# A Category-Based Framework for Privacy-Aware Collaborative Access Control

Denis Obrezkov, Karsten Sohr, and Rainer Malaka

University of Bremen, Bremen 28359, Germany
{obrezkov, sohr, malaka}@uni-bremen.de

**Abstract.** The increased availability of portable devices with high computational power gave birth to such phenomenon as Bring Your Own Device (BYOD)—a situation when an employee uses his own device for accessing enterprise sensitive resources. This situation in turn created a new conflict—an employee wants to keep his data private, and an employer want to preserve the confidentiality of their sensitive resources. Since in case of BYOD both employees' and employers' data are stored on the employee's device, a problem of distributed and collaborative access control appears.
In this paper we propose a novel framework for distributed systems with multiparty data ownership. The underlying formal model is based on the notion of Category-Based Access Control (CBAC). It is expanded with a concept of categories, representing a remote third-party policy decision point. The model is designed and evaluated against requirements for collaborative systems.

**Keywords:** usable security · privacy · BYOD · access control · CBAC

## 1 Introduction

Naturally, modern technologies create new requirements for access control systems. The high prevalence of mobile devices with high computing and communicative capabilities create new challenges in the area of access control.

**Bring Your Own Device phenomenon.** Bring Your Own Device (BYOD) is a situation when an employee can access company-sensitive resources from a device which is not controlled by an employer. To keep his data confidential the employer uses several techniques. One of them is allowing access to the data only via trusted apps installed on the employee's smartphone or laptop. Another technique is usage of corporate-owned personal-enabled mobile devices. In this case an employee is provided with a device controlled by an employer [6].

Garba et al. note that "BYOD has reorganized ownership and control of electronic devices in the present day. It is a new model that replaces the long-standing use what you are told (UWYT) model at work, in which employers define and have full control over the devices used for work-related tasks by their employees. BYOD is a simple strategy that permits staff in an organization to use the device of their choice to access and execute organizational applications

and other information resources"[4]. In other words, in the BYOD situation employers lost their ability to fully control their employees' work devices. The authors list several approaches to protecting the data: network approach, mobile device management, virtualization, and phone-centric approaches. Nevertheless, the employees' privacy issues are out of scope of the paper.

The BYOD phenomenon can be faced in many different fields, e.g. in medical and educational sectors [23, 12, 10]. Wani et al. enumerate BYOD issues in hospitals [23]. Among others, access control issues are identified. Kurniawan et al. also face access control issues while utilizing a BYOD-oriented technology for conducting exams [10]. At the same time little research has been done on the topic of privacy for employees.

Table 1: Requirements for access control models

| Req | Type | Description |
|---|---|---|
| **R1** | Policy Specification | Access control models should be able to manage the increased complexity that multiparty data ownership introduces. |
| **R2** | Policy Specification | Access control models should be able to capture the dynamics and context of relationships of multiple data owners on one system. |
| **R3** | Governance | Access control models should allow the collaborative administration of shared resources. |
| **R4** | Governance | Access control models should support conflict resolution methods. |
| **R5** | Usability | Access control systems should be unobtrusive and should not impose extra overhead on users. |
| **R6** | Usability | Access control systems should support users in the inspection and configuration of their access preferences. |
| **R7** | Transparency | Access control systems should be transparent to users and should allow users to understand collaborative decisions and their effect. |

Since the BYOD phenomenon involves interactions between two or more actors, we consider it to be of a collaborative nature. Paci et al. formulate requirements for a collaborative and community-centered system [17]. On Table 1 we list them slightly modified to better reflect the specifics of the described problem. There are four types of the requirements:

- **Policy Specification.** Given that a new access control framework should address the BYOD problem, where the employer and the employee store their confidential data on the same device, the new access control model should be able to provide data protection for each data owner.
- **Governance.** Several parties have different and sometimes competing access control policies. The employer is interested in the protection of his confidential data and in allowing access for the employee only to the data required to perform the latter's duties. The employee, in turn, is interested in protecting his privacy. Since both sides require data protection all the time (not

only during working hours), it is highly possible that conflicts are inevitable. Thereby, an access control system should be capable of conflicts resolution.
– **Usability and transparency.** In order to efficiently protect data, a user should be able to identify the system's security state, understand possible actions and their consequences, perform the required operations. The efficiency of each step depends on the transparency and usability of the access control system.

In this paper we propose an access control framework which aims to address all the listed requirements in Table 1. We introduce a Shared-CBAC access control policy which is based on the traditional CBAC model [1]. Using a BYOD use case, we demonstrate how to incorporate third-party access control policies into the decision making process by using specific mechanics of our framework and CBAC. Lastly, we evaluate our model against the usability requirement and formulate future work directions.

## 2 Background

### 2.1 Classical access control models

One of the widely known access control models is Mandatory Access Control (MAC). It was defined in The Trusted Computer System Evaluation Criteria (TCSEC) as "a means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity"[11]. Though MAC can be used in distributed systems, usually a user cannot change or create labels and cannot protect his own information in the computer system.

Another popular model is based on Role Based Access Control, or RBAC [19]. RBAC was created to better represent the organizational structure in access control systems. It assigns roles with corresponding privileges to employees and makes security decisions based on these roles. RBAC policies are usually very flexible, for example, it is often possible to create specific roles in the BYOD use case. Nevertheless, it is hardly possible to protect an employee's data on his device—each employee has his own security preferences, thus, a new role is required for protecting each employee's privacy.

In order to address the dynamic nature of computer system environments, Attribute Based Access Control (ABAC) models were created. There are several different varieties of ABAC systems [8, 25, 9]. Their common trait is the ability to change access control decisions based on environmental changes, e.g. owner's age or time of day. Those systems are capable of solving most of the problems in collaborative and distributed access control. At the same time, ABAC-based models do not usually assume the possibility of several security administrators with different needs in data protection. Moreover, it is usually assumed that there is only one security administrator (or a common ancestor node in a security hierarchy), and special steps are needed to allow collaborations [8, 20].

Category-Based Access Control model, or CBAC, is a generalization of ABAC models proposed by Barker [1]. It introduces the notion of category and defines a simple metamodel that can be changed to meet the requirements of a specific area. In CBAC, a category is considered to be a generalization of other access control concepts, like roles, clearance and attributes. In our work we present Shared-CBAC model that is aimed to fulfill the privacy requirements in collaborative environments.

### 2.2 Collaborative and concurrent access models

In order to address the issue of collaborative access control, several models and supporting access control specification languages were proposed.

The XACML language was one of the first attempts to standardize the access control for collaborative environments [24]. The most recent version, XACMLv3, recognizes that some actions require more than one subject. XACML utilizes unique attribute categories to distinguish between subjects with different capacities. At the same time, though being flexible, XACML is far from being transparent or usable for lay users. Stepien et al. enumerate the following factors for its complexity: the length of XML categories, long domain names, a vast collection of operators and structural components [21].

Mahmudlu et al. proposed a data governance model for collaborative environments. The authors classify users into different levels based on their archetypes [13]. One of the major restrictions of this work is the assumption of the existence of one collaboration platform that controls each access request. Although this is a common feature in modern social platforms, this might be not desirable for people who aim to keep their privacy.

Another collaborative access control model, CollAC, also utilizes the archetype concept [3]. It introduces a notion of mismatch—a situation when a user's access control decision differs from a collaborative policy inference. The work also assumes the existence of one central access control platform. Another possible weakness of CollAC is the fact that those mismatches can potentially create big usability issues (Will a user work with the mechanism that sometimes ignores his decisions?).

It should be noted that all previously mentioned works fail to fulfill one or more requirements from Table 1. Some systems, e.g. frameworks based on XACML, are too complex to be transparent and usable for a lay user, others, e.g. based on traditional models, are not capable of allowing efficient collaborative governance. In our work we attempt to build an access control framework that meets all the listed requirements.

## 3 Our solution for privacy-aware access control

In our work we assume that a framework based on the category-based approach will be more usable for lay users. The reason to choose the category-based approach lies in the area of human cognition. It was previously shown that categorization is one of the basic cognition features [5]. Since one of the main heuristics

of usable systems is low memory load and high learnability [16], we choose the notion of categories as the basis for our framework. The evaluation of this design decision is the future work.

### 3.1 Framework architecture

The architecture of our framework is shown in Fig. 1. The figure presents a typical collaboration scenario. There is a user and a third-party. The latter usually wants to keep confidentiality of his data, while the user is more interested in his privacy.

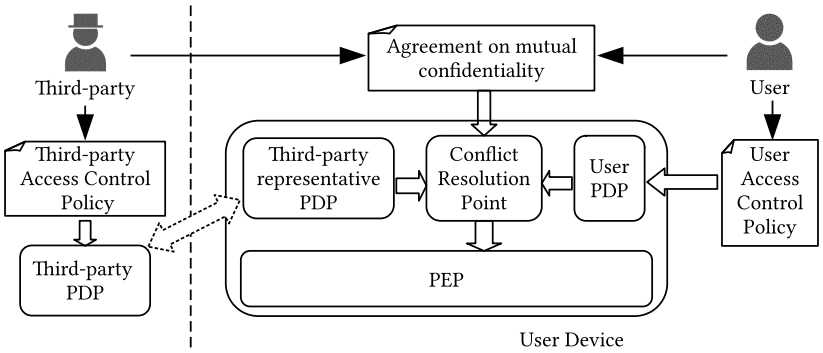



Fig. 1: Framework architecture for systems with multiple data-owners

The third-party usually represents a company with its own access control policy (it might be MAC, RBAC, etc). There is also a Third-party Representative Policy Decision Point (PDP). As the name implies, it serves to represent the third-party in a collaborative decision making. Thus, some user action requests are evaluated on the third-party side. The examples of how Shared-CBAC complements our framework can be found in Section 4.

The access request is performed as follows. A subject requests access for an object. The request is evaluated in the Conflict Resolution Point (CRP). If conflicts are not possible, for example, when the resource is exclusively owned and requested by the user, then only a decision from User PDP is taken into consideration. Otherwise, the request is also sent to all relevant Third-party PDPs and evaluated based on Third-Party Access Control Policies. In the latter case the decisions of Third-party PDPs and the User PDP are evaluated in CRP according to the "Agreements on mutual confidentiality". The final decision is then evaluated in system-wide PDP and passed to the Policy Enforcement Point (PEP).

In order to resolve conflicts, we use a predefined set of rules for conflict resolution. As an employment contract or any other agreement, "Agreement on mutual confidentiality" is made before the collaboration. This design solution

allows one to adequately capture the dynamics of some relations (BYOD), thus meeting the requirements **R2-R4** from Table 1.

To fulfill **R6**, **R7** requirements, we are not considering conflict resolution based on dynamic rules. Firstly, though, some works on collaborative access control might infer decisions opposite to the ones of the user, we believe that a user should clearly understand the consequences of his decision (**R7**). Secondly, a dynamic collaborative decision making might be opaque for a lay user. There is a possibility that a malicious company might utilize weaknesses of the conflict resolution policy, for example, by adding an additional party into a voting process, and get access to the user's data. Thus, we believe that a conflict resolution policy should be static, observable and possible to inspect at any given moment (**R6**).

Considering the aforementioned premises, we propose the following conflict resolution scheme. For each category of interest, several properties are specified in "Agreement on mutual confidentiality": `activation_event`, `release_event`, `on_conflict`. `Activation_event` and `release_event` correspond to arriving into/leaving a working environment, they can have values like *VPN.enabled/ VPN.disabled* or *8:00/17:00*. Thus, `activation_event`:*VPN.enabled* for category "Cat1" means that access requests involving category "Cat1" are processed by the "Third-party representative PDP" after the VPN is enabled. Otherwise, a conflict arises.

The property `on_conflict` specifies possible actions in case of conflict. Possible values are: *deny* and *make_request*. Thus, `on_conflict`:*deny* denies access request in case of a conflict, and `on_conflict`:*make_request* makes request via the "Third-party representative PDP" to the "Third-party PDP" (or it makes request to the "User PDP", since the third-party apps can also try to access user's data).

The weak point of the proposed conflict resolution approach is the inflexibility in big collaborative environments. In situations with multiple conflict resolution agreements, complicated conflicts might arise. The reason for this is a pairwise nature of such contracts, which is inevitable in many situations. For example, in the BYOD use case an employee might have two employers which might require the same camera access time on a user device. It is evident that it is not always possible to make the employers make their own agreement. At the same time, dynamic conflict resolution schemes might also be unsuitable in this situation—two concurring companies generally want a static 'deny' for the other party. In order to satisfy the requirement **R1**, we propose to use static pairwise contracts with priorities.

### 3.2 Shared-CBAC model

Shared-CBAC is based on the Category Based Access model [1]. It operates with the following entities: a countable set of principals $P$, a countable set of resources $R$, a countable set of categories $C$ and a countable set of actions $A$. The metamodel itself is defined with the following relationships:

- *Principal-Category Assignment*, $PCA \subseteq P \times C$, which assigns categories to principals: $(p, c) \in PCA$ iff the principal $p \in P$ is assigned to the category $c \in C$.
- *Resource-Category Assignment*, $RCA \subseteq R \times C$, which assigns categories to the resources in the system: $(r, c) \in RCA$ iff the resource $r \in R$ is in the category $c \in C$.
- *Permitted actions for principals with categories*, $PAPC \subseteq A \times C \times P$, which assigns permitted actions to categories of principals; such that $(a, c, p) \in PAPC$ iff the action $a \in A$ is permitted for the principal with the category $c_p \in C$.
- *Permitted actions for resources with categories*, $PARC \subseteq A \times C \times R$, which assigns permitted actions to categories of resources; such that $(a, c, r) \in PARC$ iff the action $a \in A$ is permitted on the resource with the category $c_p \in C$.
- *Authorizations*, $PAR \subseteq P \times A \times R$, such that $(p, a, r) \in PAR$ iff the principal $p \in P$ is allowed to perform the action $a \in A$ on the resource $r \in R$.

The distinctive feature of our model comparing to classical CBAC is the addition of new relationships: permitted actions for principals with categories ($PAPC$) and permitted actions for resources with categories ($PARC$). This makes our model capable of handling decisions from integrated third-party models in a straightforward manner.

The main idea behind the mapping of principals/resources, categories and actions can be described as follows. Firstly, a category can represent another access control policy. For example, a user can have a category "work_abac" where all decisions are made in accordance with some third-party ABAC policy. Secondly, in order to enable those third-party decisions, we allow assignment of possible actions to pairs of principles/resources and categories. Thus, our model allows us to incorporate other models into decision-making by providing high-grained control on resources via attached actions.

In order to deduce *Authorizations*, we should formulate our core axiom:

$$\begin{gathered}\forall a \in A, \forall p \in P, \forall r \in R \\ (\exists a_r, \forall c_r, \exists c_p, \\ (p, c_p) \in PCA \ \wedge \ (r, c_r) \in RCA \wedge \\ (a_r, c_r, r) \in PARC \ \wedge \ (a, c_p, p) \ \in \ PAPC \wedge \\ a = a_r \ \wedge \ c_r = c_p) \iff (p, a, r) \ \in \ PAR\end{gathered}$$

This set of relationships and the core axiom form our metamodel. In simple words, we allow a security administrator of a device to categorize principals, or subjects (e.g. programs), to categorize resources or objects (e.g. files) and to assign specific actions allowed for each category assigned to a resource or to a principal. The security decision is then made based on the following rule: if a set of a resource's categories is a subset of a principal's categories, then the intersection between actions assigned to the resource categories and to the

principal categories defines the allowed actions. In section 4, the role of a security administrator is distributed among a user and third-parties in accordance with agreements on mutual confidentiality (see Fig. 1).

Since our model can be transformed into a finite CBAC model by introducing mapping categories for the introduced relationships, the same reasoning can be applied to our model to show its decidability as done by Bertolissi et al. [2].

## 4 BYOD use case

Our goal is to show how our framework functions in BYOD scenario and how the Shared-CBAC policy in the system architecture is located.

In Fig. 2 you can see a scheme for a BYOD use case. In this scenario we have an employee (or a user), Company 1, Company 2 and the user's family. Company 1 and Company 2 are the user's employers. A User Device is a device that is used by the user for performing his duties for Company 1 and Company 2.

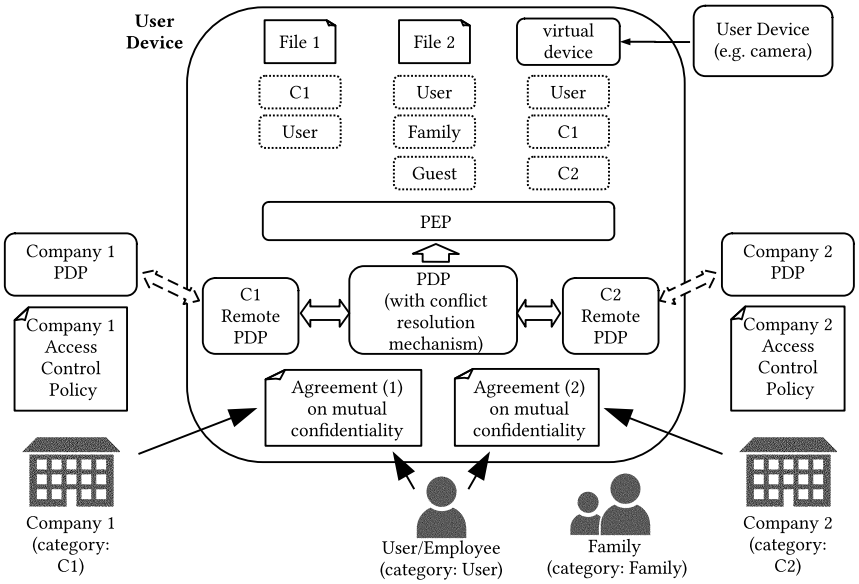



Fig. 2: Access control scheme for BYOD use case

There are several desirable properties in this scenario:

1. the User does not want to share his private information with other entities: Company 1, Company 2 and sometimes with the family
2. Company 1 and Company 2 want to keep confidentiality of their data.

Let us investigate the example. The User has three files: File 1, File 2 and a file, representing a virtual device, a camera in our case. File 1 has the security

categories $C1, User$. This means that only applications that have both categories $C1$ and $User$ can have access to the file. A more fine-grained control is possible if actions are specified: $C1(rw), User(rw)$. In this case an app requesting access should have the same category and the required set of actions for each category ($r$ and/or $w$). For example, if an app has the following security context: $C1(r), User(rw)$ then it can read from the file. At the same time, if the user wants to write to a file with the category $C1(r)$, the conflict arises. The resulting decision is based on a combination of a company response and user categories.

A similar situation occurs when a company application tries to access user files having no appropriate permissions. Let's say that a company app has the categories $C1, User$ and tries to access File 2 with the context $User, Family$, $Guest$. In that case a conflict arises and the access is either denied, or the user is asked to grant the permission to the app (it depends on the specified conflict resolution strategy). Eventually, the user has a privacy-oriented system that allows him to efficiently separate programs and files between security domains of different nature (family, work, personal).

It is worth noting that category $C1$ is just a representation of the fact that the security context is evaluated by a third-party PDP. Let us show how the access control request is processed in general. Firstly, the access control request is processed by the conflict resolution mechanism. At this step, it is decided what policies should be evaluated. Since the requested resource has both a common $User$ and a third-party representing $C1$ category, a default PDP, a remote PDP of C1 called "C1 Remote PDP" and a conflict resolution should be used. The remote PDP can evaluate the access decision on the device or send a request to the remote company server. The final decision is made according to the results of both PDPs and the conflicts resolution policy. This decision is enforced by PEP.

An interesting situation occurs with the creation of new resources. In Fig. 2 there is a virtual device, representing, for example, a camera. Since the device has categories $User$, $C1$ and $C2$, the decision is made based on the default PDP, "$C1$ *Remote PDP*" and "$C2$ *Remote PDP*". Since companies C1 and C2 can be competitors, the conflict resolution policies should be prepared carefully to define a dominance order for shared resources, for example, when one policy has exclusive rights for a speaker device. Generally, it is not a problem for BYOD—typically, employees are constrained temporarily (e.g., the policy *C1* is dominant from 8 a.m. till 1 p.m. for Company 1 and the policy *C2* from 2 p.m. till 6 p.m. for Company 2) or spatially (e.g., the policy C1 is active when NFC signal 'C1-on' was received, the policy C2 is active when NFC signal 'C2-on'; those signals represent entrance checkpoints of the companies).

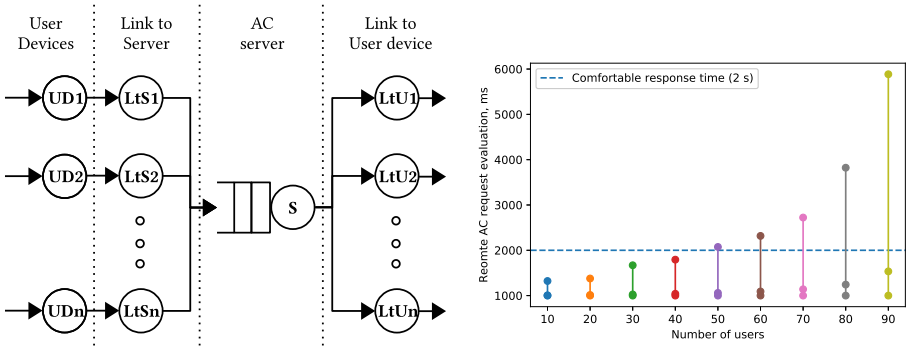
Secondly, it is worth noting that a company policy can be of any nature, e.g. MAC, RBAC and ABAC. For example, a company policy can be ABAC, and the decision can be made based on user id, file id, user age, etc. In this case a security context can only represent a security decision ($r$, $w$, $a$, $e$) and additional information should be stored on a company side.

## 5 Evaluation

It is shown that one of the most significant factors of user satisfaction is a response time [18]. Hoxmeier et al. show a significant positive relation between "ease-of-use" and user satisfaction [7]. Thus, in order to evaluate the feasibility of model implementation in terms of cognitive unobtrusiveness (**R5** requirement from Table 1), we assess how fast a user can get a response with an access control decision from a remote server. At the same time we evaluate the performance of the proposed framework.

### 5.1 Model design

For evaluation we use an $M/D/1$ queuing model. The "$M/D/1$" notion means that the arrival rate is modeled on exponential pdf with the parameter $\lambda$, and the service rate is deterministic and defined by the constant $\mu$, and $\lambda < \mu$. We use the constant rate as a worst-case scenario. The model is shown on Fig. 3a.




(a) M/D/1 queue simulation model (b) Response time from AC server

Fig. 3: Simulation model of a distributed access control environment and the results of response time analysis

There are three types of objects in the model: User Devices ($UD1...UDn$) that generate access control requests, links to and from the Access Control server ($LtS1...LtS2$ and $LtU1...LtU2$) and the Access Control server itself ($S$) with an infinite queue. In our model we use the following assumptions. Firstly, a user makes policy evaluation on its own device and then sends the access request to the Access Control server. We assume that the policy evaluation takes less than 125 ms. We base this assumption on the work of Turkmen et al., where the authors report this number as an upper bound for policy evaluation for XACML Enterprise [22]. Secondly, we assume that the communication link to/from the Access Control server has a speed of 32 kbit/s. The package size is assumed to be of the standard length—1500 bytes. Thirdly, the Access Control server has

the constant service rate. It also performs policy evaluation in 125 ms. Given that constant rate we assume that for 100 users the maximum mean arrival time should be 12500 ms (then the system is stationary). That means that in a company with 100 people each user makes one access request each 12.5 seconds. Lastly, we assume that the arrival rate is exponential. Its parameter is 12500.

### 5.2 Analysis

We ran our simulation for a period of a typical working day of 8 hours. The results for different number of users are shown on Fig. 3b (markers are for minimum, average and maximum response time).

Hoxmeier et al. show that acceptable response time in a computer system lies between 0 and 12 seconds [7]. A two-seconds interval is shown to be acceptable as a response interval for many applications [14]. We use this value as the upper bound of a comfortable response time. Other limiting values could be used as well, but since we aim to measure a feasibility of model implementation in terms of cognitive unobtrusiveness, a perception related boundary seems to be more acceptable.

Figure 3b illustrates how the response time is affected by the number of users. Each response time line is represented by three markers: minimum, average and maximum response times. It is interesting to note that even 50% load on the server gives a maximum response time worse than two seconds.

We can further examine our system in three different modes: low load (10%), high load (90%), maximum load (100%) (see Fig. 4). It can be seen that the low load gives the best response time and all access control requests are processed under two seconds. In the high load mode the majority of requests are also processed less than in two seconds, but there is a significant number of responses with higher latencies. And during the maximum load the system has the majority of response times out of two seconds.

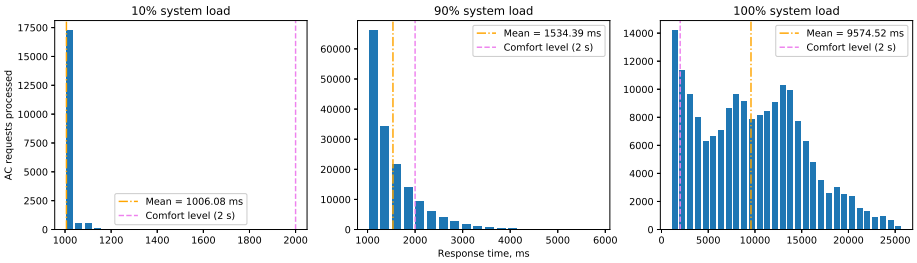



Fig. 4: Response time for different loads

The minimum response time is around 1 second. This response time can be calculated as: $UserDevicePolicyEvaluation + CommunicationLineDelays + ServerPolicyEvaluation$. Given this formula, we can see that there are several

possibilities for decreasing the response time: to lower policy evaluation time (via utilizing better hardware or more efficient policy evaluation software); to use faster communication lines; to change the AC request rate.

In real-life scenarios users/employees have different behavior patterns. Some of them generate several access requests in a small amount of time occasionally, others make requests moderately all through the working day. This leads us to the conclusion that a privacy-oriented system design cannot be a "one-fits-all" solution. At the same time, we have shown that our framework can demonstrate an acceptable response rate, thus addressing the major factor of the usability requirement. A deeper evaluation of our model is a topic of future work.

## 6 Conclusion

In this paper we have discussed problems of modern access control systems in collaborative environments with multiple data owners. We have proposed our framework and model, *Shared-CBAC*, as a solution for the revealed security challenge. We have introduced a notion of third-party decision points, which are represented as usual categories in the model. We have also proposed a static conflict resolution scheme that reflects real-life relationships (e.g. work contracts). Lastly, we have evaluated the usability of our model by measuring response times for access control decisions in a queue-based simulation.

Future work includes several directions to address the aforementioned usability requirements (see Table 1):

*Development of the prototype.* To assess the designed access control framework, a prototype should be developed. It should incorporate PDPs, PEPs and CRPs implementations. Since these modules are located on a user device, special attention should be paid to security. For example, in order to establish trust in the system, special technologies of isolated execution environments, e.g. ARM TrustZone [15], should be utilized. Also, the questions on authorizations and lifecycles of user agreements and categories should be addressed in a secure manner.

*Usability and transparency evaluation.* In our work we assumed that a category-based approach fits better human cognition. A special comparative study is required to assess this preposition. Other associated directions include assessment of means for representing a current access control state of the system and evaluation of supporting security decision mechanisms.

*Evaluation of conflict resolution strategies.* As mentioned before, there are static and dynamic strategies for conflict resolution. In our work we assumed that a static conflict resolution scheme is a better choice for privacy-aware access control systems. It might, however, not be true for all collaborative environments.

**Acknowledgments.** This work was partially supported by Klaus Tschira Foundation (Empowering Digital Media project) and the German Federal Ministry of Education and Research (BMBF) under the grant 16SV8503 (UsableSec@Home project).

## References


1. Barker, S.: The next 700 access control models or a unifying meta-model? In: Proceedings of the 14th ACM Symposium on Access Control Models and Technologies. p. 187–196. SACMAT '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1542207.1542238, https://doi.org/10.1145/1542207.1542238
2. Bertolissi, C., Fernández, M., Thuraisingham, B.: Admin-cbac: An administration model for category-based access control. In: Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy. p. 73–84. CODASPY '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3374664.3375725, https://doi.org/10.1145/3374664.3375725
3. Damen, S., Hartog, J., Zannone, N.: Collac: Collaborative access control. pp. 142–149 (05 2014). https://doi.org/10.1109/CTS.2014.6867557
4. Garba, A.B., Armarego, J., Murray, D., Kenworthy, W.: Review of the information security and privacy challenges in bring your own device (byod) environments. Journal of Information Privacy and Security **11**(1), 38–54 (2015). https://doi.org/10.1080/15536548.2015.1010985, https://doi.org/10.1080/15536548.2015.1010985
5. George, L.: Women, fire, and dangerous things: What categories reveal about the mind. Chicago: University of Chicago (1987)
6. Howell, G.E., Boeckl, K.R., Lefkovitz, N.B., Nadeau, E.M., Franklin, J.M., Shariati, B., Ajmo, J., Brown, C.J., Dog, S.E., Javar, F., et al.: Mobile device security: Corporate-owned personally-enabled (cope) (2020)
7. Hoxmeier, J.A., DiCesare, C.: System response time and user satisfaction: An experimental study of browser-based applications. AMCIS 2000 Proceedings p. 347 (2000)
8. Hu, V., Ferraiolo, D., Kuhn, D., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K.: Guide to attribute based access control (abac) definition and considerations. National Institute of Standards and Technology Special Publication pp. 162–800 (01 2014)
9. Jin, X., Krishnan, R., Sandhu, R.: A unified attribute-based access control model covering dac, mac and rbac. In: Proceedings of the 26th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy. p. 41–55. DBSecÍ12, Springer-Verlag, Berlin, Heidelberg (2012)
10. Kurniawan, O., Lee, N.T.S., Poskitt, C.M.: Securing bring-your-own-device (byod) programming exams. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education. p. 880–886. SIGCSE '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3328778.3366907, https://doi.org/10.1145/3328778.3366907
11. Latham, D.C.: Department of defense trusted computer system evaluation criteria. Department of Defense (1986)
12. Lennon, R.G.: Bring your own device (byod) with cloud 4 education. In: Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity. p. 171–180. SPLASH '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2384716.2384771, https://doi.org/10.1145/2384716.2384771
13. Mahmudlu, R., den Hartog, J., Zannone, N.: Data governance and transparency for collaborative systems. In: IFIP Annual Conference on Data and Applications Security and Privacy. pp. 199–216. Springer (2016)

14. Nah, F.F.H.: A study on tolerable waiting time: how long are web users willing to wait? Behaviour & Information Technology **23**(3), 153–163 (2004). https://doi.org/10.1080/01449290410001669914
15. Ngabonziza, B., Martin, D., Bailey, A., Cho, H., Martin, S.: Trustzone explained: Architectural features and use cases. In: 2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC). pp. 445–451. IEEE (2016)
16. Nielsen, J.: Usability engineering. Morgan Kaufmann (1994)
17. Paci, F., Squicciarini, A., Zannone, N.: Survey on access control for community-centered collaborative systems. ACM Comput. Surv. **51**(1) (Jan 2018). https://doi.org/10.1145/3146025, https://doi.org/10.1145/3146025
18. Rushinek, A., Rushinek, S.F.: What makes users happy? Communications of the ACM **29**(7), 594–598 (1986)
19. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. Computer **29**(2), 38–47 (1996)
20. Servos, D., Osborn, S.: Hgabac: Towards a formal model of hierarchical attribute-based access control. pp. 187–204 (11 2014)
21. Stepien, B., Felty, A., Matwin, S.: A non-technical xacml target editor for dynamic access control systems. In: 2014 International Conference on Collaboration Technologies and Systems (CTS). pp. 150–157 (2014)
22. Turkmen, F., Crispo, B.: Performance evaluation of xacml pdp implementations. In: Proceedings of the 2008 ACM workshop on Secure web services. pp. 37–44 (2008)
23. Wani, T.A., Mendoza, A., Gray, K.: Byod in hospitals-security issues and mitigation strategies. In: Proceedings of the Australasian Computer Science Week Multiconference. ACSW 2019, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3290688.3290729, https://doi.org/10.1145/3290688.3290729
24. XAMCL, Committee, O., et al.: extensible access control markup language (xacml) committee specification 1.0 (2003)
25. Yuan, E., Tong, J.: Attributed based access control (abac) for web services. In: IEEE International Conference on Web Services (ICWS'05). p. 569 (2005)