Dynamic Transitions of States for Context-Sensitive Access Control Decision

A. S. M. Kayes[1], Wenny Rahayu[1], Tharam Dillon[1], Syed Mahbub[1], Eric Pardede[1], and Elizabeth Chang[2]

[1]La Trobe University, Melbourne, Australia
[2]University of New South Wales, Canberra, Australia
{a.kayes,w.rahayu,t.dillon,s.mahbub,e.pardede}@latrobe.edu.au,e.chang@adfa.edu.au

Abstract. Due to the proliferation of data and services in everyday life, we face challenges to ascertain all the necessary contexts and associated contextual conditions and enable applications to utilize relevant information about the contexts. The ability to control context-sensitive access to data resources has become ever more important as the form of the data varies and evolves rapidly, particularly with the development of smart Internet of Things (IoTs). This frequently results in dynamically evolving contexts. An effective way of addressing these issues is to model the dynamically changing nature of the contextual conditions and the transitions between these different dynamically evolving contexts. These contexts can be considered as different states and the transitions represented as state transitions. In this paper, we present a new framework for context-sensitive access control, to represent the dynamic changes to the contexts in real time. We introduce a state transition mechanism to model context changes that lead the transitions from initial states to target states. The mechanism is used to decide whether an access control decision is granted or denied according to the associated contextual conditions and controls data access accordingly. We introduce a Petri net model to specify the control flows for the transitions of states according to the contextual changes. A software prototype has been implemented employing our Petri net model for detection of such changes and making access control decisions accordingly. The advantages of our context-sensitive access control framework along with a Petri net model have been evaluated through two sets of experiments, especially by looking for re-evaluation of access control decisions when context changes. The experimental results show that having a state transition mechanism alongside the context-sensitive access control increases the efficiency of decision making capabilities compared to earlier approaches.

Keywords: Context-Sensitive Access Control, Dynamic Changes to the Contextual Conditions, States, Transitions of States, Petri Net Model

1 Introduction

Access control is a cornerstone of the treatment of security of stored data that has been widely investigated in today's dynamic environments [1]. Among the available access control mechanisms in the literature, the traditional Role-Based

Access Control (RBAC) [2,3] and Attribute-Based Access Control (ABAC) [4] solutions are two representative and reliable security models for many practical applications to safeguard data and information resources. Due to the flexibility in administration when dealing with large number of users in connection with the embodiments of the user-role and role-permission associations, the traditional RBAC model [2] and the spatial and temporal RBAC models [5,6] have been widely accepted by security practitioners and scientific communities. On the other hand, the ABAC models [4] differ from the RBAC models, replacing the roles and other relevant authorities by a set of attributes and grants users' accesses to data based on the relevant attributes that are possessed by the users. However, these traditional role-based and attribute-based access control models are not adequate to incorporate the dynamic contexts into the policies.

The computing technologies have been changed over the last several years and this has created the need for the solution to the problem of controlling data access in today's dynamic environments. Many organizations nowadays have been seeking appropriate context-sensitive access control mechanisms for utilizing data resources. For example, a nurse can access a patient's daily medical records when they both are co-located in the general ward of the hospital (which is a positive policy). On the other hand, a negative policy states that a nurse cannot perform a given action (e.g., read or write) to a patient's records from the outside of the hospital. The existing context-sensitive access control approaches [7,8,9] have used such positive and negative policies [10] to grant access to data according to the contexts. In recent times, we have been moving towards the Internet of Things (IoTs) and the number of IoT sensors are growing at a rapid rate. When different types of IoT conditions which characterize different contexts are collected from these enormous numbers of sensors, which continuously generate a massive amount of data, the traditional context-sensitive access control mechanisms [11,12] (i.e., manually specifying the full set of policies) become infeasible. For example, a doctor should have access to a patient's emergency medical records to save his life from a critical heart attack while he is on the move from his office to the emergency ward.

How can we be sure that the access control decisions can be re-evaluated when there are dynamic changes to the relevant contexts? It is often easy to manually check the relevant contextual conditions that are associated with the applicable access control policies. However, building a required context-sensitive access control system for a dynamic environment is too complicated for the large number of policies. On the one hand, it is really a big challenge to specify the full set of positive and negative policies, due to the presence of the dynamic contextual conditions. On the other hand, there is a need to deal with the issue of dynamicity of contexts in real-time when context changes (e.g., making an access control decision while on the move from general ward to ICU).

In this research, our aim is to introduce a state transition model to evaluate context-sensitive access control decisions when there are dynamic changes to the contextual conditions. The significant contributions are listed as follows. We first introduce the formalization of the state transition model in the treatment of

context-aware access control (CAAC) issues. In particular, the state transition mechanism facilitates access control decision making when the context changes. Using our state transition mechanism, we build a Petri net model in a way that specifies the control flows for the transition of states according to the context changes. The fine-grained access control decisions along with a Petri net model is one of the main contributions in this paper. We implement an android studio-based software prototype along with a colored Petri net for detecting contextual changes and making required access control decisions accordingly. Through two sets of experiments, we evaluate access control decisions including the dynamic changes to the contextual conditions. The experimental results demonstrate the efficiency of decision making capabilities of our proposed approach compared to relevant earlier access control solutions.

The organization of our paper is as follows. Section 2 presents the motivation and hypotheses of this study. Section 3 discusses the formalization and methodology of our proposed state transition model. Section 4 examines the state transition mechanism using Petri net, including a prototype testing. Section 5 evaluates our proposed approach with respect to a relevant earlier approach. Section 6 briefly discusses the related work. Finally, Section 7 concludes the paper and outlines future challenges.

2 Research Motivation

In this section, we present a roadside emergency assistance scenario to motivate our work.

- *The scenario begins with patient James who has experienced a heart attack while driving in a remote area. John, who is a paramedic, has been called to investigate the patient's condition and give him necessary treatments to reduce the risk of a heart attack. John visits the roadside heart attack spot with an ambulance that consists of a body sensor network supporting a wide variety of IoT devices and monitoring applications. He needs to access James' regular medications and all of his previous medical history to treat him properly and save his life from such an emergency heart attack situation.*

The contextual conditions such as locations, request times, relationships and situations are involved in this roadside assistance scenario and consequently the access control decisions are based on the dynamically changing values of such contexts. The relevant user and resource-centric data should be used to extract relevant information about the contexts. In particular, these contexts can be extracted from people entering data like user profiles and from measurement data like IoT data. However, collecting and analyzing IoT contexts from all the sources or sensors are not feasible as there are millions of sensors connected to the Internet. In addition, an access control mechanism or application should have an awareness of relevant information about such contexts and also about dynamic changes to the contexts. For example, John, who is a paramedic and has necessary medical training, should be allowed to access a patient's regular medication and provide necessary pre-hospital treatments to the patient on

roadside environments. The relevant contextual conditions, such as John and James are co-located, need to be checked and satisfied when making an access control decision. However, someone on the roadside is not permitted to provide the required treatments to the patients without necessary medical training.

In general, we need to consider the necessary access control policies with the roles of the person and the relevant contextual conditions as policy constraints. On the one hand, we need to specify all the positive and negative policies [10]. For a given access control request, if there is no relevant policy, the default access control decision is usually taken as "*denied the access*". In absence of such negative policies, the default "denied" decisions may lead to an adverse effect like policy rule conflicts [13]. For example, John, who is normal paramedic, should be permitted to access James' medical history and provide him emergency treatments when there is a potential life-threatening situation. However, the associated contextual conditions need to be assured for granting access to the medical history of James, such as a "life-threatening" situation has occurred. As such, we need to specify corresponding context-sensitive access control policies, including all the possible contextual conditions. In addition, we need to consider the dynamically changing nature of such contextual conditions and consequently re-evaluate the access control decisions when there are dynamic changes to such contexts. For example, when the context changes (e.g., James' situation becomes normal), John should not be allowed to access James' medical history and provide him emergency treatments as roadside assistance.

In the light of the above hypothesis, we can use the basic state transition mechanism [14] to model all the possible "granted" and "denied" states, including all the values of the contextual conditions.

3 Formalization of Our State Transition Mechanism

In this section, we discuss the formalization of state transition mechanism in building our context-sensitive access control framework.

In the literature, the term "context" has been defined by many researchers and the entity-centric concept of context, that has been claimed by Dey [15] provides a general characterization of context in pervasive (context-aware) computing environments. However, Dey's and other earlier definitions of context are not enough to cover dynamic changes to the contexts. Thus, we define the following definition of context that can be used to specify the access control-specific entities and to identify the relevant contextual conditions (i.e., how an access control decision can be made by satisfying a relevant context.).

Definition 1 *(Definition of Context). Context description or simply context is the set of dynamic contextual conditions that are used to make a particular access control decision. These conditions are used to characterize the state of the access control-specific entities. Let us consider "*c_{acd}*" is a context that is composed of a set of contextual conditions "*$cc_1, cc_2, cc_3, ..., cc_i$*" , then we can specify the following expression.*

$$c_{acd} = \{cc_1, cc_2, cc_3, ..., cc_i \mid c_{acd} \in C_{ACD}\} \tag{1}$$

Example 1 *Let us consider a positive policy from our scenario, John, who is a paramedic, is granted to access necessary medical records (MR) of James to save his life from critical heart attack when they are co-located at the scene of the accident. The context (c_{acd_1}) that is associated with this positive policy* $< \mathbf{user}(\mathbf{John}), \mathbf{access}(\mathbf{granted}), \mathbf{data}(\mathbf{MR}) >$ *can be specified as follows.*

$$\begin{aligned} cc_1 &= healthStatus(Patient) = \text{“}critical\text{”} \\ cc_2 &= isColocatedWith(Paramedic, Patient) = \text{“}yes\text{”} \\ c_{acd_1} &\triangleq cc_1 \wedge cc_2 \end{aligned} \tag{2}$$

Example 2 *Let us consider a negative authorization policy from the same application scenario, John, who is a paramedic, is not permitted to access James' medical history (MH) from anywhere of the roadside location where they are not co-located. The context (c_{acd_2}) that is associated with this negative policy* $< \mathbf{user}(\mathbf{John}), \mathbf{access}(\mathbf{denied}), \mathbf{data}(\mathbf{MH}) >$ *can be specified as follows.*

$$c_{acd_2} \triangleq cc_2 \tag{3}$$

The same co-located contextual condition (cc_2) has been associated with both the contexts of c_{acd_1} and c_{acd_2}. There are two different types of contextual conditions, sensed and inferred contextual conditions.

Definition 2 *(Definition of Contextual Conditions). A sensed contextual condition is captured independently from context sources and an inferred contextual condition is derived from available conditions.*

Example 3 *Let us say the location co-ordinates or access request times can be captured directly without using any other conditions. However, the current health status of a patient can be derived from the body sensor network data. Considering an earlier research on fuzzy context information system [9], a patient's current health status is "66% normal with criticality level 34%" that is derived based on the raw contextual facts (e.g., pulse rate).*

3.1 Analysis of Dynamic Changes to the Contexts

Dynamic change to the context is typically driven by a distance measure between the new context and the old context. When there are dynamic changes to any of the contextual conditions that are associated with a context, the distance value indicates how similar the new and old contexts are.

Definition 3 *(Definition of Distance Function). The distance function is defined by the pairwise distances between the contextual conditions that are associated with the new and old contexts. This function is expressed by Equation (4).*

$$distance(c_{acd}^{new},\ c_{acd}^{old}) \triangleq distance(cc_i^{new},\ cc_i^{old}) \tag{4}$$

In the above expression, $distance(c_{acd}^{new}, c_{acd}^{old})$ is the distance between new and old contexts, $distance(cc_i^{new}, cc_i^{old})$ is the pairwise distances between new and old contextual conditions, c^{new} is a new context, c^{old} is an old context, cc_i^{new} is the new contextual condition and cc_i^{old} is an old contextual condition.

The absolute values of the pairwise distances between new and old contextual conditions is used to measure the distance between the new and old contexts. The pairwise distance between the new and old contextual condition is defined as follows (see the Equation 5), where $w_{cc_i}^{new}$ is the weight of the new contextual condition and $w_{cc_i}^{old}$ is the weight of the old contextual condition.

$$distance(cc_i^{new},\ cc_i^{old}) \ \triangleq\ |\, w_{cc_i}^{new} \ -\ w_{cc_i}^{old}\,| \tag{5}$$

We consider the corresponding distance vectors W_{cc_i} of the weights of the new ($w_{cc_i}^{new}$) and old ($w_{cc_i}^{old}$) contextual conditions. The vectors W_{cc_i} are designed to maximize dynamic range of the distances between the contextual conditions.

$$|\, w_{cc_i}^{new} \ -\ w_{cc_i}^{old}\,| \ \triangleq\ \begin{pmatrix} 0 \\ 1 \\ 0 <> 1 \end{pmatrix} \tag{6}$$

In the above equation, the pairwise distance between the new and old contextual conditions is measured based on the range from 0 to 1.

Example 4 *Based on the policy specified in Example 1, the contextual conditions "healthStatus (cc_1)" and "co-located (cc_2)" that are associated with the context. The distance between the new and old contexts is "0", for the first time when the access request is originated from the user and there are no dynamic changes to the contextual conditions at that particular situation (i.e., both the pairwise distances between the new and old contextual conditions are "0"). Based on the Equations (4), (5) and (6), we can write the following expressions.*

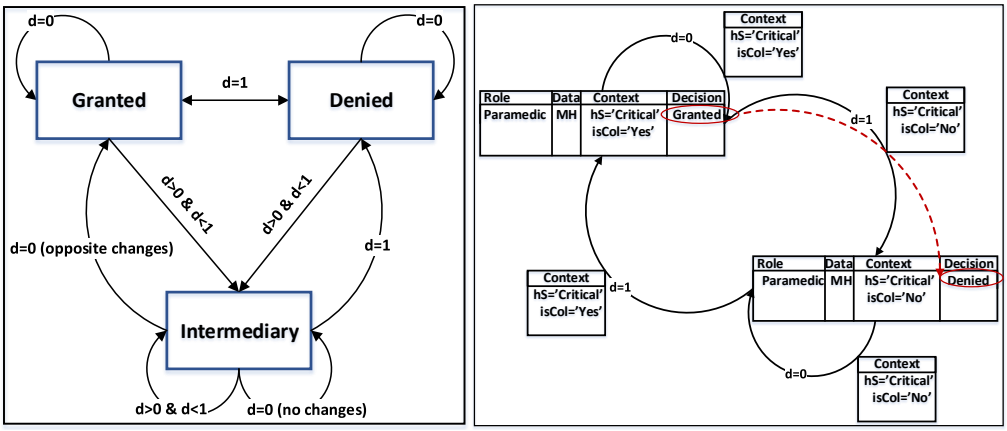
$$\begin{aligned} distance(c_{acd_1}^{new},\ c_{acd_1}^{old}) \ &\triangleq\ |\, w_{cc_i}^{new} \ -\ w_{cc_i}^{old}\,| \\ |\, w_{cc_1}^{new} \ -\ w_{cc_1}^{old}\,| \ &\triangleq\ 0 \\ |\, w_{cc_2}^{new} \ -\ w_{cc_2}^{old}\,| \ &\triangleq\ 0 \end{aligned} \tag{7}$$

3.2 Definitions of States and Analysis of the Transitions of States

We consider all the relevant access control decisions and their associated contexts to define the possible states. We represent the transition of states in terms of contexts involved in making access control decisions and also when there are dynamic changes to such contexts.

Definition 4 *(Definition of State). A state is composed of the role, data resource, context and decision (e.g., "granted" decision). It can be formally described using the following four-tuple notation.*

$$state \ \triangleq < role, data, context, decision > \tag{8}$$



(a) The State Transition Model (b) Example Transitions of States

Fig. 1: Proposed Model for Transitions of States

According to the decision values, we categorize granted, denied and intermediary states and we define these three states as follows.

Definition 5 *(Definition of Granted, Denied and Intermediary States). A granted state means a user who can play the required role can have the complete access to the data, by satisfying the associated context. A denied state means a user cannot have any access to associated data. An intermediary state means a user by playing a role and satisfying the context can have partial access to data.*

The transition of states can be determined using the possible states with the corresponding distance values of the associated contextual conditions. In particular, a state (that we call an initial state) changes to the next state based on such a distance value. The transition of states can be formalized as follows.

$$\begin{aligned} State(s_t) &\xrightarrow{\mathrm{d}} State(s_{t+1}) \\ State(s_t) &\xrightarrow{\mathrm{d}} State(s_{t-1}) \end{aligned} \tag{9}$$

In the above expression, "s_t" is an initial state, "d" is the distance variable according to the Equation (6) and "s_{t+1}" or "s_{t-1}" is the next state.

The state transitions diagram based on the all three basic states (i.e., granted, denied and intermediary) is illustrated in Figure 1(a), where rectangles represent states, arrows connecting states represent transitions and arc labels represent transition conditions. In general, the transition of states occurs as follows, depending on the values of the distance variable "d".

(i) For "*d is 1*" when there are dynamic changes to the contextual conditions, the granted state goes to denied, the denied state goes to granted or an intermediary state goes to denied state.
(ii) For "*d is 0*" when there are no changes, the granted, denied or intermediary state goes to granted, denied or intermediary state respectively.

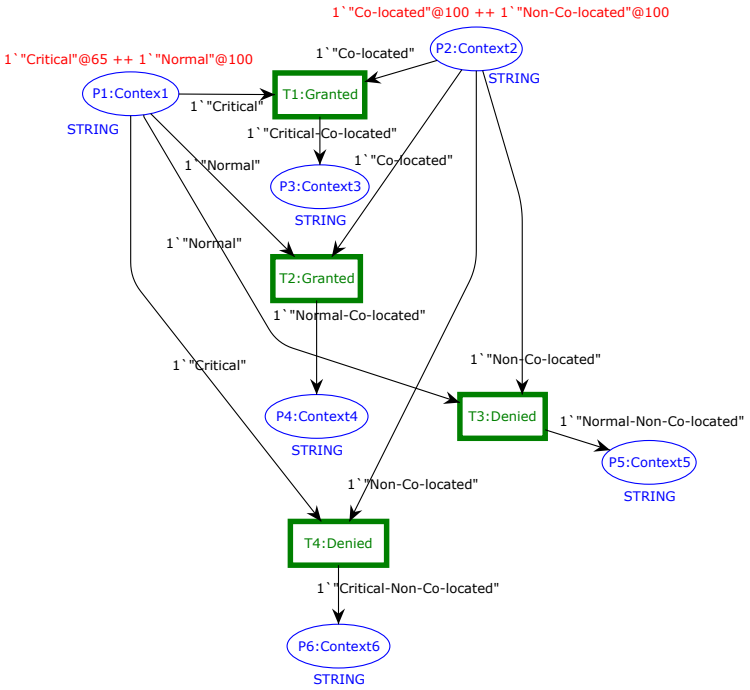


Fig. 2: A Snapshot of Our Petri Net Model

(iii) For "*d is 0<>1*" (i.e., d is greater than 0 but less than 1), the granted, denied or intermediary state goes to intermediary state.
(iv) For "*d is 0*" when there are opposite changes, an intermediary state goes to granted state.

Based on the above-specified conditions in Steps (i) to (iv), we can formalize the following expressions.

$$
\begin{aligned}
s_{granted} &\xrightarrow{\text{d=0}} s_{granted} \xrightarrow{\text{d=1}} s_{denied} \xrightarrow{\text{d>0 \& d<1}} s_{inter} \\
s_{denied} &\xrightarrow{\text{d=0}} s_{denied} \xrightarrow{\text{d=1}} s_{granted} \xrightarrow{\text{d>0 \& d<1}} s_{inter} \\
s_{inter} &\xrightarrow{\text{d=0 (no changes)}} s_{inter} \xrightarrow{\text{d=0 (same opposite changes)}} s_{granted} \\
s_{inter} &\xrightarrow{\text{d>0 \& d<1}} s_{inter} \xrightarrow{\text{d=1}} s_{denied}
\end{aligned}
\tag{10}
$$

Considering our application scenario and based on the positive and negative authorization policies specified in Examples (1) and (2), Figure 1(b) shows several specific transitions between "*Granted*" and "*Denied*" states and their corresponding distance values, according to the dynamic changes to the associated contextual conditions. In Figure 1(b), the granted state goes to denied state when the paramedic and the patient are not co-located at the scene of the heart attack (i.e., *isCol = 'No'*), which is shown by a dotted transition.

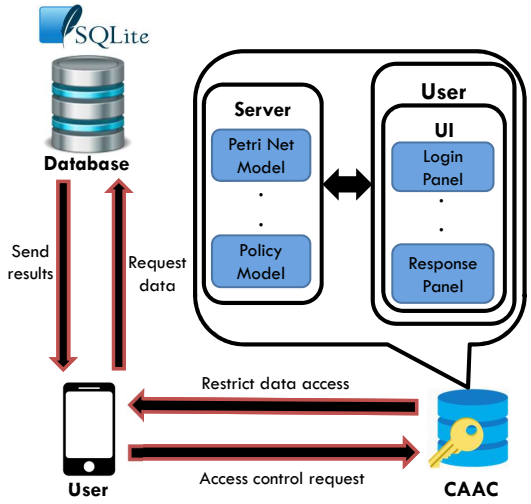


Fig. 3: Development Environment of the Prototype

4 Our Petri Net Model and Prototype Implementation

We introduce a Petri net model for our CAAC decision making process, using CPN Tools [16]. In this model, the CAAC decision making concepts are defined as inputs and outputs (i.e., places), different decisions (i.e., transitions of states), constants or conditions (i.e., guard expressions that are evaluated to fire the transition) and relations between transitions and places (i.e., arcs that link the places and transitions). Figure 2 shows a snapshot of our Petri net model based on our application scenario. For example, a transition "T1:Granted" will be enabled and fired when a relevant condition is 'Critical' and the criticality level 50% to 75% (which is a guard condition). In particular, a transition is fired when (i) the relevant tokens in the input places are equal to the weights of the arcs and these tokens are traveled from the places to the relevant transitions and (ii) the transition conditions according to the guard expressions are satisfied. After firing the transitions, the tokens are distributed to the corresponding output places. In a relevant earlier work, we have introduced a fuzzy context information system to derive fuzzy contextual conditions [9]. In this research, we adapt this fuzzy context model to implement and measure the distance vector for health status context. For example, in Figure 2, we use a temporal condition "Critical@65", which refers to a patient's current health status is "65% Critical". When a patient's current health status is transformed from a 100% normal state to 65% critical state, we measure the distance value (i.e., d) is 0.65. This Petri net model is linked to a context-sensitive access controller to make possible decisions for the users to access the requested data resources.

We have developed our prototype on the Java platform along with Android Studio IDE [17] and other widely supported open source tools. Figure 3 gives a pictorial overview of the tools and technologies that are used to implement different software components of our prototype. We have used SQLite relational database [18] to implement a data store of the patients' medical health records,

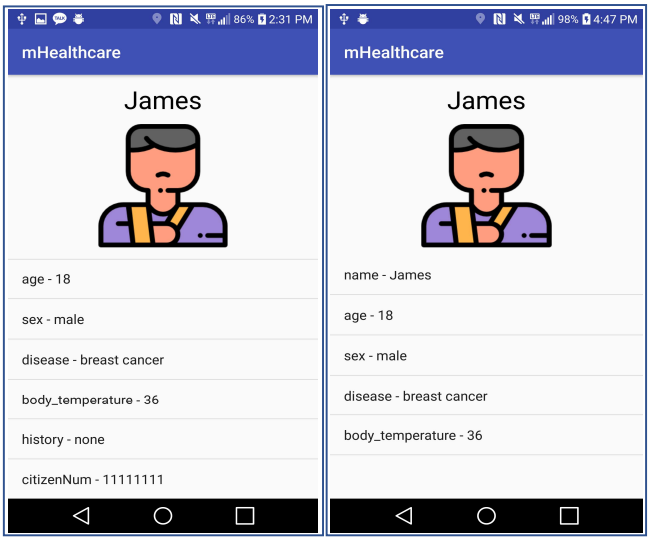
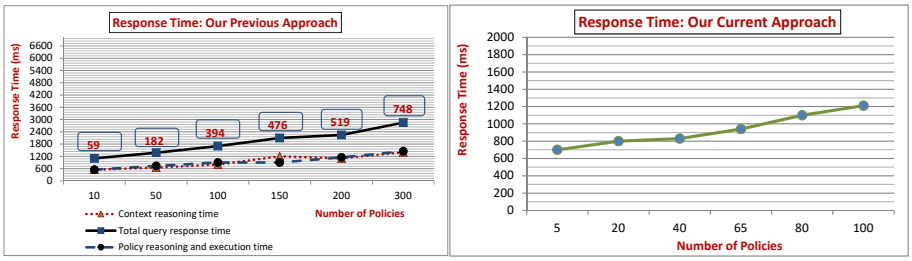


Fig. 4: John's Access Request for James' Health Records

including their previous health history. We have used different XML-based technologies to build the mobile interfaces for our prototype, including the Petri net markup language for our Petri net model [19]. Other than the health status context, we also consider different locations (e.g., the current locations of the patient and paramedic) as contextual conditions in our prototype. We have stored different location coordinates in the relational database and simulated such locations in our laboratory setup. The development environment of our mobile prototype (see Figure 3) has mainly client-side (e.g., user interfaces) and server-side (e.g., Petri-net, policies) parts. We have developed a mobile application and used our implemented user interfaces to deal with different users' requests and the responses accordingly. We actually limit the users' access to data (e.g., a paramedic's access to a patient's normal medical records) according to the associated contextual conditions. Through our proposed state transition mechanism, we also re-evaluate the relevant access control decisions when there are dynamic changes to the contextual conditions.

Figure 4 demonstrates the relevant access control decisions for a paramedic John's request, including the results of a patient James' different medical records. The left part of the Figure 4 shows John is allowed to access James' health records by satisfying the following contextual conditions: when John is co-located with James at the scene of the accident and his health status is critical. Another access request result is shown in the right part of the same Figure 4, where John has only limited access to James' medical records as his health status becomes normal. Overall, we have tested our proposed context-aware access control (CAAC) approach with regards to context changes and the prototype implementation can provide an infrastructure support for the practitioners to build relevant CAAC applications in today's dynamic environments.



(a) Earlier Approach (b) Current Approach

Fig. 5: Performance with Respect to Number of Policies

5 Experimental Evaluation and Verification

We conduct two sets of experiments to evaluate our proposed approach compared to a relevant earlier approach [7]. In particular, we evaluate the access control decisions when there are dynamic changes to the contextual conditions.

In our first set of experiments, we measure the query response time (i.e., performance overheads) with respect to different number of CAAC policies along with relevant contextual conditions. We specify the separate access control policies for the different values of the relevant contextual conditions. As such, according to a relevant earlier approach [7], we have a larger size of ontology knowledge-base, including role, data resource, context and policy ontologies. The experimental results are illustrated in Figure 5(a). In this set of experiments, we can see that the query response time is linearly increasing and for the 300 policies it measures 2.9 sec (and at that point the ontology size is 748 kilobytes). For any large policy-base, what we see that the applied approach is very expensive. This is due to the large number of policies and the complex reasoning task (context and policy reasoning) behind the data access query.

In our second set of experiments, we use our current approach to measure the query response time. In particular, we quantify the performance with respect to different number of context-sensitive access control policies in conjunction with our state transition mechanism (i.e., Petri net model). In this current setup, we don't have a larger size of the policy-base, no complex reasoning task for policy selection and reasoning is involved. Particularly, this is due to using a state transition model with Petri net. The evaluation results are illustrated in Figure 5(b), where we can see that the query response time measures 1.2 sec with respect to 100 policies. This size of the policy-base has covered the 300 policies using an earlier approach. Having a state transition mechanism, the performance can be improved using our current approach and we can achieve fine-grained access control decisions, detecting dynamic changes to the contexts.

In the above experiments that were conducted, our proposed access control approach can detect context changes and consequently facilitates the fine-grained access control decisions when context changes. Having a state transition mechanism, our current setup reduces the number of context-sensitive access control policies incorporating all the values of contextual conditions. However, based on

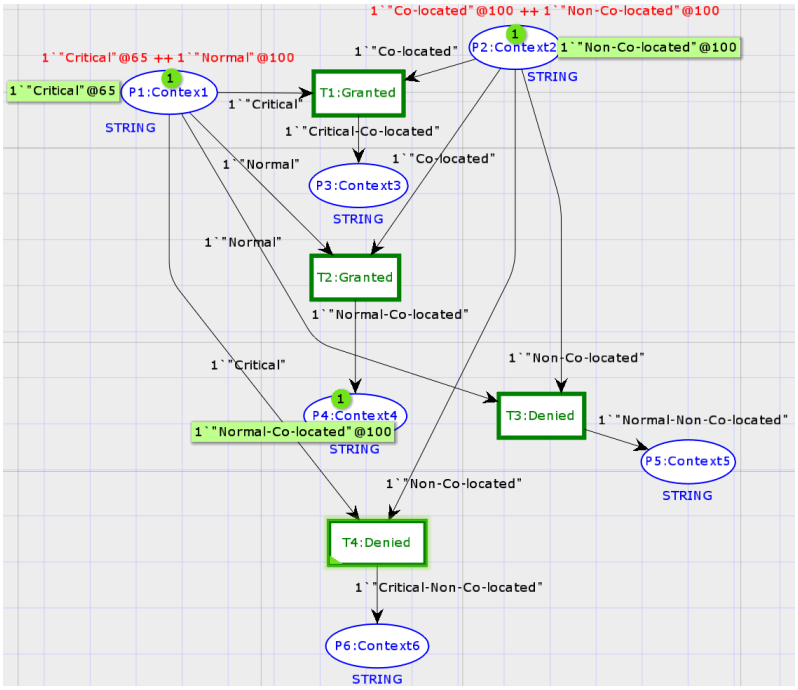


Fig. 6: Simulation Results Using Proposed Petri Net Model

a relevant earlier approach, it is really a challenging job to manually specify the policies covering all the values of contextual conditions. Overall, we can say that our current context-sensitive access control approach has acceptable response time and improves the efficiency of decision making capabilities when there are dynamic changes to the contextual conditions. There is still a possibility to improve performance further by using more powerful computers.

5.1 Verification of Our Petri Net-Based Policy Specification

In this section, we verify the Petri net model that is associated with our context-sensitive access controller by extracting necessary contextual conditions for its correctness. In particular, we verify the correctness of the model through the execution of the Petri net and firing each of the transition. In order to check the correctness of the specification, we apply the top-down approach that starts from the first level of the Petri net model and advances level by level. We know that if the model objects (i.e., places in the Petri net model) are not suitable or available, the relevant transitions cannot be fired. Conversely, if the objects are available and suitable, then the relevant transitions can be enabled for firing.

Figure 6 shows our initial simulation results in which one rule is fired based on the two tokens ("Normal" and "Co-located") and the transition "T2:Granted" is enabled accordingly. We then extract the relevant contextual conditions, changes these conditions values and passes such values as tokens. Then, we check all the transitions again in our Petri net model as well as the relevant status. The

transition "T4:Denied" can be enabled later when there are dynamic changes to the contextual conditions, i.e., when John and James are not co-located with each other and James' health condition is critical. Based on our simulation results in Figure 6, the transition "T4:Denied" is fired later when other two tokens ("Critical" and "Non-Co-located") are traveled from P1 and P2 places to T4, i.e., when there are dynamic changes to the contextual conditions.

Overall, we have checked the status of the relevant transitions (enabled or disabled) when their are the dynamic changes to the contextual conditions. We have changed some contextual conditions through passing tokens and checked the transitions accordingly to verify the correctness of the specification. Other than the above-mentioned correctness approach, we have used CPN Tools [16]) to verify the correctness of our Petri net model and specification.

6 Related Work

In this section, we include a brief review of relevant access control solutions, including the context-sensitive role-based and attribute-based approaches. In particular, we briefly analyze the contributions of our proposed solution compared to existing state-of-the-art context-sensitive access control solutions. Our analysis focuses mainly on the key aspect of our proposed approach, that is, to deal with the dynamic changes to the contextual conditions and consequently re-evaluates access control decisions.

The traditional Role-Based Access Control (RBAC) approach [2] has considered the users' identities and roles as conditions to make access control decisions. The temporal and spatial RBAC approaches [5,6] have considered further conditions like temporal and spatial information for making access control decisions. These approaches are not truly context-aware and do not provide adequate functionalities to integrate dynamic contexts into the access control policies.

On the other hand, the existing Context-Aware role-based Access Control (CAAC) approaches [11,12,7,8,9] incorporate different contextual conditions into the user-role and/or role-permission assignments policies. All the positive and negative policies are specified in these approaches in order to cover all the values of contextual conditions. In this fashion, the complexity of specifying all the policies can be increased when there are dynamic changes to the contexts and subsequently the larger the size of the policy-base. However, these approaches lack in providing support to make access control decisions when there are dynamic changes to the contexts. In our research, we have utilized the benefit of state transition mechanism to deal with such an issue of handling the dynamicity of contextual conditions and consequently making access control decisions.

Other than the RBAC approaches, the Attribute-Based Access Control (ABAC) approaches [4,20,21] are also representative security solutions for many practical applications to protect data. Colombo and Ferrari [21] have used the Oracle Virtual Private Database (VPD) security model for the benefit of context-sensitive access control. They have incorporated the contextual conditions into the relational tables to control the database resources. In the XACML approach, the contextual conditions such as the temporal and spatial information are incorpo-

rated into the policies as conditions [20]. The internal setup of these attribute-based access control approaches are also similar to above-discussed role-based approaches. These approaches also exist the same problem in specifying access control policies by incorporating all the values of relevant contextual conditions. In contrast to these existing role and attribute-based access control approaches, we have significantly reduced the complexity of specifying access control policies when context changes through the state transition mechanism.

Overall, the computing technologies have been changed over the last few years and we need a flexible policy-based solution to access required data resources in today's dynamic environments. As such, there is a need to consider the dynamically changing contextual conditions into the access control policies. The existing access control solutions are not adequate to deal with such dynamicity of contexts. However, based on the dynamic nature of the context information, it is really an important issue to detect dynamic changes to the contextual conditions and to make access control decisions accordingly. Having a state transition mechanism, our proposed context-sensitive access control solution is able to detect dynamic changes to the contextual conditions and consequently re-evaluates access control decisions when context changes.

7 Conclusion and Future Research Directions

We have addressed an important research issue with regards to accessing data when there are dynamic changes to the contexts. The existing context-sensitive access control solutions can lead to a much larger policy-base when incorporating the dynamic values of such contextual conditions into the policies. We have introduced a new context-aware access control (CAAC) solution along with the state transition mechanism to deal with the dynamicity of contexts. First, we have presented a formal model for specifying the state transition mechanism in building our CAAC approach. Then, we have implemented a mobile-based prototype including a colored Petri net model for detecting dynamic changes to the contexts and making access control decisions for the users to access required data resources. We have demonstrated the applicability of our proposed access control approach by evaluating the performance and the correctness of the specification. The experimental results show that our approach along with the state transition model has better performance than a relevant earlier approach.

In this paper, we have considered several contextual conditions and built corresponding distance vectors based on the dynamic changes to such conditions. However, while it is beyond the scope of this paper, it may require special modeling to build such distances between initial and current contextual conditions, which are domain dependent, and thus, further investigation to effectively model such distance vectors is required in the future. We have used the CPN tools to verify the correctness of the specification of our state transition mechanism. In this aspect, further investigation is also required to justify the feasibility of the underlying formalization of the specification. The main purposes of the specification of such state transition mechanism are to (i) identify the dynamic changes to the contextual conditions at runtime and (ii) relate such dynamic conditions to

the applicable policies in such a manner that the context-specific access control decisions can be evaluated.

References

1. Weiser, M.: Some computer science issues in ubiquitous computing. Commun. ACM **36**(7) (1993) 75–84
2. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer **29** (1996) 38–47
3. Wang, H., Cao, J., Zhang, Y.: A flexible payment scheme and its role-based access control. IEEE TKDE **17**(3) (2005) 425–436
4. Servos, D., Osborn, S.L.: Current research and open problems in attribute-based access control. ACM Comput. Surv. **49**(4) (2017) 65:1–65:45
5. Joshi, J.B., Bertino, E., Latif, U., Ghafoor, A.: A generalized temporal role-based access control model. IEEE TKDE **17**(1) (2005) 4–23
6. Damiani, M.L., Bertino, E., Catania, B., Perlasca, P.: GEO-RBAC: a spatially aware RBAC. ACM TISSEC **10**(1) (2007) 2
7. Kayes, A.S.M., Han, J., Colman, A.: OntCAAC: An ontology-based approach to context-aware access control for software services. Comput. J. **58**(11) (2015) 3000–3034
8. Hosseinzadeh, S., Virtanen, S., Rodríguez, N.D., Lilius, J.: A semantic security framework and context-aware role-based access control ontology for smart spaces. In: SBD@SIGMOD. (2016) 1–6
9. Kayes, A., Rahayu, W., Dillon, T., Chang, E., Han, J.: Context-aware access control with imprecise context characterization through a combined fuzzy logic and ontology-based approach. In: CoopIS 2017, Springer (2017) 132–153
10. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The ponder policy specification language. In: POLICY. Springer (2001) 18–38
11. Kulkarni, D., Tripathi, A.: Context-aware role-based access control in pervasive computing systems. In: SACMAT. (2008) 113–122
12. Schefer-Wenzl, S., Strembeck, M.: Modelling context-aware rbac models for mobile business processes. IJWMC **6**(5) (2013) 448–462
13. Sloman, M.: Policy driven management for distributed systems. Journal of network and Systems Management **2**(4) (1994) 333–360
14. Chang, E., Gautama, E., Dillon, T.S.: Extended activity diagrams for adaptive workflow modelling. In: IEEE ISORC-2001. (2001) 413–419
15. Dey, A.K.: Understanding and using context. Personal Ubiquitous Computing **5**(1) (2001) 4–7
16. CPNTools: A tool for editing, simulating, and analyzing colored petri nets: http://cpntools.org/ (2018)
17. Android-Studio-IDE: Android studio for building apps: https://developer.android.com/studio/ (2018)
18. SQLite: It is a self-contained and mostly used sql database engine in the world: https://www.sqlite.org/index.html (2018)
19. PNML: The petri net markup language (pnml) is a proposal of an XML-based interchange format for petri nets: http://www.pnml.org/ (2018)
20. Rissanen, E.: XACML v3.0 core and hierarchical role based access control (RBAC) profile version 1.0, http://http://docs.oasis-open.org/xacml/3.0/rbac/v1.0/xacml-3.0-rbac-v1.0.html. In: OASIS Standard. (2014)
21. Colombo, P., Ferrari, E.: Towards virtual private nosql datastores. In: ICDE, IEEE (2016) 193–204