Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2025 SCPE. Volume 26, Issues 3, pp. 1300–1307, DOI 10.12694/scpe.v26i3.4210

MODELING OF SECURITY AND PRIVACY ARCHITECTURE FOR PROTECTING DATABASES IN CLOUD COMPUTING INFRASTRUCTURE

XIAOHUI ZHANG [*], SONGKUN JIAO[†], JUNFENG WANG[‡], AND CUILEI YANG[§]

Abstract. In order to prevent data leakage and ensure the security of tenant's private data, and to enable tenants to have a precise understanding of the security level of their private data, the author proposes a modeling of the security and privacy architecture for protecting databases in cloud computing infrastructure. The author proposes a document database privacy protection architecture, which builds upon the existing architecture by adding a privacy protection layer between the application layer and the storage layer, forming a new service deployment architecture. Then, the author introduced the basic methods of privacy protection based on facial document databases. In order to adapt to the data structure system based on document storage for document databases, the author designed a basic method of privacy protection for document databases based on segmentation and obfuscation. By utilizing the free nature of document oriented database patterns, privacy protection data can be achieved through appropriate segmentation. For nested document structures, the author designed a document structure tree to retain document structure information. The results show that by comparing the experimental data of the 50w and 100w groups horizontally, it can be found that under the same cutoff score, as the number of database data increases, the time for data queries also increases accordingly, the query time of database A has increased by nearly 300ms compared to database B, and the additional time of the other groups is also roughly the same. By vertically comparing the experimental data of the 50w and 100w groups, it can be found that the query time of the C database is nearly 200ms longer than that of the A database, and as the sharding factor increases, the query time also increases accordingly, but the proportion of increase begins to slow down. By comparing data with the same segmentation factor but different data volumes, it can be seen that the impact of data volume on query time is positively correlated. This model can ensure that the database system is transparent to users at the view layer after privacy protection, and ensure the correctness and integrity of privacy data.

Key words: Software as a Service, Data privacy protection, Targeting document databases, Data partitioning

1. Introduction. In this era, major enterprises strive to obtain valuable information from vast amounts of data by collecting data, and data has become an important information asset [1]. Enterprises can utilize this data again to enhance market competitiveness, making its value no longer singular. Therefore, data has become a valuable asset, important economic input, and cornerstone of new business models for companies [2]. However, data has become a challenge of this era due to its massive, high-speed, and diverse characteristics [3]. The rapid development of cloud computing technology provides technical support for big data processing, providing users with computing and data storage services. Currently, there are three service models for cloud computing: Infrastructure as a Service, IaaS, users rent the infrastructure of cloud platforms, such as computer hardware resources, in order to obtain the desired services; Software as a-S service, SaaS users can rent software resources and access them from browsers, application programming interfaces, etc., users only need to configure their own application software; Platform as a Service, PaaS: Users can rent pre built software platforms in the cloud, such as operating systems, software development environments, etc. The popularity of intelligent mobile devices has led to a large number of mobile applications generating massive amounts of unstructured data [4]. Meanwhile, the increasing popularity of Web 2.0 applications such as Google and Facebook also requires the analysis and storage of large amounts of unstructured data. Ordinary relational databases are no longer able to handle these unstructured data, and database systems are undergoing a revolution. Moreover, these

[*]Department of Rail Transit, Shijiazhang Institure of Railway Technology, Shijiazhuang, Hebei, 050000, China. (XiaohuiZhang9@126.com)

[†]China nuclear power engineering Co., Ltd.Hebei branch, Shijiazhuang, Hebei, 050000, China. (SongkunJiao@163.com)

[‡]Department of Rail Transit, Shijiazhang Institute of Railway Technology, Shijiazhuang, Hebei, 050000, China. (JunfengWang6@126.com)

[§]Department of Rail Transit, Shijiazhang Institute of Railway Technology, Shijiazhuang, Hebei, 050000, China. (Corresponding author, CuileiYang9@163.com)

applications no longer only focus on issues such as consistency assurance that relational databases excel at, but also pay more attention to performance, scalability, and so on [5]. Traditional relational models use standardized SQL queries and access them through common interfaces such as JDBC and JDBC. Despite the functionality implemented by each relationship system supplier, There are slight changes in both SQL and system interfaces, but relational database systems are relatively interchangeable due to their widespread acceptance of standards. Faced with the challenges of the big data era, traditional relational databases exhibit poor scalability and slow retrieval and read/write speeds when querying simultaneously. Although the application of database distributed storage and horizontal and vertical segmentation technologies can alleviate the above-mentioned problems in processing massive data. But data migration is difficult, with high performance requirements and relatively high management costs [6].

2. Literature Review.

Encryption is a traditional and effective method of protecting data privacy. The architecture proposed by Aldawibi, O. O., and others is aimed at addressing most privacy issues in cloud computing. The main idea behind the proposed architecture is to segment data and store it on many clouds using third parties [7]. Zhou, Z. et al. proposed a practical data auditing scheme with retrievability and indistinguishable privacy protection functions, which can effectively audit the status of outsourced data. Improved reversible Bloom filter (IBF) to locally compress redundancy, which can retrieve corrupted data without prior context. In addition, an indistinguishable privacy protection model has been defined to capture the complete semantics of duplicate audit attacks and achieve indistinguishability in auditing. It has been proven that the proposed scheme is secure against adaptive message selection attacks and indistinguishable privacy protection against duplicate audit attacks [8]. Ahmadi, S. and others believe that privacy protection cloud computing is an emerging technology with many applications in various fields. The reason why cloud computing is important is because it has scalability, adaptability, and improved security. Similarly, privacy in cloud computing is also important as it ensures the integrity of data stored in the cloud remains unchanged [9]. The author referred to the traditional methods of data segmentation and data obfuscation in relational databases, and designed a basic privacy protection method for document databases based on value segmentation and key obfuscation, taking advantage of the free nature of face to face document database patterns. By utilizing the free nature of document oriented database patterns, privacy protection data is achieved by appropriately segmenting and obfuscating the segmented keys. At the same time, the author combines the privacy protection architecture proposed in Chapter 2 to provide a basic operational model for a protected database system. This model can ensure that the database system is transparent to users at the view layer after privacy protection, and ensure the correctness and integrity of privacy data.

3. Research Methods.

3.1. Privacy Protection System Architecture.

At present, the SaaS system uses a document database oriented architecture with a two-layer structure of application server and database server. The application and database interact directly, and the application can access the data in the database, Internal staff of SaaS service providers can also see the data stored by tenants in the database [10-11]. The data capture of tenants is easily obtained by internal employees of SaaS service providers, posing a data security risk. Therefore, the author's privacy protection architecture adds a privacy policy layer between the application layer and the data storage layer, as shown in Figure 3.1. The privacy protection architecture proposed by the author is divided into three layers, including three types of servers: Application Server, Privacy Policy Server, and Storage Server [12]. The application server is responsible for providing SaaS application services to tenants. The application server is a traditional SaaS application that only needs to submit data operation requests to the underlying layer, without worrying about whether tenant data needs to execute privacy protection policies or the execution methods of privacy protection policies; The privacy policy server is responsible for storing the privacy protection policies of tenants, verifying their identity after submitting operation requests, and implementing privacy protection for their data and operations based on their privacy protection policies. The privacy policy server includes an application interface module, identity authentication module, policy module, and conversion module. The storage server is responsible for storing tenant data and accepting data operation requests from the upper layer.

In this architecture, the deployment of privacy policy servers by SaaS service providers has no impact

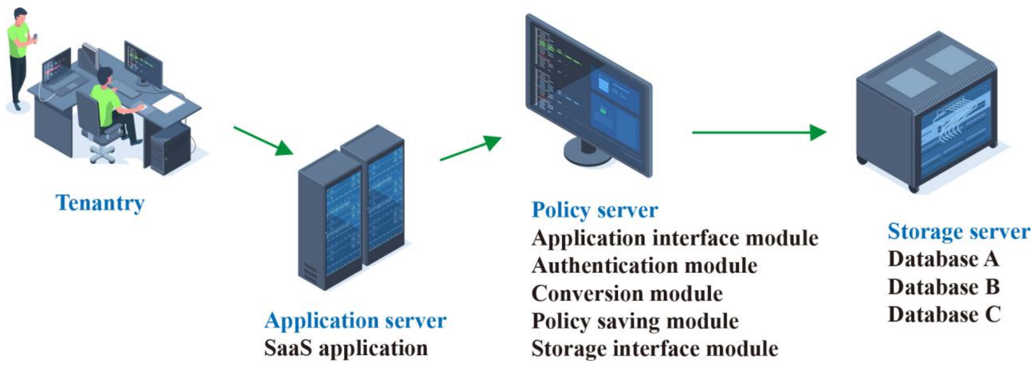

Fig. 3.1: Privacy Protection Architecture Diagram

on the execution of SaaS applications. When tenants need to manipulate data, they only need to send their identity information and operation requests to the SaaS application server, the SaaS application server executes application logic, sends tenant identity information and database operation statements to the next layer when data access is required, and waits for the return result. If a privacy policy server is deployed, the privacy policy server unlocks tenant policy information through tenant identity information and performs further operations [13]. If a privacy policy server is not deployed, the operation command will be directly transmitted to the storage server. This design solves the problem of tenant privacy protection policies and SaaS applications.

Privacy policy servers and storage servers are two independent server clusters. Tenant privacy policies are encrypted and stored in the privacy policy server. Even if SaaS service providers view tenant data in the storage server, they cannot obtain tenant privacy information because they cannot access the tenant privacy policies on the privacy policy server.

Furthermore, privacy protection servers can be provided as independent services, managed by specialized service providers. The service provider that provides privacy protection saves the tenant's privacy protection policy, but does not have tenant data. SaaS application service providers have tenant data, but cannot restore tenant data [14]. By isolating privacy policies from tenant data, we ensure the security of tenant privacy.

3.2. Basic methods of privacy protection. As is well known, the efficiency of querying encrypted data is low, so the author adopts a data segmentation method. Data segmentation is a method of protecting privacy data in relational databases, which effectively balances data read and write performance with data integrity [15]. All document data in document databases is composed of Key Value pairs, and the data of each Key Value pair is the minimum granularity for privacy data protection in document databases. If it can be guaranteed that attackers cannot obtain privacy information from Key Value pairs after obtaining data, then this privacy protection method is an available privacy protection method.

Definition 1. Value segmentation and key confusion Value segmentation is the process of segmenting the Value values of attributes that need to be protected in order to protect privacy data. $\{K : V\} \to \{K : [V_{r_1}, V_{r_2}, \cdots, V_{r_n}]\}$ While segmenting values, we will also segment each V_{r_i}; Renaming $K \to \{K_{r_1}, K_{r_2}, \cdots, K_{r_n}\}$ using the Key. In the process of renaming keys, we only ensure that each key can correspond to a fixed value, that is, there is a constraint $K_{r_i} \to V_{r_i}$ that makes $\{K : V\} \to \{\{K_{r_1} : V_{r_1}\}, \{K_{r_2} : V_{r_2}\}, \cdots, \{K_{r_n} : V_{r_n}\}\}$, while the order of keys is confidential. We refer to the renaming process as key obfuscation. The arrangement order of the confused $\{K_{r_1}, K_{r_2}, \cdots, K_{r_n}\}$ is the privacy protection strategy we obtained [16].

This segmentation divides the S SN attribute into three independent key value pairs, and the privacy protection policy after segmentation is $K_{SSN}[dd, ff, ss]$. By hiding the combination order of these three key value pairs, the purpose of protecting tenant privacy data is achieved.

Pattern freedom is a major feature of document oriented databases, and the author's approach is to utilize the feature of pattern freedom in document oriented databases to segment private data. Pattern freedom means that the length of certain attributes is not fixed [17]. We assume that the attributes to be protected for privacy are within a range, meaning that each attribute will have a minimum length and a maximum length. When formulating privacy policies, we determine the number of obfuscated keys based on the minimum length of attributes. And those big data with variable lengths are not within the author's scope of consideration.

When performing value sharding, it is necessary to determine the number of shards for each attribute. The size of the number of shards indirectly determines the degree of privacy protection and also affects the performance of the server in processing private data. The author names this number of shards as the sharding factor. The more partitions there are, the higher the level of privacy protection, but the greater the workload of reorganizing values. If the number of partitions is exactly equal to the length of the private data, it can be considered that a special encryption algorithm has been used for the data. The author determines the segmentation factor for each attribute based on the maximum length of the attribute, which ensures that each segmented attribute block has the same length. This method allows certain keys to have empty values and does not store keys with empty values.

Below is a segmentation algorithm for the basic methods of privacy protection in document oriented databases, Value segmentation can be divided into two types: fixed length segmentation and fixed number of blocks segmentation. Fixed length segmentation refers to the fixed length of the segmented blocks for a certain attribute. Fixed block segmentation refers to the fixed length segmentation used for a certain attribute, where the number of blocks generated is fixed. We use fixed length segmentation for privacy attributes, and the algorithm's parameter pl is the length of each block.

3.3. Nested Document Privacy Protection Methods. Nested document refers to treating the entire document as a key value in another document. Nested document structure is an important feature of document oriented databases, which allows data to be organized more naturally without having to be stored in a flat structure. We need to protect privacy data in object-oriented document databases, and we must fully consider how to preserve the structural information of nested documents while protecting privacy.

In order to protect the data in the embedded document, we must understand all the information about the document structure. Therefore, we generated a document structure tree, as shown in Figure 3.2. The root node of the tree is a data table for privacy protection, and the hierarchical relationship of the tree is established based on the nested relationship of documents. The author assumes that all documents in each data table contain the same structure [18]. Nodes containing value ultimately become leaf nodes, and each leaf node has a leaf node as its successor. For each leaf node, there are three attributes: Type, minimum length, and maximum length. In the diagram, type 0 represents numbers and type 1 represents characters. As shown in Figure 3, the age attribute is of numerical type, with a minimum length of 1 and a maximum length of 3. The maximum length of an attribute is used to determine the number of renamed keys in case of key obfuscation, while the minimum length of an attribute determines the maximum value of the splitting factor.

Through the document structure tree, the domain names of nested structure attributes can be obtained, which represent the structural information of attributes in the nested document. For example, if the domain name of the attribute phone is address.tel, we can obtain the full name of the attribute address.tel.phone. When we need to segment a certain attribute, the privacy policy records the full name of the attribute in order to preserve the nested information of the document in future transformations and combinations, and anonymize it in the document structure tree.

When segmenting a certain attribute in a document, first find the node of that attribute in the document structure tree, and determine the number of segmentation blocks based on the minimum length of the attribute. After the segmentation is completed, the obfuscated blocks will be stored under the root node, and the node names occupied by this attribute will be anonymized in the document structure tree, while the node attributes will be retained. For example, by segmenting the addres.zip attribute, the privacy policy after segmentation is set to $K_{address.zip}[z1, z3, z2]$.

The anonymization process of document structure tree is irreversible, and the confidence of document location after familiarizing oneself with anonymity will be saved in the privacy policy. If the application needs to view document structure information, the document structure tree can be reconstructed according to the privacy policy [19].

3.4. Privacy data operation methods. This section introduces the processing process of data operation requests on the policy server when tenants perform addition, deletion, modification, and query operations.

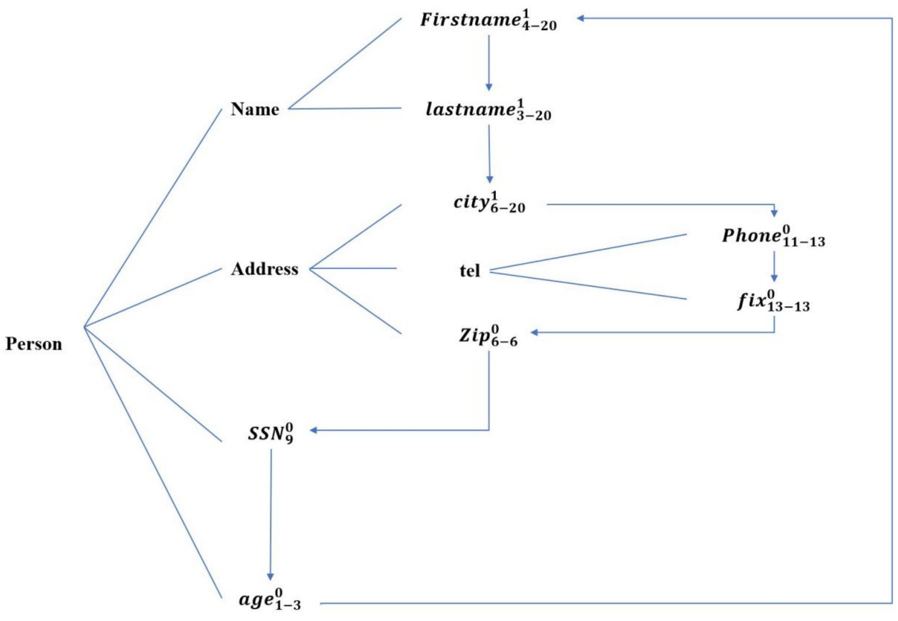

Fig. 3.2: Document Structure Tree

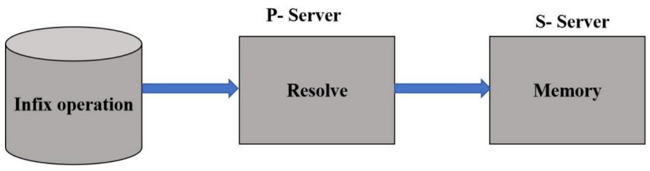

Fig. 3.3: Insertion Operation Process

When a tenant wants to store data, they submit a request to the policy server, which includes the tenant authentication information ID (T), tenant key PK, and storage operation request $op(\langle K_a, V_a >< K_\beta, V_\beta, V_\beta >< K_\theta, V_\theta\rangle)$. The policy server decrypts the request through the decryption function $D_{(PK.K)}$; Obtain the tenant's privacy policy $K_a[K_{\alpha r1}, K_{cr2}\cdots], K_p[K_{\rho r1}, K_{\rho r2}\cdots]$, then segment the tenant's data based on the tenant's privacy policy, pair the segmented data with the obfuscated Key to form $\langle K_{\alpha r1}, V_{r1} >< K_{\alpha r2}, V_{r2}\rangle \cdots, \langle K_{\beta r1}, V_{r1} >< K_{\beta r2}, V_{r2}\rangle \cdots, \langle K_\theta, V_\theta\rangle$, and finally forward the processed data to the storage server. After receiving the data sent by the policy server, the storage server stores the data (Figure 3.3).

When a tenant wants to search, update, and delete data, they submit their authentication information ID (T), tenant key PK, and data operation request $op(K_a = V_a, K_\beta = V_\beta)$ to the policy server. After decrypting the privacy policy, the policy server converts the request into $op(K_{\alpha r1} = V_{r1}\&K_{\alpha r2} = V_{r2}\&\cdots, K_{\beta r1} = V_{r1}\&K_{\beta r2} = V_{r2}\&\cdots)$ based on the privacy protection policy $K_a[K_{\alpha r1}, K_{cr2}\cdots], K_p[K_{\rho r1}, K_{\rho r2}\cdots]$. Then send the converted request to the storage server while waiting for the return information from the storage server [20]. After receiving a tenant request, the storage server processes it based on the tenant request. If it is a query request, the query result is returned to the policy server. The policy server recombines the query results based on the privacy policy and forwards them to the tenant.

When a tenant wants to update or delete data, they submit their authentication information ID (T), tenant key PK, and data operation request $op(K_a = V_a, K_\beta = V_\beta)$ to the policy server. After decrypting the privacy policy, the policy server converts the request into $op(K_{\alpha r1} = V_{r1}\&K_{\alpha r2} = V_{r2}\&\cdots, K_{\beta r1} = V_{r1}\&K_{\beta r2} = V_{r2}\&\cdots)$ based on the privacy protection policy $K_a[K_{\alpha r1}, K_{cr2}\cdots], K_p[K_{\rho r1}, K_{\rho r2}\cdots]$, then send

Table 4.1: Experimental Configuration Table

database	Data volume (10000)	Splitting factor
A	25	0
B	50	0
C	100	0
D	25	4
E	50	4
F	100	4
G	25	8
H	50	8
I	100	8

the converted request to the storage server while waiting for the return information from the storage server. After receiving a tenant request, the storage server processes it according to the tenant request. If it is a deletion operation, the storage server returns confirmation information to the policy server, which forwards the confirmation information to the tenant; If it is an update operation, return the updated data or confirmation information based on the tenant configuration.

4. Result analysis. The author designed a basic method for privacy protection in document databases based on val ue segmentation and key obfuscation. In order to maintain the nested document feature of document oriented databases, the author designed a document structure number [21]. By utilizing the free nature of document oriented database patterns, privacy protection data is achieved by appropriately segmenting and obfuscating the segmented keys. The author's experiment used a PC as the database server, with the server configuration as follows:

CPU: 4 cores Intel(R) Core(R) i5-2400 3.IOGHz
Memory: 2GB
Hard disk: 500GB
Operating System: Red Hat Enterprise Linux&. 2 32bit
Database system: MongoDB 2.4.9

The author's experiment used a power system electricity meter as the data mode. Relevant datasets were generated through simulation. The sharding factor is the number of sharding blocks for tenant privacy data. This experiment will test the impact of the size of the sharding factor on database performance and provide reference for determining the sharding factor. The experiment will establish 6 databases, and the data volume of each database is shown in Table 1. The same amount of data stored in the database for this experiment is the same, that is, database A, D. G stores the same copy of data, database B, E. H stores the same copy of data, database C, F. I stores identical copies of data. The D-I database will segment the data date attribute, and the segmentation factors are shown in Table 4.1.

The experiment will perform the same query operation on 6 databases, query the data in the data date column, and check the query time. The column has not been indexed, and the data queried for the same data copy in this experiment is the same, data date The experimental results are shown in Figure 4.1. This experiment has made every effort to eliminate the bias caused by database caching during the experimental process [22,23].

By comparing the experimental data of the 50w and 100w groups horizontally, it can be found that under the same cutoff score, as the number of database data increases, the time for data queries also increases accordingly, the query time of database A has increased by nearly 300ms compared to database B, and the additional time of other groups is also roughly the same [24]. By vertically comparing the experimental data of the 50w and 100w groups, it can be found that the query time of the C database is nearly 200ms longer than that of the A database, and as the sharding factor increases, the query time also increases accordingly, but the proportion of increase begins to slow down. By comparing data with the same segmentation factor but different data

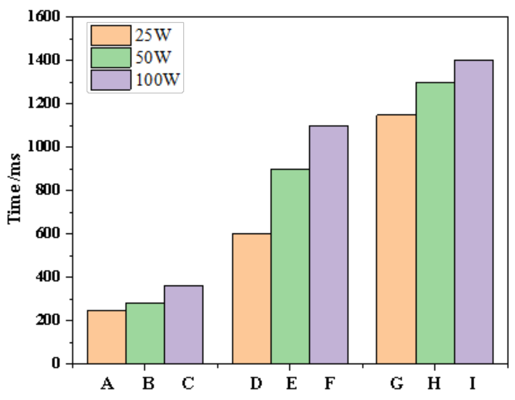

Fig. 4.1: The impact of segmentation factors on efficiency

volumes, it can be seen that the impact of data volume on query time is positively correlated [25]. Through this experiment, it can be seen that the size of the sharding factor has an impact on the search performance of a data. The larger the sharding factor, the longer the additional time required for data queries. Therefore, when setting the sharding factor, it is necessary to fully consider the performance requirements of tenants. At the same time, we also found that although data partitioning affects the query speed of data, this impact is not as significant as increasing the size of the database. The impact of data partitioning on the query performance of the database does not exceed 30%. This is still done under the premise of eliminating the impact of database performance optimization through memory. If the database is allowed to perform query optimization, the impact of this partitioning will be even lower. Therefore, the experiment in this section demonstrates the feasibility of using the privacy protection method proposed by the author in document oriented databases.

5. Conclusion. In order to adapt to the storage characteristics of document oriented databases, the author designed a basic privacy protection method for document oriented databases based on value segmentation and key obfuscation. In order to maintain the feature of nested documents in document oriented databases, the author designed a document structure tree. By utilizing the free nature of document oriented database patterns, privacy protection data is achieved by appropriately segmenting and obfuscating the segmented keys. In order to ensure the security of tenant privacy policies, the author provides a basic operation model for a protected database system. This model can ensure that the database system is transparent to users at the view layer after privacy protection, and ensure the correctness and integrity of privacy data.

REFERENCES

[1] Abba Ari, A. A., Ngangmo, O. K., Titouna, C., Thiare, O., Mohamadou, A., & Gueroui, A. M. (2024). Enabling privacy and security in Cloud of Things: Architecture, applications, security & privacy challenges. Applied Computing and Informatics, 20(1/2), 119-141.

[2] Saini, D. K., Kumar, K., & Gupta, P. (2022). Security issues in IoT and cloud computing service models with suggested solutions. Security and Communication Networks, 2022, 1-9.

[3] Bibal Benifa, J. V., & Venifa Mini, G. (2020). Privacy based data publishing model for cloud computing environment. Wireless Personal Communications, 113, 2215-2241.

[4] El Kafhali, S., El Mir, I., & Hanini, M. (2022). Security threats, defense mechanisms, challenges, and future directions in cloud computing. Archives of Computational Methods in Engineering, 29(1), 223-246.

[5] Sheikh, M. S., Liang, J., & Wang, W. (2020). Security and privacy in vehicular ad hoc network and vehicle cloud computing: a survey. Wireless Communications and Mobile Computing, 2020, 1-25.

[6] Gill, S. H., Razzaq, M. A., Ahmad, M., Almansour, F. M., Haq, I. U., Jhanjhi, N. Z., ... & Masud, M. (2022). Security

and privacy aspects of cloud computing: a smart campus case study. Intelligent Automation & Soft Computing, 31(1), 117-128.

[7] Aldawibi, O. O., Sharf, M. A., & Obaid, M. M. (2022). Cloud computing privacy: concept, issues and solutions. 2022 IEEE Symposium on Industrial Electronics & Applications (ISIEA), 33(7), 810-819.

[8] Zhou, Z., Luo, X., Wang, Y., Mao, J., Luo, F., & Bai, Y., et al. (2023). A practical data audit scheme with retrievability and indistinguishable privacy-preserving for vehicular cloud computing. IEEE Transactions on Vehicular Technology, 117(1), 109-127.

[9] Ahmadi, S., & Salehfar, M. (2022). Privacy-preserving cloud computing: ecosystem, life cycle, layered architecture and future roadmap, 34(6), 3121-3135.

[10] Lee, C., & Ahmed, G. (2021). Improving IoT privacy, data protection and security concerns. International Journal of Technology, Innovation and Management (IJTIM), 1(1), 18-33.

[11] Boerman, S. C., Kruikemeier, S., & Zuiderveen Borgesius, F. J. (2021). Exploring motivations for online privacy protection behavior: Insights from panel data. Communication Research, 48(7), 953-977.

[12] Wu, Y., Song, L., & Liu, L. (2021). The new method of sensor data privacy protection for IoT. Shock and Vibration, 2021, 1-11.

[13] Ren, Y., Liu, W., Liu, A., Wang, T., & Li, A. (2022). A privacy-protected intelligent crowdsourcing application of IoT based on the reinforcement learning. Future generation computer systems, 127, 56-69.

[14] Humayun, M., Jhanjhi, N. Z., Alruwaili, M., Amalathas, S. S., Balasubramanian, V., & Selvaraj, B. (2020). Privacy protection and energy optimization for 5G-aided industrial Internet of Things. IEEE Access, 8, 183665-183677.

[15] Zhang, Q., Li, Y., Wang, R., Liu, L., Tan, Y. A., & Hu, J. (2021). Data security sharing model based on privacy protection for blockchain-enabled industrial Internet of Things. International Journal of Intelligent Systems, 36(1), 94-111.

[16] Akter, M., Moustafa, N., Lynar, T., & Razzak, I. (2022). Edge intelligence: Federated learning-based privacy protection framework for smart healthcare systems. IEEE Journal of Biomedical and Health Informatics, 26(12), 5805-5816.

[17] Sun, Z., Wang, Y., Cai, Z., Liu, T., Tong, X., & Jiang, N. (2021). A two-stage privacy protection mechanism based on blockchain in mobile crowdsourcing. International Journal of Intelligent Systems, 36(5), 2058-2080.

[18] Li, Q., Wen, Z., Wu, Z., Hu, S., Wang, N., Li, Y., ... & He, B. (2021). A survey on federated learning systems: Vision, hype and reality for data privacy and protection. IEEE Transactions on Knowledge and Data Engineering, 35(4), 3347-3366.

[19] Diaz-Ordoñez, M., Rodríguez Baena, D. S., & Yun-Casalilla, B. (2023). A new approach for the construction of historical databases—NoSQL Document-oriented databases: the example of AtlantoCracies. Digital Scholarship in the Humanities, 38(3), 1014-1032.

[20] Istiqamah, A. N., & Wiharja, K. R. S. (2021). A schema extraction of document-oriented database for data warehouse. International Journal on Information and Communication Technology (IJoICT), 7(2), 36-47.

[21] Maicha, M. E., Ouinten, Y., & Ziani, B. (2022). UML4NoSQL: A Novel Approach for Modeling NoSQL Document-Oriented Databases Based on UML. Computing and Informatics, 41(3), 813-833.

[22] Messaoud, I. B., Alshdadi, A. A., & Feki, J. (2021). Building a document-oriented warehouse using NoSQL. International Journal of Operations Research and Information Systems (IJORIS), 12(2), 33-54.

[23] Gusarenko, A. S. (2022). Situation-oriented databases: processing heterogeneous documents of microservices in a document-based storage. Modelirovanie, Optimizatsiya i Informatsionnye Tekhnologii= Modeling, Optimization and Information Technology, 10(4), 1-16.

[24] Davardoost, F., Sangar, A. B., & Majidzadeh, K. (2020). Extracting OLAP cubes from document-oriented NoSQL database based on parallel similarity algorithms. Canadian Journal of Electrical and Computer Engineering, 43(2), 111-118.

[25] Maté, A., Peral, J., Trujillo, J., Blanco, C., García-Saiz, D., & Fernández-Medina, E. (2021). Improving security in NoSQL document databases through model-driven modernization. Knowledge and Information Systems, 63, 2209-2230.

Edited by: Bradha Madhavan
Special issue on: High-performance Computing Algorithms for Material Sciences
Received: Jun 28, 2024
Accepted: Aug 7, 2024