared dataset. At this stage, a K-pseudonym mechanism is applied to protect the privacy of each DN. Specifically, after collecting data from K DNs, pseudonyms are used in place of the actual IDs of the DNs, and these pseudonyms are calculated using the following formula.

$$P_i = \text{Hash}(D_1, D_2, D_3, \dots, D_K) \quad (4)$$

P_i represents the i -th pseudonym, and D_k represents the k -th ID of a vehicle. By using the pseudonym P_i for the entire dataset, it becomes difficult to infer individual nodes, and the linkage to the actual nodes can be reduced. Each pseudonymized data entry is used to generate a Merkle tree and is stored as a transaction along with its proof. This proof can later serve as evidence of data ownership, enabling DN authentication and tracking. Finally, the pseudonymized data are classified into sensitive data and general data using equation 1 of the iForest.

$$\text{Label}(x) = \begin{cases} \text{Sensitive}, & s(x) \geq \tau \\ \text{General}, & s(x) < \tau \end{cases} \quad (5)$$

Referring to equation 5, if the value is greater than the system threshold τ , the data are classified as sensitive data. Otherwise, they are classified as general data. Each category is then used to construct its respective dataset.

2) **General dataset process:** The general dataset performs similarity-based data shuffling to protect data privacy [6]. This process aims to prevent attackers from inferring specific users and is therefore carried out among BSs within the same shard. Each BS randomly divides the general dataset into m parts and compares each part with data from other shards. Based on the similarity results, the data are shuffled to form m similarity-combined datasets. The formula used to verify similarity is presented below. Here, SCD refers to the similarity combined dataset, and $SubData$ refers to the general sub-dataset.

$$\text{sim}_n^{\max} = \max_{k \in [1, m]} (\text{sim}(SCD_n^{\text{init}}, \text{SubData}_k^n)) \quad (6)$$

$$SCD_n^{\text{new}} \leftarrow SCD_n^{\text{new}} \cup \text{sim}_n^{\max} \quad (7)$$

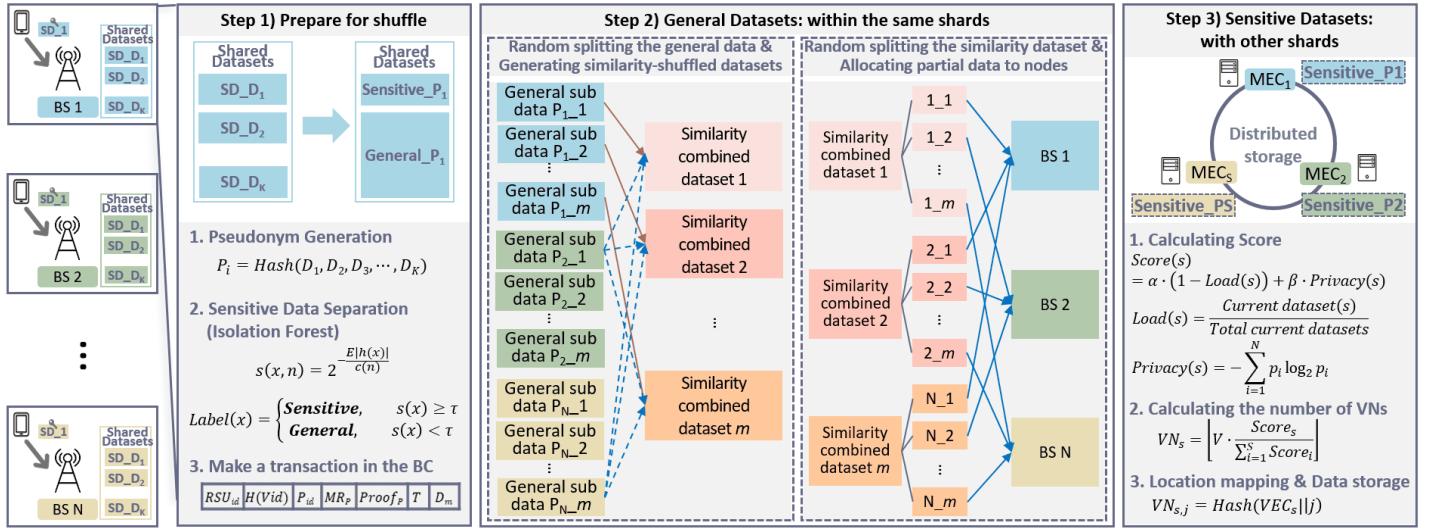


Fig. 2. The overall process of the proposed system.

Then, the generated similarity-combined datasets are randomly distributed to each BS. Data with high similarity are first gathered and then randomly divided, which helps preserve the original data characteristics. Additionally, shuffling data across regions increases the uncertainty in inferring specific nodes. Once the shuffling is completed, the data are cached in each BS and provided based on data requests. All processes are recorded as transactions. When data for the next round are generated, the current data are stored in the MEC, and the newly generated data are cached for the upcoming round.

3) Sensitive dataset process: Even if the data are encrypted during communication, there remains a risk of exposure in transit, which could lead to privacy breaches and security threats. In particular, for sensitive data, frequent communication can reveal the presence of the data and the user's behavior patterns, posing a serious threat. To address this issue, the proposed system minimizes communication and performs only cross-shard distributed storage to reduce the exposure of sensitive data patterns. Our system is designed with consideration for the sharding environment in urban computing. Accordingly, each shard manages its own data to ensure locality, while data replication is distributed across other shards to achieve stability and scalability. To address the limitations of traditional centralized systems, we utilize consistent hashing to distribute and store data. However, consistent hashing also has a limitation in that data can become concentrated on specific nodes. Therefore, we calculate the load factor of MEC servers, which serve as the nodes in the consistent hashing process, and compute the privacy diversity index. Based on these values, we determine the number of VN nodes according to the calculated ratio. The load factor of each server is calculated as shown in equation 8.

$$\text{Load}(s) = \frac{\text{Current dataset}(s)}{\text{Total current datasets}} \quad (8)$$

The goal of the proposed system is to efficiently distribute and store the overall data, with each node primarily managing the data belonging to its own shard. To achieve this, the data load factor is defined as the ratio of the amount of data stored by an individual node to the total data size. This ensures that all MEC servers maintain a balanced distribution of data in each round. For the calculation of the privacy diversity index, we utilize Shannon diversity, which is defined in equation 9.

$$\text{Privacy}(s) = - \sum_{i=1}^r p_i \log_2 p_i \quad (9)$$

A higher entropy value indicates that the data are evenly distributed across different types, with no particular type being overly concentrated. As a result, it becomes more difficult for attackers to infer individual users or detect patterns. Conversely, a low entropy value suggests that certain data types are dominant, which increases the likelihood of attackers deducing the characteristics of specific users or groups. Building on this concept, our system defines a privacy value to evaluate the level of privacy protection. A higher privacy value signifies that sensitive information is well safeguarded, as the data remain evenly distributed and more resilient to pattern analysis attacks.

$$\text{Score}(s) = \alpha \cdot (1 - \text{Load}(s)) + \beta \cdot \text{Privacy}(s) \quad (10)$$

Therefore, servers with higher privacy protection can be allocated more data for distributed storage. Equation 10 calculates the final score by considering both the load factor and the privacy value of each server. Here, α and β are weighting factors that can be adjusted by the system designer to choose between a performance-oriented or privacy-oriented policy.

$$VN_s = \left\lfloor V \cdot \frac{\text{Score}_s}{\sum_{i=1}^S \text{Score}_i} \right\rfloor \quad (11)$$

$$VN_{s,j} = \text{Hash}(VEC_s || j) \quad (12)$$

Equation 11 describes the process of determining the number of VNs assigned to each server based on the calculated Score value. Here, V means the total number of VN nodes, and VN_s means the number of VN nodes of shard s . A higher Score results in more VNs being allocated, leading to a greater amount of data being distributed to that server. This effectively assigns greater storage responsibility to servers with higher performance and privacy levels. Finally, equation 12 represents the process of mapping each VN's location based on a hash function. Here, $VN_{s,j}$ denotes the j -th VN of the MEC server in shard s . Through this approach, the system can achieve both scalability and stability.

IV. EVALUATION AND DISCUSSION

A. Experimental Setting

For our experiments, we used Intel Core i7-10700 64-bit CPUs @2.90 GHz and 64GB of RAM, NVIDIA GeForce RTX 2060 SUPER GPU on Windows 10. For the experiment, we compare Original, which uses the training data as is, with Proposal, which applies the proposed data processing method. A simple DNN model was used for the experiment, and TensorFlow 2.10 and TensorFlow-Privacy 0.8.3 were utilized. We used the United States accident dataset [7]. This dataset is a car accident dataset covering 49 states in the United States.

We conducted a membership inference attack (MIA) to evaluate the system's privacy protection performance. The evaluation metrics include the area under the ROC curve (AUC) and attacker advantage. The AUC represents the area under the receiver operating characteristic (ROC) curve, which reflects the classification performance of the trained model. Since our experiments are conducted from the attacker's perspective, a higher AUC indicates that the attacker is better at distinguishing certain data, meaning that the data are more vulnerable to attacks. Attacker advantage measures how effectively an attacker can perform an attack on the model. A higher value indicates that the attacker can more accurately identify samples belonging to the training dataset. It is calculated as the difference between the True Positive Rate (TPR) and the False Positive Rate (FPR).

B. Experimental Results

Figure 3 shows the maximum AUC from the dataset, representing the data points most vulnerable to the attacker, while Figure 4 illustrates the maximum attacker advantage. In Figure 3, random sampling in each experiment causes slight variations in the results. Nevertheless, the proposed system consistently maintains AUC values close to 0.5, even as the values of each entity increase. This indicates that when an attacker attempts to exploit the data, it becomes difficult to target specific users, demonstrating that privacy is effectively protected. Similarly, as shown in Figure 4, even when the values of all entities increase, the proposed mechanism confuses the attacker, thereby reducing the attacker's advantage compared to using the original data. Figures 3 and 4 present three experimental settings:

- (a): The number of DNs is fixed at 100 and the number of BSs is fixed at 50, while the number of shards is increased.
- (b): The number of shards is fixed at 30 and the number of DNs is fixed at 100, while the number of BSs is increased.
- (c): The number of shards is fixed at 30 and the number of BSs is fixed at 50, while the number of DNs is increased.

Table II shows the latency and migration rate of the proposed consistent hashing. In the centralized system (Central), the latency is low. However, the amount of data that must be migrated increases significantly whenever nodes are added or removed. In contrast, with consistent hashing, the latency is slightly higher, but the amount of data that needs to be migrated is greatly reduced. In particular, the proposed system assigns more VNs to servers with lower loads and stronger privacy protection, resulting in more stable performance compared to traditional consistent hashing.

C. Discussion

Recently, various approaches have been actively studied to protect data privacy, such as differential privacy (DP) and data generation using generative adversarial networks (GANs). However, in the case of DP, adding noise distorts the original data, resulting in a trade-off between model accuracy and privacy protection. For GANs, their high computational cost and long training time make them unsuitable for real-time data sharing systems. To address these limitations, we propose a data management mechanism for privacy protection that does not rely on additional security mechanisms. The proposed system enhances privacy protection by applying pseudonymization, shuffling, and distributed storage before the data are stored, while also ensuring storage efficiency. However, this approach requires additional processing time for data preprocessing compared to conventional data handling. Moreover, our system preserves locality in urban computing environments by shuffling and distributing data based on shards. Nevertheless, if the shard size or geographical scope becomes too large, locality may inevitably be lost, indicating the need for further research.

TABLE II
COMPARISON OF LATENCY AND MIGRATION RATE

Result	Number of nodes	Central	Consistent Hash	Proposal
Latency	20	0.1993	1.2502	0.9531
	40	0.2006	1.2520	0.9525
	60	0.2009	1.2551	0.9529
	80	0.2006	1.2484	0.9489
	100	0.2005	1.2543	0.9542
Migration rate	20	0.9565	0.0933	0.0870
	40	0.977	0.0437	0.0545
	60	0.9808	0.0423	0.0472
	80	0.9808	0.0423	0.0472
	100	0.9808	0.0423	0.0472

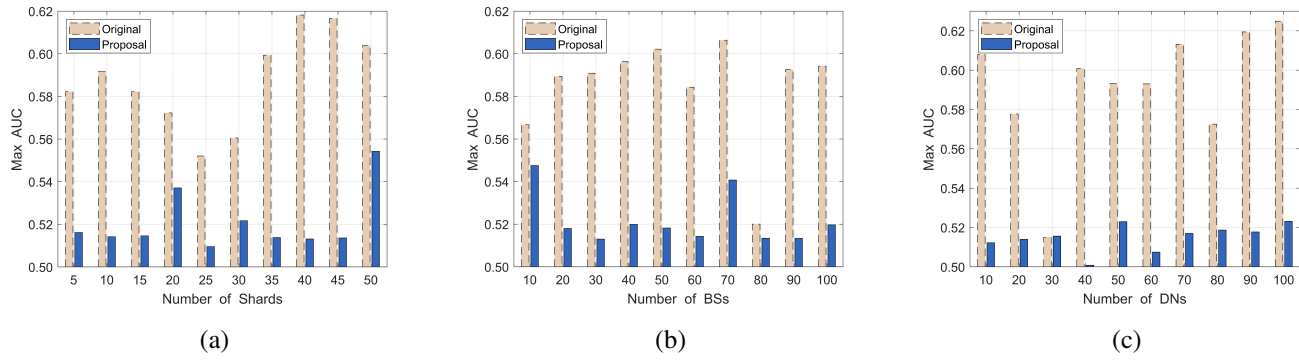


Fig. 3. Maximum AUC achieved by an attacker in membership inference attacks. Values close to 0 or 1 indicate that the attacker can distinguish members from non-members, whereas values close to 0.5 indicate no distinguishing ability. (a) Performance under different numbers of shards. (b) Performance under different numbers of base stations. (c) Performance under different numbers of device nodes.

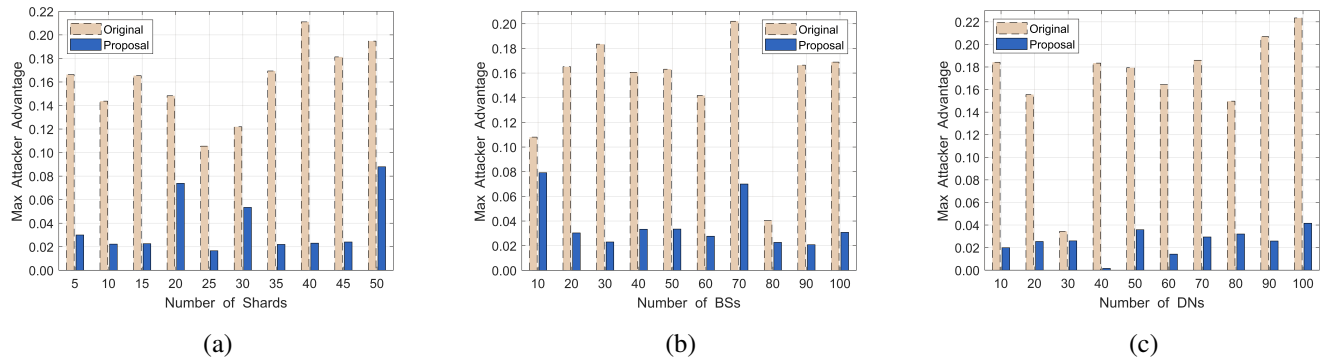


Fig. 4. Maximum attacker advantage in membership inference attacks. Values close to 0 indicate that the attacker gains no advantage in distinguishing members from non-members, whereas values close to 1 indicate perfect distinguishing ability. (a) Performance under different numbers of shards. (b) Performance under different numbers of base stations. (c) Performance under different numbers of device nodes.

V. CONCLUSION

As data sharing becomes increasingly important in urban computing, this study proposes a system that enables secure and efficient data management. To minimize the risk of exposing personally identifiable information from IoT nodes, the system adopts K-pseudonym-based identifiers. It also separates sensitive data from general data and processes them in distinct stages to protect node privacy. For sensitive data, score-based consistent hashing is employed to enhance traceability defense and improve data management efficiency. For general data, similarity-based data shuffling is applied to ensure locality while protecting privacy. While the proposed system can guarantee privacy protection without relying on additional security mechanisms, further analysis is required to address potential attack vectors. Therefore, future work will focus on investigating various attack scenarios that may arise during the data-sharing process and improving the system's robustness by incorporating the specific requirements of real-world data-sharing environments.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government,

Ministry of Science and ICT (MSIT) (RS-2023-NR076673). Inshil Doh is the corresponding author.

REFERENCES

- [1] Y. Zheng, "Urban computing: Enabling urban intelligence with big data," *Frontiers Comput. Sci.*, vol. 11, no. 1, pp. 1–3, Feb. 2017.
- [2] M. B. Farah, Y. Ahmed, H. Mahmoud, S. A. Shah, M. O. Al-Kadri, S. Taramonli, X. Bellekens, R. Abozariba, M. Idrissi, and A. Aneiba, "A survey on blockchain technology in the maritime industry: Challenges and future perspectives," *Future Gener. Comput. Syst.*, vol. 157, pp. 618–637, Aug. 2024.
- [3] X. Liu, H. Xie, Z. Yan, and X. Liang, "A survey on blockchain sharding," *ISA Trans.*, vol. 141, pp. 30–43, Oct. 2023.
- [4] C. Li, H. Pan, H. Qian, Y. Li, X. Si, K. Li, and B. Zhang, "Hierarchical sharding blockchain storage solution for edge computing," *Future Gener. Comput. Syst.*, vol. 161, pp. 162–173, Dec. 2024.
- [5] H. Xu, G. Pang, Y. Wang, and Y. Wang, "Deep isolation forest for anomaly detection," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 12, pp. 12591–12604, Dec. 2023.
- [6] H. Kim, G. Heo, and I. Doh, "Role-based federated learning exploiting IPFS for privacy enhancement in IoT environment," *Comput. Netw.*, vol. 263, pp. 111200, May 2025.
- [7] E. Tsukerman, *Machine Learning for Cybersecurity Cookbook: Over 80 Recipes on How to Implement Machine Learning Algorithms for Building Security Systems Using Python*, Packt Publishing Ltd, 2019.