

RESEARCH

Open Access



Big data security access control algorithm based on memory index acceleration in WSNs

Jianhua Peng^{1,2*} , Hui Zhou¹, Qingjie Meng¹ and Jingli Yang¹

* Correspondence: pengjh@niit.edu.cn

¹College of Computer and Software, Nanjing Institute of Industry Technology, Nanjing 210046, People's Republic of China

²Nanjing Shendi Intelligent Construction Technology Research Institute, Nanjing 210019, People's Republic of China

Abstract

The access control is used to ensure these data security when WSN (wireless sensor network) with a large number of base stations transmits huge amount of data to a data center server. Meanwhile big data systems are used to efficiently store, manage, and use data from large-scale WSNs. In big data systems for WSNs, the traditional access control technology will greatly affect the system performance. This paper first analyzes the data processing flow of the traditional access control strategy in big data systems, analyzes its time complexity, and explores how it affects system performance. Then, we propose the big data security access control algorithm based on memory index acceleration in WSNs which has better performance over the traditional ones. In our experiments, under the same test environment and security strategy, the performance has been greatly improved with the proposed algorithm.

Keywords: Wireless sensor networks, Security, Big data, Access control, Memory index

1 Introduction

A wireless sensor network (WSN) is an autonomous wireless communication system composed of a large number of micro-sensor nodes with limited computing capacity, storage capacity, and communication capability [1]. The advancement of mass data collection technology in wireless sensor networks has led to the emergence of a large number of wireless sensor applications. It is becoming increasingly important to ensure the data security of big data systems for WSNs [2]. The security is the cornerstone of the big data systems. A data breach could result in serious harm to any of the individuals to whom the information relates. Access control is an effective method to ensure data security. It is based on authentication and authorization and is the most widely used strategy for data security prevention and protection in big data systems. It can restrict access to key resources and prevent intrusion of illegal users or inadvertent operation of legitimate users [3]. The focus of access control is on authorization. In a distributed system, nodes need to be coordinated and the access rights are synchronized between nodes. After various security technologies applied to distributed

systems, each technology faces its own security challenges. The distributed systems must not only implement access control policies for data leaving each collaboration system, but must also control access to local resources. Depending on the sensitivity of the data, it needs to ensure that distributed applications on other coordination systems have access to the data they are processing. With all these factors considered, the implementation of the access control functions for distributed systems is very complicated [4].

The access control models can be divided into 5 categories: the discretionary access control (DAC) [5], the mandatory access control (MAC) [6], the role-based access control (RBAC) [7–9], the attribute-based access control (ABAC) [10], the policy-based access control (PBAC) [11, 12]. Various data security algorithms are used to improve the security of the access control model. These algorithms severely degrade the performance of big data systems. There are three ways to improve the performance of access control in big data systems. One is to improve the performance of the algorithm itself, the other is to improve the big data security model, and the third is to study the methods that affect the performance of big data systems after the algorithm is applied to the model.

It is the focus of access control research from rights rules and control policy to get rights. Due to the large amount of data and the large number of access users, the wireless sensor big data system still has major shortcomings in the way of getting and controlling access control rights. Separation of duty (SOD) is used for enforcing least privilege concept in access control model [13]. In the RBAC model, rights are assigned to roles, and roles are associated with user to form rights relationships in the access control model [14]. In the ABAC model, each attribute has a set of attribute and value definitions. The defined relationship is combined with the user to form the rights relationship of the access control model [15]. In the access control system, these rights relationships are stored in metadata files. The system gets the user's rights to access resources by parsing the metadata. The parsing process will affect the performance of access control. Binary tree method [16, 17] can reduce the parsing time. Moreover, we can also improve the performance of getting rights by including flexible authentication based on user context information in the attributes [18]. Big data systems for wireless sensor networks have lot of metadata because of the large amount of data and users. Therefore, it takes long time for access control to get rights from the metadata and judge rights. To improve its performance, the current main solution is to parameterize user function attributes to reduce the number of rights policies [19]. The method will restrict the design and application scope of access control policies. The Hadoop ecosystem provides a complete solution for big data. Therefore, it can provide all round functions such as data storage, data processing, data analysis, and data security for WSNs in the context of big data [20]. Apache Ranger is a data security management framework for the Hadoop ecosystem. It performs unified data authorization, management, and auditing for the Hadoop ecosystem. Various algorithms and methods for big data security can be easily applied to the Hadoop ecosystem through it [21]. During the development of the Apache Ranger project, we found that after enabling access control, the performance of big data systems will be greatly reduced.

The binary tree is the main method to improve the performance of access control. The binary search tree has no performance advantage when it is a single branch tree. It also takes a relatively long time when the binary tree is used to obtain permissions from

a large amount of metadata in big data systems. It can improve the performance of permission acquisition in the big data access control process by parameterizing user function attributes to reduce the number of permission policies [22]. However, this method will limit the design and application scope of access control policies. In top-level big data security engineering applications such as Apache Ranger, access control storage structured policy data. The system first loads the access control policy into memory during the access control process. The system queries the policies from the memory, obtains permissions, and judges permissions in a conditional loop when user accesses data. So this method has serious performance problems. The method which uses L2 cache to build indexes can solve current problems and significantly improve the performance of access control for big data systems.

There are three main contributions of this paper. The first contribution is to build a second-level cache to reduce the number of cycles during policy extraction. The second contribution is to build a memory index to shorten the access time of the security policy. The third contribution is to update the second-level cache and index content to ensure the effectiveness of policy changes. In this way, the access control authority efficiency will be greatly improved without affecting the security of access control.

The rest of the papers are organized as follows. Section 2 discusses the related work of the access control methods in big data system. Section 3 theoretically analyses the cause of performance degradation and proposes the big data security access control algorithm based on memory index acceleration in WSNs. Section 4 performs experimental verification and results analysis. And section 5 concludes the paper with summary.

2 Related work

2.1 Data security for WSNs

Data fusion can effectively reduce the data transmission volume and network energy consumption in wireless sensor networks (WSNs). At the same time, it can be used to improve the security of wireless sensor network data. An intelligent data fusion algorithm was proposed in [23] for wireless sensor network based on hybrid delayed perception clustering. This algorithm combines the advantages of single-layer cluster structure and multi-layer cluster structure. It effectively improves the security of data transmission while reduces network delay and network energy consumption. Haomeng Xie [24] classifies various wireless sensor networks attack detection methods based on the protocol stack layer, explains the advantages and disadvantages of those methods, and measures the security of wireless sensor networks. A game theory-based DSA algorithm was proposed in [25] to implement spectrum leasing and interference mitigation between SUs in network channels which reduces data loss in wireless sensor network data transmission and improves data security confidence. In order to improve the security of data transmission in wireless sensor networks, a decision transmission scheme was proposed in [26] to enhance collaborative spectrum sensing in industrial IoT and established a cooperative spectrum sensing mathematical model based on decision transmission, which was added packet error and packet loss factors.

With the further development of wireless communication and sensor technology, large-scale wireless sensor networks are being applied in more and more industries. The Internet of Vehicles is a typical large-scale wireless sensor network. It has the

characteristics of large data volume and high data security requirements. In order to improve the data security of large-scale wireless sensor networks, Jiliang Li [27] designed the CL-CPPA protocol which can effectively protect the data security of large-scale wireless sensor networks. Wearable devices are also a typical large-scale wireless sensor network. Hong Liu [4] designed a collaborative privacy protection scheme for wearable devices. This solution has authentication and data access control considerations in the context of space awareness and time awareness. The experimental results prove that this scheme can better protect the security of the sensor's big data.

2.2 Data security for big data system

In big data systems, storing cipher text in the cloud is one of the safest ways to store and access big data. However, verifying the user's access legitimacy and securely updating the cipher text in the cloud based on the access policy specified by the data owner are two key challenges for making cloud-based big data storage effective. Traditional approaches either completely ignore the issue of access policy updates or delegate updates to third-party agencies. Access policies update is vital to enhance security and handle the dynamics caused by user joins and leaves. Based on this, Chunqiang Hu [28] proposed a secure and verifiable access control scheme based on NTRU (Number Theory Research Unit) cryptosystem for big data storage in the cloud. This solution allows the cloud server to effectively update cipher text when the data owner specifies a new access policy. Data owner can verify the update to resist the spoofing behavior of the cloud. The test results show that the big data system using the solution can effectively prevent users from cheating and has ability to resist various attacks.

In big data systems, unstructured and semi-structured data account for the vast majority. The complicated data types make the traditional authorization mode difficult to meet the minimum authorization principle. In big data systems, fine-grained access control can meet the minimum authorization principle. With the rapid development of big data systems, the fine-grained access control model has opened up a new wave of access control research in the field of big data. After a large amount of research on the canonical system, Julian A. Padgett [29] proposed a first-order logic based on deontic to represent and infer data access strategies. Shangping Wang [17] used binary tree technology to deal with attribute revocation and grant, proposed an effective RABE (revocable and grantable attribute-based encryption) scheme. Under the assumption of error hypothesis, the security of the scheme has selective security features in the standard model.

The value of the access control can only be maximized when it is applied to big data systems. Yuqing Mo [30] discussed the security requirements of big data and extracted key technologies for big data security from authentication, authorization, access control, data hiding and encryption, network security, and system security. Based on the key technology of extraction, he designed a security management system for the Hadoop platform. Apache Hadoop is an important framework for fault-tolerant distributed big data storage and processing. The Hadoop core platform and other open source tools such as Apache Hive, Storm, and HBase provide an ecosystem that enables users to take full advantage of the potential of big data. Apache Ranger and Apache Sentry provide centralized policy management and implementation through plugins, providing fine-grained access control for components of this ecosystem [31, 32].

The access control involves various algorithms and models. In a traditional system, the impact of access control technology on system performance is not obvious. However, in big data systems, the impact is very serious because massive data is stored and processed in different nodes. In the application of access control to ensure the security of big data, especially the fine-grained access control, the specific implementation method of access control technology has a decisive impact on the performance. Through the verification of HBase, the Apache big data top-level project, the fine-grained big data access control method implemented by Apache Ranger seriously affects the performance of HBase. Based on the analysis of the impact of access control on the performance of HBase, this paper improves the implementation algorithm of Apache Ranger. The improvement algorithm significantly improved the performance of big data system.

3 Proposed method

3.1 Access control analysis in WSNs big data system

The purpose of access control is to ensure the security of data. In the WSNs big data system, the access control will perform the access control operation according to the access control policy, access control information, access control decision, and access control implementation information to obtain the access right to the access data when the visitor accesses the data. As shown in Fig. 1, the execution of access control is a complex process. It is executed in the data storage, access, and processing stages in the WSNs big data system.

The data storage structure is the basic concern of access control. It is determined by the characteristics of the data of WSNs. The resources in big data system can be determined by:

$$r = dmr(datades) \quad (1)$$

where r is the resource and $datades$ is the characteristics of the data of WSNs.

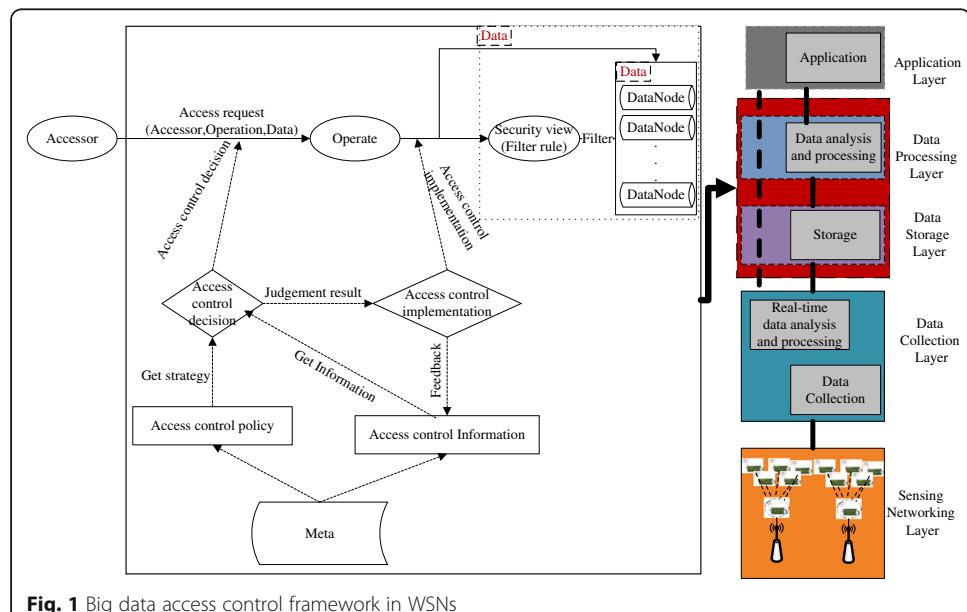


Fig. 1 Big data access control framework in WSNs

Big data system gets rights policies by loading rights resources (metadata) into memory. Its formula is as follows:

$$arp = f(r) \quad (2)$$

where r is the resource and arp is the rights policy for access control.

The system can get the rights policy of the user through the user and policy, the formula is as follows:

$$uap = getUserPolicy(u, arp) \quad (3)$$

where uap is the all rights policy of user for access control.

The accessed object, user, and the rights policy to which the user belongs determine the access control policy when a user accesses the WSNs big data system. The formula is as follows:

$$ucp = getCurAP(ao, u, uap) \quad (4)$$

where ao is the accessed object, u is the access user, uap is the all rights policy of user for access control, and ucp is the access related rights policy.

When a user accesses the WSNs big data system, he will be granted to access if the accessed object is in the access rights policy. The logic can be expressed by:

$$up = getAP(ao, ucp) \quad (5)$$

where up is the rights.

The process for users to get rights is as follows:

$$\begin{aligned} up &= getAP(ao, ucp) \\ &=> upower = getAP(ao, getCurAP(ao, u, uap)) \\ &=> upower = getAP(ao, getCurAP(ao, u, getUserPolicy(u, arp))) \\ &=> upower = getAP(ao, getCurAP(ao, user, getUserPolicy(u, f(r)))) \\ &=> upower = getAP(ao, getCurAP(ao, u, getUserPolicy(u, f(dmp(datapro))))) \end{aligned} \quad (6)$$

3.2 Problem analysis

As shown in Fig. 1, the access control for big data systems is very complex. So a unified big data security framework is needed to ensure the realization of big data access control. Apache Ranger is a data security management framework for the Hadoop ecosystem. It performs unified data authorization, management, and auditing for the Hadoop ecosystem. Various algorithms and methods for big data security can be easily applied to the Hadoop ecosystem through it [18–21]. During the development of the Apache Ranger project, we found that after enabling big data security functions in big data systems, the performance of big data systems will be greatly reduced, which will have a great impact on the application of big data systems.

Apache Ranger uses a policy-based model to express the “User-Resource-Rights” logic. It is also a fine-grained security control model that effectively manages data security on the Hadoop platform. The “User-Resource-Rights” logic has the following characteristics:

- A user is an object that accesses a resource and belongs to a group or role.

- Different components correspond to different business resources.
- *AllowACL* and *DenyACL* expressed rights. The *AllowACL* is an access control list that describes the conditions that allowed to access. The *DenyACL* is a negative access control list that describes the case of denying access.

Definition 1: Access control item (ACI): A collection of visitors (users, user groups, roles) and access types, represented by *AccessItem*.

$$AccessItem = List < User/Group/Role > + List < AccessType > \quad (7)$$

where *User/Group/Role* is the object which accesses the big data resource; *AccessType* is the type of access object requires, i.e. access rights.

Definition 2: Allow access control item (ACCI): A control that allows access to big data resources, represented by *allow*.

Definition 3: Exceptions allow access control items (EAACI): Allow access to exception access items in control items, represented by *allowException*.

Definition 4: Allow access control list (AACL): Allows the data set of the access control item set plus the exception allowed access control set, represented by *AllowACL*.

$$AllowACL = List < AccessItem > allow + List < AccessItem > allowException \quad (8)$$

Definition 5: Negative controls item (NCI): Control that do not allow access to big data resources, expressed in *deny*.

Definition 6: Abnormal negative control item (ANCI): Exception accesses in access control entries did not allow, expressed as *denyException*.

Definition 7: Negative control list (NCL): A data set in the set of negative access control items that excludes the set of abnormal negative control items, represented by *DenyACL*.

$$DenyACL = List < AccessItem > deny + List < AccessItem > denyException \quad (9)$$

Definition 8: Access control policy (ACP): A logical rule for accessing resources by a large data resource access object, consisting of a resource, an allowable control list, and a negative control list, represented by a *Policy*.

$$Policy = List < Resource > + AllowACL + DenyACL \quad (10)$$

Definition 9: Access control service (ACS): A collection of access control policies, represented by a service.

Figure 2 shows the current policy-based access control process.

As can be seen from Fig. 2, when a user AU requests access to resources Ares, Apache Ranger obtains all the rights policies of the user related to requested resources. The access right of AU will be determined by the following logic:

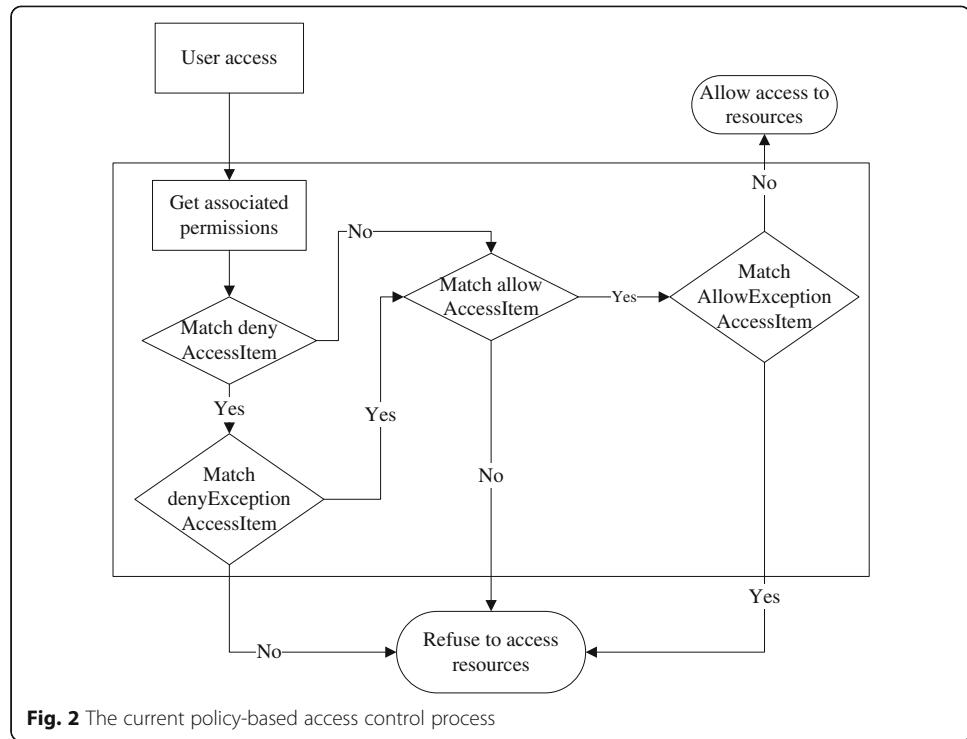
If the ARes is in the NACI and not in ANAC, AU cannot access it.

If the ARes is not in ACCI, AU cannot access it when it is in NACI and ANAC.

If the ARes is not in EAACI, AU cannot access it when it is in NACI, ANAC, and ACCI.

If the ARes is in EAACI, AU can access it when it is in NACI, ANAC, and ACCI.

If the ARes is not in ACCI, AU cannot access it when it is not in NACI.



If the ARes is not in EAACI, AU cannot access it when it is not in NACI but in ACCI.

If the ARes is in EAACI, AU can access it when it is not in NACI but in ACCI.

Figure 2 contains the query of the strategy, the analysis of the strategy, and the logical judgment based on the parsed strategy. Its logic can be decomposed to formulas (7)–(10) in the following order:

$$\begin{aligned}
 & \left[\begin{aligned}
 & \text{AccessItem} = \text{List} < \text{User/Group/Role} > + \text{List} < \text{Access Type} > \\
 & \text{DenyACL} = \text{List} < \text{AccessItem} > \text{deny} + \text{List} < \text{AccessItem} > \text{denyException} \\
 & \text{AllowACL} = \text{List} < \text{AccessItem} > \text{allow} + \text{List} < \text{AccessItem} > \text{allowException} \\
 & \text{Policy} = \text{List} < \text{Resource} > + \text{AllowACL} + \text{DenyACL}
 \end{aligned} \right] \quad (11)
 \end{aligned}$$

It can be seen from formula (11) that the user can get rights only after a quadruple loop. Since the time complexity of the nested loop is equal to the number of times the innermost statement of the line is executed, the time complexity of the algorithm is $T(n) = O(n^4)$. In big data systems, due to the large amount of data and resources being distributed in many different cluster nodes, the time to judge whether an access object has the right to access a certain resource right will be very long. Therefore, the current implementation of access control based on the “user-resource-privilege” policy will seriously affect the performance of big data systems.

3.3 Security access control algorithm based on memory index acceleration

Definition 10: Security access control algorithm based on memory index acceleration (SACABMIA): Using the principle of second-level cache to build keys, establish indexes, and place frequently accessed resources and rights on the memory accelerator

through the index. When a user requests access to a resource, system first checks the index. If there are no objects in the index, the index is then extracted and updated from the configuration resource. When the rights resource is changed, the algorithm updates the indexes in the secondary cache synchronously. Figure 3 shows the execution flow of the algorithm.

The key is constructed based on the user, the accessed object, and the type of access. The formula is as follows:

$$pk = generateKey(ao, u, at) \quad (12)$$

where ao is an accessed object, u is the access user, at is the access type, and pk is the constructed key.

The key can be parsed to obtain the accessed object, access user, and access type. The formula is as follows:

$$\langle ao, u, at \rangle = analyseKey(pk) \quad (13)$$

A memory accelerated index is built using the key and the rights obtained when a user accesses a WSN big data system.

$$indexMap = indexMap(pk, up) \quad (14)$$

where pk is the constructed key, up is the rights and $indexMap$ is the memory accelerated index.

The memory accelerated index can be parsed to a list of keys. The formula is as follows:

$$List < pk > = getAllKeyInMemoryIndex(indexMap) \quad (15)$$

where pk is the constructed key and $indexMap$ is the memory accelerated index.

The system updates the policies corresponding to all keys in the memory acceleration index when the system policy is changed. The pseudo code for the update algorithm is as follows:

```

Input: void
Output: void
1. funcation updateIndexMap()
2. arp←f(r) /*Get new rights policies*/
3. uap←getUserPolicy(u, arp) /* Assign policy to users*/
4. listPowerkey ← getAllKeyInMemoryIndex(indexMap) /* Obtain a list of keys */
5. for k to listPowerkey.size
6. pk←listPowerkey.get(k) /*Get ao, user from Powerkey*/
7. <ao, u, at>←analyseKey(pk) /*Get ao, user from Powerkey*/
8. ucp←getCurAP(ao, u, uap) /*Get user's access control policy*/
9. up←getAP(ao, ucp) /* Get access control rights */
10. indexMap←indexMap(pk, up)/* Update the memory acceleration index */
11. end for
12. return
13. end funcation

```

The process of getting permissions is as follows:

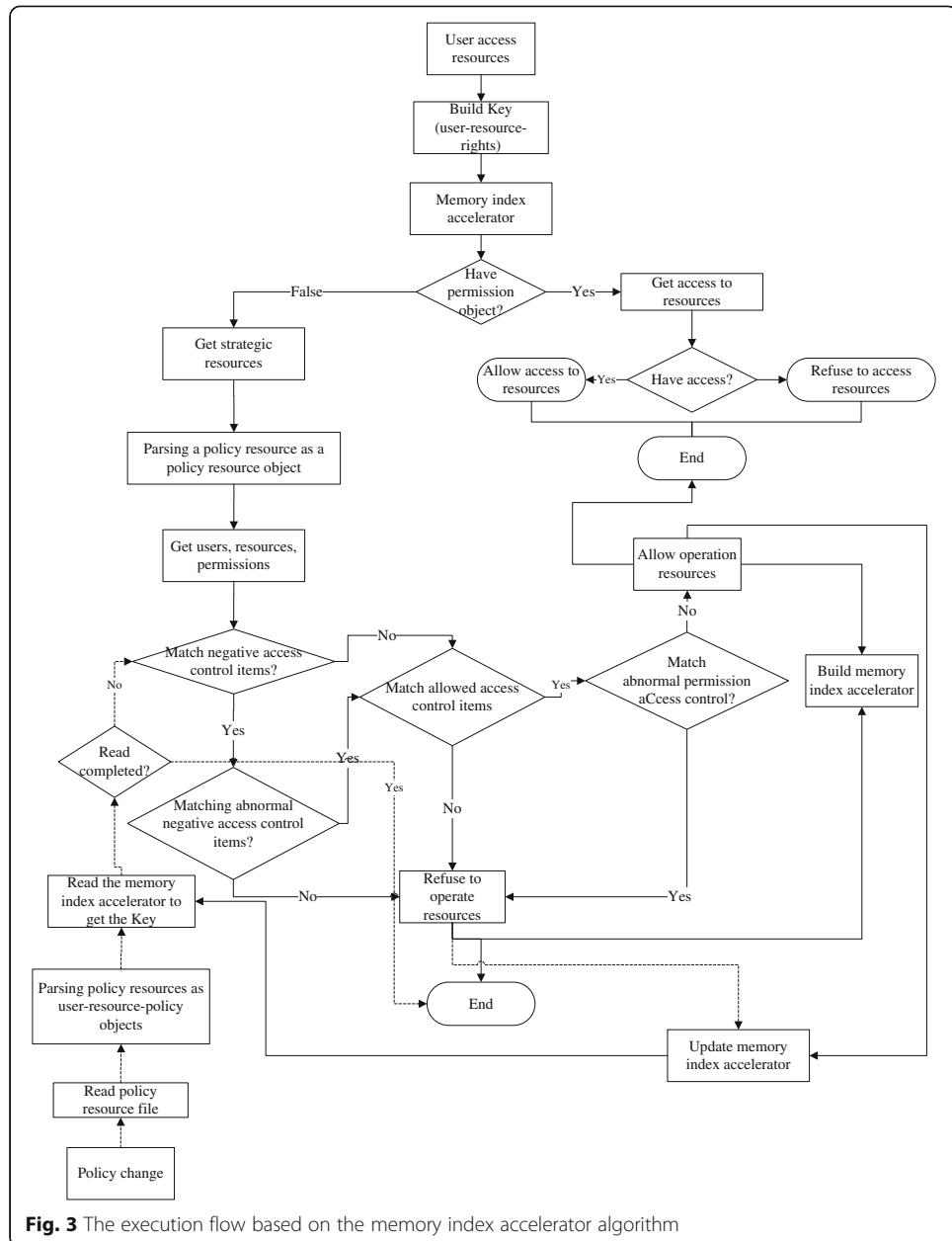


Fig. 3 The execution flow based on the memory index accelerator algorithm

$$\begin{aligned}
 up &= \text{indexMap}(pk) \\
 &= \begin{cases} \text{power}, & \text{indexMap}(pk) \neq \text{null} \\ \text{getAP}(ao, \text{getAP}(ao, \text{getCurAP}(ao, u, \text{getUserPolicy}(u, f(\text{dmp}(\text{datapro})))))), & \text{indexMap}(pk) = \text{null} \end{cases}
 \end{aligned} \tag{16}$$

In a WSNs big data system, user rights policy rarely changes once it is successfully configured [33, 34]. According to formula (16), the index relationship is relatively stable after the memory index is established, which greatly accelerates the efficiency and speed of access control in WSNs big data systems. It can be seen from formula (16) that the system can get rights when it only reads once from the memory index. So, the time complexity of the algorithm is $O(n)$.

Table 1 Test cluster node hardware configuration details

Node	Configuration
Node1	CPU: 32 core, Intel(R) Xeon(R) CPU E5-2650 v2 2.60GHz
Node2	Memory: 128GB
Node3	CPU: 48 core, Intel(R) Xeon(R) CPU E5-2670 v3 2.30GHz
Node4	Memory: 128GB

As can be seen from Fig. 3, the system builds a key based on User-Resource-Rights and accesses the memory index accelerator when the user accesses resources. If there is a User- Resource-Rights object corresponding to the key in the accelerator, the user will obtain the requested resources. Otherwise, the system obtains rights according to the logic of Fig. 1. The system re-reads the policy resource file, which is in json format, and converts it into a policy resource object when access control strategy is changed. The system also updates the memory index accelerator.

The specific method of building a key is as follows:

After receiving the user's access request, the system parses the request to obtain the user who visited this time, the resources to be accessed this time, and the rights of this access. Adds the user of this visit, the resources to be accessed this time, and the rights of this visit to get the key.

The main features of the “value” object of the in-memory index accelerator include service information, service definition list, policy details related to the user, resource, and rights status of the current access. The first layer is service information, service definition list, and policy detail list. The service definition includes the name of the service, the basic configuration related to the service, the resource details, the access type, and the policy conditions. The policy details mainly include information about policy-related resources, policy visitor objects, negative policy details, access policies, and the Exception Access Policy details.

4 Experimental results and analysis

The experiment was carried out on a four-node HBase cluster with Apache Ranger for test strategy.

The hardware configuration of the test cluster node is listed in Table 1.

A test table was created in HBase as listed in Table 2.

The YCSB (Cloud Serving Benchmark) tool was used for testing in this experiment. It is a tool developed by Yahoo to test the performance of cloud services. We generated

Table 2 Test table

Test	Score				
	Info		Score		
	Name	Age	Math	Physical	Political
00001	John	15	90	93	95
00002	Paul	15	91	92	97
00003	Carly	16	90	94	91
00004	Scott	14	92	93	96

Table 3 Current access control function test results based on the User-Resource-Rights policy

User	Policy number	Nodes	The amount of data	Execution time (s)	Data processing speed (ops/s)	Performance degradation rate
Hbase_test	0	4	100 million(102.4GB)	642.01	155687	base
Hbase_test	1	4	100 million(102.4GB)	706.26	141520	10.01%↓

102.4 GB of data using this tool and tested the current access control function based on the “User-Resource-Rights” policy.

We applied a security policy in HBase and tested performance degradation. The test results are listed in Table 3.

As shown in Table 3, the performance is reduced by 10.10% with a security policy applied for access control in HBase system. This result shows that the traditional access control based security technology greatly affects the performance of big data systems for WSN. Our previous analysis was proved by this result.

We also applied a different number of security policies in HBase to analyze their impact on system performance. The test results are listed in Table 4.

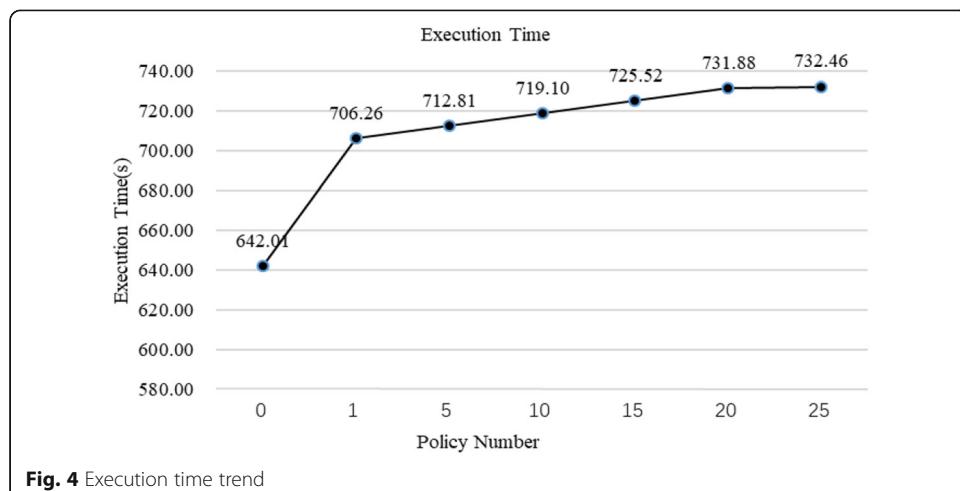
Figure 4 shows the correlation between execution time and number of strategies. The number of policies is set from 1 to 25 with an interval of 5. The execution time increases from 642.01 to 732.46 s using the old algorithm with the increase of the number of strategies. This indicates that the impact of access control security on system performance is high and the increase in the policy has also big effect on the performance when the old method is used.

Figure 5 shows the correlation between performance degradation and number of strategies. The number of policies is set from 1 to 25 with an interval of 5. The performance degradation rate decreased from 0 to 14.09% using the old algorithm with the increase of the number of strategies. This indicates that the impact of access control security on system performance is high and the increase in the policy has also big effect on the performance degradation rage when the old method is used.

As shown in Table 4, the execution time of HBase system gradually increased and its performance gradually degraded as the number of strategies increases. When the number of strategies reaches 25, the system performance drops by 14.09%. These results prove that the impact of access control algorithms on the performance of big data systems is gradually increasing with the number of policies increases. The results also

Table 4 Test results of different number of security policies based on the User-Resource-Rights policy

User	Policy number	Nodes	The amount of data	Execution time (s)	Data processing speed (ops/s)	Performance degradation rate
Hbase_test	0	4	100 million(102.4 GB)	642.01	155687	base
Hbase_test	5	4	100 million(102.4 GB)	712.81	140221	11.03%↓
Hbase_test	10	4	100 million(102.4 GB)	719.1	138994	12.01%↓
Hbase_test	15	4	100 million(102.4 GB)	725.52	137764	13.01%↓
Hbase_test	20	4	100 million(102.4 GB)	731.88	136578	14.00%↓
Hbase_test	25	4	100 million(102.4 GB)	732.46	136460	14.09%↓



verified that our analysis and judgment that the access control technology seriously affects the performance of big data systems is correct.

As a comparison, we performed experiments on the proposed new algorithm on HBase. Test results are listed in Table 5.

From the results in Table 5, we can see that with the new algorithm the performance of HBase reduced only by 2.74% when a security policy is applied for access control.

A different number of security policies in HBase is applied to test the performance of the new algorithm. The test results are listed in Table 6.

Figure 6 presents the correlation between execution time and number of strategies. The graph shows that the time for users to access 102.4 GB data is 652.40 s in the case of one security policy, the time for users to access 102.4 GB data is 652.60 s in the case of five security policies, the time for the user to access the 102.4 GB data is 652.78 s in the case of ten security policies, and the time for the user to access the 102.4 GB data is 653.29 s in the case of twenty-five security policies. It shows that the increase in the policy has little effect on the time spent in parsing the data.

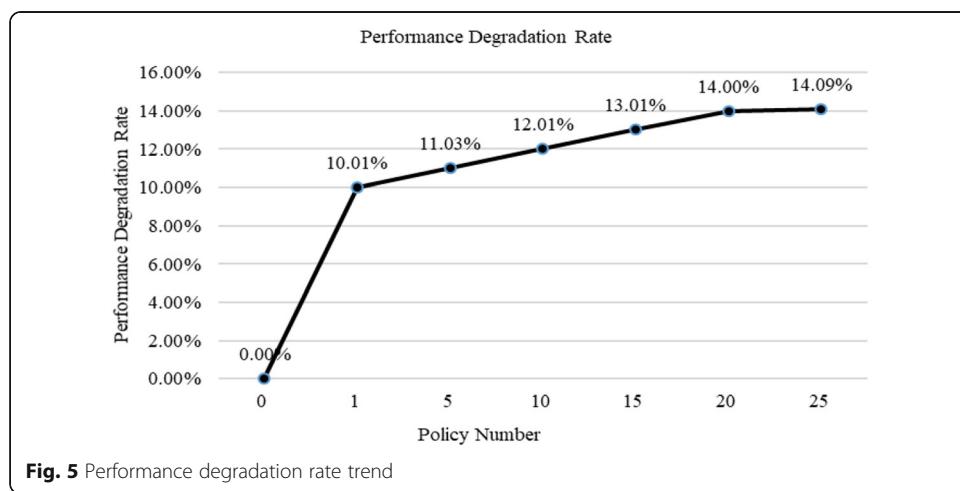


Table 5 Access control function test results using the new algorithm

User	Policy number	Nodes	The amount of data	Execution time (s)	Data processing speed (ops/s)	Performance degradation rate
Hbase_test	0	4	100 million(102.4GB)	635	157484	base
Hbase_test	1	4	100 million(102.4GB)	652	153284	2.74%↓

As shown in Table 6, when the number of strategies increases, execution time of HBase with new algorithm increases much slower than same system with traditional algorithm. System performance degradation is also much slower with new algorithm. When the number of strategies reaches 25, the system performance drops only 2.88%. The system performance is much less affected.

Figure 7 reveals the correlation between performance degradation and number of strategies. The graph shows that the performance degradation rage for users to access 102.4 GB data is 2.74% in the case of two security policy, the performance degradation rage for users to access 102.4 GB data is 2.77% in the case of three security policies, the performance degradation rage for the user to access the 102.4 GB data is 2.80% in the case of four security policies, and the performance degradation rage for the user to access the 102.4 GB data is 2.88% in the case of seven security policies. This indicates that the impact of access control security on system performance is low and the increase in the policy has little effect on the performance degradation rage.

Figure 8 shows the performance degradation trend of the new algorithm compared with the traditional algorithm. The number of policies is set from 1 to 25 with an interval of 5. The performance degradation rate decreased from 0 to 14.09% using the old algorithm, and the performance degradation rate only decreased from 0 to 2.88% using the new algorithm with the increase of the number of strategies. This indicates that the impact of access control security on system performance is low, and the increase in the policy has also little effect on the performance degradation rage when the method is used.

From the experimental results, we can get the conclusions:

The new algorithm can significantly improve the performance of big data systems where the fine-grained security policy-based access control model is applied.

Big data security technologies mainly include data asset grooming, data encryption, data security operation and maintenance, data desensitization, and data leakage scanning. These security technologies can only exert their value if applied to big data systems. The performance impact of big data systems using big data security includes two

Table 6 Test results of different number of security policies using the new algorithm

User	Policy number	Nodes	The amount of data	Execution time (s)	Data processing speed (ops/s)	Performance degradation rate
Hbase_test	0	4	100 million(102.4 GB)	635.02	157484	base
Hbase_test	5	4	100 million(102.4 GB)	652.60	153239	2.77%↓
Hbase_test	10	4	100 million(102.4 GB)	652.78	153195	2.80%↓
Hbase_test	15	4	100 million(102.4 GB)	652.91	153165	2.82%↓
Hbase_test	20	4	100 million(102.4 GB)	653.10	153120	2.85%↓
Hbase_test	25	4	100 million(102.4 GB)	653.29	153075	2.88%↓

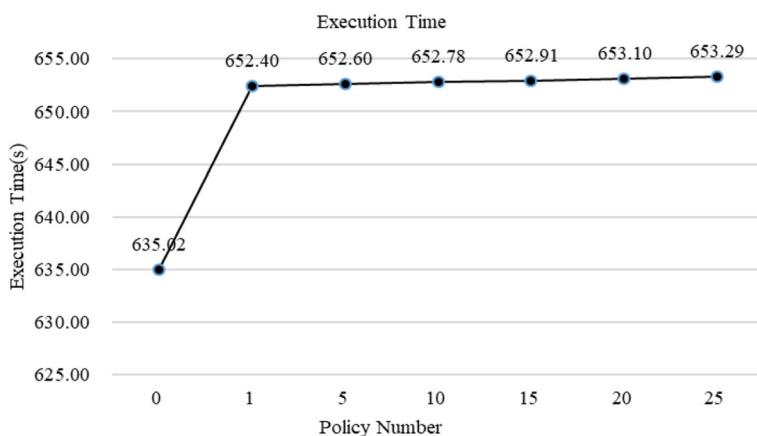


Fig. 6 The execution time trend

aspects, one is the impact of the security algorithm itself on the system, and the other is the impact of the method of applying the security algorithm on the system performance. This paper mainly studies from the second perspective to improve the performance of big data systems. The method proposed in the paper will not improve the performance of big data security algorithms.

By analyzing the results of RANGER-1729 and this experiment, we can draw the conclusion that the new algorithm can significantly reduce the impact of access control technology on the performance of big data systems. RANGER-1729 is an issue of Apache Ranger project, and its link address is <https://issues.apache.org/jira/browse/RANGER-1729>.

5 Conclusions

Performance is the life of big data systems and security is the cornerstone of big data systems. When big data systems apply access control technology to ensure the security of data, existing methods will seriously affect the system performance. This paper first

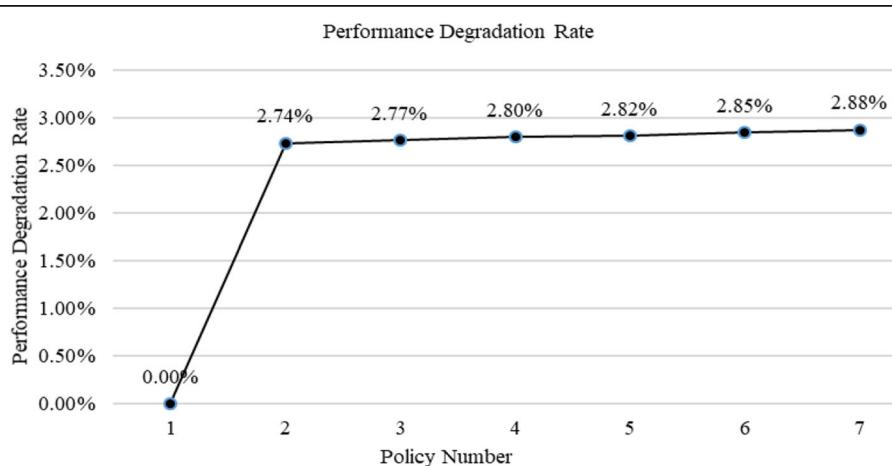
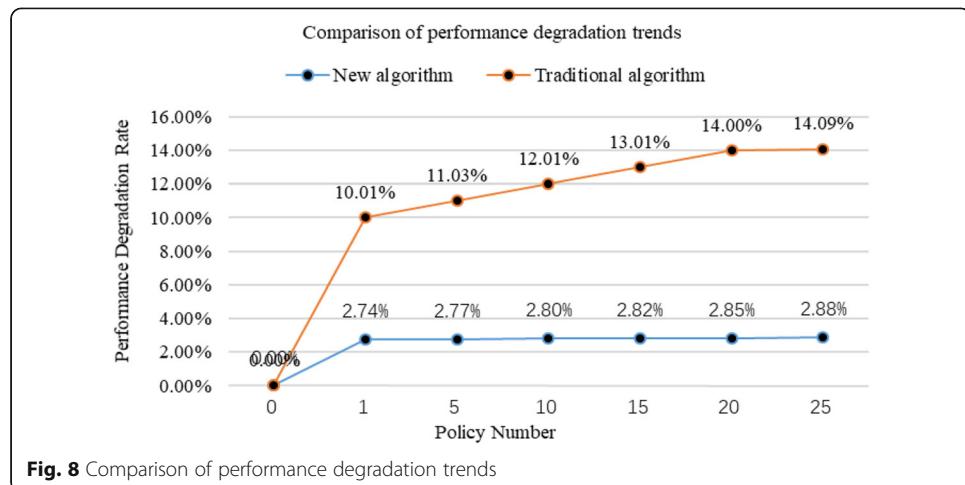


Fig. 7 The performance degradation rate trend



analyses the data processing flow of the existing access control technology in the big data system and its time complexity. Then it points out that the system performance will be greatly affected by this existing technology. We proposed a big data security access control algorithm based on memory index acceleration for WSNs in this article. We walked through its data processing flow and analyzed its time complexity. We also theoretically proved that the new algorithm has better performance. Through experiments, we further proved that compared with the traditional access control technology, the new algorithm has less impact on the performance of big data systems.

Abbreviations

YCSB: Cloud Serving Benchmark; NTRU: Number Theory Research Unit; DAC: Discretionary access control; MAC: Mandatory access control; RBAC: Role-based access control; ABAC: Attribute-based access control; PBAC: Policy-based access control; RABE: Revocable and grantable attribute-based encryption

Acknowledgements

The authors acknowledged the anonymous reviewers and editors for their efforts in valuable comments and suggestions.

Authors' contributions

J. Peng proposes the innovation ideas and theoretical analysis, and H. Zhou carries out experiments and data analysis. Q. Meng and J. Yang conceived of the study, and participated in its design and coordination and helped to draft the manuscript. All authors read and approved the final manuscript.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the school research fund of Nanjing Institute of Industry Technology (Grant No. YK18-05-03).

Availability of data and materials

Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Competing interests

The authors declare that they have no competing interests.

Received: 14 January 2020 Accepted: 23 April 2020

Published online: 07 May 2020

References

1. Z. Huang, X. Xu, J. Ni, H. Zhu, C. Wang, Multimodal representation learning for recommendation in Internet of Things. *IEEE Internet Things J.* **6**(6), 10675–10685 (2019)
2. B. Wu, T.L. Yip, X. Yan, C. Guedes Soares, Fuzzy logic based approach for ship-bridge collision alert system. *Ocean Eng.* **187**, 106152 (2019)
3. P. Alexander, L. Pike, P. Loscocco, G. Coker, Model checking distributed mandatory access control policies. *ACM Transactions on Information and System Security (TISSEC)* **18**(2), 1–25 (2015)

4. H. Liu, X. Yao, T. Yang, H. Ning, Cooperative privacy preservation for wearable devices in hybrid computing-based smart health. *IEEE Internet Things J.* **6**(2), 1352–1362 (2018)
5. Terzis, S., Wagealla, W., English, C., & Nixon, P. Trust lifecycle management in a global computing environment. In: International Workshop on Global Computing (pp. 291–313). Springer, Berlin, Heidelberg (2004).
6. Chang, R., Jiang, L., Chen, W., He, H., Yang, S., Jiang, H., & Liu, Y. (2018). Towards a multilayered permission-based access control for extending Android security. *Concurrency and Computation: Practice and Experience*, **30**(5), e4180 (2018).
7. N. Mundbrod, M. Reichert, Object-specific role-based access control. *International Journal of Cooperative Information Systems* **28**(01), 1950003 (2019)
8. M.U. Aftab, Z. Qin, N.W. Hundera, O. Ariyo, N.T. Son, T.V. Dinh, Permission-based separation of duty in dynamic role-based access Control Model. *Symmetry* **11**(5), 669 (2019)
9. C.M. Subramanian, A.K. Cherukuri, C. Chelliah, Role based access control design using three-way formal concept analysis. *Int. J. Mach. Learn. Cybern.* **9**(11), 1807–1837 (2018)
10. V. Hu, C. Kuhn, D. Richard, D.F. Ferraiolo, Attribute-based access control. *Computer* **48**(2), 85–88 (2015)
11. M. Uriarte, J. Astorga, E. Jacob, M. Huarte, M. Carnerero, Expressive policy-based access control for resource-constrained devices. *IEEE Access* **6**, 15–46 (2017)
12. J.P. Cruz, Y. Kaji, N. Yanai, RBAC-SC: role-based access control using smart contract. *IEEE Access* **6**, 12240–12251 (2018)
13. S. Aditham, N. Ranganathan, A system architecture for the detection of insider attacks in big data systems. *IEEE Transactions on Dependable and Secure Computing* **15**(6), 974–987 (2017)
14. Q. Xia, E.B. Sifah, K.O.B.O. Agyekum, H. Xia, K.N. Acheampong, A. Smahi, M. Guizani, Secured fine-grained selective access to outsourced cloud data in IoT environments. *IEEE Internet Things J.* **6**(6), 10749–10762 (2019)
15. Y. Zhu, D. Huang, C.J. Hu, X. Wang, From RBAC to ABAC: constructing flexible data access control for cloud storage services. *IEEE Trans. Serv. Comput.* **8**(4), 601–616 (2014)
16. S. Wang, X. Zhang, Y. Zhang, Efficient revocable and grantable attribute-based encryption from lattices with fine-grained access control. *IET Inf. Secur.* **12**(2), 141–149 (2018)
17. M. Zhang, D. Zhang, F. Goerlandt, X. Yan, P. Kujala, Use of HFACS and fault tree model for collision risk factors analysis of icebreaker assistance in ice-covered waters. *Saf. Sci.* **111**, 128–143 (2019)
18. M. Alam, N. Emmanuel, T. Khan, Y. Xiang, H. Hassan, Garbled role-based access control in the cloud. *J. Ambient. Intell. Humaniz. Comput.* **9**(4), 1153–1166 (2018)
19. S. Pal, M. Hitchens, V. Varadharajan, T. Rabehaja, Policy-based access control for constrained healthcare resources in the context of the Internet of Things. *J. Netw. Comput. Appl.* **139**, 57–74 (2019)
20. M. Babar, F. Khan, W. Iqbal, A. Yahya, F. Arif, Z. Tan, J.M. Chuma, A secured data management scheme for smart societies in industrial internet of things environment. *IEEE Access* **6**, 43088–43099 (2018)
21. D. Chattaraj, M. Sarma, A.K. Das, N. Kumar, J.J. Rodrigues, Y. Park, HEAP: an efficient and fault-tolerant authentication and key exchange protocol for Hadoop-assisted big data platform. *IEEE Access* **6**, 75342–75382 (2018)
22. Huang Z., Tang J., G. Shan, Ni J., Chen Y., & Wang C. An efficient passenger-hunting recommendation framework with multi-task deep learning. *IEEE Internet of Things Journal* . DOI: <https://doi.org/10.1109/IoT.2019.2901759>(2019).
23. X. Liu, R. Zhu, A. Anjum, J. Wang, H. Zhang, M. Ma, Intelligent data fusion algorithm based on hybrid delay-aware adaptive clustering in wireless sensor networks. *Futur. Gener. Comput. Syst.* **104**, 1–14 (2020)
24. H. Xie, Z. Yan, Z. Yao, M. Atiquzzaman, Data collection for security measurement in wireless sensor networks: a survey. *IEEE Internet Things J.* **6**(2), 2205–2224 (2018)
25. X. Liu, R. Zhu, B. Jalaian, Y. Sun, Dynamic spectrum access algorithm based on game theory in cognitive radio networks. *Mobile Networks and Applications* **20**(6), 817–827 (2015)
26. R. Zhu, X. Zhang, X. Liu, W. Shu, T. Mao, B. Jalaian, ERDT: Energy-efficient reliable decision transmission for intelligent cooperative spectrum sensing in industrial IoT. *IEEE Access* **3**, 2366–2378 (2015)
27. J. Li, Y. Ji, K.K.R. Choo, D. Hogrefe, CL-CPPA: certificate-less conditional privacy-preserving authentication protocol for the Internet of Vehicles. *IEEE Internet Things J.* **6**(6), 10332–10343 (2019)
28. C. Hu, W. Li, X. Cheng, J. Yu, S. Wang, R. Bie, A secure and verifiable access control scheme for big data storage in clouds. *IEEE Transactions on Big data* **4**(3), 341–355 (2017)
29. J.A. Padget, W.W. Vasconcelos, Fine-grained access control via policy-carrying data. *ACM Transactions on Internet Technology (TOIT)* **18**(3), 1–24 (2018)
30. Y. Mo, A data security storage method for IoT under Hadoop cloud computing platform. *Int. J. Wireless Inf. Networks* **26**(3), 152–157 (2019)
31. X. Fu, Y. Gao, B. Luo, X. Du, M. Guizani, Security threats to Hadoop: data leakage attacks and investigation. *IEEE Netw.* **31**(2), 67–71 (2017)
32. X. Min, Q. Yong, W. Kui, Z. Jizhong, L. Mo, Using potential to guide mobile nodes in wireless sensor networks. *Ad Hoc & Sensor Wireless Networks* **12**(3-4), 229–251 (2011)
33. Y. Yang, X. Zheng, W. Guo, X. Liu, V. Chang, Privacy-preserving smart IoT-based healthcare big data storage and self-adaptive access control system. *Inf. Sci.* **479**, 567–592 (2019)
34. K. Yang, Q. Han, H. Li, K. Zheng, Z. Su, X. Shen, An efficient and fine-grained big data access control scheme with privacy-preserving policy. *IEEE Internet Things J.* **4**(2), 563–571 (2016)

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.