Research Article

AB_sAC: Attribute-Based Access Control Model Supporting Anonymous Access for Smart Cities

Runnan Zhang ¹, Gang Liu ¹, Shancang Li ², Yongheng Wei ¹ and Quan Wang ¹

¹School of Computer Science and Technology, Xidian University, Xi'an 710071, China

²Department of Computer Science, University of the West of England, Bristol BS16 1QY, UK

Correspondence should be addressed to Gang Liu; gliu@xidian.edu.cn

Received 3 February 2021; Revised 21 February 2021; Accepted 2 March 2021; Published 22 March 2021

Academic Editor: Qi Li

Copyright © 2021 Runnan Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Smart cities require new access control models for Internet of Things (IoT) devices that preserve user privacy while guaranteeing scalability and efficiency. Researchers believe that anonymous access can protect the private information even if the private information is not stored in authorization organization. Many attribute-based access control (ABAC) models that support anonymous access expose the attributes of the subject to the authorization organization during the authorization process, which allows the authorization organization to obtain the attributes of the subject and infer the identity of the subject. The ABAC with anonymous access proposed in this paper called AB_sAC strengthens the identity-less of ABAC by combining homomorphic attribute-based signatures (HABs) which does not send the subject attributes to the authorization organization, reducing the risk of subject identity re-identification. It is a secure anonymous access framework. Tests show that the performance of AB_sAC implementation is similar to ABAC's performance.

1. Introduction

Smart cities will involve millions of autonomous smart objects around us, monitoring, collecting, and sharing data without us many times being aware of it [1]. Such pervasive and autonomous behavior can be seen as a peril without proper security and focus on user privacy preferences. Indeed, stakeholders should provide citizens with the tools to ensure their privacy and safely participate in IoT services [2].

Attribute-based access control (ABAC) could address the issues of the fine-granularity of resource protection and the user scalability of network systems, and it can provide appropriate strategies for the access control in the open network environment in the future [3]. With the fine-grained access to data, ABAC can guarantee the security of private information in the validation process of zero trust. ABAC model is considered to be identity-less in its infancy. It is the most important privacy protection-related feature of ABAC. In open computing environments, there is an urgent need for privacy protection and anonymous authorization. In many scenarios, researchers have proposed access control

models that support anonymous authorization [4], such as cloud computing [5, 6], wireless body area networks [7], payment [8], and smart grid [9]. The methods mainly use attributes to encrypt objects to support anonymous authorization in the access control model.

Ahuja and Mohanty [10] proposed a scalable attribute-based encryption (ABE) scheme in cloud environment. The scheme generates a hierarchical attribute private key for users through hierarchical authority and hierarchical ciphertext policy attribute-based encryption (CP-ABE) algorithm. The scheme supports flexible authorization, authority delegation, and authority sharing through hierarchical structure and private key of the hierarchical attribute. Yuen et al. [11] proposed an anonymous ABAC model based on an anonymous certificate to support k-times anonymous authorization for cloud services. When the subject is authenticated, its private key is randomly generated and delivered from the attribute publishing agency. When the subject accesses the service, it needs to obtain the certificate generated by the public key of the trusted institution, the private key of the user, and the number of times of access and

prove that he has the property satisfying a certain policy through its own attribute private key. If the number of accesses in the certificate is less than the maximum number of accesses, the subject can access the service. Given the serious harm of privacy disclosure in the personal health records (PHR) system, Pussewalage and Oleshchuk [12] proposed an anonymous ABAC scheme based on ABE and proxy re-encryption to protect privacy. The scheme assumes that all subjects have public key infrastructure (PKI) certificates. Subjects obtain attribute private key from attribute authorities (AAs) (including PHR owner). Proxy re-encryption reduces the computing cost of the PHR owner. If the subject has attributes that meet the access policy of PHR, it can access the PHR through his own attribute private key.

The above method supports anonymous access by using a certificate instead of a subject identity or ABE. There are some defects in these methods. In some schemes, the certificate of the subject is unique, so the subject's access may be linked to its certificate. The attacker can re-identify the identity of the subject through the access linked to the subject. Once the re-identification is successful, it will cause unexpected privacy disclosure [13]. The encryption of objects based on attributes limits the types of objects. It is friendly to access control of objects that can be moved from server to client, such as files. It is unfriendly to access control of objects that cannot be moved to client, such as web services, which needs additional mechanisms to provide access control and increases the complexity of its implementation. And using an ABE-based algorithm to access objects often needs to download the object or generate tokens, which increases the load of the network.

In this paper, we propose an ABAC model that supports anonymous access called AB_sAC. By combining homomorphic attribute-based signatures (HABSSs) [14] and transferring some functions of ABAC to the attribute authority and audit institution, we strengthen the identity-less of ABAC, so that its authorization is no longer dependent on identity. HABS supports encrypted attribute operations, and the identity of the subject is not included in the signature. This makes it the most suitable encryption algorithm to support anonymous access in ABAC. AB_sAC does not use unique certificates and is friendly for all types of objects. It inherits the features of fine-grained access control, flexible policy, and unlimited object type of ABAC. It overcomes the defect that the ABAC framework does not support anonymous access and provides an audit function that improves security.

Our contribution:

- (1) This paper proposes an ABAC framework that supports anonymous access and fine-grained control of attributes by subjects. AB_sAC is a direct extension of the ABAC framework, which has compatibility with existing ABAC implementations. It inherits many advantages of ABAC, such as fine-grained access control, flexible policy, and unlimited resource. AB_sAC supports audit function.
- (2) AB_sAC's threat model and detailed procedures are described.

- (3) An AB_sAC implementation is shown. Tests show that the performance of AB_sAC implementation is similar to ABAC's performance.

To the best of our knowledge, this paper is the first to propose an ABAC model supporting anonymous authorization by extending the authorization framework of the ABAC. Moreover, this model solves the problems of object type restriction and authorization based on a unique identifier. This paper continues Zhang's work [15].

2. Preliminary

2.1. ABAC Model. The attribute-based access control model (ABAC) is an emerging access control model that has attracted much attention from scholars. The ABAC model has many excellent features such as fine-grained, flexible, dynamic, identity-less, and rich policy expression, which makes it have a wide range of application scenarios, such as distributed computing [16], Internet of Things (IoT) healthcare [17], IoT home devices [18], financial industry [19], and so on. While there is currently no single agreed-upon model or standardization of ABAC, there are commonly accepted high-level definitions and descriptions of its function. One such high-level description is presented in the National Institute of Standards and Technology's publication (NIST), a "Guide to Attribute Based Access Control (ABAC) Definition and Considerations" [20].

Attribute-based access control is an access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environmental conditions, and a set of policies that are specified in terms of those attributes and conditions.

Many researchers use the framework which is proposed by XACML [21] shown in Figure 1, in which context handler, Policy Enforce Point (PEP), Policy Decision Point (PDP), Policy Information Point (PIP), and Policy Attribute Point (PAP) are internal services of authorization organization and are managed by an authorization organization. Context handler performs preliminary verification of access requests. PEP executes access control decisions. PDP evaluates access requests based on acquired attributes and policies. PIP manages attributes of all subjects, objects, and environment. PAP manages and maintains a policy library. In ABAC, when a subject accesses an object, it sends an access request to an authorized organization. The process of evaluating access requests by the ABAC framework is as follows:

- (1) Subject sends an access request to PEP, and PEP forwards the access request to context handler.
- (2) Context handler verifies the correctness of access request. Context handler sends the access request to PIP. The PIP performs attribute retrieval based on the unique identifier of the subject and object in the access request and returns the subject attribute set, object attribute set, and environment attribute set required for policy evaluation to the context handler.

FIGURE 1: ABAC framework with context handler.

- (3) The access request and attribute set are forwarded to PAP. PAP performs policy matching according to the attribute sets provided by context handler and returns an applicable policy set to context handler.
- (4) Context handler sends the subject attribute set, object attribute set, environment attribute set, and applicable policy set to PDP. PDP evaluates access request based on the obtained attribute sets and policy set and returns the access control decision to context handler.
- (5) Context handler sends the access control decision to PEP. PEP enforces the access control decision.

Although the context handler seems to be just forwarding messages throughout the access control process, it improves the compatibility of the framework. The context handler converts the format of messages between different points to make the points compatible. Other researchers use the ABAC framework [6] as shown in Figure 2, except that there is no context handler, and the other processes are consistent with Figure 1.

In the authorization process of ABAC, the subject has certain anonymity, that is, PDP only makes access control decisions based on attributes and policy sets. In this process, the PDP does not need to use the subject identity information contained in the access request which is called identity-less. Identity-less is considered to be the basis for implementing anonymous access in ABAC.

However, the access request used by the ABAC framework shown in Figure 1 must contain the unique identifier of the subject and object. The basis for retrieving attributes is the unique identifier of the subject and object contained in the access request. And PIP is part of the authorization organization. Obviously, the authorization organization has the subject's attributes, the subject's identity (subject's unique identifier), and access records. Once the authorization organization becomes abnormal or becomes untrustworthy, the privacy information is leaked, and the consequences are disastrous.

If the access request submitted by the subject to the authorization organization is anonymous, the authorization organization no longer has access records. The AB_sAC framework proposed in this paper separates PIP subject-related functions from the authorized institution to a trusted

third party. In this way, the subject could choose the attributes for authorization without revealing his identity. In addition, the framework also supports auditing, which can reveal the identity of the subject when necessary and ensure the security of the system. Extending the ABAC framework and giving it anonymous access and audit support can expand ABAC's scope of application and improve its practicality [22].

3. HABS Algorithm

HABS is an anonymous certification scheme based on the attribute-based signatures (ABSs). The ABS is designed for the user to sign a message with fine-grained control over identifying information, and it does not support the properties required for anonymous certification [23]. HABS has a clear identification of missing properties to serve anonymous certification objectives. HABS supports a flexible selective disclosure mechanism at no extra computation cost [14], which is inherited from the expressiveness of ABS for defining access policies.

HABS relies on four procedures based on the following that involves the inspector, subject, issuer, and verifier.

System initialization procedure is shown in Figure 3. The procedure derives a global parameter containing the inspector's public key and pairs of public and private keys for the subject, inspector, and issuer. HABS.Setup and HABS.KeyGen are executed by the trusted third party.

HABS.Setup: it takes as input the security parameter ξ and outputs the global public parameter $params$. This algorithm also derives a pair of public and private keys (pk_{ins}, sk_{ins}) for the tracing authority referred to as the inspector. In the following, public parameters $params$ are assumed to include the public key of the inspector, and all the algorithms have default input $params$.

HABS.KeyGen: this algorithm takes as input the global parameters $params$ and outputs the pair of public and private keys for subjects and the issuer. The public and private keys are noted, respectively, for subjects (pk_s, sk_s) and for the issuer (pk_{is}, sk_{is}) .

Credential issuing procedure is shown in Figure 4. It issues a certified credential for the subject based on its attributes. The HABS.Issue algorithm is executed by the issuer. The HABS.Obtain algorithm is executed by the subject.

FIGURE 2: ABAC framework without context handler.

FIGURE 3: System initialization procedure.

FIGURE 4: Credential issuing procedure.

HABS.Issue: the algorithm takes as input the public key of the subject pk_s , the set of attributes $A_s \subset A$ belonging to the subject s (where A is referred to as the attribute universe), the private key of the issuer sk_{is} , and the public key of the inspector pk_{ins} . It outputs a certified credential C over the set of attributes A_s .

HABS.Obtain: the algorithm is up to the user to verify the correctness of the received the certified credential C over its attributes. The algorithm takes as input the signed commitment C , the private key of the subject sk_s , the public key of the issuer pk_{is} , and eventually the public key of the inspector pk_{ins} . It outputs a bool to point out the correctness of the credential C .

The verifying procedure shown in Figure 5 enables the verifier to check that a subject is authorized to access an object with respect to some access policy. As such, the verifier has first to send a random message to the subject. Second, the user signs the received message based on his credential. In a nutshell, the subject signs the received message based on the subset of his attributes that satisfy the

signature predicate. The user finally sends his signature to the verifier who checks the resulting signature. The HABS.Show algorithm is executed by the subject. The HABS.Verify algorithm is executed by the verifier.

The HABS.Show algorithm takes as input the randomized message m , a signing predicate Υ , the private key of the subject sk_s , the credential C , and a subset of its attributes A'_s , such as $\Upsilon(A'_s) = 1$. This algorithm outputs a signature Σ (or an error message $\perp$).

The HABS.Verify algorithm takes as input the received signature Σ , the public key of the issuer pk_{is} , the signing predicate Υ , and the message m . It outputs a bool.

The HABS supports the inspection procedure shown in Figure 6 performed by a separate and trusted entity referred to as the inspector. It relies on two algorithms, namely, HABS.trace and HABS.judge, needed to identify the subject and give a proof of judgment. They are executed by the inspector.

The HABS.trace algorithm takes as input the secret key of the inspector sk_{ins} , the issuer public key pk_{is} , and the

FIGURE 5: Verifying procedure.

FIGURE 6: Inspection procedure.

signature Σ . It outputs the identity S of the subject that has signed the message m with respect to the predicate Y . It also outputs a proof ω .

The HABS.judge algorithm takes as input the public key of the issuer pk_{is} , the signature Σ , the identity S of the subject, and the proof ω . It outputs a bool, where True means that ω is a valid proof which proves that signature originates from subject (S) Σ .

HABS supports a flexible selective disclosure mechanism that allows the subject to sign the message m with a subset of its attributes A'_s . The verifier only obtains A'_s after successful verification and does not reveal the identity of the subject. In addition, this scheme ensures the unlinkability between sessions while maintaining the anonymity of the subject.

The flexible selective disclosure mechanism of HABS and the identity-less feature of ABAC have complementary advantages. It maintains the anonymity of subjects while providing appropriate attributes for the ABAC authorization process. At the same time, HABS adds the inspector's public key pk_{ins} when generating the anonymous attribute certificate, and when necessary, the inspector audits specific access. The HABS algorithm can be well combined with the ABAC model, giving it anonymous access and auditing capabilities.

4. AB_sAC Framework

AB_sAC is an access control framework with anonymous access and audit capabilities. The authorization agency is not malicious and curious. It tries to obtain the identity of the subject by analyzing the information carried in the subject's access request. The AB_sAC prevents the authority from acquiring the identity of the subject.

4.1. The Architecture of AB_sAC. AB_sAC shown in Figure 7 is a direct extension of the ABAC framework, which has compatibility with existing ABAC implementations. It inherits many advantages of ABAC, such as fine-grained access control, flexible policy, and unlimited object. The

functions of PDP, PEP, and PAP in the AB_sAC framework are consistent with the corresponding modules in the framework shown in Figure 2. The main functions of other modules are as follows:

- (i) PIP: in order to achieve anonymous access in AB_sAC, the subject attributes are managed by the attribute authority (AA) and the subject itself. PIP is only responsible for collecting and managing object attributes and environmental attributes. And PIP sends the corresponding attributes to the context handler according to the attribute requests.
- (ii) Context handler: repackages and redirects requests based on context.
- (iii) Key distribution center (KDC): a trusted third party responsible for generating and distributing keys.
- (iv) Attribute authority (AA): in AB_sAC, AA is responsible for collecting and managing all subject attributes. During the subject authentication process, the AA generates an attribute certificate for the subject based on the attributes owned by the subject. And AA helps the audit authority to complete the audit when the audit is needed.
- (v) Audit authority (Au): in AB_sAC, Au is responsible for auditing subject access. When the user conducts illegal operations or under other situations that require auditing, the context handler sends an audit request to Au. With the help of AA, Au can track the signature and reveal the identity of the subject.

KDC, AA, and Au should be trusted third parties. If KDC and Au are not trusted third parties, then the authorization organization can directly initiate an audit to obtain the identity of the subject. If AA is controlled by an authorization organization, it can identify the subject by issuing attributes with unique values and creating corresponding policies.

4.2. Workflow of AB_sAC. The ABAC model has four workflows, which are initialization, registration, anonymous access, and audit. Before the system starts running, the initialization workflow is executed:

- (i) (i.1) KDC runs HABS.Setup and HABS.KeyGen to generate public and private key pairs for AA and Au and distributes secret keys.

After the system is initialized, the system is started. Registration, anonymous access, and audit are all performed at runtime. When the subject enters the system for the first time, it executes the registration workflow:

- (i) (r.1) The subject sends a registration request to AA and KDC.
- (ii) (r.2) The AA assigns attributes to the subject according to the registration request and returns the attribute set. KDC generates a public and private key pair for the subject and sends it to the subject.
- (iii) (r.3) The subject sends a certificate request to the AA, and the request contains the subject's public key.

FIGURE 7: ABsAC framework; the pink module is provided by a trusted third party; the yellow module is provided by the authorization organization; the blue module is the subject and object; req. means request; res. means result; eval. means evaluation.

- (iv) (r.4) After receiving the certificate request, AA issues the certificate for the subject and returns it to the subject.
- (v) (r.5) After receiving the attribute certificate, the subject verifies whether the attribute certificate is consistent with its own attribute set and returns the verification result to the AA.

The subject can obtain a new certificate by re-registering, and the old certificate will be revoked.

When the system is running, the subject anonymously accesses the object:

- (i) (ac.1) When the subject accesses the object, it sends an access request to the context processor. The access request contains only the object's unique identifier and operation.
- (ii) (ac.2) After receiving the access request, the context handler forwards it to the PIP.
- (iii) (ac.3) The PIP returns the relevant object attributes and environment attributes to the context handler according to the unique identifier of the object in the access request.
- (iv) (ac.4) The context handler forwards the object attributes, environment attributes, and access request to PAP.
- (v) (ac.5) PAP performs policy matching based on these attributes. PAP returns the policies set suitable for these attributes to the context handler.
- (vi) (ac.6) Due to the lack of subject attributes in the access request, policy evaluation cannot be performed. The context handler generates the signature predicate γ and the random message m for each policy in the policy set and packages them into signature requests and send signature requests to the subject.
- (vii) (ac.7) After receiving the signature request, the subject selects the appropriate subset of attributes A'_s and runs HABS. Show to sign the random message m in the signature request and sends the signature Σ to the context handler.

- (viii) (ac.8) After receiving the signature, the context handler packages the signature with the corresponding policy, object attributes, and environment attributes into a policy evaluation request and sends it to the PDP.
- (ix) (ac.9) After receiving the policy evaluation requests, the PDP evaluates the policies based on the attributes in the requests. Then, PDP combines the evaluation results of these policies to form an access control (AC) decision and returns it to the context handler.
- (x) (ac.10) The context handler sends the access control decision to the subject. If the decision is denied, the workflow ends. If the decision is permitted, it is forwarded to PEP and PEP enforces the access control decision.

When abnormal access of the subject is discovered, the audit workflow can be started:

- (i) (au.1) The context sends an audit request to Au. The audit request contains the signature Σ and access request.
- (ii) (au.2) Au evaluates the audit request. If the audit decision is denied, the audit decision will be output directly. If the audit decision is permitted, the HABS. Trace and HABS. Judge algorithm is executed based on the information in the audit request to expose the corresponding subject identity. Then, Au packages the subject identity and the evidence as to the audit decision and outputs it.

Obviously, the subject identity is not involved in ABsAC's access workflow. The subject-related information obtained by the context handler or PDP in the access workflow is the signature Σ . The signature Σ only involves the predicate γ , the message m , and a subset of subject attributes A'_s . Thus, ABsAC's access workflow is identity independent, and ABsAC's access is anonymous. The message m in the signature request is randomly generated, making it impossible for the authorization organization to link multiple accesses to one credential. This allows ABsAC to overcome the drawback of using unique

credentials. The operations on objects in AB_sAC are performed by PEP, and there is no restriction on the type of objects. AB_sAC's improvement of ABAC does not involve the policies, so it inherits many advantages of ABAC, such as the fine-grained access control and flexible policy.

5. Analysis

Kaaniche and Laurent [14] proved the correctness, unforgeability, anonymity, and the anonymity removal of the HABS algorithm. The security feature of AB_sAC is the same as the HABS algorithm. AB_sAC has the characteristics of HABS and ABAC.

There are many literatures comparing multiple characteristics of different models, which can help researchers quickly understand these models. Servos et al. [22] divided the existing ABAC works into ABAC implementation, ABAC model, policy, attribute, and other categories and compared the granularity, flexibility, and dynamic characteristics of these works. Aftab et al. [24] analyzed and compared ABAC and RBAC. The evaluation in this article involves 12 issues such as granularity, flexibility, dynamics, role explosion, and interpretability. Ouaddah et al. [25] evaluated multiple models based on six characteristics: scalability, availability, flexibility, context, real-time, granularity, and delegation. Qi et al. [26] analyzed the advantages and disadvantages of RBAC, ABAC, and hybrid models and proposed objective evaluation methods for model granularity, flexibility, and decision efficiency.

Our analysis involves six characteristics, features of the model anonymous access, auditing, restricted object type, fine-grained access control, and policy flexibility. Anonymous access and auditing involve security issues; restricted object type, fine-grained access control, and policy flexibility involve availability. The characteristics included in the security issue are related to anonymous access; the characteristics included in the availability issue are always used to evaluate access control models.

The comparison between AB_sAC and the existing solution is shown in Table 1.

6. Implementation

To verify the availability of AB_sAC and to test the performance difference between AB_sAC and ABAC, AB_sAC is implemented. ABAC implementation [27] contains all the modules and steps in Figure 1. The AB_sAC implementation which contains all the modules and steps in Figure 7 is transformed from the ABAC implementation. The sample attributes and policies used in the implementation are part of the student homework management system. In this implementation, the attributes owned by the student include student number, gender, identity, grade, class, and course. The object is the homework file. Operations include read, update, create, and delete. The functionality and performance of the implementation are tested by randomly generating access requests.

6.1. Functionality Testing. When the homework management system is initializing, execute (i.1) to generate the key for the corresponding module. After the initialization of the system, the user could register. After execution of (r.1) to (r.5), the user obtains the corresponding attribute certificate and could access objects. The user access to the resource process is (ac.1) to (ac.10), resulting in between granted or denied. Users only need to initiate requests; property signatures and other works are automatically processed by the implementation. As shown in Figure 8(a), if a student with the student number 090854 wants to view their C++ homework, the user sends an access request: {object: C++ homework grade 2 class 3.pdf; operation: read}. After the system receives the user's access request, it returns the corresponding signature predicate Y to the user: {job occupation = student or teacher, grade = 2, class = 3}. The client automatically completes the signature and returns the results to the authorization authority. The authorization authority verifies the signature and gives access control results. Figures 8(a) and 8(b) are two access requests, and Figures 8(c) and 8(d) are their corresponding signature predicate and access control results. AB_sAC has achieved the desired function.

6.2. Performance Testing. In the performance test, the factors affecting AB_sAC performance are explored. The performance of the ABAC implementation and the AB_sAC implementation is compared.

Table 2 shows the environment configuration of this experiment. Table 3 shows the policy configuration of this experiment. The attributes are randomly assigned to each user. The test was conducted in 10 rounds for each scheme. Each round of testing consists of system startup, user registration, and 1000 accesses. Access requests are randomly generated. Figure 8 shows the time for the ABAC implementation and the AB_sAC implementation to complete 1000 accesses.

In scheme 1, the AB_sAC implementation takes an average of 2.3 ms to complete a policy evaluation and the evaluation time is stable. As shown in Figure 9, the ABAC implementation and the AB_sAC implementation are not much different in execution efficiency, and AB_sAC is available.

Comparing schemes 1, 2, and 3, they have no significant difference in execution time which is shown in Figure 10. This is because in ABAC implementation, the execution time of finding attributes based on the identity of the subject is constant. The execution time of the HABS increases with the number of attributes, but the impact is very small.

Comparing schemes 1, 4, and 5, the execution time increases as the number of policies increases (Figure 11). The ABAC implementation needs to traverse the policy library when searching for a suitable policy. The execution time of HABS is not affected by the size of the policy library.

In summary, there is no significant difference between the execution time of AB_sAC implementation and ABAC

TABLE 1: Solution comparison.

Literature	Feature	Anonymous access	Fine-grained AC	Policy flexibility	Audit	Restricted object type
[11]	Attribute certificate, k-times anonymity	Y	Y	N	N	N
[10]	Hierarchical CP-ABE, delegation	Y	Y	N	N	Y
[12]	CP-ABE, proxy re-encryption	Y	Y	N	N	Y
[4]	Attribute certificate	Y	Y	N	Y	N
[5]	Attribute certificate, hierarchical structure	Y	N	N	Y	N
[6]	CP-ABPRE	Y	Y	N	N	Y
AB _S AC	Nonunique identifier	Y	Y	Y	Y	N

User request	
Student number	190203
Attributes	job occupation=student , grade=2 , class=3
Object	C++ homework grade 2 class 3.pdf
☒ Read ☐ Write ☐ Delete ☐ Update	
<input type="button" value="Submit"/> <input type="button" value="Reset"/>	

(a)

User request	
Student number	190203
Attributes	job occupation=student , grade=2 , class=3
Object	C++ homework grade 2 class 1.pdf
☒ Read ☒ Write ☐ Delete ☐ Update	
<input type="button" value="Submit"/> <input type="button" value="Reset"/>	

(b)

Request evaluation result	
Evaluation result	Grant
Access control log	
Original request:	
Object : C++ homework grade 2 class 3.pdf	
Operation : Read	
Signature predicate:	
job occupation=student or teacher , grade=2 , class=3	
Access control result :	
Grant	
Time :	
2.2ms	

(c)

Request evaluation result	
Evaluation result	Deny
Access control log	
Original request:	
Object : C++ homework grade 2 class 1.pdf	
Operation : Read, Write	
Signature predicate:	
job occupation=student or teacher , grade=2 , class=1	
Access control result :	
Deny	
Time :	
2.5ms	

(d)

FIGURE 8: Sample access requests and evaluation of the requests.

TABLE 2: Environment configuration.

Operating system	Windows 7
CPU	Intel(R) Xeon(R) CPU E5530 @2.40 GHz
Memory	8 GB

TABLE 3: Policy configuration.

Scheme	Number of attributes	Number of policies
Scheme 1	5	500
Scheme 2	3	500
Scheme 3	7	500
Scheme 4	5	750
Scheme 5	5	1000

FIGURE 9: Performance of ABAC and AB_sAC; ms is milliseconds; blue bar is time consumption of native ABAC implementation; red bar is time consumption of HABS.

FIGURE 10: Performance of schemes 1, 2, and 3.

FIGURE 11: Performance of schemes 1, 4, and 5.

implementation. From a performance perspective, AB_sAC is available. The main factor affecting the execution time of AB_sAC is the number of policies.

7. Conclusion

Existing anonymous access solutions have the problems of subject re-identification and constraints on the types of objects. The ABAC with anonymous access proposed in this paper called AB_sAC inherits the features of the ABAC model, such as fine-grained authorization, policy flexibility, and unlimited object types. By combining HABS, it strengthens the identity-less of ABAC, so that the access does not involve a unique identification, reducing the risk of subject identity re-identification.

In ABAC, the authorization organization has the subject's attributes, subject's identity (subject's unique identifier), and access records. Once the authorization organization becomes abnormal or becomes untrustworthy, the privacy information is leaked, and the consequences are disastrous. The AB_sAC framework proposed in this paper separates PIP subject-related functions from the authorized institution to a trusted third party. The identity of the subject no longer participates in the process of access, which reduces the pressure on the privacy protection of authorized organizations. In the future work, we will build a variety of different IoT application environments to test the performance of AB_sAC in different environments.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported in part by the Shaanxi Key R&D Program under grant no. 2019ZDLGY13-01 and the

National Natural Science Foundation of China under grant no. 61972302.

References

- [1] A. Al-Fuqaha, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: a survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.
- [2] V. Guizani, Juan Antonio Martinez, and A. F. Skarmeta, "User-centric access control for efficient security in smart cities," in *Proceedings of the 2017 Global Internet of Things Summit (GloTS)*, IEEE, Geneva, Switzerland, 2017.
- [3] X. Wang, H. Fu, and L. Zhang, "Research progress on attribute-based access control," *Acta Electronica Sinica*, vol. 7, pp. 1660–1667, 2010.
- [4] M. Backes, C. Jan, and D. Sommer, "Anonymous yet accountable access control," in *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, Alexandria, VA, USA, 2005.
- [5] X. Yao, H. Ning, L. T. Yang, and Y. Xiang, "Anonymous credential-based access control scheme for clouds," *IEEE Cloud Computing*, vol. 2, no. 4, pp. 34–43, 2015.
- [6] Y. Liu, X. Chen, and H. Li, "Anonymous attribute-based proxy re-encryption for access control in cloud computing," *Security and Communication Networks*, vol. 9, no. 14, pp. 2397–2411, 2016.
- [7] A. Li, "Context-aware authorization and anonymous authentication in wireless body area networks," in *Proceedings of the 2018 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, Kansas City, MO, USA, 2018.
- [8] K. Shuaib, "Secure charging and payment protocol (SCPP) for roaming plug-in electric vehicles," in *Proceedings of the 2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 0173–0178, IEEE, Barcelona, Spain, 2017.
- [9] X. Li, S. Kumari, L. Xu, A. K. Sangaiah, and K.-K. R. Choo, "A provably secure and anonymous message authentication scheme for smart grids," *Journal of Parallel and Distributed Computing*, vol. 132, pp. 242–249, 2019.
- [10] R. Wu and S. Mohanty, "A scalable attribute-based access control scheme with flexible delegation cum sharing of access privileges for cloud storage," *IEEE Transactions on Cloud Computing*, vol. 8, no. 1, pp. 32–44, 2017.
- [11] T. H. Yuen, J. K. Liu, M. H. Au, X. Huang, and W. Susilo, "\$K\$ \$-times attribute-based anonymous access control for cloud computing," *IEEE Transactions on Computers*, vol. 64, no. 9, pp. 2595–2608, 2014.
- [12] H. S. Pussewalage and V. Oleshchuk, "A patient-centric attribute based access control scheme for secure sharing of personal health records using cloud computing," in *Proceedings of the 2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)*, pp. 46–53, Pittsburgh, PA, USA, 2016.
- [13] L. Sweeney, "k-anonymity: a model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.
- [14] N. Kaaniche and M. Laurent, "Attribute-based signatures for supporting anonymous certification," in *Proceedings of the European Symposium on Research in Computer Security*, pp. 279–300, Springer, Cham, Switzerland, 2016.
- [15] R. Zhang, "A³BAC: attribute-based access control model with anonymous access," vol. 344, pp. 441–447, in *Proceedings of the International Conference on Security and Privacy in New*

- Computing Environments*, vol. 344, pp. 441–447, Springer, Cham, Switzerland, 2020.
- [16] G. Fedrecheski, “Attribute-based access control for the swarm with distributed policy management,” *IEEE Transactions on Consumer Electronics*, vol. 65, no. 1, pp. 90–98, 2018.
 - [17] Y. Yang, X. Liu, and R. H. Deng, “Lightweight break-glass access control system for healthcare Internet-of-Things,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, p. 3610, 2018.
 - [18] A. L. M. Neto, “Attributed-based authentication and access control for IoT home devices: demo abstract,” in *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 112–113, Porto, Portugal, 2018.
 - [19] M. Qiu, B. Thuraisingham, L. Tao, and H. Zhao, “Proactive user-centric secure data scheme using attribute-based semantic access controls for mobile clouds in financial industry,” *Future Generation Computer Systems*, vol. 80, pp. 421–429, 2018.
 - [20] D. F. Hu, R. Ferraiolo, and D. F. Kuhn, “Guide to attribute based access control (ABAC) definition and considerations (draft),” *NIST Special Publication*, vol. 800, no. 162, 2013.
 - [21] Standard OASIS, “Extensible access control markup language (xacml) version 3.0,” 2013, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.Pdf.
 - [22] D. Servos and S. L. Osborn, “Current research and open problems in attribute-based access control,” *ACM Computing Surveys*, vol. 49, no. 4, pp. 1–45, 2017.
 - [23] H. K. Maji, M. Prabhakaran, and M. Rosulek, “Attribute-based signatures,” in *Proceedings of the Cryptographers’ Track at the RSA Conference*, pp. 376–392, Springer, Berlin, Heidelberg, 2011.
 - [24] U. Muhammad, “The evaluation and comparative analysis of role based access control and attribute based access control model,” in *Proceedings of the 2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pp. 35–39, IEEE, Chengdu, China, 2018.
 - [25] A. Ouaddah, A. Abou Elkalam, and A. Ait Ouahman, “Access control in the Internet of Things: , big challenges and new opportunities,” *Computer Networks*, vol. 112, pp. 237–262, 2017.
 - [26] H. Mousannif, X. Di, and J. Li, “Formal definition and analysis of access control model based on role and attribute,” *Journal of Information Security and Applications*, vol. 43, pp. 53–60, 2018.
 - [27] <https://github.com/JimmyKang13/TestDemoSet>.